1 2 9 0

UNIVERSIDADE Đ
COIMBRA

Catarina Filipa Lucas Menino Rodrigues Pinto

MEDICAL PRESCRIPTION IN HOSPITAL ENVIRONMENT

Dissertation in the context of the Master's in Informatics Engineering, specialization in Intelligent System, advised by Prof. Catarina Silva, by Eng. Pedro Faustino and by Eng. Henrique Batista and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July 2023

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Ð
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Catarina Filipa Lucas Menino Rodrigues Pinto

Medical Prescription in Hospital Environment

Dissertation in the context of the Master's in Informatics Engineering, specialization in Intelligent System, advised by Prof. Catarina Silva, by Eng. Pedro Faustino and by Eng. Henrique Batista and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July 2023

Acknowledgements

The first and most important thank you is for you, who, from day one, were my biggest fan, supported me in everything, and lived every achievement of mine as if it were yours. Wherever you are, you have a little bit of me and left me a little bit of you. Thank you for teaching me everything you knew and making me who I am today. For having shown me by your example what a good, beautiful, and pure human being is. For having taught me one of the most beautiful loves that can exist and always being my best friend and second mother, always proud of your little girl. This achievement is by and for you.

Secondly, I would like to thank my parents for allowing me to be where I am today and supporting me in every step of this journey. For being by my side in good and not-so-good times, always proud of my personal and academic journey. For all the love and affection they have always given me, for vibrating with all my achievements, and for showing all their pride in me.

Also, thank my brothers and my niece for all the learning, support, moments, affection, and love. In particular, thank Gonçalo for being my biggest critic and source of growth from day one, contributing significantly to my professional development. For supporting me unconditionally throughout these years and always showing me the simpler side.

I want to thank my grandparents for always supporting me, showing pride in me, and giving me all the conditions to go through this journey. To the rest of my family, thank you for all your sharing and support.

I want to thank Bono, Buddy, Fluffy, Lady, Moon, Nikki, and Yoshi for being my support and for all the love, happiness, and peace they transmit to me.

I thank my boyfriend for all his support and patience. For being my right arm. For being calm in the middle of the storms. For being with me at all times. For making everything simpler. For supporting me. For being proud of me unconditionally and for vibrating with my achievements. Also, thank Simba for the joy transmitted.

A thank you to my goddaughter Carolina, Nuno, and Rafinha, who always supported me in so many years of friendship, showed pride, and provided memorable moments.

Thank you to Anelise, Constança, David, Eric, Gonçalo, Jéssica, João Bernardo, João Vasco, Jordão, Nuno, Pires, Rui, and Tiago for making these years lighter, for always being with me and for all the good times shared.

I would also like to thank Ana Marques, Carolina, Catarina, and Gabriela, who accompanied me for almost five years. Thank you for all the sharing and good times.

I want to thank my supervisors, Catarina Silva, Henrique Batista, and Pedro Faustino, for all the support, knowledge sharing, and growth they have instilled in me. I also thank my colleagues at MedicineOne, who extraordinarily wel-

comed me. In particular, thank you to João Miguel for all the motivation and recognition and to Rute for the excellent welcome.

I also thank Professor João Fernandes for all the support and for always believing in my potential, motivating me. I thank the Introduction to Artificial Intelligence teachers who, even without realizing it, led me to take a new direction in my academic training, culminating in this master's degree and unmeasured happiness and fulfillment. I would also like to thank, in general, all the teachers who crossed my path, the Departments of Physics and Computer Engineering that welcomed me, and the University of Coimbra for all the opportunities and learning.

Finally, an exceptional thanks to the company that welcomed me and always made me feel at home. Thank you for the unconditional support, experiences, sharing knowledge, and for making me see that the world of work can be wonderful with the right people by your side.

Abstract

Medical prescription in hospital environment, also known as internal prescription, consists of a periodic task in which doctors should visit the patient, reassess them, re-evaluate the current therapy, and, when necessary, add, remove, or change some components. The doctor should repeat this task typically daily for all the hospitalized patients under his/her responsibility. In the reality of the health units that use MedicineOne's software, the M1, each patient visit is accompanied by a nursing trolley that includes, among other things, a laptop computer. This computer serves both for information acquisition and for manual insertion of new information on M1. This manual insertion of data into M1 consumes excessive time, contributing to the overload of doctors. With that, the time allocated to healthcare tasks is reduced, overwork makes doctors more susceptible to failure, and as a consequence, the quality of care is likely to be lower.

This project focuses on developing a support system to optimize the internal prescription process. This solution uses both Automatic Speech Recognition and Natural Language Processing – namely entity recognition and relationship extraction – technologies so that doctors can replace the manual insertion of information on M1 by dictating the same information to their phone in natural language. For that, the system includes a mobile application for iOS, in which the doctors record their dictation. Through transcription, interpretation, and processing of the medical dictation, the system should identify the relevant information, process it, and return it structured to the user. By its turn, the user can edit that information and, when satisfied with the result, can simulate its sending to M1, updating the patient's current therapy. When it is inappropriate to dictate the prescription, the doctor can enter it textually into the application, being that text interpreted and processed in the same way.

Testing was done during the development of the system. In these, a performance of 88.02% was achieved with the system's Automatic Speech Recognition component and the Natural Language Processing component achieved 86.33% performance for entity recognition and 83.60% for relationship detection. Besides that, the tests verified both the task's success when done using the developed system and the reduction in the time required to do it.

Keywords

Internal prescription; Automatic Speech Recognition; Natural Language Processing; Mobile application; iOS; Amazon Web Services; Microsoft Azure; Swift; Portuguese; Portugal.

Resumo

A prescrição médica em ambiente hospitalar, também conhecida como prescrição interna, consiste numa tarefa periódica na qual os médicos devem visitar o paciente, reavaliá-lo, reavaliar a terapêutica actual e, quando necessário, acrescentar, retirar ou alterar algum componente da mesma. Esta tarefa deve ser repetida tipicamente diariamente para todos os pacientes em contexto de internamento pelos quais o médico é responsável. Na realidade das unidades de saúde que utilizam o software do MedicineOne, o M1, a visita a cada um dos pacientes é acompanhada por um carrinho de enfermagem que inclui, entre outros, um computador portátil. Este computador serve tanto para a obtenção de informação como para a inserção manual de nova informação no M1. Esta inserção manual da informação no M1 consome tempo excessivo, contribuindo para a sobrecarga dos médicos. Com isto, o tempo destinado às tarefas de saúde é reduzido, o excesso de trabalho torna os médicos mais susceptíveis a falhas e, como consequência, a qualidade dos cuidados de saúde é susceptível de ser inferior.

Para optimizar o processo de prescrição interna, este projecto centra-se no desenvolvimento de um sistema de apoio à tarefa. Esta solução recorre a tecnologias de Reconhecimento Automático de Fala e de Processamento de Linguagem Natural – nomeadamente reconhecimento de entidades e extração de relações – para que os médicos possam substituir a inserção manual de informação no M1 pelo ditado em linguagem natural da mesma informação para o seu telemóvel. Para tal, o sistema inclui uma aplicação móvel para iOS, na qual é gravado o ditado do médico. Através da transcrição, interpretação e processamento do ditado médico, o sistema deve identificar a informação relevante no mesmo, processá-la e devolvê-la estruturada ao utilizador. Por sua vez, o utilizador pode editar essa informação e, quando satisfeito com o resultado, pode simular o seu envio para M1, actualizando a terapêutica actual do doente. Quando não for adequado ditar a prescrição, o médico pode introduzi-la textualmente na aplicação, sendo esse texto interpretado e processado da mesma forma.

Durante o desenvolvimento do sistema foram efectuados testes. Nestes, uma performance de 88,02% foi obtida com a componente de Reconhecimento Automático de Fala do sistema, tendo a componente de Processamento de Linguagem Natural obtido uma performance de 86,33% para o reconhecimento de entidades e de 83,60% para a deteção de relações. Além disto, os testes verificaram quer o sucesso da tarefa quando efetuada com recurso ao sistema quer a redução do tempo necessário para a sua realização.

Palavras-Chave

Prescrição interna; Reconhecimento Automático de Fala; Processamento de Linguagem Natural; Aplicação móvel; iOS; Amazon Web Services; Microsoft Azure; Swift; Português; Portugal.

Contents

1 Introduction 1
1.1 Motivation . . . 1
1.2 Goals . . . 1
1.3 Contributions . . . 2
1.4 Document structure . . . 3

2 Internal prescription 5
2.1 Introduction . . . 5
2.2 Internal prescription in Portuguese health units . . . 6
2.3 Artificial Intelligence and Computational Linguistics in medical prescription . . . 7
2.3.1 Introduction . . . 7
2.3.2 State of the art . . . 8
2.4 Conclusion . . . 20

3 Text Processing Pipeline 21
3.1 Introduction . . . 21
3.2 Speech-to-Text . . . 21
3.2.1 Introduction . . . 21
3.2.2 Main Speech-to-Text Tools . . . 22
3.3 Natural Language Processing . . . 39
3.3.1 Introduction . . . 39
3.3.2 Main Natural Language Processing Tools . . . 40
3.4 Conclusion . . . 63

4 Software Development Methodology 65
4.1 Introduction . . . 65
4.2 Cloud provider . . . 65
4.2.1 Cloud Services . . . 67
4.3 Scrum methodology . . . 75
4.4 User Interface . . . 78
4.5 Conclusion . . . 80

5 Competitors analysis 83
5.1 Introduction . . . 83
5.2 Comparison of Automatic Speech Recognition solutions . . . 84
5.2.1 Amazon Transcribe . . . 85
5.2.2 Microsoft Azure Cognitive Services for Speech . . . 85
5.2.3 Google Cloud Speech-to-Text . . . 85

5.2.4 Whisper 85
5.2.5 iOS Speech 85
5.2.6 Results and conclusion 85
5.3 International Non-proprietary Name entities detection 87
5.3.1 Default models 88
5.3.2 Customized models 94
5.4 Entity recognition and relationship extraction in prescriptions . . . 105
5.4.1 Dataset generation 106
5.4.2 Amazon Comprehend 108
5.4.3 Google Cloud Healthcare Natural Language API 110
5.4.4 Microsoft Azure Cognitive Service for Language 116
5.4.5 Microsoft Azure Text Analytics for Health 121
5.4.6 Conclusion 123
5.5 Entity recognition and relationship extraction in multi-medication prescriptions 127
5.6 Conclusion 128

6 Approach **129**
6.1 Problem definition and solution architecture 129
6.2 Comparison with the state of the art of medical prescription 133
6.3 API 136
6.3.1 Cloud Provider 136
6.3.2 Programming Language 137
6.3.3 Tools 137
6.3.4 Development 137
6.4 Mobile application 144
6.4.1 Requirements 144
6.4.2 Solution mockups 147
6.4.3 User Interface Architecture 151
6.4.4 Programming Language 151
6.4.5 Tools 151
6.4.6 Development 151

7 The System **157**
7.1 Introduction 157
7.2 Prescription 157
7.3 Suspension 158
7.4 Changing 159
7.5 Conclusion 160

8 Conclusion and Future Work **161**
8.1 Conclusion 161
8.2 Future work 163

Appendix A CMUSphinx components' architecture **181**

Appendix B Kaldi components' architecture **185**

Appendix C Whisper models' architecture **187**

Appendix D spaCy models components' architecture **189**

Appendix E NLTK components' architecture **201**

Appendix F Test dataset with single-INN prescriptions **211**

Appendix G Confusion matrices **215**
G.1 Amazon Comprehend 215
G.2 Google Cloud Healthcare Natural Language API 222
G.2.1 After post-processing 231
G.3 Microsoft Azure Cognitive Service for Language 238
G.4 Microsoft Azure Text Analytics for Health 245
G.4.1 After post-processing 253

Appendix H Application use cases illustration **261**

Acronyms

AI	Artificial Intelligence.
Amazon EC2	Amazon Elastic Compute Cloud.
API	Application Programming Interface.
ASR	Automatic Speech Recognition.
AWS	Amazon Web Services.
CL	Computational Linguistics.
CMLLR	Constrained Maximum Likelihood Linear Regression.
CPU	Central Processing Unit.
CRF	Conditional Random Fields.
CSV	Comma-Separated Value.
DL	Deep Learning.
DST	Dialog State Tracking.
EHR	Electronic Health Record System.
fMLLR	Feature-Space Maximum Likelihood Linear Regression.
FST	Finite-State Transducer.
GMM	Gaussian Mixture Model.
gRPC	Google Remote Procedure Call.
HIPAA	Health Insurance Portability and Accountability Act.
HMM	Hidden Markov Model.
INN	International Non-proprietary Name.
IR	Information Retrieval.
JWT	JSON web token.
ML	Machine Learning.

MLLR Maximum Likelihood Linear Regression.

MVC Model-View-Controller.

MVP Model-View-Presenter.

MVVM Model-View-ViewModel.

NER Named Entity Recognition.

NLG Natural Language Generation.

NLP Natural Language Processing.

NLU Natural Language Understanding.

PCM Pulse-Code Modulation.

PHI Protected Health Information.

PII Personally Identifying Information.

PMS Prescription Management Systems.

POS Part-of-Speech.

SGMM Subspace Gaussian Mixture Model.

SLU Spoken Language Understanding.

SSD Solid State Disk.

STT Speech-to-Text.

TL Theoretical Linguistics.

TTS Text-to-Speech.

UI User Interface.

UX User Experience.

VTLN Vocal Tract Length Normalization.

WER Word Error Rate.

WFST Weighted Finite State Transducer.

List of Figures

2.1 Ilustration of the internal prescription process [pressfoto] [pre, a] [pre, b] [pre, 2022]. 5

3.1 spaCy trained pipelines' schema. 59

4.1 Comparison of leader cloud providers from Infrastructure as a Service perspective [Aljamal et al., 2019]. 66
4.2 Mobile backend usage scenario. 68
4.3 Amazon API Gateway architecture. 69
4.4 Model-View-Controller (MVC) components organization. 78
4.5 Model-View-ViewModel (MVVM) components organization. . . . 79
4.6 Model-View-Presenter (MVP) components organization. 79

5.1 Flow chart for the test made for ASR solutions evaluation. 84
5.2 Flow chart for the tests made for INN entities detection evaluation. 87
5.3 Confusion matrix obtained for INN entity recognition with Amazon Comprehend Medical. 90
5.4 Confusion matrix obtained for INN entity recognition with Google Cloud Healthcare Natural Language API. 91
5.5 Confusion matrix obtained for INN entity recognition with Microsoft Azure Text Analytics for Health for entities in European Portuguese. 92
5.6 Confusion matrix obtained for INN entity recognition with Microsoft Azure Text Analytics for Health for entities in English. . . . 95
5.7 Confusion matrix obtained for INN entity recognition with Amazon Comprehend. 97
5.8 Confusion matrix obtained for INN entity recognition with Microsoft Azure Cognitive Service for Language. 100
5.9 Confusion matrix obtained for INN entity recognition with spaCy sm model. 104
5.10 Confusion matrix obtained for INN entity recognition with spaCy md model. 104
5.11 Confusion matrix obtained for INN entity recognition with spaCy lg model. 105
5.12 Flow chart for the tests made for INN entities and relations detection evaluation. 106
5.13 Confusion matrix obtained for entity recognition with Amazon Comprehend. 110

5.14 Confusion matrix obtained for relationship detection with Amazon Comprehend. 112
5.15 Confusion matrix obtained for entity recognition with Google Cloud Healthcare Natural Language API. 113
5.16 Confusion matrix obtained for relationship detection with Google Cloud Healthcare Natural Language API. 114
5.17 Confusion matrix obtained for entity recognition with Google Cloud Healthcare Natural Language API with post-processing. 116
5.18 Confusion matrix obtained for relationship detection with Google Cloud Healthcare Natural Language API with post-processing. . . 118
5.19 Confusion matrix obtained for entity recognition with Microsoft Azure Cognitive Service for Language. 119
5.20 Confusion matrix obtained for relationship detection with Microsoft Azure Cognitive Service for Language. 120
5.21 Confusion matrix obtained for entity recognition with Microsoft Azure Text Analytics for Health. 122
5.22 Confusion matrix obtained for relationship detection with Microsoft Azure Text Analytics for Health. 124
5.23 Confusion matrix obtained for entity recognition with Microsoft Azure Text Analytics for Health with post-processing. 125
5.24 Confusion matrix obtained for relationship detection with Microsoft Azure Text Analytics for Health with post-processing. 126

6.1 Architecture of the developed solution [aws, 2022] [Halfpoint, 2018]. 130
6.2 Components diagram of the solution to be developed [aws, f] [Microsoft, 2023]. 130
6.3 Example of entity detection on multi-medication prescription. . . . 132
6.4 Handy's screens [m1, 2023]. 133
6.5 MedicineOne's mockups for the solution. 148
6.6 Solution's mockups. 150
6.7 Login screen. 152
6.8 PrescriptionOrRecording screen. 153
6.9 Processed screen. 154
6.10 Changes screen. 155

A.1 Gaussian Mixture Model (GMM)s formula. 182
A.2 Gaussian density function. 182

C.1 Whisper's model architecture. 187

D.1 Default configuration of spaCy's European Portuguese pipeline Tok2Vec component. 192
D.2 Default configuration os spaCy's European Portuguese pipeline Morphologizer component. 193
D.3 Default configuration of spaCy's European Portuguese pipeline Dependency Parser component. 195
D.4 Default configuration of spaCy's European Portuguese pipeline Lemmatizer component. 197

D.5 Default configuration of spaCy's European Portuguese pipeline Sentence Recognizer component. 197
D.6 Default configuration of spaCy's European Portuguese pipeline Named Entity Recognition (NER) component. 199

G.1 Confusion matrix obtained for negative entity recognition with Amazon Comprehend. 215
G.2 Confusion matrix obtained for negative relation detection with Amazon Comprehend. 216
G.3 Confusion matrix obtained for dosage entity recognition with Amazon Comprehend. 216
G.4 Confusion matrix obtained for beginning entity recognition with Amazon Comprehend. 217
G.5 Confusion matrix obtained for ending entity recognition with Amazon Comprehend. 217
G.6 Confusion matrix obtained for duration entity recognition with Amazon Comprehend. 218
G.7 Confusion matrix obtained for time of medication relation detection with Amazon Comprehend. 218
G.8 Confusion matrix obtained for frequency entity recognition with Amazon Comprehend. 219
G.9 Confusion matrix obtained for frequency of medication relation detection with Amazon Comprehend. 219
G.10 Confusion matrix obtained for medication entity recognition with Amazon Comprehend. 220
G.11 Confusion matrix obtained for route entity recognition with Amazon Comprehend. 220
G.12 Confusion matrix obtained for route of medication relation detection with Amazon Comprehend. 221
G.13 Confusion matrix obtained for negative entities recognition with Google Cloud Healthcare Natural Language API. 222
G.14 Confusion matrix obtained for negative relationship detection with Google Cloud Healthcare Natural Language API. 222
G.15 Confusion matrix obtained for anatomical structure entities recognition with Google Cloud Healthcare Natural Language API. . . . 223
G.16 Confusion matrix obtained for dosage entities recognition with Google Cloud Healthcare Natural Language API. 223
G.17 Confusion matrix obtained for dosage of medication relationship detection with Google Cloud Healthcare Natural Language API. . 224
G.18 Confusion matrix obtained for beginning entities recognition with Google Cloud Healthcare Natural Language API. 224
G.19 Confusion matrix obtained for ending entities recognition with Google Cloud Healthcare Natural Language API. 225
G.20 Confusion matrix obtained for duration entities recognition with Google Cloud Healthcare Natural Language API. 225
G.21 Confusion matrix obtained for time of medication relationship detection with Google Cloud Healthcare Natural Language API. . . . 226

G.22 Confusion matrix obtained for dose entities recognition with Google Cloud Healthcare Natural Language API. 226
G.23 Confusion matrix obtained for frequency entities recognition with Google Cloud Healthcare Natural Language API. 227
G.24 Confusion matrix obtained for frequency of medication relationship detection with Google Cloud Healthcare Natural Language API. 227
G.25 Confusion matrix obtained for route entities recognition with Google Cloud Healthcare Natural Language API. 228
G.26 Confusion matrix obtained for route of medication relationship detection with Google Cloud Healthcare Natural Language API. . . . 228
G.27 Confusion matrix obtained for status of medication entities recognition with Google Cloud Healthcare Natural Language API. . . . 229
G.28 Confusion matrix obtained for medication strength entities recognition with Google Cloud Healthcare Natural Language API. . . . 229
G.29 Confusion matrix obtained for medication unit entities recognition with Google Cloud Healthcare Natural Language API. 230
G.30 Confusion matrix obtained for medication entities recognition with Google Cloud Healthcare Natural Language API. 230
G.31 Confusion matrix obtained for negative entities recognition with Google Cloud Healthcare Natural Language API after post-processing. 231
G.32 Confusion matrix obtained for negative relationship detection with Google Cloud Healthcare Natural Language API after post-processing. 231
G.33 Confusion matrix obtained for dosage entities recognition with Google Cloud Healthcare Natural Language API after post-processing. . . 232
G.34 Confusion matrix obtained for dosage of medication relationship detection with Google Cloud Healthcare Natural Language API after post-processing. 232
G.35 Confusion matrix obtained for beginning entities recognition with Google Cloud Healthcare Natural Language API after post-processing. 233
G.36 Confusion matrix obtained for ending entities recognition with Google Cloud Healthcare Natural Language API after post-processing. . . 233
G.37 Confusion matrix obtained for duration entities recognition with Google Cloud Healthcare Natural Language API after post-processing. 234
G.38 Confusion matrix obtained for time of medication relationship detection with Google Cloud Healthcare Natural Language API after post-processing. 234
G.39 Confusion matrix obtained for frequency entities recognition with Google Cloud Healthcare Natural Language API after post-processing. 235
G.40 Confusion matrix obtained for frequency of medication relationship detection with Google Cloud Healthcare Natural Language API after post-processing. 235

G.41 Confusion matrix obtained for route entities recognition with Google Cloud Healthcare Natural Language API after post-processing. . . 236
G.42 Confusion matrix obtained for route of medication relationship detection with Google Cloud Healthcare Natural Language API after post-processing. 236
G.43 Confusion matrix obtained for medication strength entities recognition with Google Cloud Healthcare Natural Language API after post-processing. 237
G.44 Confusion matrix obtained for medication entities recognition with Google Cloud Healthcare Natural Language API after post-processing. 237
G.45 Confusion matrix obtained for negative entity recognition with Microsoft Azure Cognitive Service for Language. 238
G.46 Confusion matrix obtained for negative relationship detection with Microsoft Azure Cognitive Service for Language. 238
G.47 Confusion matrix obtained for dosage entity recognition with Microsoft Azure Cognitive Service for Language. 239
G.48 Confusion matrix obtained for dosage of medication relationship detection with Microsoft Azure Cognitive Service for Language. . 239
G.49 Confusion matrix obtained for beginning entity recognition with Microsoft Azure Cognitive Service for Language. 240
G.50 Confusion matrix obtained for ending entity recognition with Microsoft Azure Cognitive Service for Language. 240
G.51 Confusion matrix obtained for duration entity recognition with Microsoft Azure Cognitive Service for Language. 241
G.52 Confusion matrix obtained for time of medication relationship detection with Microsoft Azure Cognitive Service for Language. . . . 241
G.53 Confusion matrix obtained for frequency entity recognition with Microsoft Azure Cognitive Service for Language. 242
G.54 Confusion matrix obtained for frequency of medication relationship detection with Microsoft Azure Cognitive Service for Language. 242
G.55 Confusion matrix obtained for medication entity recognition with Microsoft Azure Cognitive Service for Language. 243
G.56 Confusion matrix obtained for route entity recognition with Microsoft Azure Cognitive Service for Language. 243
G.57 Confusion matrix obtained for route of medication relationship detection with Microsoft Azure Cognitive Service for Language. . . . 244
G.58 Confusion matrix obtained for negative entity recognition with Microsoft Azure Text Analytics for Health. 245
G.59 Confusion matrix obtained for negative relationship detection with Microsoft Azure Text Analytics for Health. 245
G.60 Confusion matrix obtained for body structure entity recognition with Microsoft Azure Text Analytics for Health. 246
G.61 Confusion matrix obtained for date entity recognition with Microsoft Azure Text Analytics for Health. 246
G.62 Confusion matrix obtained for dosage entity recognition with Microsoft Azure Text Analytics for Health. 247

G.63 Confusion matrix obtained for dosage of medication relationship detection with Microsoft Azure Text Analytics for Health. 247
G.64 Confusion matrix obtained for beginning entity recognition with Microsoft Azure Text Analytics for Health. 248
G.65 Confusion matrix obtained for ending entity recognition with Microsoft Azure Text Analytics for Health. 248
G.66 Caption 249
G.67 Confusion matrix obtained for duration entity recognition with Microsoft Azure Text Analytics for Health. 249
G.68 Confusion matrix obtained for time of medication relationship detection with Microsoft Azure Text Analytics for Health. 249
G.69 Confusion matrix obtained for frequency entity recognition with Microsoft Azure Text Analytics for Health. 250
G.70 Confusion matrix obtained for frequency of medication relationship detection with Microsoft Azure Text Analytics for Health. . . 250
G.71 Confusion matrix obtained for medication entity recognition with Microsoft Azure Text Analytics for Health. 251
G.72 Confusion matrix obtained for route entity recognition with Microsoft Azure Text Analytics for Health. 251
G.73 Confusion matrix obtained for route of medication relationship detection with Microsoft Azure Text Analytics for Health. 252
G.74 Confusion matrix obtained for time entity recognition with Microsoft Azure Text Analytics for Health. 252
G.75 Confusion matrix obtained for negative entity recognition with Microsoft Azure Text Analytics for Health after post-processing. . . . 253
G.76 Confusion matrix obtained for negative relationship detection with Microsoft Azure Text Analytics for Health after post-processing. . 253
G.77 Confusion matrix obtained for dosage entity recognition with Microsoft Azure Text Analytics for Health after post-processing. . . . 254
G.78 Confusion matrix obtained for dosage of medication relationship detection with Microsoft Azure Text Analytics for Health after post-processing. 254
G.79 Confusion matrix obtained for beginning entity recognition with Microsoft Azure Text Analytics for Health after post-processing. . 255
G.80 Confusion matrix obtained for ending entity recognition with Microsoft Azure Text Analytics for Health after post-processing. . . . 255
G.81 Confusion matrix obtained for duration entity recognition with Microsoft Azure Text Analytics for Health after post-processing. 256
G.82 Confusion matrix obtained for time of medication relationship detection with Microsoft Azure Text Analytics for Health after post-processing. 256
G.83 Confusion matrix obtained for frequency entity recognition with Microsoft Azure Text Analytics for Health after post-processing. . 257
G.84 Confusion matrix obtained for frequency of medication relationship detection with Microsoft Azure Text Analytics for Health after post-processing. 257

G.85 Confusion matrix obtained for medication entity recognition with Microsoft Azure Text Analytics for Health after post-processing. . 258
G.86 Confusion matrix obtained for route entity recognition with Microsoft Azure Text Analytics for Health after post-processing. . . . 258
G.87 Confusion matrix obtained for route of medication relationship detection with Microsoft Azure Text Analytics for Health after post-processing. 259

H.1 Prescribe use case illustration. (I) 261
H.2 Prescribe use case illustration. (II) 262
H.3 Prescribe use case illustration. (III) 263
H.4 Prescribe use case illustration. (IV) 264
H.5 Prescribe use case illustration. (V) 265
H.6 Prescribe use case illustration. (VI) 266
H.7 Prescribe use case illustration. (VII) 267
H.8 Prescribe use case illustration. (VIII) 268
H.9 Prescribe use case illustration. (IX) 269
H.10 Prescribe use case illustration. (X) 270
H.11 Suspend use case illustration. (I) 271
H.12 Suspend use case illustration. (II) 272
H.13 Suspend use case illustration. (III) 273
H.14 Suspend use case illustration. (IV) 274
H.15 Suspend use case illustration. (V) 275
H.16 Suspend use case illustration. (VI) 276
H.17 Change use case illustration. (I) 277
H.18 Change use case illustration. (II) 278
H.19 Change use case illustration. (III) 279
H.20 Change use case illustration. (IV) 280

List of Tables

2.1 Structured actions for medical prescription and their additional information. . . . 6
2.2 Advantages associated with the state of the art solutions. 17
2.3 Evaluation metrics of the state of the art systems. 18

3.1 Comparison of the various ASR tools. (I) 22
3.2 Comparison of the various ASR tools. (II-I) 23
3.3 Comparison of the various ASR tools. (II-II) 23
3.4 Comparison of the various ASR tools. (II-III) 23
3.5 Comparison of the various ASR tools. (II-IV) 23
3.6 Comparison of the various ASR tools. (II-V) 24
3.7 Comparison of the various ASR tools. (III) * When using multiple content URLs as input. 24
3.8 Comparison of the various ASR tools. (IV) /hour means per audio hour /model/hour means per model per hour 25
3.9 Features provided by Amazon Transcribe. 25
3.10 Features provided by Amazon Transcribe for European Portuguese. 26
3.11 Standard batch transcription pricing for Amazon Transcribe. 27
3.12 Data limits for the Amazon Transcribe features provided for European Portuguese. 27
3.13 Microsoft Azure Cognitive Services for Speech features (all but Text-to-speech). 28
3.14 Features provided by Microsoft Azure for Speech to Text service for European Portuguese. 29
3.15 Microsoft Azure Cognitive Services for Speech pricing. 29
3.16 Microsoft Azure data limits for Speech-to-text service, for Free (F0) plan in real-time transcription. 30
3.17 Microsoft Azure data limits for Speech-to-text service, for Standard (S0) plan in real-time transcription. 30
3.18 Microsoft Azure data limits for Speech-to-text service, for Free (F0) plan for model customization. 30
3.19 Microsoft Azure data limits for Speech-to-text service, for Standard (S0) plan for model customization. 30
3.20 Google Cloud Speech-to-Text features. 32
3.21 Description of the templates provided by Google Cloud Speech-to-Text service for European Portuguese. 32
3.22 Default model features for European Portuguese. 32
3.23 Latest long model features for European Portuguese. 33

3.24 Google Cloud Speech-to-Text pricing. 33
3.25 Google Cloud data limits for Speech-to-Text service. 33
3.26 Whisper models provided and respective specifications. 36
3.27 Whisper's pipeline features. 36
3.28 Automatic Speech Recognition (ASR) solutions to be tested in experiments section. 39
3.29 Comparison of the various NLP tools. (I) *Except at the Healthcare Natural Language API level. 41
3.30 Comparison of the several NLP tools (II-I) *For English text only. . . 41
3.31 Comparison of the several NLP tools (II-II) *For English text only. . . 42
3.32 Comparison of the various NLP tools. (III-I) u means unit or units . 42
3.33 Comparison of the various NLP tools. (III-II) u means unit or units . 42
3.34 Comparison of the various NLP tools. (III-III) *(Per 1,000 text records) **(available in containers) *** text records 43
3.35 Comparison of the various NLP tools. (III-IV) u means unit or units 43
3.36 Comparison of the various NLP tools. (IV-I) 44
3.37 Comparison of the various NLP tools. (IV-II) 44
3.38 Amazon Comprehend features. 45
3.39 Amazon Comprehend features provided for Portuguese. 46
3.40 Amazon Comprehend pricing (I). 1 unit = 100 characters 46
3.41 Amazon Comprehend pricing (II). UI: inference unit One unit = 100 characters. Minimum of three units per request 46
3.42 Quotas associated with the features of interest provided by Amazon Comprehend for European Portuguese. 47
3.43 Amazon Comprehend Medical features. 48
3.44 Amazon Comprehend Medical pricing. 1 unit = 100 characters 49
3.45 Amazon Comprehend Medical input data limits. 49
3.46 Google Cloud Natural Language API features. 50
3.47 AutoML features. 50
3.48 Natural Language API features. 50
3.49 Healthcare Natural Language API features. 51
3.50 Google Cloud Natural Language API features provided for European Portuguese. 52
3.51 Cloud Natural Language pricing. 52
3.52 Healthcare Natural Language API pricing. 52
3.53 VertexAI pricing. (AutoML) 52
3.54 Google Cloud Natural Language API AutoML limits. 53
3.55 Google Cloud Natural Language API limits. 53
3.56 Microsoft Azure Cognitive Service for Language features. 54
3.57 Microsoft Azure Cognitive Service for Language features for European Portuguese. 55
3.58 Features provided by Text Analytics for Health. 55
3.59 Microsoft Azure Cognitive Service for Language pricing (Standard (S) instance). *Per 1,000 text records **(available in containers) 56
3.60 Microsoft Azure Cognitive Service for Language data limits. *across all documents 57
3.61 IBM Watson Natural Language Understanding features. 57

3.62 IBM Watson Natural Language Understanding features provided for European Portuguese. 58
3.63 IBM Watson Natural Language Understanding pricing. 59
3.64 IBM Watson Natural Language Understanding data limits. 59
3.65 spaCy statistical and rule-based models features. 60
3.66 spaCy engines and rule-based components' features. 60
3.67 Features provided by spaCy's models for European Portuguese. . . 61
3.68 NLTK features. 62
3.69 Solutions to be tested in preliminary experiments section. 63

4.1 Features provided by Amazon Web Services (AWS) Lambda. . . . 68
4.2 Features provided by Amazon API Gateway. 70
4.3 Comparison between MVC, MVP, and MVVM. 80
4.4 Advantages of MVP and MVVM. 80
4.5 Disadvantages of MVP and MVVM. 80

5.1 Results obtained for ASR solutions comparison. 86
5.2 Results obtained for INN entity recognition with Amazon Comprehend Medical. 89
5.3 Results obtained for INN entity recognition with Google Cloud Healthcare Natural Language API for entities in English. 91
5.4 Results obtained for INN entity recognition with Microsoft Azure Text Analytics for Health for entities in European Portuguese. . . . 92
5.5 Results obtained for INN entity recognition with Microsoft Azure Cognitive Service for Language - Text Analytics for Health for entities in English. 93
5.6 Default models performance in the INN recognition for entities in European Portuguese. 94
5.7 Default models performance in the INN recognition for entities in English. 94
5.8 Different values used for each prescription component for the prescriptions dataset generation process. 96
5.9 Results obtained for INN entity recognition with Amazon Comprehend customized model. 97
5.10 Results obtained for INN entity recognition with Amazon Comprehend customized model. 99
5.11 Execution time in seconds obtained for different dropout rates with the three spaCy's pipelines available for European Portuguese. . . 101
5.12 Entities detection performance obtained for different dropout rates with the three spaCy's pipelines available for European Portuguese. 101
5.13 Execution time in seconds obtained for different batch sizes with the three spaCy's pipelines available for European Portuguese. . . 102
5.14 Entities detection performance obtained for different batch sizes with the three spaCy's pipelines available for European Portuguese. 102
5.15 Entities detection performance obtained for different batch sizes with the three spaCy's pipelines available for European Portuguese. 103

5.16 Entities detection performance obtained for different batch sizes with the three spaCy's pipelines available for European Portuguese. . . . 103
5.17 Customized models performance in the INN recognition. 103
5.18 Different values used for each prescription component for the test dataset generation process. 107
5.19 Results obtained for entity recognition and relationship detection in prescriptions with Amazon Comprehend. 109
5.20 Results obtained for INN entity recognition in prescriptions with Google Cloud Healthcare Natural Language API. 111
5.21 Results that could be obtained for entity recognition and relationship detection in prescriptions with Google Cloud Healthcare Natural Language API after post-processing. 115
5.22 Results obtained for INN entity recognition in prescriptions with Google Cloud Healthcare Natural Language API with post-processing. 116
5.23 Results obtained for INN entity recognition in prescriptions with Microsoft Azure Text Analytics for Health. 117
5.24 Results obtained for INN entity recognition in prescriptions with Microsoft Azure Text Analytics for Health. 121
5.25 Results obtained for INN entity recognition in prescriptions with Microsoft Azure Text Analytics for Health, after post-processing. . 122
5.26 Results obtained for INN entity recognition in prescriptions with Microsoft Azure Text Analytics for Health with post-processing. . 123
5.27 Models performance for entities recognition and relationship extraction in prescriptions. 127
5.28 Results obtained for INN entity recognition in multi-medication prescriptions with Microsoft Azure Text Analytics for Health. . . . 128

6.1 Example of relationships detection on multi-medication prescription. 131
6.2 Comparison of the state of the art with the current work. (I) 134
6.3 Comparison of the state of the art with the current work. (II) 134
6.4 Comparison of the state of the art with the current work. (III) . . . 134
6.5 Comparison of the state of the art with the current work. (IV) . . . 134
6.6 Comparison of the state of the art with the current work. (V) 135
6.7 Comparison of the state of the art with the current work. (VI) . . . 135
6.8 Comparison of the state of the art with the current work. (VII) . . . 135
6.9 Comparison of the state of the art with the current work. (VIII) . . 135
6.10 Comparison of the state of the art with the current work. (IX) . . . 135
6.11 Comparison of the state of the art with the current work. (X) 136
6.12 Comparison of the state of the art with the current work. (XI) . . . 136
6.13 Results obtained by making several requests to the developed API. 141
6.14 Results obtained by making several requests to the developed API. 142
6.15 Functional requirements of the developed solution. 145

7.1 Prescription orders and time consumed to do them. 158
7.2 Suspension orders and time consumed to do them. 159
7.3 Changing orders and time consumed to do them. 160

8.1 Comparison of the results obtained with the state of the art results. 162

F.1 Test dataset for the NLP solution's test with single-INN prescriptions. 213

Chapter 1

Introduction

The current document presents the work developed in MedicineOne, Life Sciences Computing, S.A., in the scope of the curricular unit Dissertation / Internship in Intelligent Systems of the second year of the Master in Informatics Engineering of the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

In this chapter, first, the driving force behind this project, the motivation for its realization is presented. Then, the achieved objectives and the strategy followed are illustrated. Also, the contributions are described, and finally, the structure of this document is presented.

1.1 Motivation

The process of medical prescription in hospital environment, also called internal prescription, requires the doctor to visit each of the inpatients for whom they are responsible, reassess their condition, reassess their current therapy, and, finally, adapt it by adding, removing or changing any of its components when necessary. Typically repeated daily, the visit to each patient is accompanied by a nursing cart which provides the doctor with a laptop computer. There, the doctor can access the patient's information and enter any new information, such as desired changes in therapy. However, the information insertion into the system is totally manual and, consequently, time-consuming.

The optimization of this task is associated with the reduction of the time allocated to it, allowing its reallocation to health tasks, the relief of the doctors' overload, and, consequently, the provision of better health care.

1.2 Goals

Prescription using natural language dictation consists of an action of typically less than one-minute duration. In turn, manually entering the information into M1 is much more time-consuming. Therefore, the goals of the present work correspond to the following:

- Development of a system capable of transcribing a dictated medical prescription
- Development of a system capable of processing a prescription text identifying relevant information on it
- Development of a mobile application for iOS that allows the final user, i.e., the doctor, to access the system
- Optimization of the internal prescription task, minimizing the time allocated to it

To achieve these objectives, the strategy adopted corresponded to the workflow:

- Study of the state of the art at the level of:
  - Internal prescription in the health units that make use of M1
  - Artificial Intelligence (AI), Computational Linguistics (CL) and medical prescription
  - ASR and Natural Language Processing (NLP)
  - Cloud provider, Scrum methodology, and User Interface
- Test of the ASR and NLP solutions
- Solution proposal, implementation, and testing

1.3 Contributions

With this work, several contributions were achieved:

- A positive and impactful contribution in terms of the state of the art of medical prescription both in European Portuguese and other languages
- An evolution in the European Portuguese, for which this type of system does not exist
- The identification of gaps, for example, at the level of tools available for European Portuguese, signaling the need for investment and development of resources for it
- The speeding up of the task of internal prescription, making available a pocket system that intends to complete the task more straightforward and less time-consuming
- The relief of the overloaded workload of doctors, improving, in the big picture, the quality of care in the health unit that makes use of the developed solution

1.4 Document structure

This document is divided into eigth chapters: Introduction, Internal prescription, Text Processing Pipeline, Software Development Methodology, Competitors analysis, Approach, The System, and Conclusion and Future Work.

The current chapter frames the work developed by presenting the scope of the work, that is, the motivation and objectives of its realization. It also explains the strategy followed and the contributions achieved with the project's development. Finally, it is responsible for presenting the structure of this document.

Chapter 2 (Internal prescription) provides a theoretical background and overview of the state of the art of internal prescription in Portugal, namely in the context of the health units that use M1. Besides that, it gives a theoretical background of AI and CL and presents their usage for medical prescription optimization.

Chapter 3 (Text Processing Pipeline) provides an introduction of ASR and NLP technologies, exploring their leading solutions.

Chapter 4 (Software Development Methodology) introduces the cloud provider, the cloud services, the work methodology, and the mobile application development design pattern.

Chapter 5 (Competitors analysis) presents the experiments done for the selection of both the ASR solution and the NLP solution used. In addition to the flow followed in each test, the results obtained, their analysis, and the conclusions drawn are presented.

Chapter 6 (Approach) recapitulates the problem definition, introducing the architecture and context of the developed solution in its sequence. In addition, it offers a brief comparison between this solution and the state of the art. Finally, the solution's components are explored individually, being their development presented.

Chapter 7 (The System) presents the system in operation, illustrating its use cases.

Finally, chapter 8 (Conclusion and Future Work) summarises the work presented in this document and gives a final reflection on it. Also, it shows the future work that should be done out of the scope of the internship.

Chapter 2

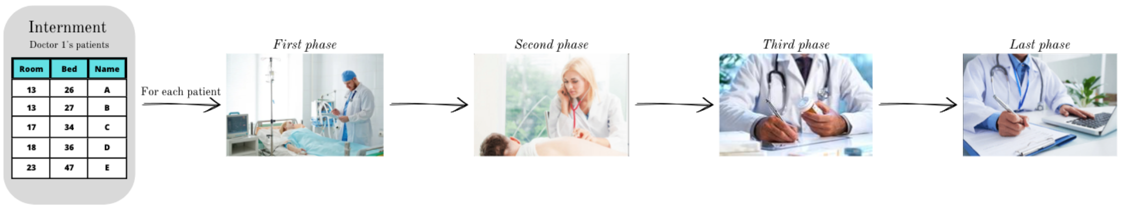
Internal prescription

This chapter provides a theoretical background and overview of the state of the art of internal prescription in Portugal, namely in the context of the health units that use M1. Besides that, it gives a theoretical background of Artificial Intelligence (AI) and Computational Linguistics (CL) and presents their usage for medical prescription optimization.

First, theoretical concepts related to internal prescription and its state of the art for European Portuguese are presented. Then, the fundamental concepts associated with AI and CL are presented, followed by state of the art regarding their application to medical prescription.

2.1 Introduction

When a patient is admitted for hospitalization, the doctor should determine the therapy that should be followed. Then, if the hospitalization lasts long enough, that therapy should be reviewed periodically, typically daily. For that, the doctor should visit the patient, re-evaluate them and their response to the actual therapy, re-evaluate the current therapy, and, when necessary, change it. This process, illustrated in Figure 2.1, is called internal prescription, and the doctors must repeat it for all the patients under their responsibility. The actual therapy can be changed by adding - prescribe -, removing - suspend - or changing - change - any component of it. These modifications have associated additional information presented in Table 2.1.



Room	Bed	Name
13	26	A
13	27	B
17	34	C
18	36	D
23	47	E

Figure 2.1: Ilustration of the internal prescription process [pressfoto] [pre, a] [pre, b] [pre, 2022].

Action	Additional information
Prescribe	Active principle Route of administration Dosage Frequency Beginning Ending
Suspend	Active principle
Change	Active principle New route of administration New dosage New frequency New beginning New ending

Table 2.1: Structured actions for medical prescription and their additional information.

Changes must be reflected in the patient's process whenever therapy is changed; for that, the doctor must update it.

2.2 Internal prescription in Portuguese health units

In the reality of the health units that use M1, the doctor's visit to the patient is accompanied by a nursing cart that, among others, includes a laptop computer with M1 on it. The doctor uses this computer to get all the needed information about the patient and its therapy, giving the context required for a safe prescription. Besides that, it allows the doctor to insert new data into the system, including new prescription actions. There, the introduction of International Non-proprietary Name (INN) is done following the process:

- Open M1
- Open hospitalization module
- Open desired patient's process
- Open therapeutic tab
- Select the add button
- Search the INN in a search bar
- Select the desired INN from the list returned
- Select the select button

- Select the desired type of medication packaging from a list with the available for the selected INN and similar INN
- Select the prescribe button, which opens a posology window
- Definition of the posology by selecting an existing one, defining a new one by free-text, or defining a new one by editable fields filling. The posology includes the pharmaceutical form, the route, the frequency, and the duration of the medication. In this window, some additional information can be inserted
- Select the button add to recipe
- Repetition of the process for each INN that should be on the final prescription
- Select the button issue followed by authentication

This time-consuming task takes up much of the doctors' valuable time and contributes to their overload. If, on the one hand, excessive time consumption prevents greater allocation of time to medical care, contributing to overloading doctors makes them more susceptible to failure. With this, the lack of task optimization contributes, in the big picture, to sub-optimal medical care.

To our knowledge, there is yet to be a solution for internal prescription optimization in Portugal.

2.3 Artificial Intelligence and Computational Linguistics in medical prescription

This section introduces fundamental concepts related to AI and CL and presents the state of the art regarding their use in the medical prescription process.

2.3.1 Introduction

AI is a branch of computer science aimed at automating intelligent behavior. The definition of intelligence is widely explored, and there are countless ways to define it. A compact way to define intelligence is the junction of perceiving, analyzing, and reacting. The basic materials in this area include data structures, techniques for representing knowledge, algorithms for applying knowledge and language, and the programming techniques needed to implement all of the above [**?**].

Several questions must be asked to have an idea of intelligence, for example, "What is the mechanism for representing knowledge in living cells?". The goal of AI is, in part, to answer these questions by using the tools that AI provides. These tools are related to the fact that AI provides not only the medium but also the testbed for theories of intelligence that can be expressed in computer programs. Through their execution, these programs can be both tested and verified [?].

In turn, CL corresponds to the study of computer systems for the understanding and generation of natural language being linguistics constituted by two branches: CL and Theoretical Linguistics (TL). It focuses on applying quantitative and statistical methods to understand how humans model language and computational approaches to answering linguistic questions. Its concept is commonly used interchangeably with the concept of Natural Language Processing (NLP), although the difference relates to the underlying motivation.

As far as CL is concerned, this branch focuses on developing algorithms capable of handling a good range of natural language as input. On the other hand, the branch concerning TL focuses mainly on the aspect of linguistic performance called grammatical competence, the way people accept sentences as sentences that do or do not correctly follow the grammar. Still, there is a concern with universal language, that is, with finding the grammatical principles that apply to all natural languages.

CL can refer to both written and spoken natural language, the former being a symbolic representation of spoken or sign language. The processing of the former is typically called text analysis, and that of the latter is called speech analysis. CL focuses on studying natural language analysis and language generation and can be further divided into two domains: sentence analysis and discourse and dialogue structure. Much more is currently known about processing individual sentences than determining discourse structure. However, this requires a prerequisite, such as analyzing the meaning of individual sentences. Sentence analysis, whose primary goal is to determine the meaning of a sentence, can be further divided into syntactic analysis and semantic analysis. Achieving the primary purpose of this analysis depends on translating the natural language input into a simple semantic language like formal logic or a database command language. Most systems have, as their first phase, syntax analysis.

At last, it should be noted that CL concerns an interdisciplinary field of theoretical and applied science that combines multiple areas such as linguistics, psychology, neuroscience, philosophy, computer science, and mathematics [Chowdhary, 2020a] [Kamath et al., 2019a].

2.3.2 State of the art

Although they are widely used in several markets and, within them, for different purposes, to the best of our knowledge, AI and CL are not yet used to support medical prescription in European Portuguese. However, several works are already underway on the medical side for other languages. In this section, four studies are addressed, being relative to Indian and French realities. Those studies refer to the ones presented in [Shaikh et al., 2021], [Kocabiyikoglu et al., 2019] and [Kocabiyikoglu et al., 2020], [Dhokley et al., 2021], and [Mahatpure et al., 2019].

Language support

Of the four previously mentioned studies, three were developed in India and focused on Indian English [Shaikh et al., 2021] [Dhokley et al., 2021] [Mahatpure et al., 2019]. The remaining one was developed in France and focuses on French [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020].

Studies motivation and objectives

In [Mahatpure et al., 2019], the authors present the Indian reality. According to it, the patient record addition and retrieval using Electronic Health Record System (EHR) systems consumes 49% of the doctor's time. In addition, handwritten documents are still the gold standard in the country, forcing the insertion into the system of much information regarding different patients in short periods. An identifier is used to drive this information insertion, being the task susceptible to errors. By the time of the study, these questions were not widely considered by the systems in the market that, according to [Shaikh et al., 2021], are typically focused on symptom-based prescribing, diagnosis-based prescribing or scanning, supervision, and analysis of handwritten prescriptions approach.

In turn, in [Kocabiyikoglu et al., 2019] and [Kocabiyikoglu et al., 2020] the authors present the French reality. There, the Prescription Management Systems (PMS) are used, but according to the authors, these systems entail a major disadvantage. It corresponds to the need for manual insertion of information into the system by the doctor, which reduces the time dedicated to medical care.

With that, the goals enumerated through the multiple studies include:

- Mediation of purely digitally-based medical prescription process [Shaikh et al., 2021]
- Eradication of the problem of the significant number of deaths in India due to errors in interpreting handwritten prescriptions [Shaikh et al., 2021]
- Complement of the current portable medical services conveyance system [Shaikh et al., 2021]
- Reduction of the time spent on accessing [Shaikh et al., 2021], making [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020], or accessing and making [Mahatpure et al., 2019] medical prescription records
- Support medical prescription process in French in a hospital environment [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- Avoid manual information insertion into systems [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020], and handwritten or typed prescriptions [Dhokley et al., 2021], that reduces the healthcare-dedicated time
- Enable the optimization of medical prescription [Mahatpure et al., 2019]

The proposed systems

To meet the previously mentioned objectives, all the state of the art studies include the development of a mobile application to allow the interaction of the doctor with the developed system. In addition, the system developed at [Mahatpure et al., 2019] can also be accessed using a web application. Any of these solutions are based on speech recognition technologies, and the type of input common to all solutions is dictation. At this level, the solution developed in [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] is distinguished from the others by the fact that it also allows the input of information by the doctor in free-text. Also, there is another level of divergence between the various projects. This concerns the fact that, unlike the other systems, the system developed in [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] does not exist in isolation but rather mediates the insertion of information into a PMS. Finally, except [Shaikh et al., 2021], all state of the art systems make use of solutions of NLP in addition to solutions of Automatic Speech Recognition (ASR). The system developed in [Mahatpure et al., 2019] also uses blockchain technology for information storage. Of these systems, only the one developed in [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] does not present the processed information in tabular format to its user, presenting it in the form of a dialog.

Focusing on each system individually and starting with the system purposed in [Shaikh et al., 2021], the developed solution is partly based on voice commands. Those must be used either to initiate the prescription dictation or to request that this final one be issued. These voice commands are detected by several online and offline speech recognition APIs combined using a Random Forest model. Furthermore, this system can automatically learn and improve its performance based on experience without being explicitly programmed. According to the authors, their solution model should be trained over Indian names, and even when the model is below a given threshold of certainty for a given word, it must highlight that same word for the user to validate. It should be compatible with INN names since it should use speech recognition APIs for medical domains, having the advantage of being associated with a medicine database provided by the government of India. This system has four components and seven technologies, highlighting the use of Google STT API, Dragon Natural Speaking, Sphinx4, and Kaldi for processing the dictation.

The system presented in [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020], in turn, works in a dialogue basis between the interface and the user, being the dialogue initiated by the user. According to the authors, a dialogue system is composed of ASR, Spoken Language Understanding (SLU), Dialog State Tracking (DST), dialogue policy, Natural Language Generation (NLG) and Text-to-Speech (TTS) components, being the ASR and Natural Language Understanding (NLU) components already explored by the authors. The authors make dialogue modeling using interactive learning, a Machine Learning (ML) technique in which humans are involved in ML model building [Varangaonkar, 2018]. The proposed system corresponds to a modular one where each model is independent and followed by another, enabling the adaptation and training of each one individually. Although this sequential processing powers the error propagation, the authors use multiple validations to mitigate it. Also, the results obtained with the system made the authors assume the necessity of improvement associated with their system, namely at the NLU component level. Besides that, the authors show concern about keeping confidentiality, integrity, and conformity.

The system developed in [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020], in turn, uses a toolset named Rasa X as its ASR component. This toolset provides a simple web interface that allows the NLU, dialogue scenarios, and domain definition changing using only a web interface. For the NLU component, on the other hand, the authors explored four models compared to a baseline model also developed by the authors. Those models were chosen because of the state of the art analyzed by the authors, from which it is possible to draw the various approaches followed in this type of problem: rule-based, based on machine learning classifiers, based on Conditional Random Fields (CRF) models, based on Deep Learning (DL) and based on speech processing. Whit that, the tested models corresponded to the following:

- RASA, that follows the CRF and Machine Learning classifier-based approaches, having a linear chain CRF, a lookup table, and, separately, a linear SVM based on pre-trained word embeddings
- Tri-CRF, that follows the CRF approach
- Att-RN, that follows the DL approach, having an initial embedding layer consisting of 128 neurons, followed by a bi-directional LSTM encoder and decoder, each one with 128 neurons
- Seq2Sqeq, that follows the DL approach, having a recurrent encoder-decoder architecture with attention that makes use of a single bi-directional LSTM encoder and decoder layer with 128-sized encoder, decoder, and embeddings layer
- Baseline, that follows the rule-based approach

Since Tri-CRF had the best F-measure in the authors' testing, they identified it as the best model among the tested ones. When applying the class weights, however, it can be seen that the RASA model is less affected than the other models. The former, not so data dependent, manages to be less affected by a problem that significantly affects data-driven approaches, such as the DL approach.

In addition, the system developed in [Dhokley et al., 2021] corresponds to a task-based dialogue system that uses Stack-Propagation and has three components: an encoder and two decoders.

- Encoder: corresponds to a self-attentive encoder composed of a BiLSTM with attention mechanism. It is shared between intent detection and slot-filling tasks, being its output passed to the second decoder.
- First decoder: a unidirectional LSTM. It is used for intent detection for each of the tokens, being the intent of the input utterance calculated taking into

account the intent of each token on it. Similar to the previous one, its output is passed to the second decoder, and it can have one of two values: prescriptive or non-prescriptive.

- Second decoder: a unidirectional LSTM. This decoder is used for slot-filling, being considered by the authors 22 different slot labels.

Focusing only on the development of a model for the structuration of information on a tabular format, this study does not focus on the ASR component, being that indicated by the authors as future work.

Finally, the study conducted in [Mahatpure et al., 2019] proposes a system that needs the inclusion of keywords anticipating the information of interest. For example, the prescribed medication should be preceded by the keyword medication or medicine. The authors also assume that a prescription comprises four basic and two optional components. The first corresponds to the patient's name, age, symptoms, and medicines with dosage, while the second corresponds to laboratory tests and notes. This solution is associated with patient identification throw a QR code that, being read by the medical staff, can map to the first two basic components of the prescription. With that, the system focuses on the other basic and optional components. This system is composed of five modules, being relevant to highlight the:

- Python Django REST APIs, responsible for the NLP component of the system
- React-Native Mobile Application, that facilitates the use of the system by its users, and can be used by:
  - Doctors, that can register a patient, identify a patient and create e-prescriptions. This application provides an interface for ASR and NLP to fill in all the details of a prescription form.
  - Patients, that can access their prescriptions.
- React JS Web Application, which also allows access to the system by the user. It allows the user to:
  - List participants and prescriptions according to user's permissions
  - Register new participants
  - Create prescriptions with ASR and NLP

It should also be noted that this solution uses Google's Speech Recognition API for ASR since, according to the authors, it is the most suitable option because of its support for Indian English.

Systems' features

With regard to the functionalities of the systems explicitly expressed by their authors, these include:

- Store patient-related information [Shaikh et al., 2021]
- Store information regarding the patient's present and past prescriptions [Shaikh et al., 2021]
- Generate alerts if the prescription includes INNs that interact adversely [Shaikh et al., 2021] [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- Generate alerts if the prescription includes INNs that react adversely with INNs from other prescriptions [Shaikh et al., 2021] [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- Generate alerts in case of a lack of stock of INNs in the pharmacy [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- Communicate with the PMS to verify if the prescription can trigger adverse reactions according to the patient's history [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- Reduce the time spent in producing a prescription [Shaikh et al., 2021]
- Update the prescription in real-time through requests made by the dialogue-based system to its user so it follows the e-prescribing regulations [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- Inclusion of other relevant information on the system, like health unit policies regarding available medication and best practices [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- Be able to be used on personal devices, facilitating the identification process [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]

Systems-mediated prescription

Each state of the art solution has a different associated workflow, being them detailed for [Shaikh et al., 2021] and [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] .

In [Shaikh et al., 2021], the generation of the prescription and the subsequent acquisition of the INNs in the pharmacy can be done through the workflow:

1. Evaluation and diagnosis of the patient by the doctor
2. Activation of the system with a simple voice command
3. Dictation of the prescription by the doctor, including symptoms, medication, notes, and others
4. Processing of the dictation by the system by the transcription of the dictation followed by string comparison to filling the fields of the prescription

5. Presentation of the processed information in tabular form, with the name, age, gender, symptoms, diagnosis, medication, and notes associated with the patient

6. Doctor's validation of the information processed by the system

7. Doctor asks for the prescription to be issued using a simple voice command

8. Generation of a prescription automatically signed by the doctor, available in digital format or printed, in the cases the patient does not have a smartphone

9. Obtaining the patient's ID or reading the prescription directly from the patient's cell phone or paper – in emergency cases only – by the pharmacist

10. If necessary, send the prescription to the doctor by the pharmacist for confirmation or request a review

By turn, in [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020], the Natural Language interface proposed allows medical prescription following the workflow:

1. The doctor initiates the dialogue with the system

2. The doctor dictates or writes a free-text prescription

3. The system interprets the prescription for the extraction of information of interest

4. The result of this interpretation is returned to the doctor

5. Request for information by the system, which is optional and occurs only if something is not according to the standard

6. Request by the system for doctor's validation of the final proposal prescription

7. The doctor validates the processed information

8. The prescription is filled in the PMS without the intervention of the doctor

9. The PMS validates the prescription and issues it

In this system, the initial utterance of the doctor should be understood, disambiguated, and completed through a goal-oriented dialogue. The final proposal prescription is presented to the doctor for validation when all the information is filled in.

Prescription filling methodology

The four studies analyzed rely on string comparison [Shaikh et al., 2021], slot-filling [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] [Dhokley et al., 2021], and keyword search [Mahatpure et al., 2019] to fill the prescription's information.

In [Shaikh et al., 2021], the authors organize the transcribed text in strings, comparing them with other strings already in the system database. This comparison allows the assignment of the new strings to their respective fields.

In turn, in [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] the authors base the data extraction on slot-filling, a two-phase approach. First, the statement's intention, which reflects its user's intention, is identified. Then the most important elements of that statement are identified, being these elements called slots. These slots relate to the entities and relationships of interest for the task associated with the statement in which they are found. The authors consider the types of intentions Medical Prescription and None and 39 slot labels. They consider the introduction of intent recognition relevant so the system can deal with scenarios that capture instructions that do not concern prescriptions. Similarly, in [Dhokley et al., 2021] the authors base the prescription filling process on slot-filling and intent recognition.

Finally, in [Mahatpure et al., 2019] the authors resort to the identification of keywords and extraction of the information associated with them. For that, the processing follows the workflow:

1. Stop-words and unnecessary words removal

2. Lowercase of the resultant text

3. Symptom and Symptoms keywords search
   In all the keywords search, when the keyword is found, it is removed from the sentence, and the sentence is split in the keyword's position, resulting in two new segments. The keywords are searched by order, and the others are not searched when a keyword is found.

4. Medication, Medications, Medicine and Medicines keywords search
   If this search is insufficient, a regex condition is used to search for the keyword.

5. Symptoms extraction
   This extraction can follow one of two ways based on medication identification:

   - If a medication was found, a search for symptoms is done in the segment of the sentence relative to them. This search is based on a list with 1111 symptoms, and the result of it is a list of the symptoms present in the segment.

- Otherwise, lab test, lab tests, test, tests, note and notes keywords search is done. With that, two segments are generated again, and from the first, a list of symptoms is equally extracted.

6. Once the medication was identified, a lab test, lab tests, test, tests, note, notes keywords search is done. In the first segment, a name keyword search is done. Then, a string array is created with two elements containing the information in each of the two segments.

7. Medicines extraction
   The previously created array is traversed, and a dose and dosage keywords search is done. The first element of the array is searched on a cached list of medicines details retrieved from Healthos API, and if not found, a request is sent to Healthos API, being the best match returned. This array is then associated with the prescription object.

8. If no symptoms or medication are found, a lab test, lab tests, test, tests, note, notes keywords search is done.

9. If lab test or lab tests is found, a note and notes keyword search is done. If found, the first segment of the sentence is assigned to the lab tests of the prescription object and the second to the notes.

10. If, otherwise, note or notes is found, the segment is assigned to the notes attribute of the prescription object.

Advantages of the systems

Throughout their studies, some authors highlight advantages associated with their systems including:

Level	Advantage
Cost	In [Shaikh et al., 2021] only open-source solutions are used. In turn, the technologies used in [Mahatpure et al., 2019] are mostly open-source. This brings benefits to the systems that are, then, cost-less
Portability	The systems are focused on portable devices, bringing benefits in terms of ease of use, portability, time-saving, and mobility, allowing the prescription at the point of care
Single dictation	In [Shaikh et al., 2021] and [Mahatpure et al., 2019], a single dictation is needed to fill all fields of the prescription to be issued, allowing the reduction of time spent in the process. It is reduced concerning both manual prescriptions and solutions in which the fields to be filled are dictated one by one

Security	In [Shaikh et al., 2021], medical records are stored following Health Insurance Portability and Accountability Act (HIPAA), which is a US federal law that requires national standards to be created to protect against disclosure of sensitive patient health information without patient consent or knowledge [for Disease Control and Prevention]
Information	The storage of present and past prescriptions in [Shaikh et al., 2021] allows the doctor to make a more informed choice when issuing a new prescription
Faster identification	Since the users use their devices to access the state of the art systems, the identification is faster than made with standard login
No learning	The system in [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] was designed so the users do not have to learn new knowledge to be able to deal with it
Versatility	The system in [Mahatpure et al., 2019] can be accessed with a mobile application and a web application, being in both cases accessible through a smartphone

Table 2.2: Advantages associated with the state of the art solutions.

Difficulties of the studies

Some of the authors also point out some difficulties faced in their systems' development. Those difficulties include:

- Indian language has variants by region, which extends to medicine names [Shaikh et al., 2021]
- Some NLP models, including the best ones, get confused about the name of the INNs and the name of the trademarks of the INN in question [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- There are great difficulties in the identification of the temporal expressions that relate to the duration of the prescription [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- Increasing the size of the prescriptions degrades the performance of the models [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- The INNs' names are ambiguous [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]
- The models must have difficulties identifying names they never faced before [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]

- The prescriptions data is scarce. It is a reality for English and gets worse in the remaining languages [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]. Because of it, the authors of [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] needed to create a dataset that included:

  1. Prescriptions extracted from a book where there were realistic prescriptions
  2. Artificial prescriptions generated based on a grammar to solve the paucity of data, and the class imbalance
  3. Statements not related to prescriptions, corresponding to the type of intent None

- There are few projects concerning the ASR of dictated prescriptions for medical prescription support [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]. The authors of [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] mention a single study on this aspect, called FreePharma™, with no technical details. However, it is indicated that this solution should be able to extract medical prescriptions from speeches captured using PDAs.

- The privacy, ethic, and effectiveness of a solution for health purposes [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]

Results and metrics

The evaluation of some systems was made by its authors and resulted in the evaluation metrics illustrated in table 2.3.

Metric	Value
Slot labeling F1 [Shaikh et al., 2021]	96.00%
Intent recognition accuracy [Shaikh et al., 2021]	99.00%
ASR Word Error Rate (WER) for doctors [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]	3.40%
ASR WER for naive users [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]	17.35%
NLU F-measure for doctors [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]	75.00%
NLU F-measure for naive users [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020]	43.00%
Slot labeling F1 [Dhokley et al., 2021]	91.00%
Intent recognition accuracy [Dhokley et al., 2021]	10.00%

Table 2.3: Evaluation metrics of the state of the art systems.

Besides the previous evaluation metrics, the authors of [Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] also pointed some conclusions from the tests. The tests were made in a zero-noise environment with doctors and naive user, and the conclusions drawn include:

- The dialogue was quickly interrupted by the system with a "drug not found" response
- The system had difficulties in recognizing the frequency and duration at the NLU level
- When the medication was correctly associated, then the prescription process took 20 to 30 seconds, which is truly reasonable when compared with the manual prescription process
- The system performed better with doctors because of the difference in language used
- The task rate success, that is, the ratio of validated prescriptions, was equal to 45% for doctors and 16.6% for naive users

According to the authors of [Dhokley et al., 2021], the values previously presented were achieved with a low quantity of data. The authors are confident that, with more data, the values could be improved to 96.00% for the F1 score for slot labeling and 99.00% accuracy for intent recognition.

Although the authors of [Mahatpure et al., 2019] do not provide evaluation metrics of their system, a test comparing the time needed for prescribing using their system and the conventional EHR systems was done. This test conferred that the manual insertion of information into the EHR systems is slower than the prescription mediated by ASR and NLP, namely as the complexity and size of the prescription increase.

Relevant studies

The authors of [Dhokley et al., 2021] also present some interesting studies conclusions, including:

- Handwritten prescriptions, besides typically offering difficulties in interpreting their text, have much missing information.
- There are errors related to INN interactions in many prescriptions.
- The dictation of a prescription is faster than handwriting or typing, being five times faster than typing and almost two times faster than writing.
- The patient's visit time is about 22% – about three minutes and a half – occupied by the writing of the prescription, on average. Besides that, more 5% of the time is also used in tasks related to the prescription, namely in clarifying and providing guidelines associated with the same.

2.4 Conclusion

There is no evidence of any systems dedicated to optimizing the internal prescription process for European Portuguese at the time of this study. In the development of such systems for other languages, it can be seen that the trend is towards portable systems based on ASR. The filling of prescription information tends to be done using different approaches, with mostly good values being obtained. Evidence of the relevance of this type of system is also presented, namely by minimizing the time allocated to the task they are associated with. Thus, the state of the art presented allows not only to have a perception of existing systems and approaches followed but also to internalize the high added value they symbolize.

Chapter 3

Text Processing Pipeline

The present chapter introduces Automatic Speech Recognition (ASR) and Natural Language Processing (NLP) technologies, exploring their leading solutions.

First, the concept of ASR is introduced, and its leading solutions are presented and analyzed. Then, the concept of NLP is introduced, and its leading solutions are equally introduced and analyzed.

3.1 Introduction

The developed system intended to replace the manual insertion of information into M1 by its dictation, processing, and sending to M1. With that, it needed ASR and NLP capabilities. The following sections introduce the concepts associated with the technologies, also presenting their leading solutions. With that, suitable solutions for this project were selected.

3.2 Speech-to-Text

This section first presents the fundamental concepts of ASR. Then, the most relevant solutions in this area, i.e., the solutions corresponding to the state of the art, are presented. A brief analysis of these solutions and a comparison between them is offered, aiming at selecting the solutions likely to be used in the present work. Each platform is analyzed in terms of its description, services, functionalities, support for European Portuguese, price charged, and limits applied to the input data.

3.2.1 Introduction

ASR corresponds to the NLP task responsible for the computational transcription of spoken language in real-time. Although it has been at the forefront of studying human-computer interfaces since the 1950s, its importance has increased more recently with the advent of Artificial Intelligence (AI) personal assistants such as Siri, Alexa, or Cortana. This task aims to transcribe speech at a human level (almost 100%). However, until 2019 it was only possible to approximate 95%, and the conditions needed to be perfect. The evolution of ASR allows, nowadays, to recognize speech in several conditions, such as noise, diction, and tone. These conditions are still a problem for computers because they cannot deal with them [Kamath et al., 2019b].

3.2.2 Main Speech-to-Text Tools

According to [Iancu, 2019], there are five leading providers of ASR services: Google, Apple, Microsoft, Amazon, and IBM. For the first four, the authors highlight the assistants they provided, and the interest lies in the technology on which they are based. Although Apple has promising technologies at the base of its assistant, Siri, it does not provide a service where these technologies can be used, losing its interest in the project. Besides these services, there are two sets of tools that can be used in Python and that, according to [Kamath et al., 2019d], are two of the most popular resources for ASR: Sphinx and Kaldi. Also, besides the previous solutions, a more recent solution, competitive with state of the art ASR systems in long-form transcription, is called Whisper [Radford et al.]. In addition to these solutions, there are also those native to Android and iOS.

Each solution is explored in the next sections, being a first comparative analysis between the relevant solutions presented in Tables 3.1 to 3.8. Table 3.1 compares the tools at the level of European Portuguese support.

Tool	Has support
Amazon Transcribe	Yes
Amazon Transcribe Medical	No
Microsoft Azure Cognitive Services for Speech	Yes
IBM Watson Speech to Text	No
Google Cloud Speech-to-Text	Yes
Whisper	Yes
iOS Speech	Yes
Android Speech	Yes

Table 3.1: Comparison of the various ASR tools. (I)

By its turn, Tables 3.2 to 3.6 compare the solutions at the level of supported features for European Portuguese.

Tool	Digit Transcription	Language Detection
Amazon Transcribe		
Microsoft Azure Cognitive Services for Speech	x	x
Google Cloud Speech-to-Text	x	
Whisper	x	x

iOS Speech	x	
Android Speech	x	

Table 3.2: Comparison of the various ASR tools. (II-I)

Tool	**Customized Recognition**	**Speech Translation**
Amazon Transcribe	x	
Microsoft Azure Cognitive Services for Speech	x	x
Google Cloud Speech-to-Text	x	
Whisper	x	x
iOS Speech		
Android Speech		

Table 3.3: Comparison of the various ASR tools. (II-II)

Tool	**Spoken Punctuation Detection**	**Word-level Confidence Translation**
Amazon Transcribe		
Microsoft Azure Cognitive Services for Speech		
Google Cloud Speech-to-Text	x	x
Whisper		
iOS Speech		
Android Speech		

Table 3.4: Comparison of the various ASR tools. (II-III)

Tool	**Segmentation**	**Silences Detection**
Amazon Transcribe		
Microsoft Azure Cognitive Services for Speech		
Google Cloud Speech-to-Text		
Whisper		x
iOS Speech	x	
Android Speech	x	

Table 3.5: Comparison of the various ASR tools. (II-IV)

Tool	**Time-stamps Prediction**	**Content Filtering**
Amazon Transcribe		
Microsoft Azure Cognitive Services for Speech		
Google Cloud Speech-to-Text		x
Whisper	x	
iOS Speech		

Android Speech		

Table 3.6: Comparison of the various ASR tools. (II-V)

Also, Table 3.7 compares the solutions at the level of input data limit.

Tool	Input data limit		
Amazon Transcribe	Audio file length		Up to 4 hours
	Audio file		Up to 2 GB
	Size of a custom vocabulary		Up to 50 KB
	Length of a custom vocabulary phrase		Up to 256 characters
	Minimum audio file duration		500 milliseconds
Microsoft Azure Cognitive Services for Speech	Batch transcription	Max audio input file size	1 GB
		Max input blob size	2.5 GB
		Max blob container size	5 GB
		Max number of blobs per container	10000
		Max number of files per transcription request*	1000
	Model costumization	Max number of speech datasets	500
		Max acoustic dataset file size for data import	2 GB
		Max language dataset file size for data import	1.5 GB
		Max pronunciation dataset file size for data import	1 MB
		Max text size when using the text parameter in the creation of a new template	500 KB
Google Cloud Speech-to-Text	Synchronous component		Audio up to 1 minute
	Asynchronous component		Audio up to 480 minutes (4 hours)
Whisper	Does not have		
iOS Speech	Does not have		
Android Speech	Does not have		

Table 3.7: Comparison of the various ASR tools. (III)
* When using multiple content URLs as input.

Finally, Table 3.8 compares the solutions at the level of their costs.

Tool	Price	
Amazon Transcribe	First 250,000 minutes/month	0.02400 $/minute
	Next 750,000 minutes/month	0.01500 $/minute
	Next 4,000,000 minutes/month	0.01020 $/minute

	More than 5,000,000 minutes/month	0.00780 $/minute
Microsoft Azure Cognitive Services for Speech	Free (F0) - Standard	5 hours free/month
	Free (F0) - Custom	5 hours free/month 1 model free/month
	Pay as You Go - Standard	€1.001/audio hour
	Pay as You Go - Custom	€1.401/hour €0.0538/model/hour
Google Cloud Speech-to-Text	0-60 minutes/month	Free
	Over 60 and up to 1 Million minutes/month	$0.006/15 seconds
Whisper	0	
iOS Speech	0	
Android Speech	0	

Table 3.8: Comparison of the various ASR tools. (IV)
/hour means per audio hour
/model/hour means per model per hour

Since the aim of this project is internal prescription dictation transcription, the most relevant features of these systems should be speech and digits transcription. The masking of sensitive information, not an indispensable component, could be an added value for the solution that makes it available.

Amazon Transcribe

Amazon Transcribe is an ASR service that uses Machine Learning (ML) models to convert audio to text. This service can be used standalone or to add Speech-to-Text (STT) transcription capabilities to any application and offers several features, illustrated in Table 3.9.

Feature
Language identification in single-language audio
Language identification in multi-language audio
Channel identification
Job queueing
Speaker diarization
Transcribing digits
Custom language models
Custom vocabularies
Tagging
Identifying personally identifiable information
Redacting transcripts
Vocabulary filtering
Subtitles

Table 3.9: Features provided by Amazon Transcribe.

This service allows users to transcribe multimedia content in real-time or by previously uploading media files into an Amazon S3 service container [Amazon Web Services, 2022c], being transcribed next [Amazon Web Services, 2022g]. Also, it supports European Portuguese, offering the functionalities illustrated in Table 3.10 for the language [Amazon Web Services, 2022d] [Amazon Web Services, 2022b] [Amazon Web Services, 2022a].

Feature	Description
Data input	The data input can be passed in batch.
Custom vocabularies	Used to improve the accuracy of the transcription of specific words - usually domain-specific words - by providing hints. For European Portuguese, it can be used with acronyms and in batches.
Call Analytics	Designed for call center's audios, this functionality is used to get insight into customer-agent interaction. For European Portuguese, it can be used in post-call mode, that is, after the interaction finishes and its recording is loaded into S3.

Table 3.10: Features provided by Amazon Transcribe for European Portuguese.

This service does not support the transcription of digits in European Portuguese, the transcription of speech in real-time, and redaction. Redaction allows masking or removing sensitive content from transcripts in the form of Personally Identifying Information (PII). Its absence is not a significant drawback, but making it available would be an added value [Amazon Web Services, 2022f]. Although this service allows the use of acronyms in personalized vocabularies, these feature is not particularly relevant in the present work since it is not a recurrent practice to use acronyms to mention the information intended to be collected from the discourse of interest. Since the work does not focus on customer-agent interactions, call analytics is also not interesting in this context.

The functionality provided by this service for European Portuguese is available for batch transcription only. The potential use of this service in the present work should involve asynchronous requests to the API where one and only one file corresponding to one medical dictation should be sent each time. Although this does not impede its usage, real-time processing is preferable.

Information on monetary costs associated with this service is available for three European countries: London, Frankfurt, and Ireland. Prices in these three countries are very similar and are defined by level, with each level relating to a given number of minutes of audio transcribed per month. The difference between the three locations is that Ireland has one more level than the others. This service then presents four price levels for Europe and, considering the prices for the standard batch transcription, the costs shown in Table 3.11 are charged [Amazon Web Services, 2022e].

Tier	Volume (minutes/month)	Price (USD/minute)
1	First 250.000	0.02400
2	Next 750.000	0.01500
3	Next 4.000.000	0.01020

4	More than 5.000.000	0.00780

Table 3.11: Standard batch transcription pricing for Amazon Transcribe.

For batch processing and considering the features provided by the service for European Portuguese, Amazon Transcribe imposes the data input limits presented in Table 3.12 [Amazon Web Services, h].

Description	Quota
Audio file length	Up to 4 hours
Audio file size	Up to 2 GB
Size of a custom vocabulary	Up to 50 KB
Length of a custom vocabulary phrase	Up to 256 characters
Minimum audio file duration	500 milliseconds

Table 3.12: Data limits for the Amazon Transcribe features provided for European Portuguese.

Finally, no information is publicly disclosed regarding the model on which this service is based and the data used to train it.

Amazon Transcribe Medical

Like the previous service, Amazon Transcribe Medical is provided by Amazon Web Services (AWS). It is an ASR service designed for medical-related speech transcription, such as doctor notes or doctor-patient conversations. This service can be used for real-time multimedia transcription or transcription of previously uploaded files in batch mode. Regarding security, it operates under a shared responsibility model where AWS protects the infrastructure that manages the service, and the service user is responsible for managing their data.

Despite being a service specifically targeted at the medical field and similar to Amazon Transcribe in terms of the functionalities made available, this service has the disadvantage in this work of only supporting speeches in the English language [Amazon Web Servoces, 2022]. The users of the solution developed in the current project should speak European Portuguese, so this service was not considered for comparison with the others.

Besides that, regarding the model used by Amazon Transcribe Medical and the data used to train it, no information is publicly disclosed similar to Amazon Transcribe.

Microsoft Azure Cognitive Services for Speech

Microsoft Azure Cognitive Services for Speech is a service provided by Microsoft that can transcribe audio into text – ASR –, convert text into human-synthesized speech – Text-to-Speech (TTS) –, translate speech, identify the languages in a given speech, recognize speakers in a speech, assess speech pronunciation and recognize the user's intentions from the transcribed speech. Among other usage scenarios of this service, the dictation in Office 365 can be highlighted.

The functionality of transcribing audio into text is called Speech. It can transcribe audio in real-time and asynchronously, and the audio to be transcribed can have multiple sources, including microphones, audio files, and Microsoft's object storage solution for the cloud, blob storage. This service provides several features, presented in Table 3.13 [mic, 2022h].

Feature
Speaker diarization
Get readable transcripts with automatic formatting and punctuation
Custom speech models
Speech translation
Language Identification
Speaker recognition
Pronunciation assessment
Intent recognition
Custom base vocabularies

Table 3.13: Microsoft Azure Cognitive Services for Speech features (all but Text-to-speech).

Speech can be used either in the cloud or locally, and local use may bring the service closer to user data for compliance, security, or other operational reasons. Government entities and their partners use this service through its deployment in sovereign clouds. An example is the Azure Government cloud that makes this service available to US government entities and their partners.

This service can also be integrated into applications using Speech Studio. This is a set of tools based on the User Interface (UI) to build and incorporate Speech functionalities into applications. Speech Studio allows projects to be created according to a code-free approach, with reference then being made in the application to the resources created in it using the Speech SDK, Speech CLI, and REST APIs. The Speech SDK is available in various programming languages and on all platforms. By its turn, Speech CLI is a command-line tool that allows the Speech service to be used without developing code, having most of the features of the Speech SDK and some advanced features and simplified customizations. Also, REST APIs are used mainly in cases where the Speech SDK cannot or should not be used [mic, 2022h].

This service provides support for European Portuguese and, for the language, it provides the features presented in Table 3.14 [mic, 2022g] [mic, 2022c].

Feature	Description
Translate audio	Translate speech in one language to another or speech in one language to text in another one.

Language identification	Identifies the language in which the speech is.
Customized speech recognition	Customize templates for the specific user use case's domain by providing the template with plain text or pronunciation data. For European Portuguese, it can be used with plain text and pronunciation.

Table 3.14: Features provided by Microsoft Azure for Speech to Text service for European Portuguese.

None of the functionalities previously presented are of particular interest. The audio is recorded in European Portuguese and must be transcribed, so the translation is unnecessary. On the other hand, the language of the audio can be passed to the system, avoiding the need for language identification. In its turn, speech recognition customization requires data and time. Because of the data scarcity, it is irrelevant in this context. With that, only the default transcription is of interest in this project. Although Microsoft Azure does not say that Speech can transcribe digits in European Portuguese, its demo application [mic, 2022b] enables it to demonstrate that it can do it. This feature is a significant advantage since it allows the direct transcription of digits, removing the need for post-processing to convert string numbers to digits. Similarly to Amazon Transcribe, the Speech solution does not provide sensitive information filtering.

Attending to the prices charged for this service, it can be understood that Europe is divided into Northern Europe and Western Europe, being the prices in both regions similar. These prices are divided into three modes: the Free mode (F0), the Pay as You Go mode: pay only for what you use, and the Commitment Tiers mode. The latter is similar to the Pay as You Go modality but offers discounts. To access this modality, it is necessary to submit an online application to evaluate it for a future possible selection. For this reason, this last modality was not considered for price comparison purposes [mic, 2022a]. If we look at Europe, in the Free and Pay as You Go plans, in the Speech to Text category, and in the functionalities made available for European Portuguese, the prices charged correspond to those shown in the Table 3.15 [mic, 2022e].

Modality	Feature	Price
Free (F0)	Standard	5 audio hours free/month
	Custom	5 audio hours free/month Endpoint hosting: 1 model free/month
Pay as You Go	Standard	€1.001/audio hour
	Custom	€1.401/audio hour Endpoint hosting: €0.0538/model/hour

Table 3.15: Microsoft Azure Cognitive Services for Speech pricing.

In this project, the intent is to transcribe audio as it is recorded. This solution provides real-time transcription, which enables audio transcription as it is recognized from a microphone or file. Then, this is an advantage of the current service [mic, 2022f].

Therefore, the limits for data input associated with real-time transcription are presented in Tables 3.16 and 3.17 [mic, 2022d]. The limits associated with the Free plan are not adjustable, but, in turn, the ones regarding the Standard plan are.

Description	**Quota**
Concurrent request limit - base model endpoint	1
Concurrent request limit - custom endpoint	1

Table 3.16: Microsoft Azure data limits for Speech-to-text service, for Free (F0) plan in real-time transcription.

Description	**Quota**
Concurrent request limit - base model endpoint	100 (default value)
Concurrent request limit - custom endpoint	100 (default value)

Table 3.17: Microsoft Azure data limits for Speech-to-text service, for Standard (S0) plan in real-time transcription.

Regarding the customization of models, the limits presented in Tables 3.18 and 3.19 are imposed [mic, 2022d].

Description	**Quota**
REST API limit	300 requests per minute
Max number of speech datasets	2
Max acoustic dataset file size for data import	2 GB
Max language dataset file size for data import	200 MB
Max pronunciation dataset file size for data import	1 KB
Max text size when using the text parameter in the creation of a new template	200 KB

Table 3.18: Microsoft Azure data limits for Speech-to-text service, for Free (F0) plan for model customization.

Description	**Quota**
REST API limit	300 requests per minute
Max number of speech datasets	500
Max acoustic dataset file size for data import	2 GB
Max language dataset file size for data import	1.5 GB
Max pronunciation dataset file size for data import	1 MB
Max text size when using the text parameter in the creation of a new template	500 KB

Table 3.19: Microsoft Azure data limits for Speech-to-text service, for Standard (S0) plan for model customization.

Finally, no information regarding this solution's model and the data used to train it is publicly disclosed.

IBM Watson Speech to Text

IBM Watson Speech to Text is an IBM Watson service that provides speech transcription capabilities to applications. It uses ML to combine knowledge of grammar, language structure, and the composition of audio and speech signals to make accurate transcriptions of the human voice. The transcription is continually updated and enhanced as more speech is received by the service, which provides APIs and can be used in various cases, such as dictating messages and notes. It enables customers across a range of industries, including customers in the healthcare industry, to develop cloud-native applications for a diversity of uses [IBM, a].

As reported by Speech to Text, only Brazilian Portuguese is encompassed [IBM, 2022d]. This fact represents a significant disadvantage for the present service since it should not be able to be used for audio transcription in European Portuguese or, at least, not with the desired quality, given the significant discrepancy in phonemes, namely. This service was not, therefore, considered for the development of this project.

No information is publicly available regarding the model and the training data used by IBM in the present solution.

Google Cloud Speech-to-Text

Speech-to-Text is a service provided by Google Cloud that allows audio transcription and has three main methods for performing speech recognition: synchronous, asynchronous, and real-time recognition. In synchronous speech recognition, audio data is sent to the Speech-to-Text API, which performs speech recognition on the data transmitted and returns the results when all audio processing is completed. In asynchronous recognition, data is also sent to the Speech-to-Text API, but a long-running operation is triggered in this case, allowing the user to poll periodically to obtain recognition results. Finally, the real-time recognition option performs recognition on audio data provided within a bi-directional Google Remote Procedure Call (gRPC) stream. For example, this type of recognition is used in real-time audio transcription captured by a microphone [Cloud, a] [Cloud, 2022a].

This service provides several features, presented in Table 3.20 [Cloud, c].

Feature
Global vocabulary
Streaming speech recognition
Speech adaptation
Speech-to-Text On-Prem
Multichannel recognition
Noise robustness
Domain-specific models
Content filtering
Transcription evaluation

Automatic punctuation (beta version)
Speaker diarization (beta version)

Table 3.20: Google Cloud Speech-to-Text features.

This service supports European Portuguese, offering four templates for the language: command and search, standard, latest long, and latest short. Besides these templates, Speech-to-Text offers other templates for other languages, such as medical dictation and medical conversation templates. The medical dictation template is used for transcribing notes dictated by medical professionals, while the medical conversation template is used for transcribing conversations between a health professional and a patient. These models would be of interest in the present work but are unavailable for European Portuguese, preventing their use. Table 3.21 illustrates the usage of the four available templates for European Portuguese [Cloud, 2022a] [Cloud, 2022b].

Template	Usage
Command and search	Useful in cases where the audios to be transcribed are shorter, for example, audios relating to voice commands or voice search
Latest Long	More appropriate for long-form content, such as spontaneous conversations
Latest Short	Appropriate for short utterances, a few seconds long, and is a useful model in capturing voice commands or other one-time directed speech use cases
Default	Should be used in cases where the audio that is to be transcribed does not fit into any of the other available models

Table 3.21: Description of the templates provided by Google Cloud Speech-to-Text service for European Portuguese.

Since the speeches to be transcribed tend to be long, as they do not correspond to a single voice command, the models applicable in the present work correspond to the default model and the latest long model. For European Portuguese, the functionalities provided by these models are presented in Tables 3.22 and 3.23 [Cloud, 2022b] [Cloud, 2022d] [Cloud, 2022e] [Cloud, 2022f].

Feature	Description
Boost	Customized word recognition
Profanity filter	Filtering out profane words in the final text, being them represented by their first letter and asterisks
Spoken punctuation	Detection of spoken punctuation in the speech, which is converted to the respective punctuation in the transcription

Table 3.22: Default model features for European Portuguese.

Feature	Description
Word-level confidence	Adding a confidence level to transcribed words (in addition to the confidence level associated with the total transcription)
Profanity filter	Filtering out profane words in the final text, being them represented by their first letter and asterisks
Spoken punctuation	Detection of spoken punctuation in the speech, which is converted to the respective punctuation in the transcription

Table 3.23: Latest long model features for European Portuguese.

By testing the Speech-to-Text service in its demo [Cloud, b], it can be seen that it can transcribe digits in a speech in European Portuguese. As previously mentioned, this is a feature of high interest for the current project, given the need to identify complementary information regarding numbers. The profane word filtering is irrelevant since the presence of profane words in medical dictations is not expected. Once again, no sensitive data filtering is available.

The prices charged for this service vary according to whether or not the option of logging data is used. This option is related to the supply of data to the service so that it can successively improve its results and has associated discounts [Cloud, 2022c]. Although this program could be appealing due to the discounts provided, working with medical data that may contain sensitive information removes the use of such a program from the hypothesis. Therefore, focusing on speech recognition without data logging, and in the models of interest in this project, the prices charged are presented in Table 3.24 [Cloud, d].

Minutes/month	Price
0-60	Free
Over 60 up to 1 Million	$0.006 for each 15 seconds

Table 3.24: Google Cloud Speech-to-Text pricing.

Regarding data input limits, the system imposes the ones presented in Table 3.25 [Cloud, 2022a].

Description	Quota
Max audio input file size for the synchronous recognition component	1 minute in duration
Max audio input file size for the asynchronous recognition component	480 minutes (4 hours) in duration

Table 3.25: Google Cloud data limits for Speech-to-Text service.

Finally, no information is publicly available regarding the model and training data used by Google in the present solution.

CMUSphinx

CMUSphinx is a toolkit that is among the most popular open-source toolkits for ASR. It was developed by Carnegie Mellon University, focusing on application production and development [Kamath et al., 2019d] and provides several tools that can be used to build applications involving speech. Currently, it offers two components: Pocketsphinx and Sphinxtrain. Pocketsphinx corresponds to a lightweight speech recognition library written in C, while Sphinxtrain corresponds to acoustic model training tools. This set of tools is still under development, and much needs to be added. Examples are the absence of sense extraction and the absence of post-processing of the decoding result. This solution currently imposes the development of a post-processing system by the user so the decoding result is converted to natural language [CMUSphinx, e] [CMUSphinx, a]. Using CMUSphinx, it is necessary to develop the language model for the domain the user is working in, a post-processing system, an adaptation system, and a user identification system [CMUSphinx, f].

As this toolkit provides the option of developing language models, it will always be possible to use it for European Portuguese by training a new model, which requires data and time availability. The training of language models is done using Pocketsphinx, and this model is responsible for defining which sequences of words are likely to be detected. This model can have one of the types keyword lists, grammars, statistical language models, and phonetic language models. The first allows a list of keywords to be specified to search for in speech. In the second, a grammar describes a straightforward type of command and control language and allows possible inputs to be specified with great precision. Statistical language models, in turn, describe more complex language and contain probabilities of words and word combinations. The model type determines the capabilities and performance properties, which vary between the various types. It is possible to choose the model best suited to the use case and switch between modes at runtime [CMUSphinx, d]. In addition to linguistic models, acoustic models refer to models trained with recourse to data corresponding to audio files. These can be used in any language and trained from scratch or adapted from existing models [CMUSphinx, b].

Pocketsphinx allows the user to read Pulse-Code Modulation (PCM) audio from a standard 16-bit input channel or one or more files and then attempt to recognize speech on it using standard acoustic and linguistic models [Huggins-Daines and Solovets, 2022a]. At no point in the documentation is it said that this set of tools could perform speaker recognition or other functionalities present in the previous services. However, it is possible to perform segmentation and diarization using external tools. Segmentation refers to the division of audio into manageable and distinct homogeneous audio parts, while diarization corresponds to the identification of unique speakers in a given audio file. It can be done by combining LIUM tools with CMUSphinx [CMUSphinx, c]. Apart from these, no other functionality is mentioned.

CMUSphinx provides templates for Brazilian Portuguese only [CMU, 2019]. The fact that no pre-trained model is available for European Portuguese is a disadvantage since it is impossible to leverage knowledge from pre-existing models, which tends to lead to higher performance models, namely when the original model was trained on a large dataset. This implies the need to spend time and have data to develop the whole model from scratch. Besides the tendency to create models with lower performance, there is also a significant investment in their development. The need to develop a complementary system is also a significant disadvantage of the system. Since one of the current project's goals is the time to market, and since the scarcity of data will result in a low performance model, this solution loses its interest in the context.

Besides that, CMUSphinx is free, and no limits regarding input data are mentioned for this set of tools. An analysis of the architecture of its underlying models is made in appendix A.

Kaldi

Kaldi is a speech recognition toolkit that has been developed in C++ and is intended for use by researchers in the field. Like the previous toolkit, it is one of the most popular open-source speech recognition toolkits [Kamath et al., 2019d] [Kaldi, a]. Kaldi makes a wide range of tools available to its users, allowing them to do grouping and calculate errors, among others [Kaldi, c]. Observing this solution's models, it is possible to build speech recognition, speaker identification, speech activity detection, and speaker diarization systems [Kaldi, b]. However, there is no mention in the documentation of models available in European Portuguese. Still, since Kaldi is a set of tools that allows the user to develop his models, it is perfectly possible to build and train any of the systems previously mentioned for the language. The developable system with the most significant interest in the context of this project concerns the speech recognition system. Similarly to the previous set of tools, however, leveraging knowledge from pre-designed models is impossible, and the model building has potentially high costs in terms of time and computational resources. Still, the possibility of identifying and filtering sensitive information is also not mentioned.

This service is free, and no reference is made to limitations in the input data size. Similarly to CMUSphinx, and due to the non-availability of pre-trained models in European Portuguese, Kaldi was not considered for future comparisons. Regarding its components architecture, an analysis is made in B.

Whisper

Whisper is an ASR solution made available by OpenAI in September 2022. It was trained on 680.000 hours of supervised multilingual and multitasking data collected from the web, being usable in 96 languages. This solution is open-source at the models and inference code levels, being developed to serve as a basis for developing applications of interest and for further research on robust speech processing. Its architecture consists of a simple end-to-end approach implemented as an encoder-decoder Transformer, and its generalized training, not adjusted to specific cases, makes it not the best model in particular use cases. However, there is the possibility to adapt it to specific use cases [Gandhi, 2022]. Besides that, in general use cases, this solution has shown satisfactory results [Radford et al., 2022].

This solution offers five sizes of templates for multilingual tasks and four English-only versions. The different models are associated with different counterparts of speed and accuracy, presented in Table 3.26 [whi, 2022].

Model	**Numer of parameters**	**English version**	**Multilingual version**	**Required VRAM**	**Relative speed**
Tiny	39 million	tiny.en	tiny	1GB	32x
Base	74 million	base.en	base	1GB	16x
Samll	244 million	small.en	small	2GB	6x
Medium	769 million	medium.en	medium	5GB	2x
Large	1550 million	-	large	10GB	1x

Table 3.26: Whisper models provided and respective specifications.

Although the main speech recognition is the central task of ASR systems, other tasks of interest must be added to these systems. Whisper adds them sequentially in a single pipeline, being those features illustrated in Table 3.27 [Radford et al.].

Feature	**Description**
Detection of spoken language	The model predicts the language of the speech and assigns the unique token associated with it, which comes from the VoxLingua model107.
Detection of unspoken segments (silences)	The system is trained to predict a specific token indicative of the absence of speech in the audio segment in question when this absence occurs.
Transcription and translation of speech	It is possible to add a token to the text transcription that indicates whether the task to be performed should correspond to a transcription and/or a translation of the speech
Prediction of timestamps	It is possible to add a token that determines whether or not timestamps should be included

Table 3.27: Whisper's pipeline features.

Of the functionalities provided by this solution, only speech recognition is of genuine interest in the context of this project. Once again, no masking of sensitive data is available, which, not being a disadvantage, would be an added value for the system which would make it available. Although it is not said by the authors, by using the system, it can be seen that it can transcribe digits too. In terms of protection, encryption techniques are used by the system for data transmitted between the endpoint and the server to which it is making requests. This data transmission is the riskiest component regarding data security, being Whisper concerned with that. Regarding the customization of the model in use, it is already possible to adjust the models provided by Whisper by training them on specific datasets of the area in which they should be applied. Whisper is trained on a vast amount of labeled data, which allows it to train directly on the supervised speech recognition task and to learn a STT mapping from these labeled data. Due to this fact, Whisper requires little additional tuning to produce a performing ASR model [Gandhi, 2022]. The possibility of adjusting the models already developed to the medical area could be a positive aspect of the context in question. However, this possibility depends on time and computational resources and the existence of datasets with labeled data corresponding to audio-transcribed pairs. This model adjustment work is much smaller than that of the open-source solutions previously presented, but the scarcity of data also makes the fine-tuning unusable in this context.

As seen in [Radford et al.], European Portuguese is included in the multilingual Whisper models. Observing Tables 10, 11, and 13 of the paper, it is possible to understand that not only it is possible to apply all the potentialities of the multilingual models to the language, but also that the Word Error Rate (WER) for it is quite satisfactory for the tests made by the authors since it tends to be in average or below the WER of the other languages.

Besides that, this solution is entirely free, and no limits are mentioned for the input data. Finally, the architecture of its model's architecture is provided in D.

iOS Speech

iOS Speech concerns a system provided by Apple that allows speech structure to do spoken word recognition in recorded or live audio. This system can be used, for example, to recognize verbal commands or to handle text dictation, making available an object to each language it supports. That object is responsible for allowing speech recognition in the language. A device has speech recognition available for some languages, but the framework is still dependent on Apple's servers for speech recognition [1].

The idea of using a native speech recognition system would be to use the potentialities made available by the mobile phone. By only allowing access to the microphone within the final application and then applying the native system, it would be possible to do this speech transcription transparently without developing and integrating a speech recognition functionality. However, this is a static engine in that it cannot be adapted to use cases, and the result obtained using it is directly dependent on its knowledge of the specific terms of the domain. Besides that, this service presents a positive point in the context of this project which corresponds to the fact that it is possible to transcribe numbers in European Portuguese, as it could be tested using an Apple device.

For the context in which the final application is to be used, i.e., in health units in Portugal where cell phones tend to be in European Portuguese, this service should be available in the language. As the aim is to transcribe the medical speeches at the moment they are dictated so that the doctor can confirm what has been transcribed, this system could be used to transcribe live audio.

[1] https://developer.apple.com/documentation/speech

Since this system uses free functionalities intrinsic to the mobile device, integrating them into the final application should not cost extra. Also, no limits imposed on the system's input data are presented for this system. Finally, no information is publicly available regarding the model and training data used by iOS Speech.

Android Speech

Android Speech is a namespace for a rich set of APIs that developers can use to enable a device to understand speech. This is provided by Google and allows both STT transcription and TTS conversion. This service has limitations associated with the hardware that uses it, and it is unlikely that a device will successfully interpret everything spoken to it in all available languages [And, 2021].

Similarly to the engine previously presented, the idea of using a native speech recognition system would be to make use of the potentialities made available by the mobile phone. Like the previous one, this service has the added value of allowing the transcription of dictated digits in European Portuguese, which can be tested on any Android device.

European Portuguese should also be available in this service, considering the context in which the final application is used, similar to the previous service. Also, Android Speech can be used to transcribe the audio in real-time within the application itself [2].

Since this system also uses free functionalities intrinsic to the mobile device, its integration into the final application should not cost extra. Besides, no limits are presented at the input data level for the present system. Despite its advantages, this system can not be considered for an iOS mobile application development, losing its interest in the context of the current project. Finally, no information is publicly available regarding the model and the training data used by Google in Android Speech.

Conclusion

From the first comparative analysis made at the level of European Portuguese support, it was already possible to exclude the following:

- Amazon Transcribe Medical, that only supports English
- IBM Watson Speech to Text, that only supports Brazilian Portuguese

This is a crucial question of the project since the final users of the developed system should be European Portuguese speakers.

Regarding the most important features of these systems, all the solutions transcribe speech, but not all transcribe digits in European Portuguese, and none pro-

[2]https://www.android.com/accessibility/live-transcribe/

vides sensitive information filtering. Although the transcription of digits is relevant, more than its absence is needed to rule out any solution since the conversion of the string numbers to digits can be done with post-processing. However, solutions capable of transcribing digits should be preferred.

Based on the analysis and comparison made for the various ASR solutions, the solutions selected for testing correspond to those listed in Table 3.28.

NLP Tool
Amazon Transcribe
Microsoft Azure Cognitive Services for Speech
Google Cloud Speech-to-Text
Whisper
iOS Speech

Table 3.28: ASR solutions to be tested in experiments section.

3.3 Natural Language Processing

This section first presents the fundamental concepts of NLP and then delivers the most relevant solutions in this area, i.e., the solutions corresponding to its state of the art. A brief analysis of these solutions and a comparison between them is offered, aiming at selecting the solutions likely to be used in the present work. Each platform is analyzed in terms of its description, services, and/or functionalities, the support offered for European Portuguese, the price charged, and the limits applied to the input data.

3.3.1 Introduction

NLP corresponds to an area that deals with human communication, encompassing several approaches that aim to help machines perceive, interpret, and generate human language. These approaches are sometimes outlined as Natural Language Understanding (NLU) and Natural Language Generation (NLG) methods. According to [Chowdhary, 2020b], NLP is the subject of Computational Linguistics (CL), previously presented in this document, and it is a theory-driven field, an academic research domain that is technology-driven and encompasses several computational techniques to deal with human languages, namely to represent and automatically analyze those languages. But this automatic analysis of human languages requires that the machines have a fairly deep knowledge of them which makes machines not yet capable of understanding natural language and imposes considerable challenges in the development of programs for NLU for multiple reasons:

- Natural languages are typically large and complex, which leads to an enormous number of possible sentences in each language

- The natural languages are ambiguous since a single word can change its meaning according to the sentence where it appears, and some sentences can have multiple meanings according to the context

The human brain works in quite complex ways. It can represent each problem and the necessary knowledge in several ways, successively finding new methods to solve a given problem upon the failure of the previously designed methods. For computers to understand humans, it should be necessary to equip them with the appropriate knowledge. However, the most suitable representation structure for a given purpose is unknown even to humans, who, as stated earlier, formulate various reasonings for the same problem until they arrive at the winning approach, i.e., the approach that provides the answer to the problem. Therefore, a uniform way of representing knowledge should not be sought. Instead, it is necessary to use different representations depending on the difficulty of the problem in question, and additional knowledge about the situation should be required to find its solution. Although the human language's enormous richness and complexity cannot be underestimated, there is an increasing need for algorithms capable of understanding this language. The existence of NLP aims to fill this gap.

NLP can be divided into two types of methods: traditional and modern. In traditional methods, the approach is based on linguistics, built on a language's basic semantic and syntactic elements, like Part-of-Speech (POS). On the other hand, modern deep learning approaches can bypass the need for intermediate elements present in traditional methods and can learn their hierarchical representations for generalized tasks [Kamath et al., 2019c].

NLP focus on multiple areas – like machine translation, Information Retrieval (IR), i.e., question answering, text summarization, topic modeling, and opinion mining – and its applications are countless, such as the classification of text into categories according to its content, the automatic translation of texts, their automatic summarization, the extraction of information from them, the possibility of generating question-answering systems, and the acquisition of knowledge in text. In several applications of NLP, it is common to have a first pre-processing phase in which program modules are used on the documents. These program modules include text zoners, segmenters, filters, tokenizers, lexical analyzers, disambiguators – in which both POS and semantic taggers are included – and stemmers [Chowdhary, 2020b].

3.3.2 Main Natural Language Processing Tools

According to [Koneru et al., 2018], the five main NLP cloud service providers in the market are AWS, Salesforce, IBM, Microsoft Azure, and Google Cloud Platform. In the present study, four of these platforms are considered, plus two Python libraries that, according to [Kamath et al., 2019d], are two of the most popular toolkits for NLP. The NLP services studied correspond to those provided by Amazon, IBM, Microsoft, and Google, and the Python libraries to spaCy and NLTK. Their study is detailed below. The choice of these four particular cloud service providers is because they are successively mentioned in the literature as the best service providers in the area of interest.

Since the analysis of the transcribed text is intended to identify, on the one hand, the entities of interest contained therein and, on the other hand, the relationship between them, like the association of an administration frequency to a specific active principle, these should be the two most relevant features in the context of this project. Still, in the medical area, information filtering may be relevant, but this feature should not be of extreme importance, given the underlying tendency not to include sensitive patient information in the dictation of the prescription.

Each solution is explored in the following sections, being a first comparative analysis between the relevant solutions presented in Tables 3.29 to 3.37. Table 3.29 compares the solutions at the level of European Portuguese support.

Tool	Has support
Amazon Comprehend	Yes
Amazon Comprehend Medical	No
Google Cloud Natural Language API	Yes*
Microsoft Azure Cognitive Service for Language	Yes
IBM Watson Natural Language Understanding	Yes
spaCy	Yes

Table 3.29: Comparison of the various NLP tools. (I)
*Except at the Healthcare Natural Language API level.

By its turn, Tables 3.30 and 3.31 compare each service's features available for European Portuguese. NER denotes Named Entity Recognition (NER), and PHI denotes Protected Health Information (PHI).

Tool	NER	Custom NER
Amazon Comprehend	x	x
Amazon Comprehend Medical	x*	
Google Cloud Natural Language API	x	x
Microsoft Azure Cognitive Service for Language	x	x
IBM Watson Natural Language Understanding	x	x
spaCy	x	x

Table 3.30: Comparison of the several NLP tools (II-I)
*For English text only.

Tool	Relationship detection	PII or PHI detection
Amazon Comprehend		
Amazon Comprehend Medical	x*	x*
Google Cloud Natural Language API	x*	
Microsoft Azure Cognitive Service for Language	x	x

IBM Watson Natural Language Understanding	x	
spaCy		

Table 3.31: Comparison of the several NLP tools (II-II)
*For English text only.

In Tables 3.32 to 3.35, a comparison is made at the level of the prices charged by each solution.

Tool	Price				
Amazon Comprehend	Feature		Up to 10 million u	10 to 50 million u	Over 50 million u
	Entity Recognition		0.0001 US$	0.00005 US$	0.000025 US$
	Asynchronous NER		0.0005 US$/u		
	Synchronous NER		0.0005 US$/UI/second		
	Model training		3 US$/hour training		
	Model management		0.50 US$/month		
Amazon Comprehend Medical	API	Free Tier	Up to 1M u	1M to 2M u	Over 2M u
	NERe	85.000 u	$0.01/u	$0.005/u	$0.001/u
	PHI		$0.0014/u	$0.0005/u	$0.00025/u
	ICD10CM		$0.0005/u	$0.0005/u	$0.00025/u
	SNOMED CT		$0.0075/u	$0.00375/u	$0.00075/u
	RxNorm		$0.00025/u	$0.00025/u	$0.00025/u

Table 3.32: Comparison of the various NLP tools. (III-I)
u means unit or units

Tool	Price				
Google Cloud Natural Language API	Feature	0 – 5K	5K – 1M	1M – 5M	5M– 20M
		u/month			
	Entity Analysis	Free	$1.00	$0.50	$0.25
	Entity extraction (AutoML)	First 50.000 u/month/ project		Next 950.000 up to 1.000.000 u/month/ project	
		$86/1.000 units/human labeler		$60/1.000 units/human labeler	
Google Cloud Healthcare Natural Language API	Feature	0 – 5K	5K – 1M	1M – 5M	5M– 20M
		u/month			
	Entity analysis	$0.10 text records/month			

Table 3.33: Comparison of the various NLP tools. (III-II)
u means unit or units

Tool	Price			
Microsoft Azure Cognitive Service for Language - Free instance	Features	Inferencing*	Training	Model endpoint hosting

	NER, including PII Custom NER	5.000 text records free per month	-	
			Up to 1 hour free	Up to 1 model free
Microsoft Azure Cognitive Service for Language - Standard instance	NER, including PII	0.0M-0.5M: €0.9618 0.5M-2.5M: €0.7214 2.5M-10.0M: €0.2886 10.0M+: €0.2405	-	
	Custom NER	€4.809	€2.886/ hour	€0.481/ model/ month
	Text Analytics for Health**	0M-0.005M*** – Free 0.005M-0.5M***: €26 0.5M-2.5M***: €16 2.5M+***: €11	-	

Table 3.34: Comparison of the various NLP tools. (III-III)
*(Per 1,000 text records)
**(available in containers)
*** text records

Tool	**Price**		
IBM Watson Natural Language Understanding	Usage	Lite	Standard
	1-250K items/month	First 30K Free	0.003$/ item
	250K-5M items/month	–	0.001$/ item
	5M+ items/month	–	0.0002$/ item
	Custom entities and relations model trained with WKS (USD/model/month)	one free custom model	800$
	Custom classification model (USD/model/month)	1 free custom model	25$
spaCy	0		

Table 3.35: Comparison of the various NLP tools. (III-IV)
u means unit or units

Finally, Tables 3.36 and 3.37 present a comparison of the solutions at the level of input data limits.

Tool	**Input data limit**		
Amazon Comprehend	Character encoding		UTF-8
	Maximum document size		100 KB
	Custom analysis - API	Maximum size UTF-8 text documents	10 KB
		Maximum size PDF documents	10 MB
		Maximum size word documents	10 MB
		Maximum size image files	10 MB
		Maximum size textract output files	1 MB

	Custom analysis - Console	Maximum size UTF-8 text documents	10 KB
		Maximum size PDF documents	5 MB
		Maximum size word documents	5 MB
		Maximum size image files	5 MB

Table 3.36: Comparison of the various NLP tools. (IV-I)

Tool	Input data limit		
Amazon Comprehend Medical	Character encoding		UTF-8
	Maximum document size	Entity and PHI detection	20 KB
		Inference of ontology to ICD10-CM and RxNorm	10 KB
		Inference of ontology links to SNOMEDCT	5 KB
Google Cloud Natural Language API	AutoML	50 to 100.000 training items 1 to 100 labels per dataset Names from 1 to 32 characters for labels Annotated range length from 1 to 100 characters 100 to 100.000 training items per label Training items up to 128 KB (text format) or 20 MB (PDF format) Up to 20 MB of items sent for prediction	
	Natural Language API	Up to 1 MB of input text Input text with up to 100.000 tokens Input text with up to 5.000 entity names	
Microsoft Azure Cognitive Service for Language	Personalised recognition of named entities	10 to 100.000 input documents Input documents with 1 to 128.000 characters 1 to 200 types of entities Entities with 1 to 500 characters 0 to 10 trained models per project 0 to 10 deployments per project	
spaCy	Does not have		

Table 3.37: Comparison of the various NLP tools. (IV-II)

Amazon Comprehend

Amazon Comprehend is an AWS service that uses NLP to develop knowledge about a document's content. It uses a pre-trained model to examine and analyze a document or a set of documents to extract knowledge, being this model continually trained on a large body of text, so the user does not need to provide any training data. This service can extract knowledge through the features enunciated in Table 3.38.

Feature
Entities Recognition
Key phrases Recognition

Language Recognition Sentiment Analysis PII Recognition Targeted Sentiment Analysis Syntax Analysis Custom classification Custom entity recognition Document clustering

Table 3.38: Amazon Comprehend features.

The functionality of this service can be used using the Amazon Comprehend console or the service Application Programming Interface (API)s. It allows the user to perform real-time analysis for small documents or document sets or initiate asynchronous analysis jobs for large document sets. In addition, this service offers the possibility to use pre-trained models available there or to train custom models for document classification and entity recognition, allowing the user to customize models for specific requirements without requiring in-depth knowledge of developing ML-based NLP solutions. For that, the user can use AutoML since it allows them to create customized NLP models without requiring their intervention in the development and using the data the user already has to train the customized model [Amazon Web Services, g].

Amazon Comprehend supports multiple languages, so it should be noted that the text analysis features available vary depending on the language in question. It offers support for European Portuguese and, with this language, it provides the features presented in Table 3.39 [Amazon Web Services, g] [Amazon Web Services, a].

Feature	**Description**
Language recognition	Determination of the dominant language of a given document
Entity recognition	Recognition of entities that refer to names of people, places, items, and locations contained in the document
Key phrases extraction	Recognition of the phrases that appear in a given document and are intrinsically related to the topic of the document
Sentiment analysis	Determination of the dominant sentiment present in the document. Sentiments can be divided into positive, neutral, negative, and mixed
Syntax Analysis	Identification of the parts of speech to which each of the words in a given document refers
Topic modeling	Examination of a corpus of documents and consequent organization of the documents into clusters based on similar keywords among them
Custom classification	Classification of documents into specific, user-defined categories

Custom entity recognition	Identification of specific terms and phrases based on names

Table 3.39: Amazon Comprehend features provided for Portuguese.

Of the functionalities offered by this service for European Portuguese, only entity recognition and custom entity recognition are relevant to the context of this project. The possibility of customization is an added value to this service because the standard models were insufficient for extracting the entities of interest in the present work. However, the absence of identification of relations between the identified entities is a disadvantage of this system since it is one of the most relevant features to have in the model of this project and one of the most difficult ones to train a model to do.

The price charged for work that may be done with this service varies from resource to resource. For the resources of interest in this project, the respective prices are summarised in Tables 3.40 and 3.41 [Amazon Web Services, f].

Feature	**Price**		
	Up to 10 million units	**10 to 50 million units**	**Over 50 million units**
Entity Recognition	0.0001 USD	0.00005 USD	0.000025 USD

Table 3.40: Amazon Comprehend pricing (I).
1 unit = 100 characters

Feature	**Price**
Asynchronous entity recognition	0.0005 USD/unit
Synchronous entity recognition	0.0005 USD/UI/second
Model training	3 USD/hour training
Model management	0.50 USD/month

Table 3.41: Amazon Comprehend pricing (II).
UI: inference unit
One unit = 100 characters. Minimum of three units per request

In turn, regarding data limits and focusing on the features of interest for this project and on synchronous jobs, which are suitable for small documents – that should correspond to the type of document used in this work – in which a call is made to the API for each document intended to be processed, this service imposes the limits present in Table 3.42 [Amazon Web Services, b]. The augmented manifest files referred in the table correspond to JSON files in which each line corresponds to a complete JSON object containing both the training document and the associated labels [Amazon Web Services, e].

Description	**Quota**
Character encoding	UTF-8

	Maximum document size	100 KB
Custom analysis	Maximum size UTF-8 text documents (API)	10 KB
	Maximum size UTF-8 text documents (console)	10 KB
	Maximum size PDF documents (API)	10 MB
	Maximum size PDF documents (console)	5 MB
	Maximum size word documents (API)	10 MB
	Maximum size word documents (console)	5 MB
	Maximum size image files (API)	10 MB
	Maximum size image files (console)	5 MB
	Maximum size textract output files (API)	1 MB
Plaintext entity recognition – training	Number of entities per model/custom entity recognizer	1–25
	Document size (UTF-8)	1–5.000 B
	Number of documents	3–120.000
	Document corpus size (all docs in plaintext combined)	5 KB – 100 MB
	Minimum number of annotations per entity	25
	Number of items in entity list	1–1 million
	Length of individual entry (post-strip) in entry list	1–5.000
	Entity list corpus size (all docs in plaintext combined)	5 KB – 100 MB
PDF or Word text entity recognition – training	Number of entities per model/custom entity recognizer	1–25
	Maximum annotation file size (UTF-8 JSON)	5 MB
	Number of documents	250–10.000
	Document corpus size (all docs in plaintext combined)	5 KB–1 GB
	Minimum number of annotations per entity	100
Augmented manifest files entity recognition – training	Maximum number of augmented manifest files	5
	Maximum number of attribute names for each augmented manifest file	5
	Maximum length of attribute name	63 characters

Table 3.42: Quotas associated with the features of interest provided by Amazon Comprehend for European Portuguese.

Finally, no information is found in official sources regarding the models/algorithms and the training data used by this solution. Thus, this information should not be publicly available and is not exploited.

Amazon Comprehend Medical

Like the previous service, Amazon Comprehend Medical is also provided by AWS. It can detect and return useful information in unstructured clinical texts such as clinical notes or test results by using NLP models. Those provide the features enumerated in Table 3.43.

Feature	Description
Entity recognition	Identification of textual references to medical information such as medical conditions, drugs, or protected health information, or PHI
Entities linking	Link of the detected entities to standard medical knowledge bases (for example, RxNorm, which encompasses all the medicines available in the United States of America) through ontological links

Table 3.43: Amazon Comprehend Medical features.

This service can only detect medical entities in English-language texts. Still, this fact should not hinder its use on this project since the initially transcribed text can be translated and, consequently, Amazon Comprehend Medical can be used with all its potential, specifically focused on the medical area. It should be noted, however, that the system's performance will be affected not only by the quality of the transcription of the medical discourse but also by the translation of the transcribed text.

This service works based on confidence scores, which indicate the level of confidence that the service has in the accuracy of the entities it detects. The interpretation of these scores should be based on a confidence threshold characteristic of the use case. Cases that require high accuracy should use a higher threshold. Also, in some cases, the system might need human review and verification of the results by properly trained human reviewers.

This service values data security by being a HIPAA-eligible service, i.e., it can be configured to meet HIPAA compliance requirements, which refers to the United States of America federal law that requires national standards to be created to protect against the disclosure, without the consent or knowledge of the patient, of sensitive patient health information [for Disease Control and Prevention]. Furthermore, all connections to the service containing protected health information must be encrypted, and by default, they use HTTPS over TLS. Also, the service does not persistently store its client's content, so it is unnecessary to configure at-rest encryption within the service.

Amazon Comprehend Medical can be accessed in three different ways, being them through the AWS Management Console, the AWS Command Line Interface, or the AWS SDKs. AWS Management Console provides a web interface to access this service. On the other hand, AWS Command Line Console provides commands for several AWS services, including Amazon Comprehend Medical, and is supported on Windows, macOS, and Linux. Finally, AWS SDKs, or software development kits, are libraries and sample code for various programming languages and platforms such as Python, iOS, and Android. These provide a convenient way to create programmatic access to the service provided by AWS and to AWS itself [Amazon Web Services, c].

The functionalities provided by Amazon Comprehend Medical are the exact ones that are of interest in the present work, allowing not only the identification of the desired entities but also the linking of them to standard medical knowledge bases and, then, the identification of the relations between entities, like a relation between a International Non-proprietary Name (INN) and its dosage. Also, it provides security and protection of health data, which, as indicated before, is a major concern of medical software.

Regarding the price for this service, the same values are specified for both Ireland and London, being these the only European countries included in the regions. The fee charged varies according to the API used, with a free tier available for the first month of use of any of the APIs. The information regarding the price of the service is summarized in Table 3.44.

API	**Free Tier**	**Price (per unit)**		
		Up to 1M units	**1M to 2M units**	**Over 2M units**
NERe	85.000 units (8.5M characters, or about 1000 5-page 1700-character per page documents)	\$0.01	\$0.005	\$0.001
PHI		\$0.0014	\$0.0005	\$0.00025
ICD10CM		\$0.0005	\$0.0005	\$0.00025
SNOMED CT		\$0.0075	\$0.00375	\$0.00075
RxNorm		\$0.00025	\$0.00025	\$0.00025

Table 3.44: Amazon Comprehend Medical pricing.
1 unit = 100 characters

In turn, regarding the limits for input data and focusing on synchronous jobs, which are best suited to make individual requests for each file, Amazon Comprehend Medical imposes the limits enumerated in Table 3.45 [Amazon Web Services, d].

Description	**Quota**
Character encoding	UTF-8
Maximum document size for entity detection and protected health information detection operations	20 KB
Maximum document size for inference of ontology links to ICD10-CM and RxNorm	10 KB
Maximum document size for inference of ontology links to SNOMEDCT	5 KB

Table 3.45: Amazon Comprehend Medical input data limits.

Finally, no information is found in official sources regarding the models/algorithms and the training data used by this solution. Thus, this information should not be publicly available and is not exploited.

Google Cloud Natural Language API

Google Cloud Natural Language API is a service provided by Google that uses ML to provide NLU technologies. This service offers several methods for analyzing and annotating user text, providing essential information for understanding the language at each level of analysis it provides. These methods are illustrated in Table 3.46.

Feature
Syntax analysis
Entity analysis
Custom entity extraction
Sentiment analysis
Custom sentiment analysis
Content classification
Custom content classification
Custom models
Powered by Google's AutoML models
Spatial structure understanding

Table 3.46: Google Cloud Natural Language API features.

Natural Language API provides three key features: AutoML, Natural Language API, and Healthcare Natural Language API. AutoML allows the user to train its custom ML models, offering the features presented in Table 3.47. This feature corresponds to an approach where the effort by the user is minimal, there is no need for code development by the user, and ML expertise is used through the use of Vertex AI.

Feature
Custom classification
Custom entity extraction
Custom sentiment detection

Table 3.47: AutoML features.

Natural Language API, in turn, provides its user with pre-trained models that allow the integrations of NLU into their applications. The included functionalities are illustrated in Table 3.48.

Feature
Sentiment analysis
Entity analysis
Entity sentiment analysis
Content classification
Syntactic analysis

Table 3.48: Natural Language API features.

There is also the AutoML Entity Extraction for Healthcare model that allows the construction of custom knowledge extraction models for healthcare and life science applications without the need to write any code. It corresponds to a resource included within AutoML, namely in its entity analysis functionality.

By its turn, Healthcare Natural Language API corresponds to the branch of Natural Language API focused on the medical area and allows real-time analysis of knowledge contained in unstructured medical texts, making it possible to extract machine-readable medical knowledge [Cloud, i] [Cloud, h]. With Healthcare Natural Language API, the features enumerated in Table 3.49 are provided. However, this resource does not support documents in European Portuguese. Similarly to what was said for Amazon Comprehend Medical, the absence of support for the language does not prevent its use in the present work. However, it is required to add a translation component to the designed solution. Once again, the quality of the processing will depend on both the quality of the transcript and the quality of the translation [Cloud, f] [Cloud, g].

Feature
Extract recognized medical knowledge entities
Extract functional characteristics
Extract relationships between known entities
Extract contextual attributes
Extract mappings of medical knowledge entities into standard terminologies

Table 3.49: Healthcare Natural Language API features.

Regarding the ability of the resources provided by Google Cloud Natural Language API to process documents written in European Portuguese, the features presented in Table 3.50 are provided [Cloud, e] [Cloud, j] [Cloud, l] [Cloud, k].

		Feature	**Description**
AutoML		Custom entity analysis	Identification of entities in the user's documents labeling them based on domain-specific keywords or phrases
AutoML	Natural Language API	Content classification	Content analysis of the input text, and return of the content category in which the content in question fits
AutoML	Natural Language API	Entity analysis	Inspection of the input text to identify known entities, which may correspond to proper names or common names. Information on these entities is returned
AutoML	Natural Language API	Sentiment analysis	Inspection of the input text and identification of the predominant emotional opinion, particularly to determine whether the attitude of the text author is positive, negative or neutral
	Natural Language API	Syntactic analysis	Extraction of linguistic information with the input text being partitioned into a series of sentences and tokens. A more detailed analysis is provided in these tokens

Table 3.50: Google Cloud Natural Language API features provided for European Portuguese.

Of the functionalities made available for European Portuguese, only the analysis and the customized analysis of entities are relevant in this work. This service also presents two advantages regarding its resources focused on the medical area: the Healthcare Natural Language API and the AutoML Entity Extraction for Healthcare. The former only supports English text, which, as mentioned before, does not hinder its use but leads to conditioned final performance. Since the health-focused AutoML is part of AutoML itself, it should be possible to use it to develop healthcare models that use European Portuguese since this is a language supported by AutoML. An advantage of this service is that it can relate entities between them using the Healthcare Natural Language API component, one of this project's most essential features. Still, this feature is not available in European Portuguese. In this context, the most interesting among the resources provided by this service should be Healthcare Natural Language API, which has the disadvantage of depending on a translation but offers the two most significant features in this project.

Regarding the price associated with using the functionalities of interest provided by this service, their use is calculated in units, each corresponding to 1.000 characters. Each document sent to the API corresponds to at least one unit, and multiple units are considered when they exceed 1.000 characters, one unit per every 1.000 characters. The prices charged are shown in Tables 3.51 and 3.52 [Cloud, o] [Cloud, p].

Feature	**Price (units/month)**			
	0 – 5K	**5K+ – 1M**	**1M+ – 5M**	**5M+ – 20M**
Entity Analysis	Free	$1.00	$0.50	$0.25

Table 3.51: Cloud Natural Language pricing.

Feature	**Price (text records/month)**
Entity analysis	$0.10

Table 3.52: Healthcare Natural Language API pricing.

As far as AutoML is considered, the price is based on the number of annotation units, and for text data, the values are illustrated in table 3.53 [Cloud, q].

Feature	**Unit**	**Price (per 1,000 units per human labeler)**	
		First 50,000 units/month/project	**Next 950,000 up to 1,000,000 units/month/project**
Entity extraction	Entity	$86	$60

Table 3.53: VertexAI pricing. (AutoML)

Regarding the AutoML limitations, the service imposes the ones listed in Table 3.54 [Cloud, n]. Since AutoML Entity Extraction for Healthcare corresponds to a part of AutoML itself, the limits associated with the former correspond to those associated with the service where it is inserted.

Type of limit	Entity extraction
Training items	50 to 100,000
Labels per dataset	1 to 100
Length of label name	1 to 32
Length of annotated span	1 to 100 characters
Training items per label	100 to 100.000
Training item size	128 KB (text); 20MB (PDF) 10 to 300.000 characters (text)
Item sent for prediction	20MB
Items per batch request	10.000

Table 3.54: Google Cloud Natural Language API AutoML limits.

On the other hand, the Natural Language API imposes some limits regarding input data. Those are enumerated in Table 3.55 [Cloud, m].

Content Quota	Value
Text Content	1.000.000 bytes
Token Quota	100.000 tokens
Entity Mentions	5.000 (explicit or in the form of pronouns)

Table 3.55: Google Cloud Natural Language API limits.

Finally, no information is found in official sources regarding the models/algorithms and the training data used by this solution. Thus, this information should not be publicly available and is not exploited.

Microsoft Azure Cognitive Service for Language

Microsoft Azure offers, within Cognitive Services, the Language service aimed at understanding conversations and unstructured text [Microsoft, a]. This cloud-based service provides NLP capabilities to understand and analyze text. It can aid the development of applications with a NLP component by using web-based Language Studio, REST APIs, or client libraries.

The service includes Text Analytics, QnA Maker, and LUIS (Language Understanding), with various functionalities that can be pre-configured or customizable. In the first case, the user only sends their data and uses the output of the service functionalities in their application, not customizing the AI model to be used by the service. In the second case, the user trains the AI model to adjust it precisely to their data using the tools provided by the service. It should be noted that according to [azu, 2022b], LUIS will be withdrawn in 2025, with no new resources being allowed to be created as of April 2023. This second date is before the finalization of the current project, removing interest from the present solution. Also, QnA Maker will be retired in 2025, with a new version of the question and answering capability already available. This feature is, however, not particularly relevant in this project where it is not a focus to have a natural conversational layer [azu, 2021].

This service provides many different functionalities, listed in Table 3.56 [azu, 2022a].

Feature
NER
Personally identifying (PII) and PHI detection
Language detection
Sentiment Analysis and opinion mining
Summarization
Key phrase extraction
Entity linking
Text analytics for Health
Custom text classification
Custom NER
Conversational language understanding
Orchestration workflow
Question answering

Table 3.56: Microsoft Azure Cognitive Service for Language features.

It also offers support for many languages, varying the features offered. This service supports European Portuguese, and, for the language, the functionalities listed in 3.57 are available [azu, 2022a].

Feature	Description
Custom text classification	Development of personalised AI models for documents classification
Custom NER	Development of customized AI models that allow them to extract personalized entity categories
Conversational language understanding	Development of custom models of natural language understanding so that it is possible to predict the general intent of a given statement and extract important information from it
Language detection	Detection of the language in which a text is written. It returns the language code for the language, variant, dialect, or regional/cultural language concerned
Key phrase extraction	Evaluation and return in list form of the main concepts in an unstructured text

NER	Identification of entities in unstructured texts using predefined categories (for example, people, places, dates, among others)
PII detection	Identification, categorization, and redaction of sensitive information (such as mobile phone numbers or email addresses) either in unstructured text documents or in transcripts of conversations
Sentiment analysis and opinion mining	Use text extraction to get clues about positive or negative feelings, which can also be associated with specific aspects of the text under analysis
Summarization	Extractive text summarization, that is, the sentences that collectively represent the most relevant or important information of the original content is extracted so that a summary of documents or transcripts of conversations can be produced

Table 3.57: Microsoft Azure Cognitive Service for Language features for European Portuguese.

One of the functionalities that would be of interest in this work is the text analysis functionality for health. This feature is only available in Brazilian Portuguese, but because of its main interest in the current project, it will be tested and compared with the other solutions. Text Analytics for Health provides the features enumerated in Table 3.58 [Azu, 2022].

Feature
Named entity recognition
Relation extraction
Entity linking
Assertion detection

Table 3.58: Features provided by Text Analytics for Health.

From the features provided for European Portuguese, only the NER and custom NER are interesting in this project. The detection of PII adds value to this solution, but it's not a major requisite for this work. Text Analytics for Health is the main interest of this service in this context since it allows its user to identify and relate medical entities in unstructured text [Azu, 2022].

Attending to the prices practiced for this service, it may be understood that Europe is divided into Northern Europe and Western Europe, and the prices in the two regions are similar, being both considered for the following analysis. For this service, there are two ways of making resources available: through the Web or by joining commitment tiers, whose access is limited. The use of the functionalities made available by this service based on a commitment tiers plan will not be considered, given its requirements to be able to request access to it. Considering only the Web part of this service, two different levels are available: Free and Standard. If we look at Europe, the Web plans and functionalities of interest available for European Portuguese and the Text Analytics for Health, the prices charged correspond to those on Table 3.59 [Microsoft, b].

Features	**Inferencing***	**Training**	**Model endpoint hosting**
NER, including PII	0.0M-0.5M – €0.9618 0.5M-2.5M – €0.7214 2.5M-10.0M – €0.2886 10.0M+ – €0.2405	-	
Custom NER	€4.809	€2.886/hour	€0.481/model/month
Text analytics for health**	0M-0.005M text records – Included 0.005M-0.5M text records – €26 0.5M-2.5M text records – €16 2.5M+ text records – €11	-	

Table 3.59: Microsoft Azure Cognitive Service for Language pricing (Standard (S) instance).
*Per 1,000 text records
**(available in containers)

Focusing on the functionalities of interest made available for Portuguese of Portugal and Text Analytics for Health, the limits enumerated in 3.60 are imposed [azu, 2022c] [aahill, 2023].

Feature	**Item**	**Lower Limit**	**Upper Limit**
Custom NER	Documents count	10	100.000
	Document length in characters	1	128.000 characters
	Count of entity types	1	200
	Entity length in characters	1	500
	Count of trained models per project	0	10
	Count of deployments per project	0	10
Text Analytics for health	Characters per document		125.000

	Request size		1MB
	Documents per request (synchronous)		25 (web-based API) 1000 (container)
	Documents per request (asynchronous)		25
NER	Characters per document		5.120 (synchronous) 125.000* (asynchronous)
	Documents per request		5 (synchronous) 25 (asynchronous)
PII detection	Documents per request (synchronous)		5
	Documents per request (asynchronous)		25

Table 3.60: Microsoft Azure Cognitive Service for Language data limits.
*across all documents

Finally, no information is found in official sources regarding the models/algorithms and the training data used by this solution. Thus, this information should not be publicly available and is not exploited.

IBM Watson Natural Language Understanding

IBM Watson Natural Language Understanding is a service provided by IBM Watson that allows its users to analyze semantic features of text input, including the features presented in Table 3.61.

Feature
Categories analysis
Concepts analysis
Emotion analysis
Entities analysis
Keywords analysis
Metadata analysis
Relationships analysis
Semantic roles analysis
Sentiment analysis
Syntax analysis
Summarisation analysis (Experimental)

Table 3.61: IBM Watson Natural Language Understanding features.

By sending a request to the API using text, HTML, or a public URL, it is possible to use one or more of the previously mentioned features [IBM, 2022a]. The service offers support for many languages, varying the features according to the considered language. For European Portuguese, the service offers the functionalities presented in Table 3.62 [IBM, 2022a] [IBM, 2021] [IBM, 2022c].

Feature	Description	Support
Categories analysis	Categorisation of the input content according to a 5-level classification hierarchy	Standard
Concepts analysis	Identification of high-level concepts that do not necessarily have to be referenced directly in the text	Standard
Entity analysis	Identification of various types of entities mentioned in the text (for example, people, places, events, among others)	Standard and Customized models
Keyword analysis	Search for relevant words in the document content	Standard
Metadata analysis	Finding the author, title, and publication date of a web page passed to the system through HTML or URL entries	Standard
Relationships analysis	Recognition and identification of the type of relationship between two entities when they are related	Standard and Customized models
Sentiment analysis	Retrieve information about the sentiment related to specific target sentences, the whole document, or related to detected entities and keywords (in this semantic feature, it is required that their option of sentiment is enabled)	Standard
Syntax analysis	Identification of sentences and tokens in the input text	Standard
Classifications (or tone analytics)	Language tone detection in the written text (tone can be sad, frustrated, satisfied, excited, polite, impolite, and nice) based on a model of pre-constructed classifications	Customized models

Table 3.62: IBM Watson Natural Language Understanding features provided for European Portuguese.

This service offers the two features that are most relevant for the context of this project, which is a significant advantage of it.

Regarding the price for commercializing the functionalities of interest for this project, there are two different price plans: Lite and Standard. Both plans are described in Table 3.63 [IBM, b] [IBM, c].

Usage	Lite	Standard ($)
1-250K items/month	First 30K Free	0.003/item
250K-5M items/month	–	0.001/item
5M+ items/month	–	0.0002/item

Custom entities and relations model trained with WKS (USD/model/month)	One free custom model	800
Custom classification model (USD/model/month)	1 free custom model	25

Table 3.63: IBM Watson Natural Language Understanding pricing.

Besides that, this service also imposes limits associated with the input data, presented in Table 3.64 [IBM, 2022b].

Description	Limit
Maximum input text size	50,000 single-byte or multibyte characters

Table 3.64: IBM Watson Natural Language Understanding data limits.

Finally, no information is found in official sources regarding the models/algorithms and the training data used by this solution. Thus, this information should not be publicly available and is not exploited.

spaCy

spaCy is a free, open-source library that allows its user to do advanced NLP in Python. It was specifically developed for use in the production and development of applications that process and "understand" large volumes of text, and it can be used both for the development of systems capable of extracting information from text written in natural language and for the development of systems for understanding this language [spa, a].

spaCy provides trained pipelines that can be installed as Python packages, i.e., these pipelines are a component of the application under development, just like any other module. These trained pipelines refer to input text processing sequences that have already been trained. They typically include a tagger, a lemmatizer, an interpreter, and an entity recognizer, continuously processing the input text and then passing it on to the next pipeline component. The schema of the pipeline can be seen in Figure 3.1 [spa, c] [spa, d].

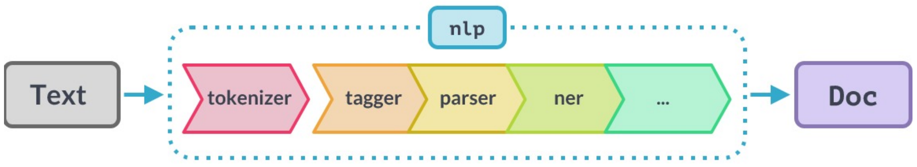


Figure 3.1: spaCy trained pipelines' schema.

spaCy focuses on the fact that it is generally better to use linguistic knowledge when processing raw text to add helpful information to it and, with this library, it is possible to input raw text that will be returned in the form of a Doc object, which will already have a variety of annotations. This library then offers a variety of functionalities of interest in NLP, being them enumerated in Table 3.65.

Feature
Part-of-text tagging
Morphology analysis
Lemmatization
Dependency analysis
NER
Entity binding
Tokenization
Join and split
Sentence segmentation
Mappings and exceptions
Vectors and similarity
Language data

Table 3.65: spaCy statistical and rule-based models features.

Within each of these functionalities, several features are also offered [spa, e]. The functionalities previously presented provide statistical models, rule-based models, or both, the former being strongly dependent on their training phase. In this phase, the model weights under training for the use case are defined for the labels specified in the model in question [spa, f].

Other features offered by spaCy correspond to its engines and rule-based components that not only allow finding the words and phrases the user wants to find in the text, as in the previously presented functionalities but also allow access to the tokens within the document in question and their relations. This allows the user to easily access and analyze the surrounding tokens, merge the ranges into single tokens, or even add entries to the named entities, and several engines are available for this purpose. Those engines are enumerated in Table 3.66.

Feature
Token-based matching
Efficient sentence matching
Dependency matching
Rule-based entity recognition
Rule-based interval matching

Table 3.66: spaCy engines and rule-based components' features.

These models can be used independently, or statistical models can be combined with rule-based models in various ways, and like statistical methods, rule-based models each offer a variety of methods [spa, g].

In addition, with spaCy, it is possible to perform several transfer learning and multi-task learning workflows that can help improve the efficiency or accuracy of a pipeline. Considering transfer learning, tasks such as word vector tables and language model pre-training can be highlighted, bringing raw text knowledge into a given pipeline and allowing the model in question to have a greater ability to generalize from the user's annotated examples. The transformed model or other contextual embedding model developed by the user could be shared across the various components of their pipeline, potentially making long pipelines several times more efficient. In the context of spaCy, transfer learning always requires at least some annotated examples relating to what is to be predicted [spa, h].

The models offered by spaCy are based on neural networks, allowing users to define or change their configuration and include neural networks as sub-layers of another neural network [spa, i]. This library offers, for European Portuguese, three distinct packages. All three offer the same capabilities, illustrated in Table 3.67.

Feature	Description
Part-of-text tagging	Phase where the trained pipeline and its statistical models come into play to allow predictions to be made concerning the label or tag that best applies to the context in question
Dependency analysis (parser)	Syntactic dependency analysis, sentence boundary detection and allows iteration over basic nominal phrases or chunks of text [spa, k]
Lemmatization	Reduction of the words in the text being processed to their root form
NER	Assignment of labels to entities based on a statistical entity recognition system. Standard trained pipelines are capable of recognizing a variety of named and numeric entities, and it is also possible to add new (user-defined) categories to the entity recognition system and update the model by providing new examples

Table 3.67: Features provided by spaCy's models for European Portuguese.

Entity recognition is done here through prediction requests to the model, which, being statistical and heavily dependent on the examples they were trained with, may not work perfectly and must be fine-tuned depending on the use case.

The three offered packages differ because, of the three, one does not provide support for word vectors, and among the two that offer this support, the differences translate into the number of keys and unique vectors supported. Besides, all three packages offer pipelines trained in news and media texts [spa, j] [spa, b].

Among the features provided for European Portuguese, only NER is relevant in this context. Nevertheless, this solution does not support identifying relations between recognized entities, which is a disadvantage in this project. The possibility of customization of the detection of entities is an advantage of the solution since the standard models are insufficient for the goal of the present work.

spaCy is free of charge, and no limits are indicated for the input data for this library, so it should have no restrictions on the input data size. Finally, regarding the architecture of the components of the models provided by spaCy for European Portuguese, an analysis is made in E.

NLTK

NLTK is a platform for developing Python programs to work with human language data. It is an open-source, free project that provides many usable interfaces to over 50 corpora and lexical resources. It also provides a set of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. It also provides an active discussion forum and wrappers for industrial-strength NLP libraries [NLT, a]. This platform supports several language processing tasks, as illustrated in Table 3.68 [Loper et al., 2009].

Language processing task
Accessing corpora
String processing
Collocation discovery
Part-of-speech tagging
Machine learning
Chunking
Parsing
Semantic interpretation
Evaluation metrics
Probability and estimation
Applications
Linguistic fieldwork

Table 3.68: NLTK features.

It is important to note that NLTK allows the recognition of named entities being done inside chunking, as it can be seen in the examples made available by the authors of the solution [3].

This platform allows the analysis of text in European Portuguese, and at [NLT, b], it is possible to see an example where, among other analyses, it is possible to perform a text search, the representation of text as a list of words, access and read the corpora, generate a concordance for a given word taking into account a given context, mark parts of the text, segment sentences, reduce words to their root form and identify stop words.

With that, NLTK provides one of the desired features: entity recognition. Nevertheless, it does not give pre-trained models. This solution genuinely depends on big data, time, and computational resources. This makes the time to market a solution deployed with NLTK above the one expected in the present work. The intent is to produce a solution to go to the market as fast as possible, with good

[3]https://www.nltk.org/some-simple-things-you-can-do-with-nltk

quality. For that reason, and because it can't be accomplished using NLTK, this solution was not further explored.

This solution is free, and, like the previous library, no size limits are mentioned for the input data, so this platform should not present restrictions at this level. Finally, regarding the architecture of the components provided by NLTK, an analysis is provided in E.

Conclusion

Although it is more convenient to use a tool that offers support for European Portuguese, the fact that they do not support it does not prevent its use. The translation of the transcribed text should allow the usage of tools that do not support the language, as the system's performance depends on both the quality of the transcribed text and the quality of the translated text.

Besides that, compared to the high price of IBM Watson's NLP and Google Cloud's AutoML services compared with other solutions, these were not considered for future tests or used in this project.

Finally, based on the analysis and comparison made for the various NLP solutions, the ones tested in the following section correspond to those listed in Table 3.69.

NLP Tool
Amazon Comprehend
Amazon Comprehend Medical
Microsoft Azure Cognitive Service for Language
Google Cloud Natural Language API
spaCy

Table 3.69: Solutions to be tested in preliminary experiments section.

3.4 Conclusion

From the previous study, it was possible to identify both ASR and NLP solutions suitable for the current project. These include Amazon Transcribe, Microsoft Azure Cognitive Services for Speech, Google Cloud Speech-to-Text, Whisper, and iOS Speech for ASR, and Amazon Comprehend, Amazon Comprehend Medical, Microsoft Azure Cognitive Service for Language, Google Cloud Natural Language API, and spaCy for NLP, which were tested and compared in chapter 5.

Chapter 4

Software Development Methodology

The present chapter introduces the cloud provider, the cloud services, the work methodology, and the mobile application development design pattern.

First, the cloud provider is explained, and the cloud services are analyzed. Then, the work methodology is introduced. Besides that, the most popular design patterns in mobile application development and a comparative analysis between them are presented, culminating with the option adopted in the project.

4.1 Introduction

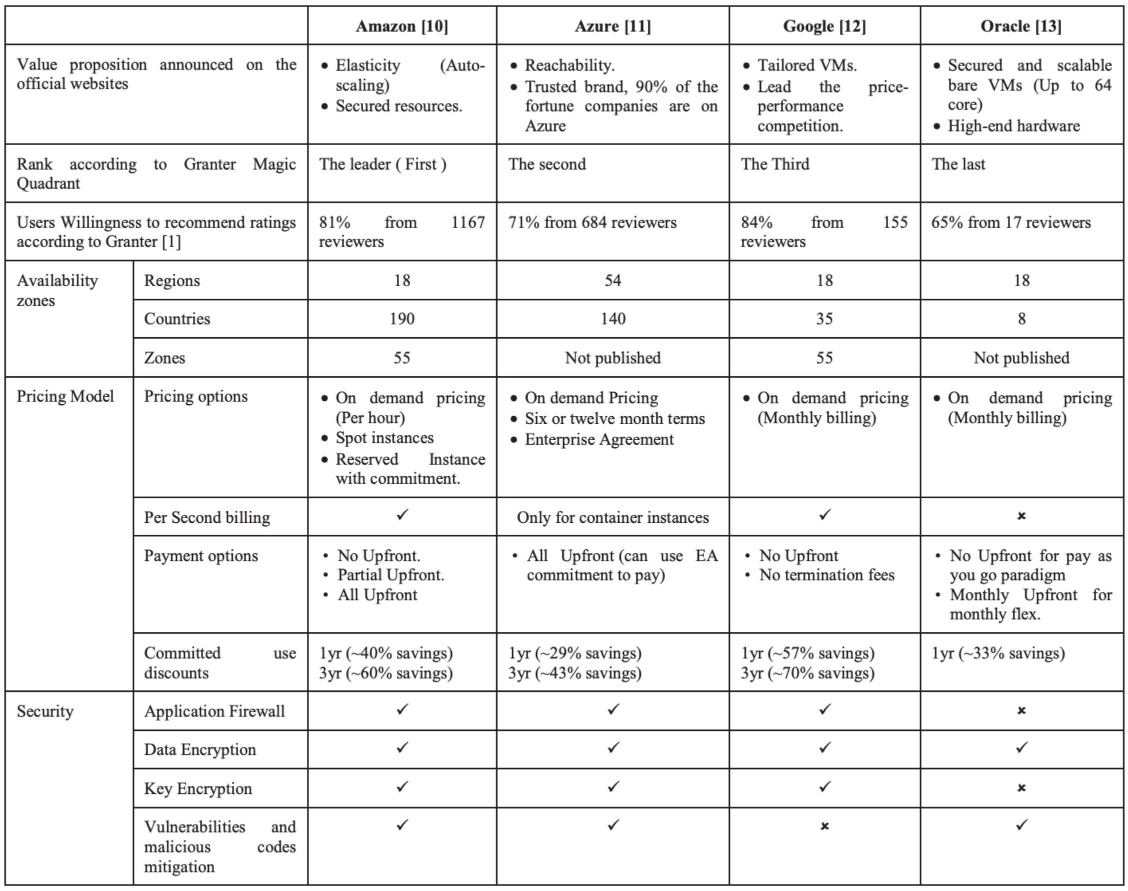
For the development of the system proposed in the current project, it was necessary to have the following:

- A cloud provider, to host the services and data needed
- Cloud services, to deal with the data, hosted services, and user requests
- A work methodology, to organize the work
- A design pattern, for the mobile component development

The following sections illustrate each of these components.

4.2 Cloud provider

For the orchestration of the Natural Language Processing (NLP) service to be used in this project, a cloud provider was needed, and, according to [Aljamal et al., 2019], there are four leading cloud providers in the market: Amazon, Microsoft Azure, Google Cloud, and Oracle. Figure 4.1 corresponds to Table 1 in [Aljamal et al., 2019] and compares the top cloud providers previously mentioned from Infrastructure as a Service perspective.



		Amazon [10]	Azure [11]	Google [12]	Oracle [13]
Value proposition announced on the official websites		• Elasticity (Auto-scaling) • Secured resources.	• Reachability. • Trusted brand, 90% of the fortune companies are on Azure	• Tailored VMs. • Lead the price-performance competition.	• Secured and scalable bare VMs (Up to 64 core) • High-end hardware
Rank according to Granter Magic Quadrant		The leader (First)	The second	The Third	The last
Users Willingness to recommend ratings according to Granter [1]		81% from 1167 reviewers	71% from 684 reviewers	84% from 155 reviewers	65% from 17 reviewers
Availability zones	Regions	18	54	18	18
	Countries	190	140	35	8
	Zones	55	Not published	55	Not published
Pricing Model	Pricing options	• On demand pricing (Per hour) • Spot instances • Reserved Instance with commitment.	• On demand Pricing • Six or twelve month terms • Enterprise Agreement	• On demand pricing (Monthly billing)	• On demand pricing (Monthly billing)
	Per Second billing	✓	Only for container instances	✓	×
	Payment options	• No Upfront. • Partial Upfront. • All Upfront	• All Upfront (can use EA commitment to pay)	• No Upfront • No termination fees	• No Upfront for pay as you go paradigm • Monthly Upfront for monthly flex.
	Committed use discounts	1yr (~40% savings) 3yr (~60% savings)	1yr (~29% savings) 3yr (~43% savings)	1yr (~57% savings) 3yr (~70% savings)	1yr (~33% savings)
Security	Application Firewall	✓	✓	✓	×
	Data Encryption	✓	✓	✓	✓
	Key Encryption	✓	✓	✓	×
	Vulnerabilities and malicious codes mitigation	✓	✓	×	✓

Figure 4.1: Comparison of leader cloud providers from Infrastructure as a Service perspective [Aljamal et al., 2019].

Since this work is in the medical field and considering that a major concern is data security, one of the most relevant issues in choosing the cloud provider corresponds to its security. For this reason, both Google and Oracle should not be considered as hypotheses for this work since they fall short of the other two solutions regarding security. This table also shows that Amazon is the market leader and the most recommended cloud provider, and one of its main focuses is security. Also, it has higher forwarding rates, as pointed out by the authors in Table 2 [Aljamal et al., 2019]. For this reason, Amazon was the cloud provider used in the present work.

Amazon Web Services

Amazon Web Services (AWS) is a cloud platform that provides more than 200 fully featured services from data centers globally to its users. It is described as the cloud provider with more services and features, the largest and most dynamic community, the most secure, the fastest pace of innovation, and the most proven operational expertise [aws, e]. In this project, it was useful to encompass the services needed for the system to work well, being those services addressed below.

4.2.1 Cloud Services

Since the cloud provider used is AWS, the four primary cloud services for the system development corresponded to Amazon API Gateway, AWS Lambda, Amazon DynamoDB, and Amazon Cognito.

AWS Lambda

AWS Lambda is a service provided by Amazon that allows its users to develop and execute code without needing to provision or manage servers.

All computer resource management is the responsibility of this service, with the code executed in a high-availability infrastructure. Computer resource management includes server and operating system maintenance operations, capacity provisioning, and automatic scaling and recording. When using this service, the user is only responsible for its code since the service manages a fleet of computers to offer a balance of several resources, such as memory and Central Processing Unit (CPU). This resource from AWS executes different operational and administrative activities by the user, for example, the capacity management, monitoring, and the log of the users' lambda functions. This service allows the execution of code for several types of applications or backend services, being only necessary to provide the code to be executed in one of the programming languages supported by the service. Also, it supports several different runtimes, including them in terms of programming languages, Node.js, Python, Java, .NET Core, .NET, Go, Ruby, and Custom Runtime [aws, d].

This service does the automatic scheduling, and the functions are executed

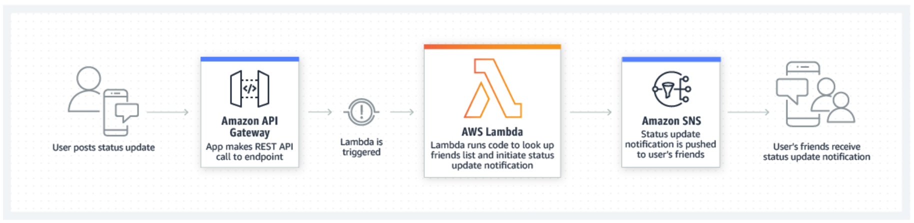

User posts status update
Amazon API Gateway
App makes REST API call to endpoint
Lambda is triggered
AWS Lambda
Lambda runs code to look up friends list and initiate status update notification
Amazon SNS
Status update notification is pushed to user's friends
User's friends receive status update notification

Figure 4.2: Mobile backend usage scenario.

only when it is effectively needed, supporting from a few daily requests to thousands of requests every second. Its use is charged only when there is code in execution, being still a highly available service.

In cases where it becomes necessary to manage the computer resources to be used in the project, AWS has other services that satisfy this need. Those services refer to Amazon Elastic Compute Cloud (Amazon EC2) and AWS Elastic Beanstalk[1] [2].

This service is also suitable for a variety of application scenarios. All that is required is to run the application code using the Lambda standard runtime environment and within the resources provided by Lambda. Such scenarios include file processing, data stream processing, web applications, IoT, and mobile backends. Among these, we should highlight the last use case, which corresponds to the case of interest of this work. Figure 4.2 illustrates an example where Amazon API Gateway, AWS Lambda, and Amazon SNS are used to detect status updates in a given mobile application. This update should derive an action of sending a notification to other users of that same application.

Regarding the features provided by this service, a summarization is presented in Table 4.1 [aws, c].

Feature
Concurrency and scaling controls
Functions defined as container images
Code signing
Lambda extensions
Function blueprints
Database access
File systems access

Table 4.1: Features provided by AWS Lambda.

Amazon API Gateway

This service provided by AWS allows its user to create, publish, maintain, and monitor the security of APIs (REST, HTTP, and WebSocket) at any scale. With

[1]https://aws.amazon.com/pt/ec2/
[2]https://aws.amazon.com/pt/elasticbeanstalk/

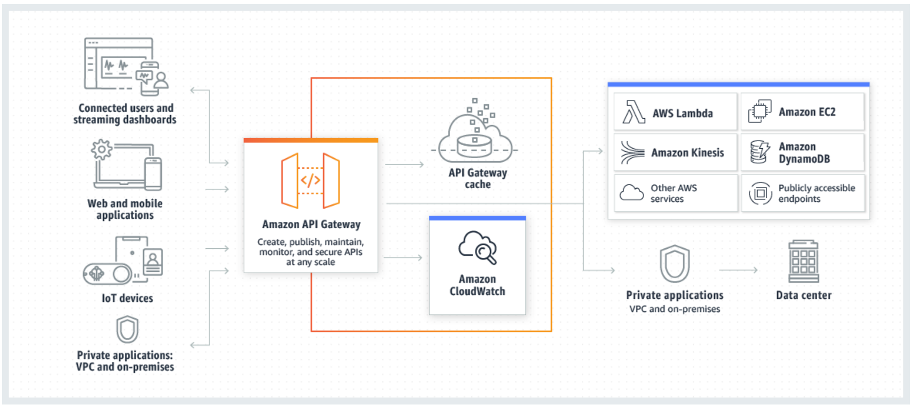

Connected users and streaming dashboards
Web and mobile applications
IoT devices
Private applications: VPC and on-premises
Amazon API Gateway
Create, publish, maintain, monitor, and secure APIs at any scale
API Gateway cache
Amazon CloudWatch
AWS Lambda
Amazon EC2
Amazon Kinesis
Amazon DynamoDB
Other AWS services
Publicly accessible endpoints
Private applications
VPC and on-premises
Data center

Figure 4.3: Amazon API Gateway architecture.

this service, APIs can be created to access AWS services or other web services and data stored in the AWS Cloud. The user can also create APIs both for use in their client applications and to make their APIs available to other application developers.

This service creates RESTful APIs based on HTTP, which allow stateless communication between client and server and implement standard HTTP methods, such as the GET and POST methods. On the other hand, it creates WebSocket APIs that enable stateful, full-duplex communication between client and user – since they adhere to the WebSocket protocol – and forward incoming messages considering their content.

In Figure 4.3, the architecture of this service is illustrated. This diagram presents a serverless application development approach, being the API Gateway responsible for handling the tasks related to the acceptance and processing of simultaneous API calls, up to hundreds of thousands of simultaneous calls being supported. These tasks include traffic management, authorization and access control, monitoring, and API version management. This service is between the application and the content it wants to access. It refers to data, business logic, and backend service functionalities, such as the code executed in Lambda AWS. It acts like a front door for the applications' access to resources.

Besides that, the features provided by Amazon API Gateway are enumerated in Table 4.2. Between the flexible authentication mechanisms mentioned in the table, there are Amazon Cognito user pools, which should be presented next.

Feature
Stateful (WebSocket) and stateless (HTTP and REST) APIs support
Flexible authentication mechanisms
Developer portal for APIs publishing
Canary release deployments for safely rolling out changes
Logging and monitoring of API usage and API changes with CloudTrail
Access logging and execution logging, including the ability to set alarms with CloudWatch
Ability to use AWS CloudFormation templates to enable API creation
Custom domain names support

Integration with AWS WAF – protection against common web exploits
Integration with AWS X-Ray – performance latencies understanding and triaging

Table 4.2: Features provided by Amazon API Gateway.

Amazon API Gateway service can be accessed in several different ways. Throw AWS Management Console, which provides a web interface for the creation and management of APIs, throw AWS SDKs, API Gateway V1 and V2 APIs, AWS Command Line Interface and AWS Tools for Windows Powershell.

This service is part of the AWS serverless infrastructure, including Amazon API Gateway and AWS Lambda. The latter service can enable communication between an application and publicly available AWS services through the exposure of the Lambda functions using API methods in the Gateway API. To enable serverless applications, this service allows simplified proxy integration with Lambda and HTTP endpoints [aws, a].

Several usage scenarios of the Amazon API Gateway, which enables the generation of REST APIs, HTTP APIs, and even WebSocket APIS, are highlighted by AWS [3]. As far as WebSocket APIs are concerned, these maintain persistent connections between the client and the server. In the usage scenario of the present work, there is no need for these types of connections, which is why this type of API was not chosen. Compared to HTTP APIs, REST APIs encompass more features and are more expensive. Since the additional features of REST APIs are not relevant in the context of this paper, this type of API was not chosen either. With this, the generated API was then inserted into the HTTP APIs [4].

HTTP APIs can send requests to AWS Lambda functions or any publicly routable HTTP endpoint. For example, an HTTP API can be created to integrate with a Lambda function in the backend. When the client calls the API, the API Gateway then sends the client's request to the Lambda function. This returns a response that is then forwarded by the API Gateway back to the client [5].

The HTTP APIs suport OpenID Connect e OAuth 2.0 authorization. OpenID Connect corresponds to a simple identity layer on top of the OAuth 2.0 protocol that allows the verification of an end-users entity based on authentication by an Authorization Server, as well as obtaining basic profile information regarding the end-user in an interoperable and REST-like manner. This can be used by clients of all types – including mobile and web-based clients – and allows them to request and receive information about authenticated sessions and end-users. Optional features, such as encryption of identity data, are also permitted when it makes sense for the client to use them [6]. OAuth 2.0, on the other hand, considers the industry-standard protocol for authorization. It focuses on simplicity as far as

[3]https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-overview-developer-experience.html

[4]https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html

[5]https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-overview-developer-experience.html

[6]https://openid.net/connect/

the client developer is regarded, yet provides specific authorization flows for web and desktop applications, mobile phones, and living room devices [oau, 2020].

These APIs also support cross-origin resource sharing (CORS) and automatic deployments. CORS corresponds to a browser security feature restricting HTTP requests initiated from scripts running in the browser. Using this feature allows requests to be made from a web application hosted in a different domain than the developed API [7]. CORS refers to an HTTP-header-based mechanism that allows a server to specify sources other than its own from which the browser should allow loading resources, and these sources may relate to domain, schema, or port. This mechanism also presents a strategy by which browsers make a "preflight" request to the server that hosts the cross-origin resource. The purpose of this is to verify if the server will allow the request in question, and in this preflight request, the browser sends headers that indicate both the HTTP method and the headers that will be used in the actual request [Mozilla, 2023].

Regarding the Application Programming Interface (API) testing made, it was based on Postman, an API platform for building and using APIs. It simplifies each step of the API lifecycle and the collaborations to create better APIs faster. This platform offers several layers that relate to [8]:

- API repository allows storage, cataloging, and collaboration around all user APIs on a single central platform. Postman can store and manage all aspects of APIs, such as specifications, documentation, workflow recipes, test cases, results, and metrics.
- Tools, responsible for providing a comprehensive set of tools that help the user accelerate the API lifecycle. It encompasses the design, testing, documentation, and mocking and the sharing and discovery of the user's APIs.
- Governance, related to Postman's complete lifecycle approach to governance. This approach allows the development of better quality APIs by enabling the change of development practices by its users. It is also related to the promotion of collaboration between development teams and API design teams.
- Workspaces, responsible for helping in the organization of API work and collaboration both in the user organization and the world. Four types of workspaces exist: personal, team, partner, and public.
- Integrations, responsible for integrating Postman with the most essential tools in the user software development pipeline, enabling API-first practices. Postman is further extensible through the Postman API or open-source technologies.

Since this study aims to test the developed API, the most relevant component in this context corresponds to Tools. In this component the following tools are included [9]:

[7]https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-cors.html
[8]https://www.postman.com/product/what-is-postman/
[9]https://www.postman.com/product/tools/

- API client, this platform's fundamental tool, allows users to explore, debug and test its APIs. This enables complex API requests to be defined for different APIs - HTTP, REST, SOAP, GraphQL, and WebSockets.

  This component automatically detects response and link language, formats the text within the body for easy inspection, and supports authentication protocols such as OAuth 2.0 and AWS Signature.

  Through the API client, the user can organize requests into Postman Collections that help manage requests for reuse. The user collections can also contain JavaScript code to link requests or automate common workflows, and scripting can be used to visualize the user API responses as charts or graphs.

- API design, which allows users to design their API specifications using OpenAPI, RAML, GraphQL or SOAP, these being tools used to describe APIs [ope, 2022] [10] [Foundation, 2012] [11].

  Besides Postman's schema editor making it easy to work with specification files of any size and validate user's specifications with a built-in linting engine, the user can also generate Postman Collections for various stages of their API lifecycle – for example, for mocks and tests – based on the specification file, all in sync.

- API documentation, which provides automatic documentation features. Postman supports markdown-enabled and machine-readable documentation through the Postman Collection format and allows documentation generation through user OpenAPI files. These documents will automatically contain details of user requests with code examples in various client languages. Users can share the documentation with their team or the world using workspaces or publish it to a dedicated portal.

- API testing allows the user to build and run tests directly in Postman or as part of the user's CI/CD pipeline. In the second case, Newman is used as a Collection Runner that allows the user to run and test a Postman Collection directly on the command line. Postman can be used to write various tests, including functional, integration, and regression. In addition, Postman's Node.js-based runtime supports common patterns and libraries that the user can use for rapid test building.

- Mock servers allow the user to visualize precisely how its API will work before it is even in production. This component allows mock servers to be created in Postman to simulate endpoints of the API under development, and it is also possible to simulate network latency on the mock server by specifying custom delays for responses.

  Because they are hosted in the Postman Cloud, these servers are available wherever needed, from local to test or staging environments.

[10] https://raml.org/
[11] https://www.w3schools.com/xml/xml$_soap.asp$

- Postman also offers a wide range of API monitors that help users keep track of the health and performance of its APIs, which can run in different geographic regions and be integrated with alert systems and dashboards provided by third parties, such as Slack.

  Like the previous tool, the monitors are also hosted in the Postman Cloud to be installed quickly.

- API detection, responsible for capturing requests and cookies from the user's browser to Postman to speed up its debugging flow through the Postman Interceptor – which allows the capture of requests and responses – or the Postam proxy - which, besides running inside the Postman application, can be used with HTTP and HTTPS sites.

Amazon Cognito

Amazon Cognito is a service provided by AWS that provides customer authentication, authorization, and user management for their web and mobile applications. This service allows the customer to provide its user with a login done directly – with a username and password – or indirectly – using a third party such as Facebook, Amazon, Google, or even Apple. This service is composed of two main components corresponding to its features [12]:

- User pools, which refer to user directories in Amazon Cognito. These provide login and user registration options for the customers' web and mobile applications. All members of a user pool have a directory profile accessible from an SDK. User pools provide
  - User login and registration services
  - A web interface that is both integrated and customizable and is intended for user login
  - Social login using Facebook, Google, Amazon, and Apple, and through identity providers Security Assertion Markup Language (SAML) and OpenID Connect (OIDC) from the user pool.
  - User directory and user profile management
  - Security features include multi-factor authentication (MFA), verification of compromised credentials, account takeover protection, and phone and email verification.
  - Custom workflows and user migration using AWS Lambda trigger.
- Identity pools, which in turn allow the customer to grant access to other AWS services to their users, as they can be provided with temporary AWS credentials to access them. In addition to anonymous guest users being supported for user authentication for identity pools, identity providers, including:

[12]https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

- Amazon Cognito user pools
- Social login using Facebook, Google, Amazon login, and Apple login
- OIDC Providers
- SAML identity providers
- Developer authenticated identities

SAML refers to an open standard that allows identity providers to pass authorization credentials to service providers. Put another way, it is possible to use one set of credentials to log in to many different websites. It uses Extensible Markup Language (XML)-based transactions for standardized communications between identity providers and service providers, falling somewhere between authenticating a user's identity and authorizing that user to use a service [Buckbee, 2022].

Amazon Cognito components can be used either in isolation from each other or together. However, It should be noted that to store user profile information, the identity pool has to be integrated with a user pool. This service is also:

- Compatible with:
  - SOC 1-3
    System and Organization Controls (SOC) refer to a set of service offerings that Certified Professional Accountants (CPAs) can provide in connection with the system-level controls of a service organization or the entity-level controls of other organizations [13].
  - PCI DSS
    The Payment Card Industry Data Security Standard is a global security standard designed to improve control over users' payment card data, with operational and technical requirements. This standard and its respective standards are intended to ensure that there is a safe environment for entities and users who process data from payment cards, avoiding fraud [14].
  - ISO 27001
    ISO 27001 corresponds to the norm that is the International standard and reference in information security management. Its general principle corresponds to the adoption of a set of requirements, processes, and controls by the organization, aiming at the mitigation and adequate management of its risk [15].
- HIPAA-BAA eligible
  AWS, like other cloud service providers, is considered a business associate under HIPAA regulations. The Business Associate Addendum (BAA) refers to an AWS contract that is required under HIPAA rules to ensure that AWS adequately safeguards Protected Health Information (PHI). In addition, this

[13]https://www.aicpa-cima.com/resources/landing/system-and-organization-controls-soc-suite-of-services

[14]https://www.integrity.pt/pt/pci-dss-compliance.html

[15]https://www.27001.pt/

contract also aims to clarify and appropriately limit the permissible uses and disclosures of PHI by AWS, taking into account the relationship between AWS and its customers as well as the activities or services being performed by the cloud provider [16].

Amazon DynamoDB

Amazon DynamoDB, on the other hand, is a fully managed NoSQL database service that offers fast, predictable performance and seamless scalability. With this service, the user does not have to deal with distributed database administration tasks. In addition, encryption is offered at rest, eliminating a further operational burden and the complexity associated with protecting sensitive information [aws, b].

With DynamoDB, the user can organize its data in database tables, which can organize and recover any data and serve any request traffic level. The production capacity of the tables can be increased or decreased without a stop time or associated performance degradation. DynamoDB also offers an on-demand backup capability and allows the creation of complete security copies of users' tables and archiving. To protect the users' tables from accidental writing or elimination, this service also provides point-in-time recovery, restoring a table to any point in time in the last 35 days. To help the user reduce storage usage and storage cost, DynamoDB also allows the automatic deletion of expired items from the table, avoiding costs associated with irrelevant data. Also, DynamoDB has both high availability and durability.

With DynamoDB, data and table traffic are automatically spread over multiple servers to handle the user's production and storage requirements while maintaining consistent and fast performance. The data is stored in Solid State Disk (SSD)s and automatically replicated over multiple available zones in AWS Region. Whit that, a high availability and built-in data durability is provided to the users [17] [18].

4.3 Scrum methodology

The current project's development was based on an agile methodology: Scrum. This refers to a framework that helps teams work together, encouraging experience-based learning, self-organization at work, and reflection aimed at continuous improvement. This methodology was chosen, particularly since the company adopts this.

Software development teams commonly use Scrum, but its principles and lessons can be applied to every kind of team. It is commonly thought of as an agile project management framework, and it describes a set of meetings, tools,

[16]https://aws.amazon.com/compliance/hipaa-compliance/
[17]https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html
[18]https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

and roles that, by working together, help teams to structure and manage their work.

Starting with Scrum teams, there are three key roles: product owner, scrum master, and development team.

- The product owner role focuses on understanding the business, the customers, and what the market demands. Based on this, they focus on the engineering team's work to ensure that the greatest possible value is delivered to the business.
  This role, which a single individual should perform, is associated with the tasks of building and managing the product backlog, establishing a close partnership with the business and the team to ensure that everyone understands the various tasks in the backlog, providing clear guidance by highlighting which features are to be delivered next, and finally deciding when the product should be delivered with a predisposition for more frequent delivery.
- Scrum masters, on the other hand, are responsible for coaching both teams and product owners and the business while also looking for ways to improve their scrum practice.
  A scrum master must be able to deeply understand the work being done by the team and be able to help the team optimize its transparency and delivery flow. This role is also tasked with scheduling the resources needed for each sprint planning, stand-up, sprint review, and sprint retrospective (human and logistical).
- Scrum teams are responsible for developing the product or service intended to be delivered. These teams are typically five to seven members, having different skills and cross-training each other so the work flows. This team defines the plan for each sprint, predicting how much work they believe could be done in the sprint, considering the historical data from the previous sprints. These teams are typically small, and within the development team role are testers, designers, User Experience (UX) specialists, operations engineers, and programmers.

In this methodology, there are also artifacts. Those correspond to relevant information to help define the product to be developed and the work to create it. There are three different artifacts related to the three constants on which there should be a reflection by the team both during sprints and over time: product backlog, sprint backlog, and increment.

- In this methodology, the product or service intended to be developed is produced iteratively throw sprints. Those correspond to short periods in which the team works to make a defined amount of work. Being the key of this methodology, a well-organized sprint is the key to better products with fewer difficulties in the development process. With these sprints, the Agile principle of delivering working software is satisfied. Also, the value of giving a response to changes instead of only following a plan is respected. The concepts of transparency, inspection, and adaptation are associated with this framework, particularly with this sprint concept [Rehkopf, 2019]. In each sprint, a valuable increment should be delivered.

- The product backlog corresponds to a prioritized list of the work the development team should do, derived from the roadmap and its requirements. The items are organized in the list from the most important to the least, being the items pulled from this backlog to the sprint backlog until the maximum capacity of the team in each sprint is achieved. By its turn, the sprint backlog deals with the work assigned to the sprint [Radigan, 2019].
- Finally, the increment defines the usable end-product achieved from a sprint.

This methodology also includes several key ceremonies such as the backlog organization, the sprint planning, the sprint itself, the daily scrum, the sprint review, and the sprint retrospective [Drumond]:

- The backlog organization corresponds to a product owner's task. As its main function is driving the product towards its product vision, continuously dealing with the market and the customer, the product owner should maintain the product backlog according to the users' feedback and the development team. With that, it should be ensured that the list is always prioritized, clean, and ready for the development team to work on it at any time.
- Sprint planning, by its turn, corresponds to a meeting involving all the development team in which the work that must be performed in the current sprint, i.e., its scope, should be defined. The scrum master is responsible for leading the meeting, and the development team decides the goal of the sprint. With that in mind, the sprint backlog is filled with specific user stories selected from the product backlog. Besides the agreement of those stories with the goal of the sprint, they are agreed by the scrum team as being doable in the current sprint. By the end of the meeting, the scrum members should know clearly what can be delivered and how.
- Similarly, the daily scrum is also a meeting. This one, however, corresponds to a really short daily meeting intended to guarantee that every team member is in tune with the sprint goals and to define the plan until the next meeting. In these meetings, the members should present any concern or blocker to achieving the sprint goal. Finished each sprint, the teams meet informally to view a demo or to inspect the increment delivered. The items from the backlog that were done are presented both to the rest of the team and to stakeholders, aiming at receiving feedback. The product owner decides the release of the increment. This meeting is also used to rework the product backlog based on the current sprint, being that done by the product owner. This rework can feed into the next sprint planning meeting.
- Finally, the sprint retrospective consists of the meeting where the team documents and discusses what worked and what did not in the last sprint, project, people, relationships, tools, and certain ceremonies. With that, the

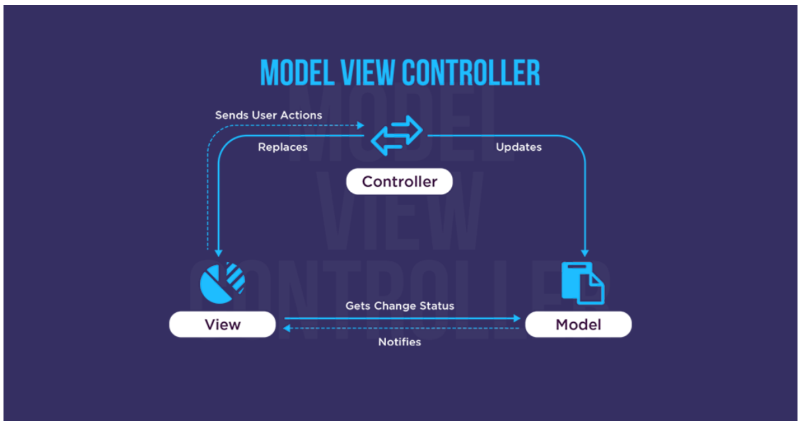


Figure 4.4: MVC components organization.

team should be able to focus on what went well and what needs improvement for the next time.

This methodology was used for the API and mobile application development in the current project context.

4.4 User Interface

Regarding the architecture adopted for mobile application development, the three most popular design patterns were analyzed and compared, being them Model-View-Controller (MVC), Model-View-ViewModel (MVVM), and Model-View-Presenter (MVP). Each has its components, advantages, disadvantages, and particular use cases that are more suitable. All three models have two similar components, the Model and the View. Although, all of them vary in the third component, which distinguishes them. The first component, the Model, is responsible for storing and managing the application data in all three design patterns. In turn, the View component displays the information received from the Model in all design patterns. The third component varies between the design patterns and has different roles.

Starting with the MVC, the most used and traditional design pattern, the three components are organized as presented in Figure 4.4. The third component in this design pattern corresponds to the Controller, the component responsible for the interactions between the other two (the communication between them), coordinating it. This design pattern is beneficial when the application under development requires storage and access of the data in several locations, also requiring the code to be highly reusable.

In turn, MVVM corresponds to a variation of the previous design pattern, and its three components are organized as illustrated in Figure 4.5. For this design pattern, the third component corresponds to the ViewModel, which is responsible for both the communication between the other components, for user interactions' handling, and the update of the View component. This is a helpful design

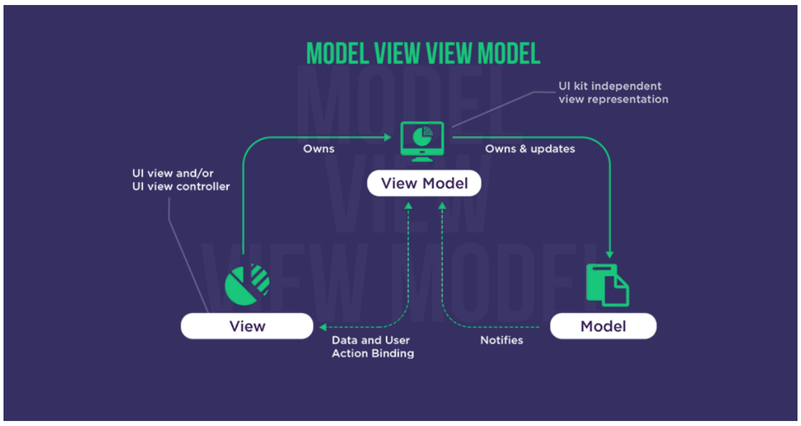

MODEL VIEW VIEW MODEL
UI kit independent view representation
Owns
Owns & updates
UI view and/or UI view controller
View Model
View
Model
Data and User Action Binding
Notifies

Figure 4.5: MVVM components organization.

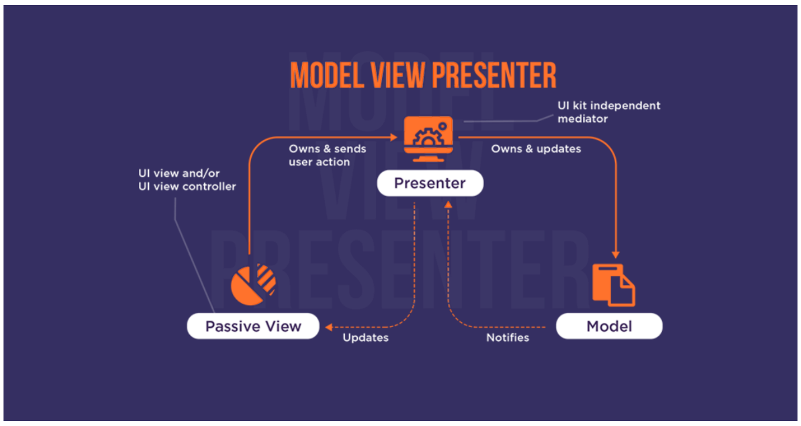


Figure 4.6: MVP components organization.

pattern, namely when the application under development requires a significant degree of user interaction and frequent updates of the View component based on the user's input. Also, it is exceptional when the application requires high data manipulation and, also, high communication between its components (namely, the Model and the View).

Finally, the MVP is the newest of the three design patterns, and its three components are organized as presented in Figure 4.6. For this design pattern, the third component corresponds to the Presenter responsible for both user interactions' handling, View's update, the application's logic, and the way the other two components interact between them. Like the previously presented design pattern, this one is useful, namely when the application under development requires a significant degree of user interaction and, also, frequent updates of the View component based on the user's input, being also a suitable architecture for the cases in which the application requires many interactions at the level of the Model and View interactions.

All these design patterns are different, and the decision of the one that is more suitable for each application development process should be based both on the requirements of the application and the user interactions, namely, the type the application has to support. In Table 4.3, each model is associated with its kind of approach, implementation, the type of application, and user interaction it is more suitable for.

Model	Approach	Implementation	Application	Interaction
MVC	Most basic	Simpler	Smaller and simpler	Few
MVP	More complex	More flexible	Larger and more complex	More complex
MVVM	Most modern	More flexible	Larger and more complex	Complex

Table 4.3: Comparison between MVC, MVP, and MVVM.

Analyzing the previous table, it can be seen that MVC, despite its more straightforward implementation, is unsuitable for the current project since it is more adequate for applications with fewer user interactions.

Comparing the MVP and MVVM design patterns in terms of each one's advantages and disadvantages, tables 4.4 and 4.5 are obtained.

MVP	MVVM
Easy debugging	Developers can work and debug even the most fundamental lines of code
Code reusability	Straightforward unit tests' writing
Concerns clearly separated	Faster development The UI can be modified with no code modifications

Table 4.4: Advantages of MVP and MVVM.

MVP	MVVM
Too complex for simple, straightforward arrangements	May require expensive memory resources
May have assembly issues	View models and their state may be difficult to operate (when nested views and complex interfaces are used)

Table 4.5: Disadvantages of MVP and MVVM.

MVP has the disadvantage of not being able to use it when the arrangement is simple, which should be the case for the current project, so it should not be suitable. With that, MVVM was the design pattern for the current project [Shah, 2020].

4.5 Conclusion

In summary, AWS was the cloud provider for the developed system, and the cloud services referring to Amazon Cognito, AWS Lambda, Amazon API Gateway, and Amazon DynamoDB were used. Furthermore, the Scrum work methodology was adopted, and the development of the mobile component of the system followed the MVVM design pattern.

Chapter 5

Competitors analysis

The present chapter describes the tests performed to decide the Automatic Speech Recognition (ASR) and Natural Language Processing (NLP) solutions used in this project and their results.

First, a context is given about the tests made. Then the approach followed to test, and the results obtained for each ASR solution are presented. Equally, the approaches followed to test, and the results obtained with the NLP solutions are presented. Finally, a brief conclusion is provided.

5.1 Introduction

From the findings in the previous chapter, five ASR solutions were selected for testing. These solutions are Amazon Transcribe, Microsoft Azure Cognitive Services for Speech, Google Cloud Speech-to-Text, Whisper, and iOS Speech. Their test chose the most suitable solution for this project, corresponding to the solution integrated into the system.

Also, from the findings in the previous chapter, seven NLP solutions were selected for testing. These include Amazon Comprehend, Amazon Comprehend Medical, Google Cloud Natural Language API, Google Cloud Healthcare Natural Language API, Microsoft Azure Cognitive Service for Language, Microsoft Azure Text Analytics for Health, and spaCy. In their first test, these solutions were tested for their ability to recognize International Non-proprietary Name (INN) entities in isolation. This test was divided into testing the default models and testing custom models corresponding to adaptations of the default ones. The custom models were generated and trained in the context of this project.

A second test was performed with the solutions that proved to be the most suitable for this project in the previous test. These solutions include Amazon Comprehend, Google Cloud Healthcare Natural Language API, Microsoft Azure Cognitive Service for Language, and Microsoft Azure Text Analytics for Health. Those were tested for entity recognition in single-INN prescriptions, now considering not only INN entities but also the dosage, route of administration, fre-

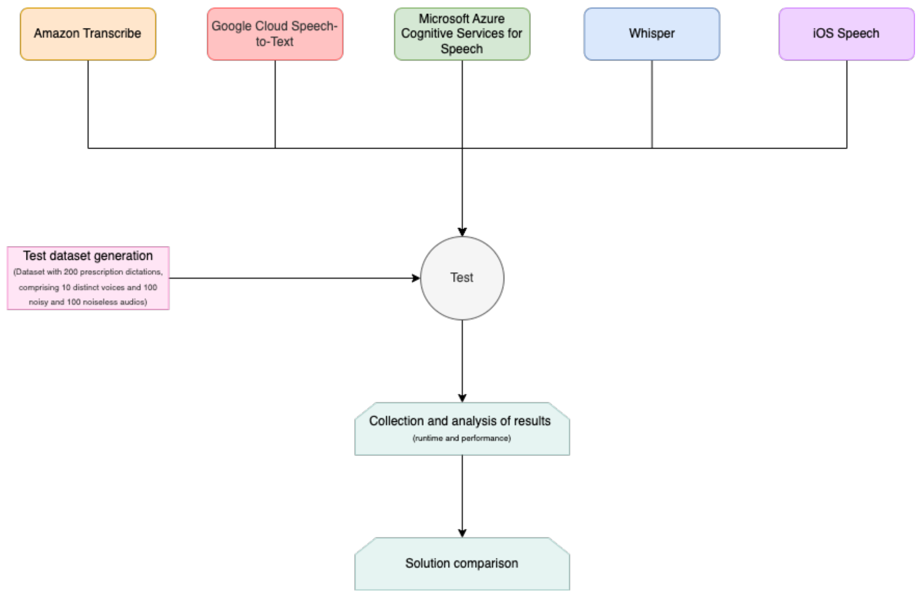

Amazon Transcribe
Google Cloud Speech-to-Text
Microsoft Azure Cognitive Services for Speech
Whisper
iOS Speech
Test dataset generation
(Dataset with 200 prescription dictations, comprising 10 distinct voices and 100 noisy and 100 noiseless audios)
Test
Collection and analysis of results
(runtime and performance)
Solution comparison

Figure 5.1: Flow chart for the test made for ASR solutions evaluation.

quency, start, and end, or duration of treatment.

A final test was performed with the most suitable solution for this project. Microsoft Azure Text Analytics for Health was tested for identifying the various entity types in the context of prescriptions with multiple INN.

The following sections present the results obtained in each test and their analysis.

5.2 Comparison of Automatic Speech Recognition solutions

The study conducted in the previous chapter allowed the identification of five ASR solutions that were best suited for this project and should therefore be compared. These solutions include Amazon Transcribe, Microsoft Azure Cognitive Services for Speech, Google Cloud Speech-to-Text, Whisper, and iOS Speech. Only the default models of these solutions were tested, although some allow model customization. Still, the lack of data and the impossibility of creating a considerable amount of it in the context of the project make it impossible to train them with quality. The flowchart shown in Figure 5.1 was followed to test these solutions.

For this test, ten single-medication prescriptions were randomly chosen for recording. These were checked to verify that each entity type covers different values among the various prescriptions. With this, a dataset of recordings was generated in which there are 18 different voices. Nine are human voices, and the remaining nine are computer-generated, generated using the site [Limited]. A hospital environment noise [war] was added to the previous recordings, giving rise to a new set of audios. Thus, the final dataset had 360 recordings, 180 without noise and 180 with. Four of the nine human voices are female, and five are male, with a northern accent evident in two male voices. Six of the nine computer-generated voices are female, and the remaining three are male. With this, each solution was tested individually.

5.2.1 Amazon Transcribe

Amazon Transcribe is available for European Portuguese only for batch transcription, thus requiring the prior loading of audio files into an S3 bucket. Therefore, the present test required first loading the various audios into an S3 bucket, and then a Python file was developed and executed to get results from the system.

5.2.2 Microsoft Azure Cognitive Services for Speech

This solution was tested by developing and running a Python script.

5.2.3 Google Cloud Speech-to-Text

Likewise, the current service was tested by developing and running a Python script.

5.2.4 Whisper

Similarly to the previous service, Whisper was tested by developing and running a Python script. This solution encompasses the tiny, base, small, medium, and large multi-language models in order of complexity, and all of them were tested.

5.2.5 iOS Speech

In contrast to the previous solutions, this solution was tested through the development and execution of a Swift project.

5.2.6 Results and conclusion

Table 5.1 shows the results obtained with the various solutions tested.

Solution	Performance	Execution meantime
Google Cloud Speech-to-Text	69.70%	2.61 seconds
Microsoft Azure Cognitive Services for Speech	88.02%	3.44 seconds

Whisper	Tiny model	38.94%	4.47 seconds
	Base model	51.22%	5.44 seconds
	Small model	57.95%	12.49 seconds
	Medium model	69.97%	34.79 seconds
	Large model	77.53%	72.26 seconds
iOS Speech		42.46%	3.96 seconds
Amazon Transcribe		65.07%	22.85 seconds

Table 5.1: Results obtained for ASR solutions comparison.

The calculation of the solutions' performance followed the following logic:

1. A pre-processing was made to standardize the transcripts – for example, to convert milligrams to mg so all the transcripts conform to the gold standard.

2. For each transcribed prescription, a value was calculated. This value corresponds to a weighted sum of the words and plus signals in the prescription. The weight associated with those words and plus signals was equaled to one except for those which were part of a INN entity, for which the weight applied corresponded to ten.

3. The value associated with each transcription was divided by the value associated with the corresponding gold standard, calculated with the same logic.

4. For each solution, the values produced in the previous step were summed and divided by 360, producing the performance values presented.

The average execution time, in turn, corresponds to the average execution times obtained by each solution for the various recordings.

Although Google and Microsoft are widely indicated as the market leaders in ASR, in the present context, there is an apparent discrepancy between their solutions. Analyzing the results, Google's solution is likely to be more sensitive to both the speaker's accent and the audio quality, leading it to be outperformed by Microsoft's solution, which proved to be more flexible in these cases. As an example, for a male voice with an accent and for noiseless recording of the prescription "Tomar Telmisartan 20 mg, vaginal, ao almoço, a começar amanhã e durante 7 dias.", Google's solution identified "20 miligramas vaginal almoço a começar amanhã e durante 7 dias" while Microsoft's solution identified "Tomar telmisartan 20 miligramas vaginal ao almoço a começar amanhã e durante 7 dias.". In turn, for a female voice where the audio has low quality and for the prescription "Fazer Amoxicilina + Ácido clavulânico 500 mg + 125 mg, intravenosa, às refeições, a começar hoje e até dia 12-01-2023." Google's solution could not identify any words while Microsoft's solution identified "Fazer amoxicilina mais ácido clavulânico, 500 miligramas, +125 miligramas intravenosa às refeições a começar Hoje e até dia 12/01/2023."

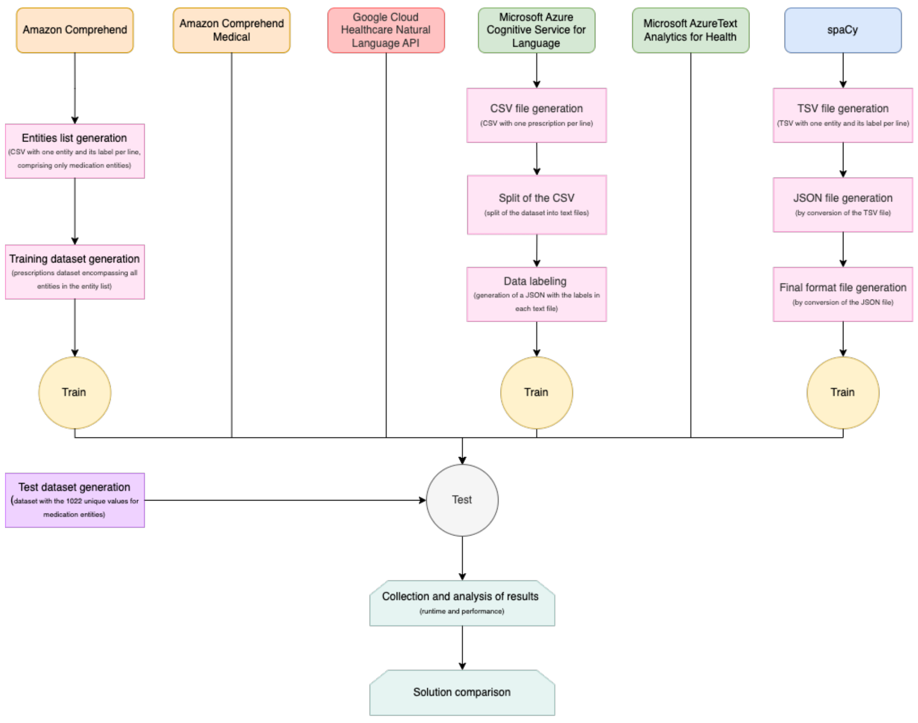

Amazon Comprehend
Amazon Comprehend Medical
Google Cloud Healthcare Natural Language API
Microsoft Azure Cognitive Service for Language
Microsoft AzureText Analytics for Health
spaCy
Entities list generation
(CSV with one entity and its label per line, comprising only medication entities)
Training dataset generation
(prescriptions dataset encompassing all entities in the entity list)
CSV file generation
(CSV with one prescription per line)
Split of the CSV
(split of the dataset into text files)
Data labeling
(generation of a JSON with the labels in each text file)
TSV file generation
(TSV with one entity and its label per line)
JSON file generation
(by conversion of the TSV file)
Final format file generation
(by conversion of the JSON file)
Train
Train
Train
Test dataset generation
(dataset with the 1022 unique values for medication entities)
Test
Collection and analysis of results
(runtime and performance)
Solution comparison

Figure 5.2: Flow chart for the tests made for INN entities detection evaluation.

With the results shown, it can be understood that the most suitable ASR solution in the context of the present project corresponds to Microsoft Azure Cognitive Services for Speech. With that, this solution was integrated into the final application.

5.3 International Non-proprietary Name entities detection

The INN entities are the core of medical prescriptions, and the other entities should be identified and related to them. Therefore, these entities should be the most relevant in this project, and the systems' performance in identifying them should have a word in the solution choice. For that, the performance of each system was evaluated based on its ability to correctly identify the active substances, i.e., its capability of identifying each INN in the same way they appear in the Infarmed database and to assign the correct entity type to them.

The conclusions drawn from the previous chapter were that the solutions corresponding to Amazon Comprehend, Google Cloud Natural Language API, Google Cloud Healthcare Natural Language API, Microsoft Azure Cognitive Service for Language, Microsoft Azure Text Analytics for Health, and spaCy should be tested and compared to define the most appropriate one for the present work. To this end, this first test was performed following the procedure illustrated in Figure 5.2.

Amazon Comprehend, Microsoft Azure Cognitive Service for Language, and spaCy needed a training phase to recognize INN entities. Those solutions had two additional phases compared to those that were not training-dependent. Those phases refer to generating the training data and the training phase. In the testing phase, the unique values for active substances of Infarmed's database [1] were used. A Comma-Separated Value (CSV) file with one INN entity per line was passed to each solution in this phase, and the results are presented in the following sections. Also, the train datasets do not overlay the test dataset.

Since the test dataset only encompasses the INN entities, the system's performance corresponds to its accuracy. Besides the model's evaluation through its accuracy, the macro F1 score was also used to consider class imbalance. Finally, the Translator function from googletrans [Han, 2020] was used when a translation was necessary.

5.3.1 Default models

The first phase of the test aimed to study the default models provided by each solution under discussion. The test proceeding and the results are presented in the following sections.

Amazon Comprehend

To test Amazon Comprehend, its synchronous component – the real-time analysis – was used, and the API was invoked through a Python script [2]. For that, the following process was done:

1. Installation of the AWS CLI through the command line command msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
2. Verification of the installation with the command aws –version [3]
3. Update of the environment variables based on the command provided in the Amazon Web Services (AWS) private account for command line or programmatic access
4. Update the credentials file with the values provided by the AWS private account. For that, the command aws configure was executed, and the following information was specified:
   - AWS Access Key ID
   - AWS Secret Access Key
   - Default region name
   - Default output format

[1]https://www.infarmed.pt/web/infarmed/servicos-on-line/pesquisa-do-medicamento
[2]https://docs.aws.amazon.com/comprehend/latest/dg/using-api-sync.html
[3]https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

5. Installation of the AWS SDK for Python with the execution of the command pip install boto3

This process enables the usage of Amazon Comprehend and Amazon Comprehend Medical, being a distinct function called to invoke each solution.

Although Amazon Comprehend allows its user to recognize entities in the input text, the default model has defined types in which it classifies them. Those types correspond to COMMERCIAL_ITEM, DATE, EVENT, LOCATION, ORGANIZATION, PERSON, QUANTITY, TITLE, and OTHER [4]. It can be seen that none of these types contemplates medication or similar term. For that reason, the default model will never be able to identify the INN entities as such, which negates its interest in this project. Because it cannot detect the INN entities as such, its performance should be equal to 0%.

Amazon Comprehend Medical

Amazon Comprehend Medical has, opposite to the last service, an entity type MEDICATION [5], but the service only supports English. Because the initial dataset is in European Portuguese, it was necessary to translate it before passing it as input to the system. A script was then developed to translate the data, pass it to the system, and evaluate its performance, being the results obtained presented in Table 5.2.

Execution time
280.86 seconds
Performance
29.21%
Macro F1
45.22%

Table 5.2: Results obtained for INN entity recognition with Amazon Comprehend Medical.

Figure 5.3 presents the confusion matrix obtained for this solution. These results show that Amazon Comprehend Medical fails to detect most of the INN entities, reducing its interest in this context.

Google Cloud Natural Language API

To use both Google Cloud Natural Language API and Google Cloud Healthcare Natural Language API, a single command needed to be executed. This command refers to installing the Python package named google through the command pip install google.

[4]https://docs.aws.amazon.com/comprehend/latest/dg/how-entities.html

[5]https://docs.aws.amazon.com/comprehend-medical/latest/dev/textanalysis-entities.html

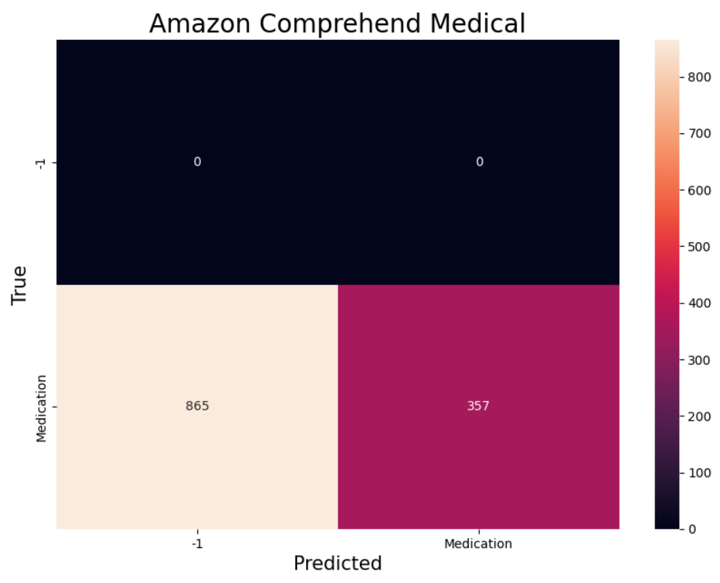


Figure 5.3: Confusion matrix obtained for INN entity recognition with Amazon Comprehend Medical.

Focusing on Google Cloud Natural Language API, it can recognize various entity types. Those types include UNKNOWN, PERSON, LOCATION, ORGANIZATION, EVENT, WORK_OF_ART, CONSUMER_GOOD, OTHER, PHONE_NUMBER, ADDRESS, DATE, NUMBER, and PRICE [ent, 2023]. It can then be understood that none relate to medication or similar. For this reason, this default model should have 0% performance in recognizing INN entities since it will never assign the respective entity type.

Google Cloud Healthcare Natural Language API

Unlike the previous service, Google Cloud Healthcare Natural Language API can extract medication information from text. It maps the text to a set of predefined medical knowledge categories, these categories being ANATOMICAL_STRUCTURE, BF_RESULT, BM_RESULT, BM_UNIT, BM_VALUE, BODY_FUNCTION, BODY_MEASUREMENT, LABORATORY_DATA, LAB_RESULT, LAB_UNIT, LAB_VALUE, MEDICAL_DEVICE, MEDICINE, MED_DOSE, MED_DURATION, MED_FORM, MED_FREQUENCY, MED_ROUTE, MED_STATUS, MED_STRENGTH, MED_TOTALDOSE, MED_UNIT, PROBLEM, PROCEDURE_RESULT, PROCEDURE, PROC_METHOD, SEVERITY, and SUBSTANCE_ABUSE [hea, 2023]. Among these, it is relevant to highlight the categories MEDICINE, MED_DOSE, MED_DURATION, MED_FORM, MED_FREQUENCY, MED_ROUTE, MED_STATUS, MED_STRENGTH, MED_TOTALDOSE, and MED_UNIT, since they are the ones that are intrinsically related to the identification of INN and related entities.

To test the performance of this system, a script was then developed and executed, and the results are presented in Table 5.3. Because it only supports documents in English, the input data was translated before being passed to the system.

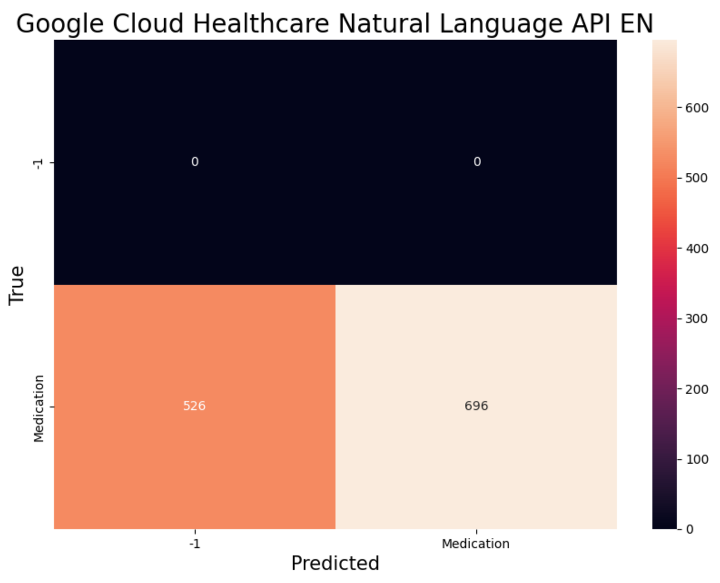


Figure 5.4: Confusion matrix obtained for INN entity recognition with Google Cloud Healthcare Natural Language API.

Execution time
810.16 seconds
Performance
56.96%
Macro F1
72.58%

Table 5.3: Results obtained for INN entity recognition with Google Cloud Healthcare Natural Language API for entities in English.

The system has a performance of about 57%, being its confusion matrix illustrated in Figure 5.4. The results show that the system can recognize more than half of the INN entities, remaining interesting for further study.

Microsoft Azure Cognitive Service for Language

To use Microsoft Azure Cognitive Service for Language and Microsoft Azure Text Analytics for Health, it was necessary to install a Python package. For that, the command pip install azure-ai-textanalytics==5.2.0 was executed [jboback, 2023].

The current language service supports several types of entities for entity recognition being them Person, PersonType, Location, Organization, Event, Product, Skill, Address, PhoneNumber, Email, URL, IP, DateTime, and Quantity [azu, 2022d]. As can be seen, those types do not include medication or similar, so this service should have a performance equal to 0%.

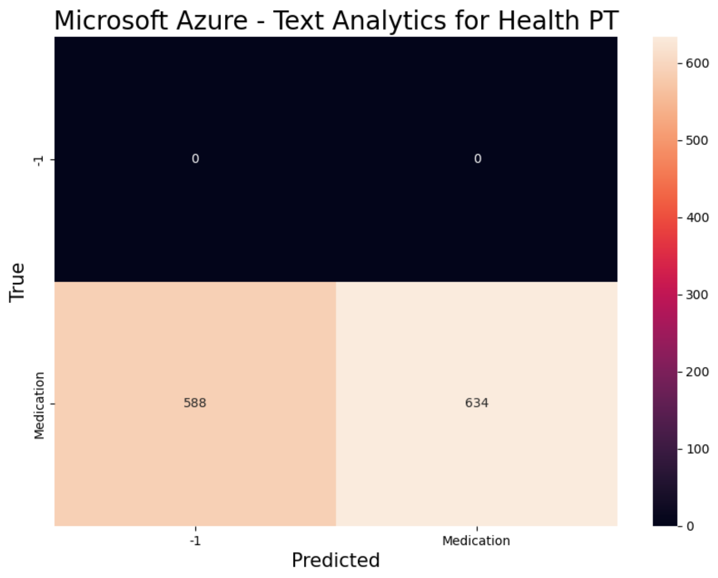


Figure 5.5: Confusion matrix obtained for INN entity recognition with Microsoft Azure Text Analytics for Health for entities in European Portuguese.

Microsoft Azure Text Analytics for Health

In opposite to the previous service, Text Analytics for Health can recognize medical-related entities like Anatomy, Demographics, Examinations, External Influence, General attributes, Genomics, Healthcare, Medical condition, Medication, Social, and Treatment [azu, 2023]. Therefore, a script was developed to determine the performance of this system, and the results are presented in Table 5.4.

Execution time
6599.94 seconds
Perfomance
51.88%
Macro F1
68.32%

Table 5.4: Results obtained for INN entity recognition with Microsoft Azure Text Analytics for Health for entities in European Portuguese.

Figure 5.5 presents the confusion matrix obtained. With these results, it can be seen that this system can also recognize more than half of the entities. However, this service is not available for European Portuguese but only for Brazilian Portuguese. Because of that, it was also tested in English, and the results are presented in Table 5.5.

Execution time
6582.50 seconds
Perfomance

63.09%
Macro F1
77.37%

Table 5.5: Results obtained for INN entity recognition with Microsoft Azure Cognitive Service for Language - Text Analytics for Health for entities in English.

Figure 5.6 illustrates the confusion matrix obtained with the last test. The results show that the system performs better in English, recognizing more INN entities in the language. Also, these results show the interest of the solution in the current project, which continued for further tests.

spaCy

To test spaCy, it was necessary to install it. To do so, the instructions in [6] were used, namely the following commands:

- pip install -U pip setuptools wheel
- pip install -U spacy
- python -m spacy download pt_core_news_sm
- python -m spacy download pt_core_news_md
- python -m spacy download pt_core_news_lg

These commands correspond to the choice of Windows as the operating system, x86 as the platform, pip as the package manager, CPU as hardware, and Portuguese for trained pipelines. spaCy provides three models for European Portuguese, and all of them were downloaded for tests.

Those three models have four different named entity types that they can recognize: ORG, LOC, PER e MISC [K] [Hien.Ha, 2021]. With that, the provided models for the language should not be able to recognize the entities of interest in this project. So, the default models have no interest in this project and should have a performance equal to 0%.

Conclusion

The results obtained with the several solutions are summarized in tables 5.6 and 5.7.

	Performance

[6]https://spacy.io/usage

Amazon Comprehend	00.00%
Google Cloud Natural Language API	00.00%
Microsoft Azure Cognitive Service for Language	00.00%
Microsoft Azure Text Analytics for Health	51.88%
spaCy	00.00%

Table 5.6: Default models performance in the INN recognition for entities in European Portuguese.

	Performance
Amazon Comprehend Medical	29.21%
Google Cloud Healthcare Natural Language API	56.96%
Microsoft Azure Text Analytics for Health	63.09%

Table 5.7: Default models performance in the INN recognition for entities in English.

From the previous tests, the models studied have more difficulty identifying INN entities that correspond to compound names or have multiple substances. It can be understood that the results obtained in this phase could be better since the maximum performance achieved is lower than 64%. This revealed the need to fine-tune the default models to adjust them to the needs of this project.

With the results shown, it can be understood that the most suitable default model in the context of this project corresponds to Microsoft Azure Text Analytics for Health. Google Cloud Healthcare Natural Language API is the second better model, being faster than the first.

5.3.2 Customized models

Because of the poor results obtained with the default models, it needed customization to achieve better results. The customized models were then tested being the test proceeding, and the results obtained presented in the following sections.

Amazon Comprehend

Amazon Comprehend entity recognition can be customized in two ways: with an entity list or with annotations. The two input type entail different effort and complexity, also leading to different results in terms of performance. Annotations are more expensive to make but lead to better results, while entity list is simpler to make, but the results can be worse. Looking at the cases in which Amazon advises using one or the other type of input, it can be understood that the most suitable for this project is entity list since there is already a list of the entities

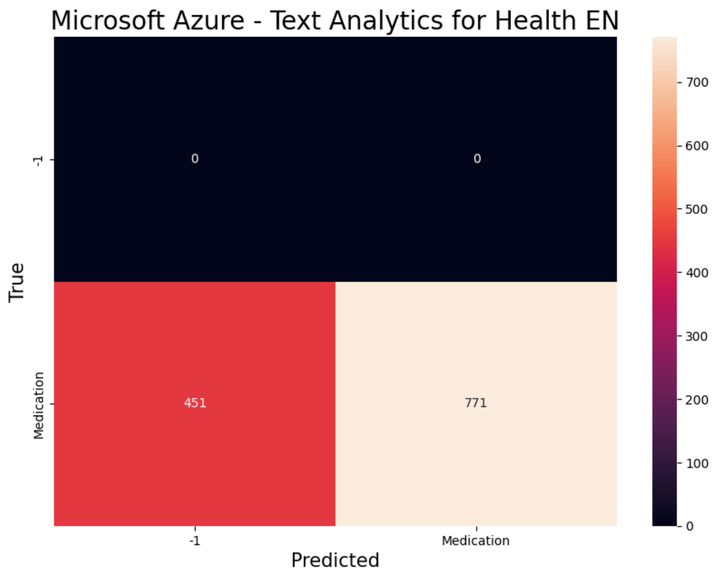


Figure 5.6: Confusion matrix obtained for INN entity recognition with Microsoft Azure Text Analytics for Health for entities in English.

intended to be recognized, the list is complete, and this is a project of first-time user [7].

Focusing on the model training using an entity list, it was necessary to provide two files: the list of entities intended to be recognized along with their corresponding entity types and a set of unannotated documents where the entities appear. The entity list should be a UTF-8 encoding CSV file with two columns for (1) Text, that corresponds to the text of an entry example and should be exactly like it appears in the accompanying document corpus, and (2) Type, that refers to the customized entity type, defined by the user. Each model can be trained for up to 25 entity types, and a minimum of 200 entity mentions per entity type is required, so it is possible to train the model for custom entity recognition [8]. So, the test of the customized model was divided into six phases:

- Generation of the CSV file
- Generation of the prescriptions dataset
- Loading of the documents into an S3 bucket
- Train of the customized model for MEDICATION entity recognition
- Endpoint creation

[7] https://docs.aws.amazon.com/comprehend/latest/dg/prep-training-data-cer.html
[8] https://docs.aws.amazon.com/comprehend/latest/dg/cer-entity-list.html

- Test of the model and evaluation of the results

The first step was generating the CSV file with the multiple INN available in the Infarmed database and the respective entity type associated with them, MEDICATION. Then, the prescriptions dataset was generated. This generation was needed due to the scarcity of this kind of data, namely for non-English languages, as stated by the authors of [Kocabiyikoglu et al., 2019]. To this end, a Python script was executed. This script iterated over different values for the verb used for prescription, the INN to administrate, the dosage, the administration route, the frequency of administration, the beginning and the ending or duration of the administration. The different values used can be seen in Table 5.8. Also, this service limits the training dataset length to a maximum of 200.000 lines.

Component	**Value**
Verbs	Prescrever Iniciar
INN	1222 unique values contained in Infarmed database
Dosage	20 ml 40 mg + 10 mg 40 mg + 10 mg + 10 mg 250 mg + 3 mg + 10 mg + 36 mg 0.5 mg/ml + 0.6 mg/ml
Route	Oral Via intravenosa Anal Via vaginal Subcutânea Via intramuscular
Frequency	de 2 em 2 horas 1 vez por dia 2 vezes ao dia Às refeições Ao jantar Ao almoço e ao jantar Em jejum
Beginning	A começar hoje A começar a 01-11-2024
Ending	E sem fim definido. E até dia 23-12-2024.
Duration	E durante 7 dias.

Table 5.8: Different values used for each prescription component for the prescriptions dataset generation process.

Following the example present in [9], a recognizer was trained to detect entities of type MEDICATION. The programming language used was Python.

[9]https://docs.aws.amazon.com/comprehend/latest/dg/get-started-cer.html

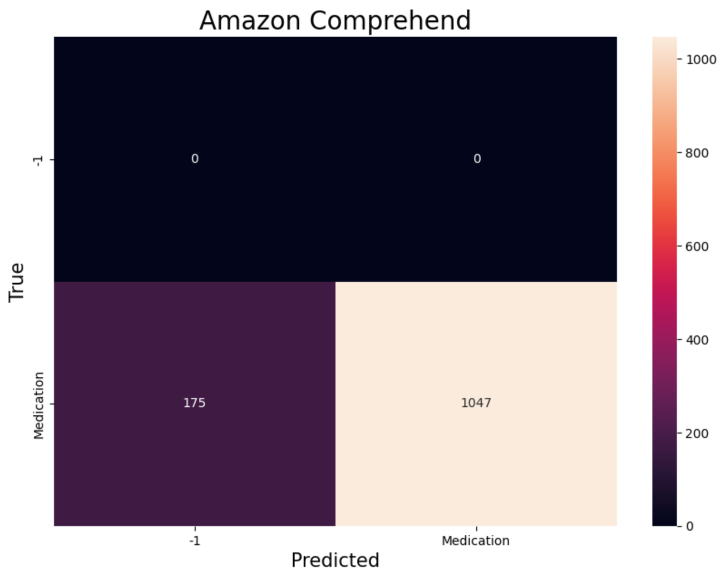


Figure 5.7: Confusion matrix obtained for INN entity recognition with Amazon Comprehend.

The test phase was based on the guidelines to make a real-time entity detection analysis and encompassed the creation of an endpoint and then the test of the model itself [10]. The number of inference units to be used in the endpoint creation represents the throughput, corresponding each inference unit to 100 characters per second, being its number limited between one and ten. For the present test, three inference units were used [11]. After the endpoint creation, the developed script for the test was executed, and the results obtained are presented in Table 5.9.

Execution time
221.00 seconds
Performance
85.68%
Macro F1
92.29%

Table 5.9: Results obtained for INN entity recognition with Amazon Comprehend customized model.

The confusion matrix obtained is presented in Figure 5.7. These results show a big improvement from the results obtained with the previous models, being the present model able to identify almost all of the INN entities present in the test dataset. With that, the model was considered for further study.

[10] https://docs.aws.amazon.com/comprehend/latest/dg/detecting-cer-real-time-api.html
[11] https://docs.aws.amazon.com/comprehend/latest/dg/detecting-cer-real-time.html

Amazon Comprehend Medical

Despite the poor results obtained with the Amazon Comprehend Medical default model, no further studies could be done with the service since it is not customizable. Therefore, no tests were made on the level of this service in this section.

Google Cloud Natural Language API

Similarly to the previous service, Google Cloud Natural Language API is also not customizable. For this reason, it lost all interest in the context of the present work, and no testing was done.

Google Cloud Healthcare Natural Language API

This solution is also not customizable, so there is no way to fine-tune it. That said, there is no space to improve the results obtained with its default model.

Microsoft Azure Cognitive Service for Language

Microsoft Azure Cognitive Service for Language allows the creation of custom models for Named Entity Recognition (NER). To develop one of these models, Microsoft's tutorial was followed [aahill et al., 2023]. With that, the model generation followed the flow:

1. Creation of a new resource for Custom text classification and Custom NER from the Azure portal
2. Generation and labeling of the training data
3. Uploading of the training data and labels file to the blob container
4. Creation of a custom NER project
5. Model training
6. Model deployment
7. Model test

Regarding the training data generation and labeling, it considered the fact that Microsoft Azure imposes:

- A limit of 100.000 documents used to train the model
- A limit of 128.000 characters per document
- That all documents are text files
- A limit of 15 MB for the labels JSON file

A training dataset with 100 prescriptions per INN entity was generated, considering the same data generation procedure as Amazon Comprehend. The 122.200 resultant prescriptions were divided into 111 documents, each containing prescriptions related to different INN entities. Also, the JSON file with the model's specifications – namely, the entity types it must recognize and the language – and the annotations associated with each document, identifying the location of the entities of interest and their label, was generated. The data labeling was done using a Python script, which returned the JSON needed.

With the data generated, it was uploaded to the blob storage. After that, a training job was started considering 111 documents for training and one for testing, corresponding to the test dataset. The results presented in Table 5.10 were achieved.

Execution time
9203.41 seconds
Performance
100.00%
Macro F1
100.00%

Table 5.10: Results obtained for INN entity recognition with Amazon Comprehend customized model.

Figure 5.8 presents the confusion matrix obtained. These results show the ability of the system to correctly identify all the INN entities as it, being, as the previous model, a huge improvement when compared with the non-customized models. This system was then further studied.

Microsoft Azure Text Analytics for Health

Regarding Text Analytics for Health, as with Amazon Comprehend Medical and Google Cloud solutions, the customization of the base model is not allowed. For this reason, no testing of this solution was done in this section.

spaCy

Like Amazon Comprehend and Microsoft Azure Cognitive Service for Language, spaCy requires training data to fine-tune its base models. This training data refers to a set of prescriptions with annotations that should identify, besides others,

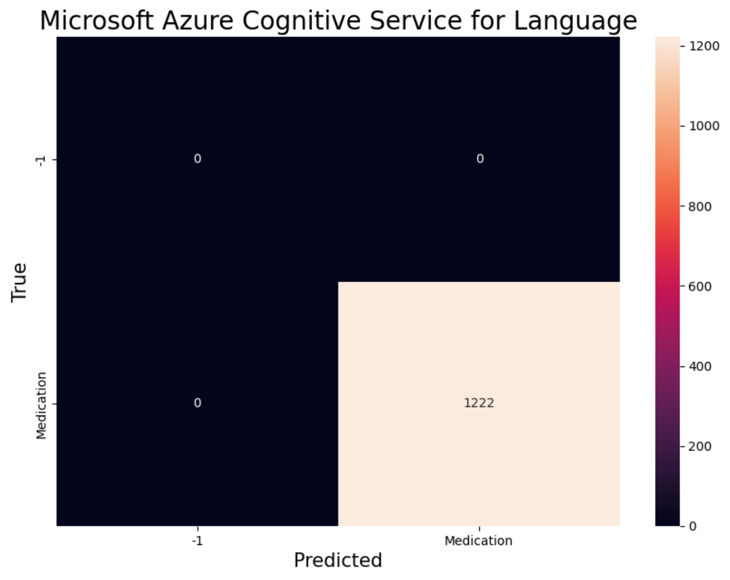


Figure 5.8: Confusion matrix obtained for INN entity recognition with Microsoft Azure Cognitive Service for Language.

the entities in the prescription and its entity type. With that, to fine-tune the models provided by spaCy for European Portuguese, it should then be necessary to follow the following steps:

- Generation of the training data
- Training the model using the data previously generated
- Test the model and evaluate its results

A dataset was generated using the data generation procedure adopted by Amazon Comprehend. Following the tutorial in [Jaiswal, 2019], a TSV file was created based on the previous dataset. The TSV file included, in each line, a non-INN text with the associated tag 0 or an INN with the associated tag MEDICATION. This TSV file was converted into a JSON file that, in turn, was converted into the final format required by spaCy for its training data. Having said that, and having the training data ready to be used, the three default models for European Portuguese were trained and tested. Also, all the tutorial scripts were adapted to be aligned with this project's objectives.

The models have four adjustable parameters, so multiple combinations were tested. Those parameters include the number of prescriptions per entity in the training dataset, the number of iterations, the dropout rate, and the batch size used.

The first models were trained based on a dataset with 100 prescriptions per INN entity and a batch size of 1000, being tested with multiple dropout rates for a variable number of iterations. The results are shown in Table 5.11 and Table 5.12. The values marked with an correspond to the converged tests and should not be executed with more iterations.

Model	Iterations	Dropout rate				
		10%	30%	50%	70%	90%
sm	50	4.05*	5.56	6.20	5.75	5.33
	100		3.70*	3.56*	4.47	4.39
	150				3.82	3.40
	200				3.78	3.66
	250				3.39	
md	50	4.12*	5.43*	6.26	5.63	5.07
	100		4.60*	5.15	5.04	4.35
	150			4.21*	4.11	4.04
	200				3.76	
lg	50	4.16*	4.18	6.82	4.89	5.37
	100		4.96*	3.98	3.58	3.84
	150			4.40	3.98	5.30
	200			3.97*		

Table 5.11: Execution time in seconds obtained for different dropout rates with the three spaCy's pipelines available for European Portuguese.

Model	Iterations	Dropout rate				
		10%	30%	50%	70%	90%
sm	50	24.14%*	24.30%	21.85%	23.08%	23.24%
	100		22.42%*	15.30%*	24.14%	24.30%
	150				22.50%	24.30%
	200				23.16%	24.30%
	250				24.14%	
md	50	24.80%*	24.14%*	24.88%	24.88%	24.88%
	100		20.62%*	24.88%	24.88%	24.88%
	150			24.96%*	24.96%	24.88%
	200				24.88%	
lg	50	22.59%*	23.00%	23.00%	23.00%	23.00%
	100		22.59%*	23.00%	23.00%	23.00%
	150			22.91%	23.00%	23.00%
	200			23.00%*		

Table 5.12: Entities detection performance obtained for different dropout rates with the three spaCy's pipelines available for European Portuguese.

Tables 5.11 and 5.12 make it possible to understand that none of the models converged for 70% and 90% of dropout. They were not further tested because a pattern of losses - namely oscillations on them - and performance was detected, indicating that the models should not be able to converge with those dropout rates. It should be noted that only the results about performance in entity detection were presented because all the experiments have zero performance for relationship extraction since that functionality is not provided by spaCy.

From this first phase of tests, it could be noted that the best combination of parameters for each model should correspond to the following:

- sm model: 50 iterations with a dropout of 30%.
- md model: 150 iterations with a dropout of 50%.
- lg model: 50 iterations with a dropout of 30%.

Those combinations of parameters were then used to make tests with a batch of different sizes, being the results presented in tables 5.13 and 5.14.

Model	Batch size				
	500	1000	1500	2000	2500
sm	3.83	5.56	4.27	3.46*	
md	3.65*	4.21	3.59*	4.01*	4.22*
lg	3.63*	4.18	3.80*	3.73	

Table 5.13: Execution time in seconds obtained for different batch sizes with the three spaCy's pipelines available for European Portuguese.

Model	Batch size				
	500	1000	1500	2000	2500
sm	12.03%	24.30%	14.08%	24.22%*	
md	24.80%*	24.96%	24.88%*	25.04%*	24.88%*
lg	22.59%*	23.00%	19.72%*	22.67%	

Table 5.14: Entities detection performance obtained for different batch sizes with the three spaCy's pipelines available for European Portuguese.

From the previous study, it can be concluded that the combination of parameters that achieves better results corresponds to the following:

- sm model: 50 iterations with a dropout of 30% and 1000 elements per batch.
- md model: 150 iterations with a dropout of 50% and 2000 elements per batch.
- lg model: 50 iterations with a dropout of 30% and 1000 elements per batch.

With that, a last set of tests were made to improve the models' performance. This test focused on the number of prescriptions per entity present in the training dataset that, by now, has been equal to 100 in all the cases. A dataset with 50, 150, and 200 prescriptions per entity was then used for the present test, and the results are shown in tables 5.15 and 5.16.

Model	Prescriptions per entity			
	50	100	150	200

sm	4.98*	5.56	3.97*	3.67*
md	4.56*	3.52	3.75*	3.58*
lg	5.28*	4.18	4.35*	5.74*

Table 5.15: Entities detection performance obtained for different batch sizes with the three spaCy's pipelines available for European Portuguese.

Model	**Prescriptions per entity**			
	50	**100**	**150**	**200**
sm	23.00%*	24.30%	22.59%*	23.40%*
md	24.80%*	24.88%	24.63%*	22.42%*
lg	22.01%*	23.00%	23.24%*	23.00%*

Table 5.16: Entities detection performance obtained for different batch sizes with the three spaCy's pipelines available for European Portuguese.

From these tests, it can be seen that spaCy can not overcome 25% of performance. This value is meager, so this solution was not further considered for the final solution for this project. This performance value could be caused by the fact that some of the INN entities are too long and, on the other hand, by the fact that simple entities like Amoxicillin or Clavulanic Acid appear in multi-substance INN entities, what could make the recognizer work harder. As said by spaCy authors, those conditions mean that the entity recognition component provided by them is not a good fit for the problem, justifying the bad results [12].

By way of example, figures 5.9, 5.10, and 5.11 illustrate the confusion matrices obtained along the tests. It can be seen that all of the models perform poorly, failing to recognize the majority of the INN entities.

Conclusion

The summarization of the results obtained with the several customized solutions is presented in Table 5.17, being the spaCy value equal to the highest performance achieved with the system.

	Performance
Amazon Comprehend	85.68%
Microsoft Azure Cognitive Service for Language	100.00%
spaCy	25.04%

Table 5.17: Customized models performance in the INN recognition.

The results obtained with Amazon Comprehend and Microsoft Azure Cognitive Service for Language were better than the ones obtained with the default

[12]https://spacy.io/api/entityrecognizer

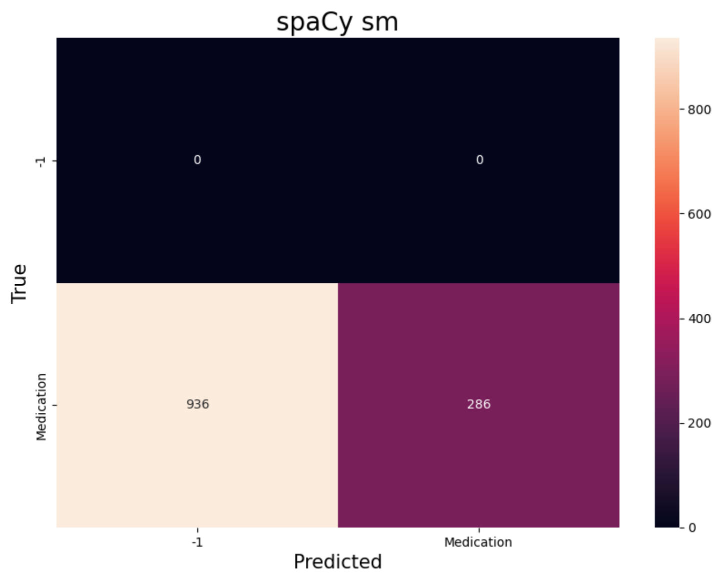


Figure 5.9: Confusion matrix obtained for INN entity recognition with spaCy sm model.

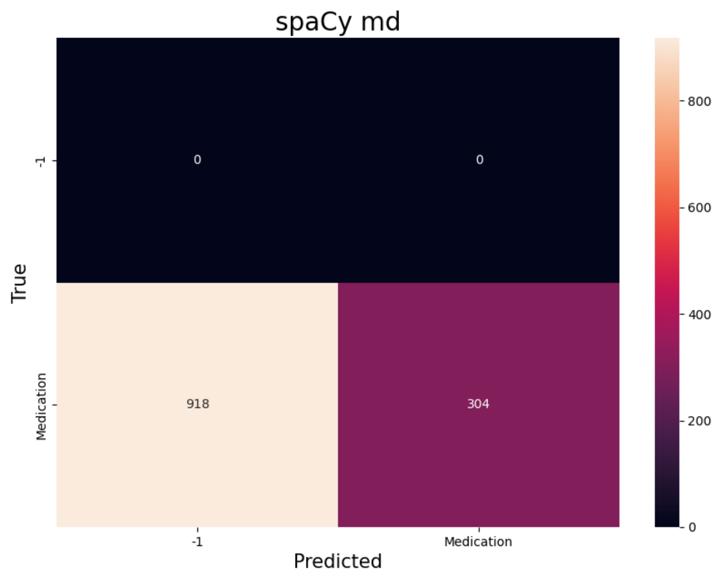


Figure 5.10: Confusion matrix obtained for INN entity recognition with spaCy md model.

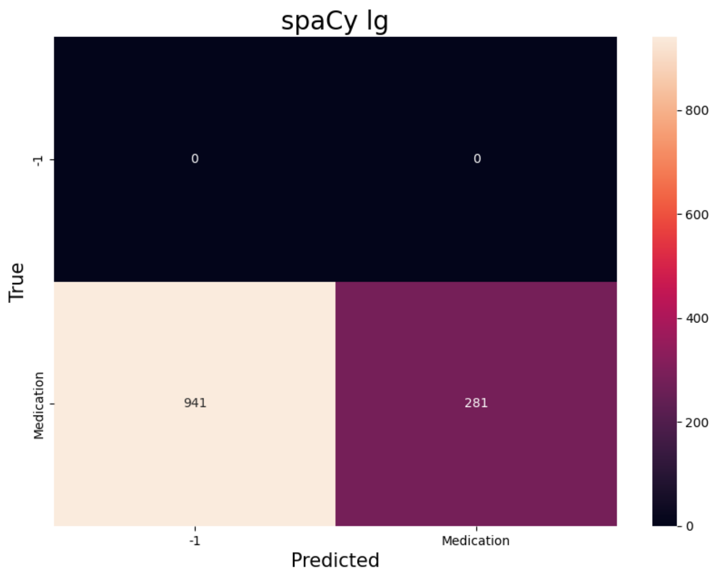


Figure 5.11: Confusion matrix obtained for INN entity recognition with spaCy lg model.

models. Nevertheless, identifying relationships between the entities is also relevant in this context. It is also relevant to see that Amazon Comprehend achieves much better results using fewer data and taking less training time than spaCy. Also, Microsoft Azure Cognitive Service for Language tends to be slower than the other solutions – it takes 9203.41 seconds in the testing phase, while the second slower takes 6599.94 seconds and corresponds to Microsoft Azure Text Analytics for Health default model.

With the results shown, the most suitable customized model in the context of this project corresponds to Microsoft Azure Cognitive Service for Language. Amazon Comprehend corresponds to the second better model and is faster than the better.

5.4 Entity recognition and relationship extraction in prescriptions

Because the relationship detection feature is also interesting in the current project, the performance of the previously determined most suitable solutions was also evaluated. The results obtained in 5.3.1 make it possible to understand that, among the default models explored, the one provided by Microsoft Azure is the most suitable in this context. From the results presented in 5.3.2, it can be concluded that the most suitable customized solution corresponds to Microsoft Azure Cognitive Service for Language customized model. Those solutions were compared at the level of prescriptions analysis for entity and relationship detection. Although Google Cloud Healthcare Natural Language API performs worse than Microsoft Azure Text Analytics for Health, it is much faster. Equally, Ama-

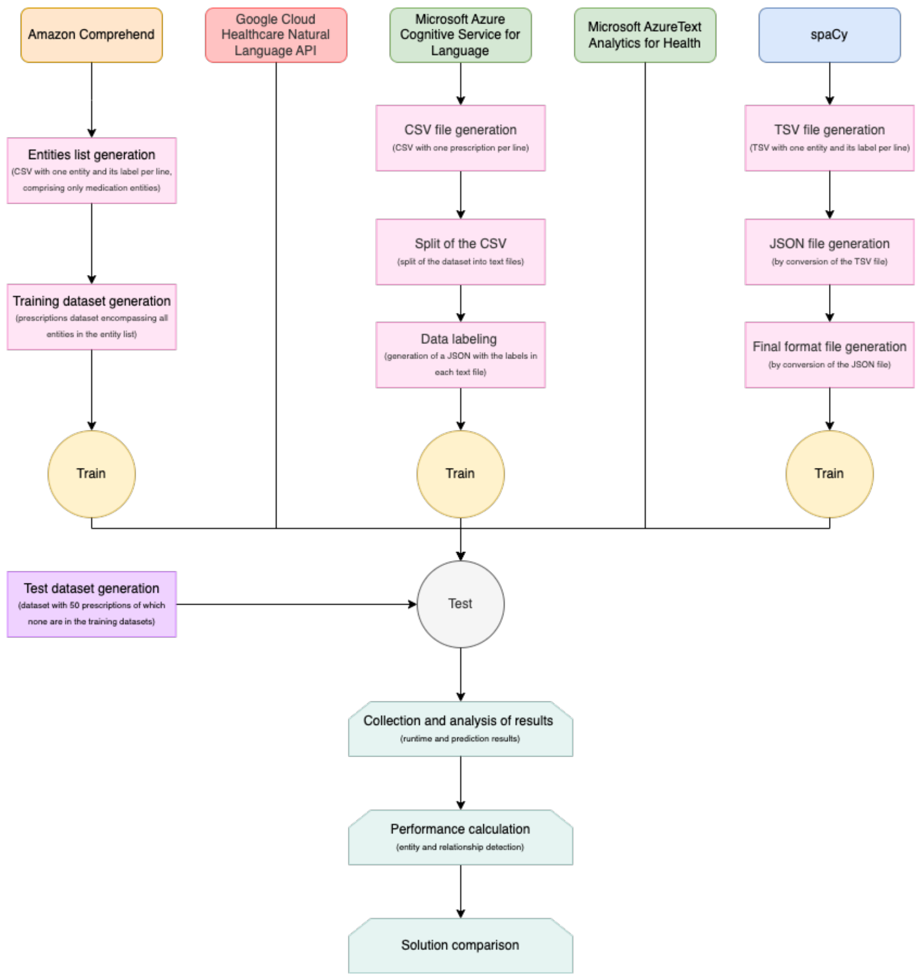

Amazon Comprehend
Google Cloud Healthcare Natural Language API
Microsoft Azure Cognitive Service for Language
Microsoft AzureText Analytics for Health
spaCy
Entities list generation
(CSV with one entity and its label per line, comprising only medication entities)
CSV file generation
(CSV with one prescription per line)
TSV file generation
(TSV with one entity and its label per line)
Split of the CSV
(split of the dataset into text files)
JSON file generation
(by conversion of the TSV file)
Training dataset generation
(prescriptions dataset encompassing all entities in the entity list)
Data labeling
(generation of a JSON with the labels in each text file)
Final format file generation
(by conversion of the JSON file)
Train
Train
Train
Test dataset generation
(dataset with 50 prescriptions of which none are in the training datasets)
Test
Collection and analysis of results
(runtime and prediction results)
Performance calculation
(entity and relationship detection)
Solution comparison

Figure 5.12: Flow chart for the tests made for INN entities and relations detection evaluation.

zon Comprehend's customized model is worse but faster than Microsoft Azure Cognitive Service for Language customized model. For this reason, these solutions were also tested in this phase. To test the solutions, the testing procedure illustrated in Figure 5.12 was followed.

For each of the systems under study, the performance, the accuracy, and the macro F1-measure were calculated. Each system has two performance measures: entity recognition performance and relationship detection performance. The first one is based on the ability of the system to identify the entities correctly and assign them the correct type. The second one examines the ability of the system to identify the relations correctly and assign them the correct label. Also, the macro F1 score was considered because there is a class imbalance since the negative cases surpass the positives for each entity type.

5.4.1 Dataset generation

To accomplish the present test phase, a test dataset was developed considering 50 unique combinations of the values presented in Table 5.18. The test dataset generated is illustrated in F.

Component	Value
Verbs	Prescrever Iniciar Fazer Tomar
INN	1222 unique values contained in Infarmed database
Dosage	Values associated with each INN entity in Infarmed database
Route	Via oral Via intravenosa Via anal Via vaginal Via subcutânea Via intramuscular Oral Intravenosa Anal Vaginal Subcutânea Intramuscular
Frequency	1 vez por dia 1 vez ao dia 2 vezes por dia 2 vezes ao dia 3 vezes por dia 3 vezes ao dia 4 vezes por dia 4 vezes ao dia Às refeições Ao almoço Ao jantar Ao almoço e ao jantar Ao deitar Em jejum
Beginning	A começar amanhã A começar hoje A começar depois de amanhã A começar a 03-01-2023
Ending	E sem fim definido. E até dia 12-01-2023.
Duration	E durante 7 dias. E durante 3 tomas.

Table 5.18: Different values used for each prescription component for the test dataset generation process.

This dataset thus encompasses prescriptions with one INN entity each and five related entities, resulting in five relationships by prescription. With that, it includes 300 entities and 250 relationships to be identified by each solution. The entities considered refer to:

- INN
- Dosage
- Route
- Frequency
- Beginning
- Ending or duration

On the other hand, the relationships include:

- Dosage of the medication
- Route of administration
- Frequency of administration
- Time of the treatment, that includes beginning, ending, and duration

It should be highlighted that none of the prescriptions of this dataset appear in any of the training datasets presented in the following sections. Also, the test dataset was used with all the solutions under study.

5.4.2 Amazon Comprehend

An Amazon Comprehend trained model can only detect the type of entities it is trained on [13]. For that reason, it was necessary to train a new model for the recognition of all the entities involved in the present study. This test followed the same process previously presented for Amazon Comprehend, again with 160 prescriptions per INN entity in the training dataset. Once again, the generated endpoint encompassed three inference units to ensure that at least one prescription was processed per second.

The results obtained with this solution are presented in Table 5.19.

Execution time	
15.99 seconds	
Entities	
Performance	86.00%

[13]https://docs.aws.amazon.com/comprehend/latest/dg/training-recognizers.html

Global accuracy	73.97%
Global macro F1	80.75%
Negative entities accuracy	73.97%
Negative entities F1	61.35%
Beginning entities accuracy	93.80%
Beginning entities F1	70.00%
Dosage entities accuracy	99.59%
Dosage entities F1	98.04%
Duration entities accuracy	99.17%
Duration entities F1	90.91%
Ending entities accuracy	100.00%
Ending entities F1	100.00%
Frequency entities accuracy	100.00%
Frequency entities F1	100.00%
Medication entities accuracy	88.84%
Medication entities F1	57.81%
Route entities accuracy	92.56%
Route entities F1	67.86%
Relationships	
Performance	00.00%
Global accuracy	48.35%
Global macro F1	13.04%
Negative relations accuracy	48.35%
Negative relations F1	65.18%
Dosage of medication relations accuracy	89.67%
Dosage of medication relations F1	00.00%
Frequency of medication relations accuracy	89.67%
Frequency of medication relations F1	00.00%
Route of medication relations accuracy	89.67%
Route of medication relations F1	00.00%
Time of medication relations accuracy	79.34%
Time of medication relations F1	00.00%

Table 5.19: Results obtained for entity recognition and relationship detection in prescriptions with Amazon Comprehend.

In figures 5.13, 5.14, and in appendix G, the confusion matrices obtained both for entity recognition and relationship detection are presented. Once again, it shows the ability of the system to accurately identify the entities of interest and its inability to identify relationships between them. The results obtained with this service are really good both for execution time and entity recognition. The system is pretty fast and achieves 86% of performance, being the model still improvable through training and post-processing. Although, it requires a permanently available endpoint to make requests to the trained model. These endpoints are expensive – for example, the endpoint used in this test cost 5.40$ per hour – and the service is not used permanently. With that, many costs are associated with periods in which the service is not being used, and the monthly fee becomes

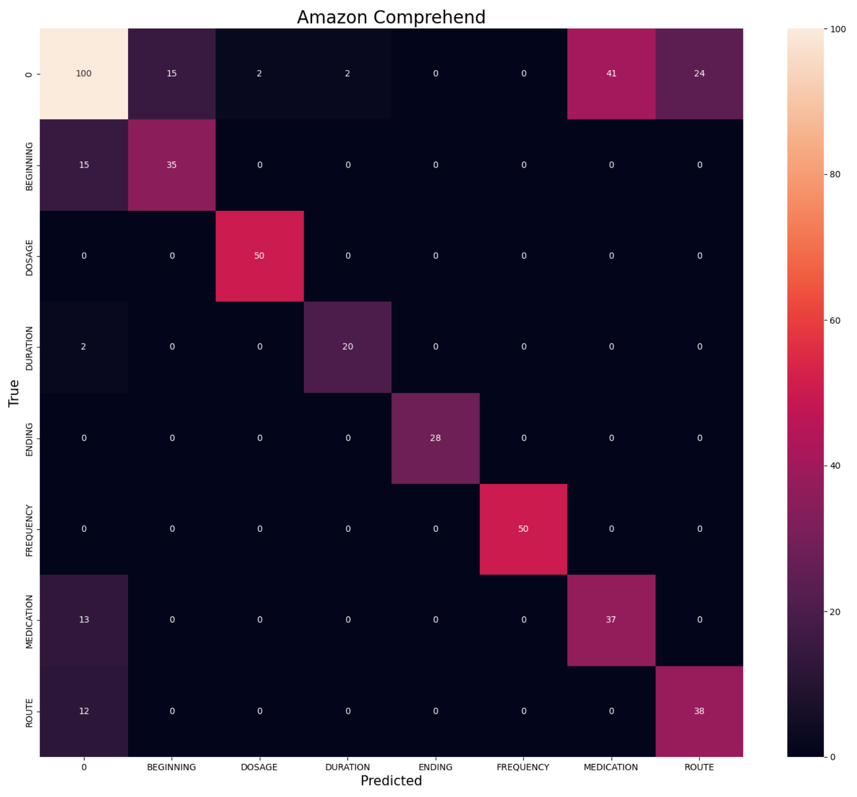


Figure 5.13: Confusion matrix obtained for entity recognition with Amazon Comprehend.

enormous. With that and the fact that the present solution does not support relationship detection, its cost does not justify its usage. So, the service is no longer particularly interesting for the current work and was not considered for the final solution.

5.4.3 Google Cloud Healthcare Natural Language API

This solution, capable of recognizing INN entities and related entities except for dates - beginning, ending, and duration - was tested through the execution of a script, and the results obtained are presented in Table 5.20.

Execution time	
35.87 seconds	
Entities	
Performance	50.00%
Global accuracy	45.29%
Global macro F1	29.51%
Negative entities accuracy	45.29%

Negative entities F1	39.84%
Beginning entities accuracy	90.94%
Beginning entities F1	00.00%
Dosage entities accuracy	90.94%
Dosage entities F1	00.00%
Duration entities accuracy	97.83%
Duration entities F1	62.50%
Ending entities accuracy	94.93 %
Ending entities F1	00.00%
Frequency entities accuracy	99.09%
Frequency entities F1	94.74%
Medication entities accuracy	97.46%
Medication entities F1	86.54%
Route entities accuracy	100.00%
Route entities F1	100.00%
Relationships	
Performance	38.00%
Global accuracy	71.92%
Global macro F1	56.07%
Negative relations accuracy	71.92%
Negative relations F1	79.58%
Dosage of medication relations accuracy	90.94%
Dosage of medication relations F1	00.00%
Frequency of medication relations accuracy	98.37%
Frequency of medication relations F1	90.11%
Route of medication relations accuracy	99.28%
Route of medication relations F1	95.83%
Time of medication relations accuracy	83.33%
Time of medication relations F1	14.81%

Table 5.20: Results obtained for INN entity recognition in prescriptions with Google Cloud Healthcare Natural Language API.

In figures 5.15, 5.16, and in appendix G, the confusion matrices obtained in this test are presented. While Table 5.20 only includes the entities of interest in the current project, these matrices also encompass other entities identified by the system. The results are poor, as the system cannot identify many entities or relations.

In some cases, the system cannot recognize INN entities that include multiple active principles as a single entity. Also, the dosage is always divided into strength and unit. Besides that, some entities are identified, but the wrong type is assigned to them. With that, post-processing should be done to achieve better results. With the post-processing of the results obtained with this solution, the maximum performances for entity recognition and relationship detection obtainable are the ones shown in Table 5.21.

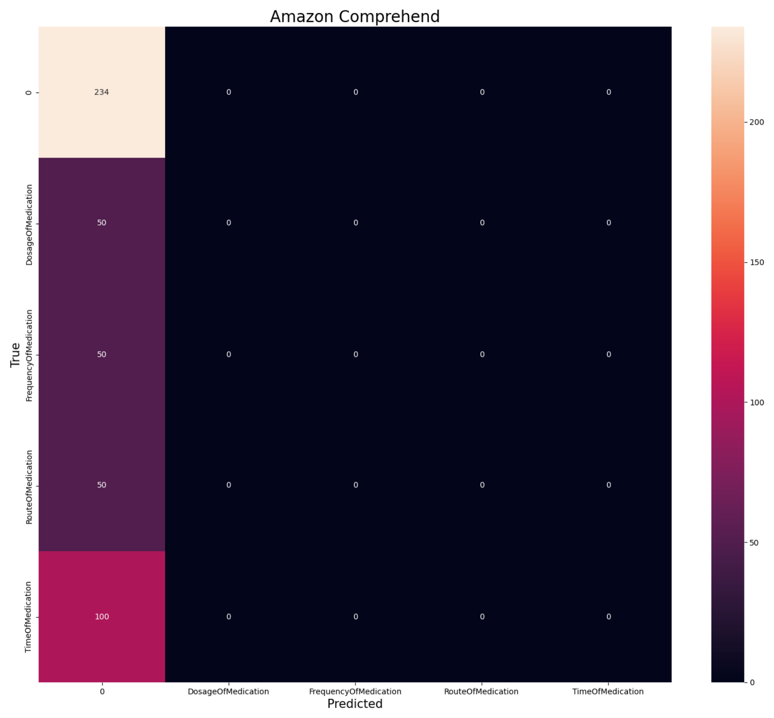


Figure 5.14: Confusion matrix obtained for relationship detection with Amazon Comprehend.

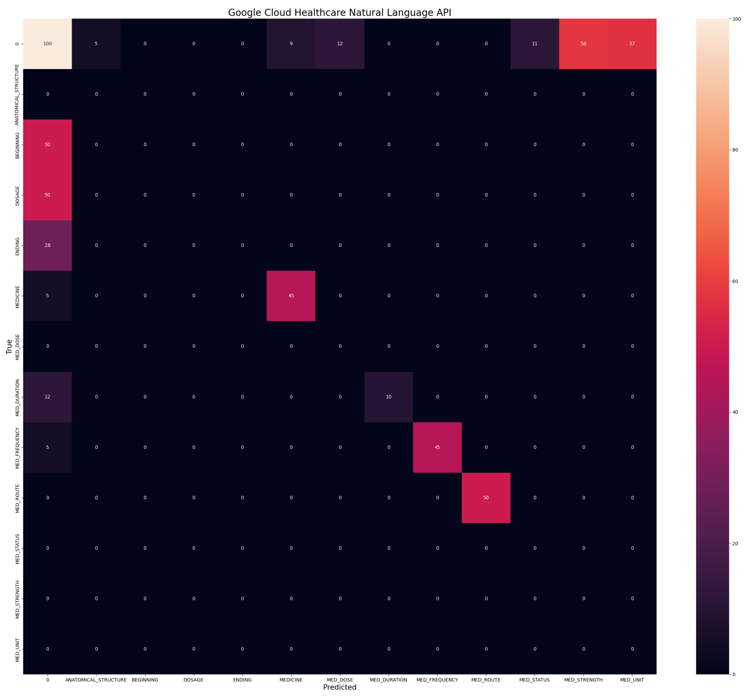


Figure 5.15: Confusion matrix obtained for entity recognition with Google Cloud Healthcare Natural Language API.

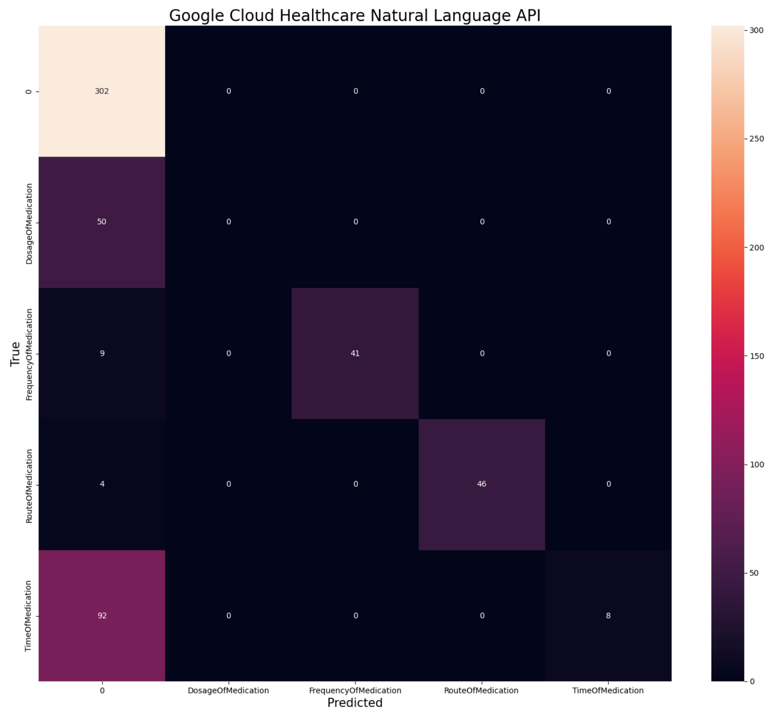


Figure 5.16: Confusion matrix obtained for relationship detection with Google Cloud Healthcare Natural Language API.

Performance - Entities
82.67%
Performance - Relationships
77.20%

Table 5.21: Results that could be obtained for entity recognition and relationship detection in prescriptions with Google Cloud Healthcare Natural Language API after post-processing.

When applied to the system, the post-processing produces the values presented in 5.22.

Execution time	
272.28 seconds	
Entities	
Performance	82.67%
Global accuracy	76.61%
Global macro F1	71.31%
Negative entities accuracy	76.61%
Negative entities F1	67.07%
Beginning entities accuracy	83.26%
Beginning entities F1	22.00%
Dosage entities accuracy	98.71%
Dosage entities F1	94.00%
Duration entities accuracy	100.00%
Duration entities F1	100.00%
Ending entities accuracy	95.28%
Ending entities F1	60.71%
Frequency entities accuracy	100.00%
Frequency entities F1	100.00%
Medication entities accuracy	99.57%
Medication entities F1	98.00%
Route entities accuracy	100.00%
Route entities F1	100.00%
Relationships	
Performance	77.20%
Global accuracy	76.39%
Global macro F1	82.99%
Negative relations accuracy	76.39%
Negative relations F1	74.77%
Dosage of medication relations accuracy	98.50%
Dosage of medication relations F1	92.93%
Frequency of medication relations accuracy	99.79%
Frequency of medication relations F1	98.99%
Route of medication relations accuracy	99.79%
Route of medication relations F1	98.99%

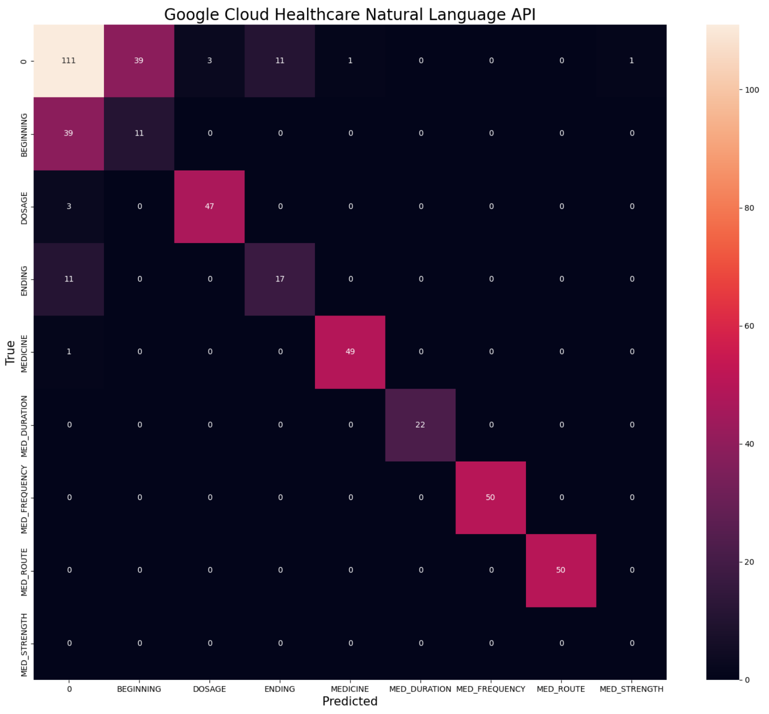


Figure 5.17: Confusion matrix obtained for entity recognition with Google Cloud Healthcare Natural Language API with post-processing.

Time of medication relations accuracy	78.33%
Time of medication relations F1	49.25%

Table 5.22: Results obtained for INN entity recognition in prescriptions with Google Cloud Healthcare Natural Language API with post-processing.

Figures 5.17, 5.18, and appendix G illustrate the confusion matrices obtained for the results with post-processing.

5.4.4 Microsoft Azure Cognitive Service for Language

A new model was trained following the same workflow as before to test the present solution. This time, and because of the limit of 15 MB labels JSON file imposed by the system, 15 prescriptions per INN entities were used, resulting in 15 documents. Those documents were used to train the model, being the test dataset used in this test passed to the trained model in the testing phase. The results shown in Table 5.23 were achieved with that.

Execution time
273.05 seconds

Entities	
Performance	92.00%
Global accuracy	83.37%
Global macro F1	83.26%
Negative entities accuracy	83.37%
Negative entities F1	72.73%
Beginning entities accuracy	98.23%
Beginning entities F1	92.31%
Dosage entities accuracy	99.33%
Dosage entities F1	97.09%
Duration entities accuracy	95.57%
Duration entities F1	54.55%
Ending entities accuracy	96.45%
Ending entities F1	76.47%
Frequency entities accuracy	98.45%
Frequency entities F1	93.33%
Medication entities accuracy	95.34%
Medication entities F1	79.61%
Route entities accuracy	100.00%
Route entities F1	100.00%
Relationships	
Performance	00.00%
Global accuracy	44.57%
Global macro F1	12.33%
Negative relations accuracy	44.57%
Negative relations F1	61.66%
Dosage of medication relations accuracy	88.91%
Dosage of medication relations F1	00.00%
Frequency of medication relations accuracy	88.91%
Frequency of medication relations F1	00.00%
Route of medication relations accuracy	88.91%
Route of medication relations F1	00.00%
Time of medication relations accuracy	77.83%
Time of medication relations F1	00.00%

Table 5.23: Results obtained for INN entity recognition in prescriptions with Microsoft Azure Text Analytics for Health.

Figures 5.19, 5.20, and appendix G illustrate the confusion matrices obtained for this test. It can be seen that, although the system has truly good results in entity recognition, achieving a performance of 92.00%, it does not provide relationship detection.

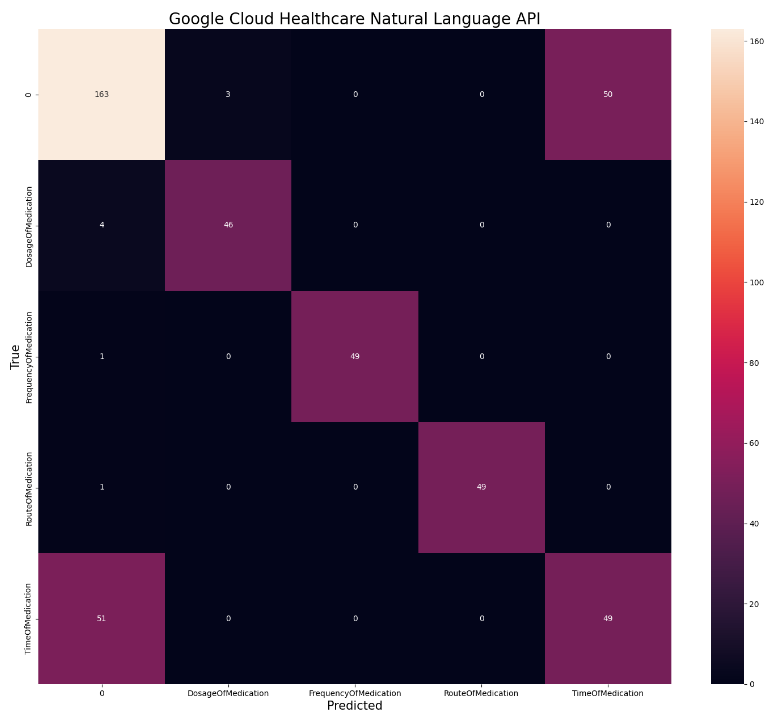


Figure 5.18: Confusion matrix obtained for relationship detection with Google Cloud Healthcare Natural Language API with post-processing.

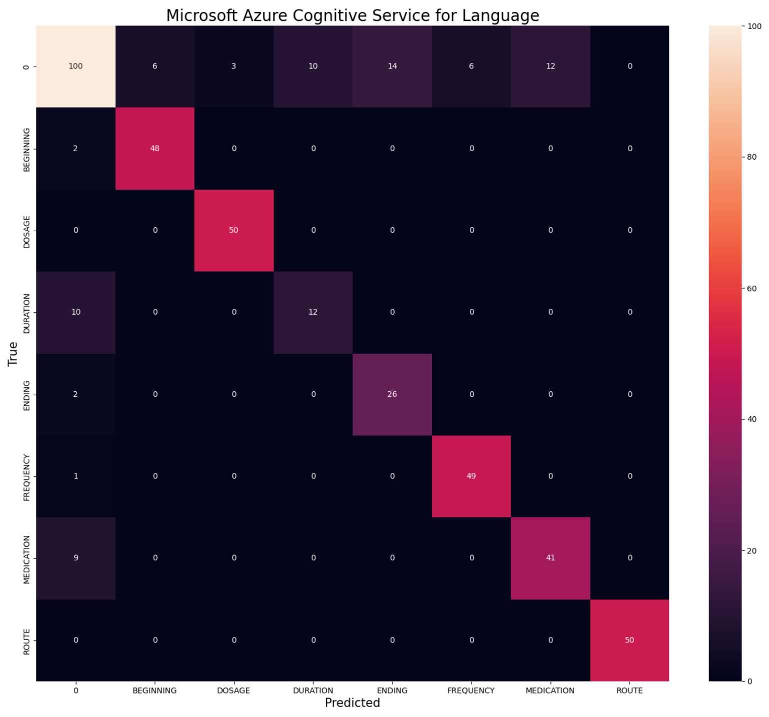


Figure 5.19: Confusion matrix obtained for entity recognition with Microsoft Azure Cognitive Service for Language.

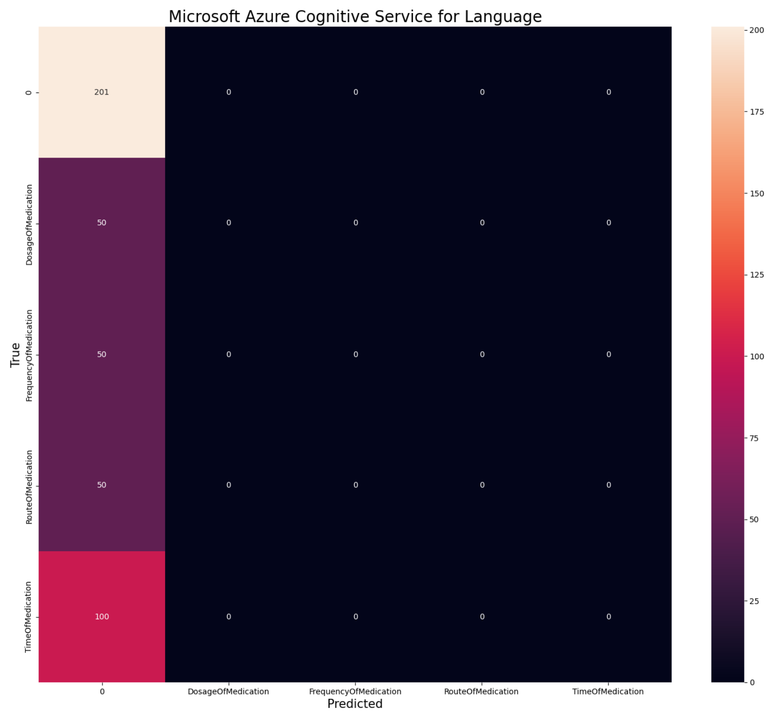


Figure 5.20: Confusion matrix obtained for relationship detection with Microsoft Azure Cognitive Service for Language.

5.4.5 Microsoft Azure Text Analytics for Health

The present solution was tested through a Python script, and the results obtained are presented in Table 5.24.

Execution time	
272.28 seconds	
Entities	
Performance	57.67%
Global accuracy	51.70%
Global macro F1	34.16%
Negative entities accuracy	51.70%
Negative entities F1	43.96%
Beginning entities accuracy	90.53%
Beginning entities F1	00.00%
Dosage entities accuracy	90.53%
Dosage entities F1	59.68%
Duration entities accuracy	95.83%
Duration entities F1	00.00%
Ending entities accuracy	94.70%
Ending entities F1	00.00%
Frequency entities accuracy	98.67%
Frequency entities F1	92.47%
Medication entities accuracy	95.83%
Medication entities F1	79.63%
Route entities accuracy	100.00%
Route entities F1	100.00%
Relationships	
Performance	46.80%
Global accuracy	67.80%
Global macro F1	62.23%
Negative relations accuracy	67.80%
Negative relations F1	73.93%
Dosage of medication relations accuracy	90.53%
Dosage of medication relations F1	59.68%
Frequency of medication relations accuracy	97.54%
Frequency of medication relations F1	85.06%
Route of medication relations accuracy	98.67%
Route of medication relations F1	92.47%
Time of medication relations accuracy	81.06%
Time of medication relations F1	00.00%

Table 5.24: Results obtained for INN entity recognition in prescriptions with Microsoft Azure Text Analytics for Health.

Figures 5.21, 5.22, and appendix G illustrate the confusion matrices obtained for this test. In opposition to Table 5.24, these figures include all the entity types

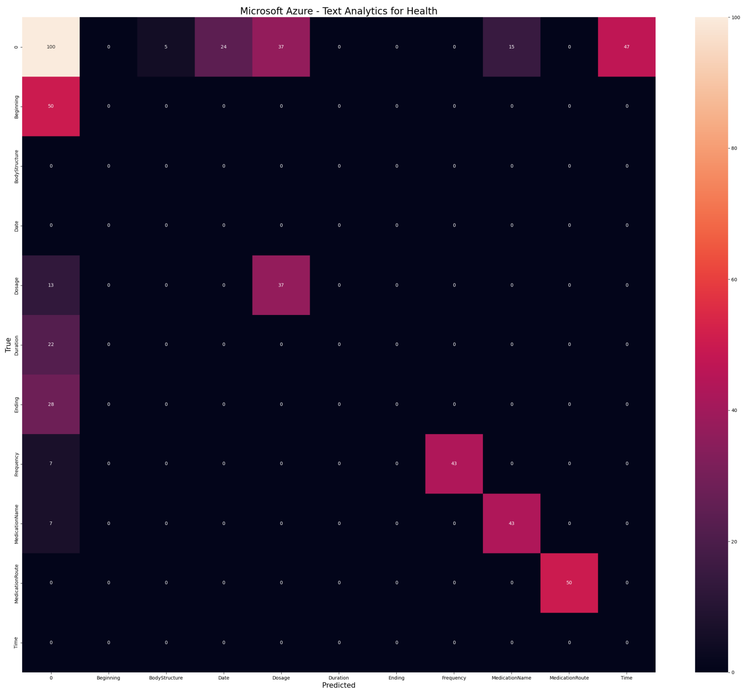


Figure 5.21: Confusion matrix obtained for entity recognition with Microsoft Azure Text Analytics for Health.

identified by the system. Once again, the results are poor, as the system can recognize only about 60% of the entities and 50% of the relations.

This solution can not identify most of the compound INN entities' names and compound dosages as a single entity. Besides that, some entities are recognized but associated with the wrong type. With that, a post-processing cloud be done to achieve better results. With the post-processing of the results obtained with this solution, the maximum performances for entity recognition and relationship detection obtainable are the ones shown in Table 5.25.

Performance - Entities
91.00%
Performance - Relationships
89.20%

Table 5.25: Results obtained for INN entity recognition in prescriptions with Microsoft Azure Text Analytics for Health, after post-processing.

When applied to the system, the post-processing produces the values presented in 5.26.

Execution time

272.28 seconds	
Entities	
Performance	86.33%
Global accuracy	81.41%
Global macro F1	81.99%
Negative entities accuracy	81.41%
Negative entities F1	70.92%
Beginning entities accuracy	93.20%
Beginning entities F1	70.00%
Dosage entities accuracy	98.64%
Dosage entities F1	94.00%
Duration entities accuracy	100.00%
Duration entities F1	100.00%
Ending entities accuracy	90.47%
Ending entities F1	25.00%
Frequency entities accuracy	99.09%
Frequency entities F1	96.00%
Medication entities accuracy	100.00%
Medication entities F1	100.00%
Route entities accuracy	100.00%
Route entities F1	100.00%
Relationships	
Performance	83.60%
Global accuracy	81.41%
Global macro F1	86.51%
Negative relations accuracy	81.41%
Negative relations F1	78.53%
Dosage of medication relations accuracy	98.64%
Dosage of medication relations F1	94.00%
Frequency of medication relations accuracy	99.10%
Frequency of medication relations F1	96.00%
Route of medication relations accuracy	100.00%
Route of medication relations F1	100.00%
Time of medication relations accuracy	83.67%
Time of medication relations F1	64.00%

Table 5.26: Results obtained for INN entity recognition in prescriptions with Microsoft Azure Text Analytics for Health with post-processing.

Figures 5.23, 5.24, and appendix G illustrate the confusion matrices obtained for this test.

5.4.6 Conclusion

Table 5.27 summarizes the results obtained in these tests.

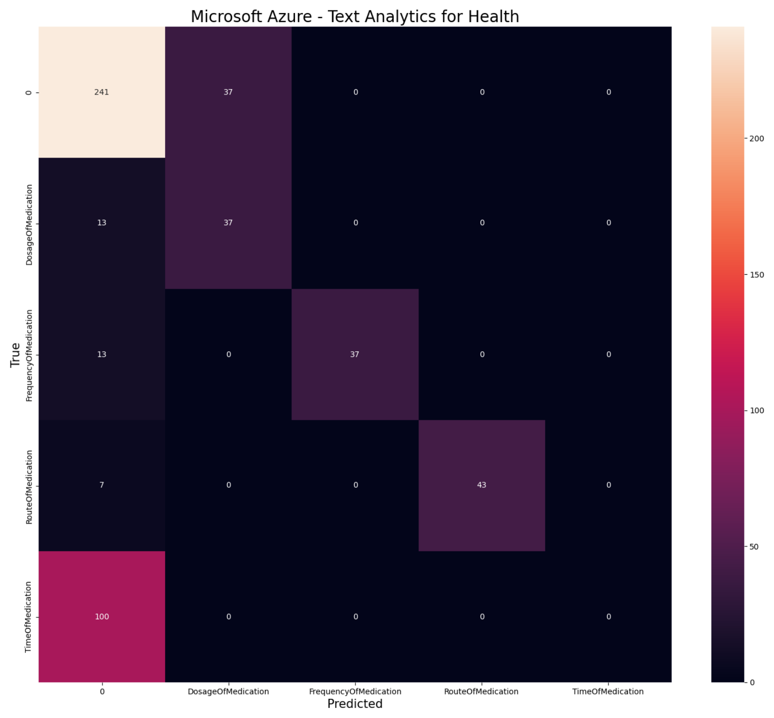


Figure 5.22: Confusion matrix obtained for relationship detection with Microsoft Azure Text Analytics for Health.

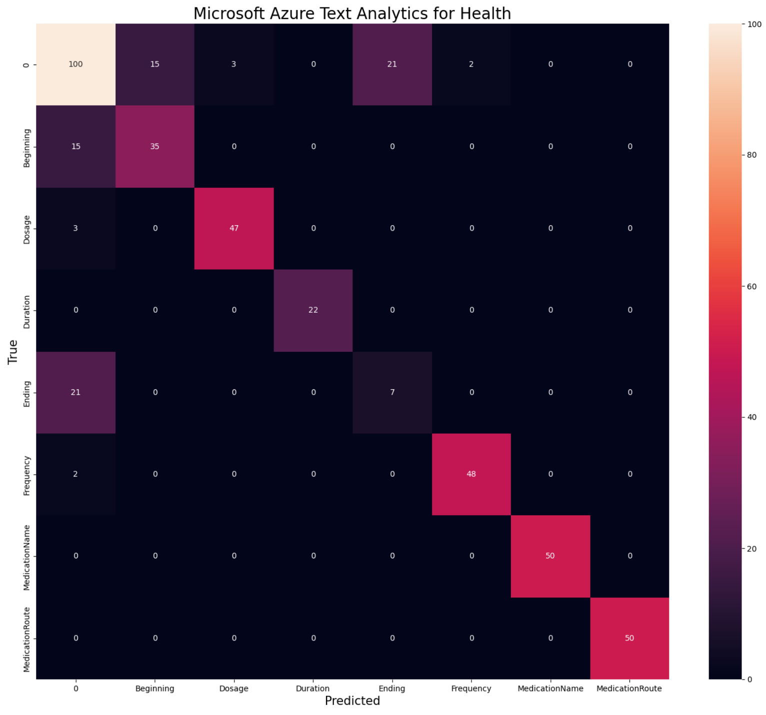


Figure 5.23: Confusion matrix obtained for entity recognition with Microsoft Azure Text Analytics for Health with post-processing.

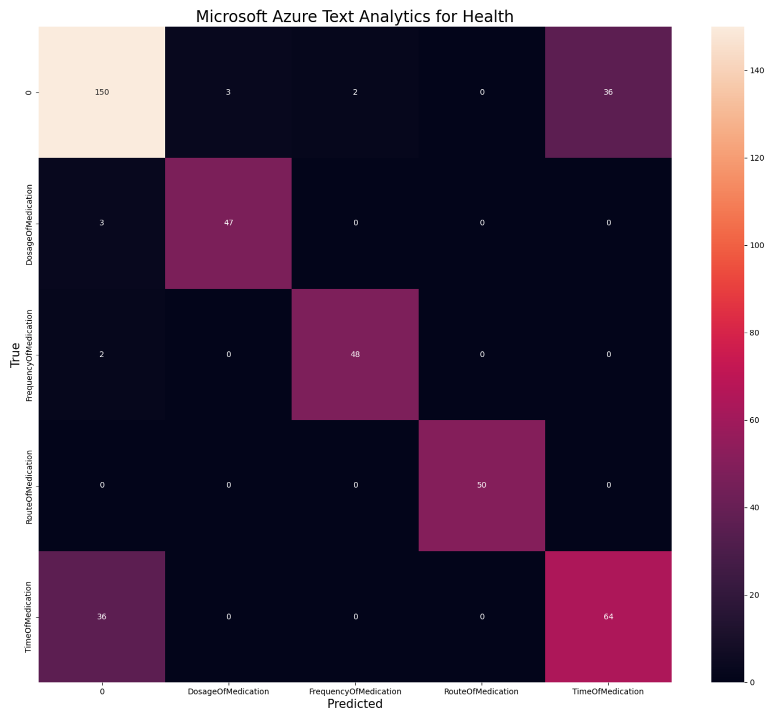


Figure 5.24: Confusion matrix obtained for relationship detection with Microsoft Azure Text Analytics for Health with post-processing.

	Performance - Entities	Performance - Relationships
Amazon Comprehend	86.00%	00.00%
Google Cloud Healthcare Natural Language API	50.00%	38.00%
Microsoft Azure Cognitive Service for Language	92.00%	00.00%
Microsoft Azure Text Analytics for Health	57.67%	46.80%

Table 5.27: Models performance for entities recognition and relationship extraction in prescriptions.

As said before, Amazon Comprehend has become too expensive compared to the other solutions because of its endpoint costs. Considering the performances achieved and the maximum performances achievable by the other solution with post-processing, Microsoft Azure Text Analytics for Health outperforms Google Cloud Healthcare Natural Language API in the previous tests. Comparing the two solutions of Microsoft Azure, it can be seen that Microsoft Azure Cognitive Service for Language has a higher performance in entity recognition. Analyzing the results, it could be understood that date translation errors and the inability of the system to identify open-ended as an ending of a prescription justify this difference. With that, translator and post-processing levels could be adjusted to achieve better performance values. Also, observing the macro F1 scores obtained for both models, it can be seen that the values are pretty close – 83.26% for Microsoft Azure Cognitive Service for Language and 81.36% for Microsoft Azure Text Analytics for Health. In addition, Microsoft Azure Text Analytics for Health has higher accuracy and macro F1 scores for INN entity recognition. With that, the performance difference becomes insignificant when Microsoft Azure Text Analytics for Health includes the two most essential features, with truly good results. With that, Microsoft Azure Text Analytics for Health should be the most suitable solution for the present work. It was then submitted to a last test regarding the detection and correct relation of entities in the context of multi-INN prescriptions, that is, prescriptions including several INN, each with their respective related entities.

5.5 Entity recognition and relationship extraction in multi-medication prescriptions

Defined that the most suitable solution for the present context corresponds to Microsoft Azure Text Analytics for Health, it was important to guarantee that this solution performs well in the context of multi-INN prescriptions. To do so, the present test followed the workflow:

1. Prescriptions dataset generation with each prescription including more than one INN entity and respective entities related to them. This dataset was generated by the aggregation of the 50 prescriptions used in the test dataset of the previous test in groups of three and one group of two. This way, the

results in a single-INN prescription and multi-INN prescription contexts could be compared.

2. Test of the system with the generated dataset as input.

3. Collection of the results obtained.

4. Post-processing development and application to the results.

5. Analysis of the results and performances' calculation.

Table 5.28 presents the results obtained with this test.

Execution time
101.76378980005393 seconds
Performance - Entities
86.33%
Performance - Relationships
83.60%

Table 5.28: Results obtained for INN entity recognition in multi-medication prescriptions with Microsoft Azure Text Analytics for Health.

From this test, it was possible to realize that even when prescriptions with several sentences are passed, each one related to the prescription of a INN entity, the system can identify both the entities and the relationships between them. This is notable because the performance is maintained from the previous test to the current one, that is, the change of context does not degrade the system performance. Furthermore, the values obtained are both above 80%, which is very good. Moreover, the values for accuracies and F1 scores, as well as the confusion matrices obtained, coincide with those of the test presented for single-glsinn prescriptions.

5.6 Conclusion

In conclusion, the previous experiments conclude that the most suitable ASR and NLP solutions for this project correspond to Microsoft Azure Cognitive Services for Speech and Microsoft Azure Text Analytics for Health, respectively. Those were the solution used in the development of the system, which is detailed in the next chapter.

Chapter 6

Approach

The current chapter recapitulates the problem definition, introducing the architecture and context of the developed solution in its sequence. In addition, it offers a brief comparison between this solution and the state of the art. Finally, the solution's components are explored individually, being their development presented.

First, the context of the problem is revisited, and the solutions' architecture and context are explored. Then, a brief comparison of the present solution with the state of the art studied is offered. After, the Application Programming Interface (API) component of the solution is presented, and its development is described. Following it, the mobile application component is analyzed, and its development is detailed.

6.1 Problem definition and solution architecture

The task of internal prescription, performed periodically by doctors for each patient for whom they are responsible, allocates excessive time, namely in the information entering into M1. This task contributes to the doctor's overload, making them more susceptible to failure and reducing the time dedicated to healthcare tasks, culminating in sub-optimal medical care.

The optimization of this task allows the minimization of the time required for its execution, allowing the reallocation of time to the provision of healthcare. Focused on this, the main goal of this project is the development of a system accessible through a mobile application for iOS that, using Automatic Speech Recognition (ASR) and Natural Language Processing (NLP), allows the optimization of the task by replacing the manual entering of information into M1 by the dictation in natural language. Even though the solution is focused on ASR-based prescription, it also allows the prescription entered in free-text to be sensitive to the cases in which the doctor is unable or unwilling to dictate it.

To realize this solution, the architecture followed is illustrated in Figure 6.1. This figure shows the organization between the mobile application, the API responsible for the communication between the mobile application and the cloud,

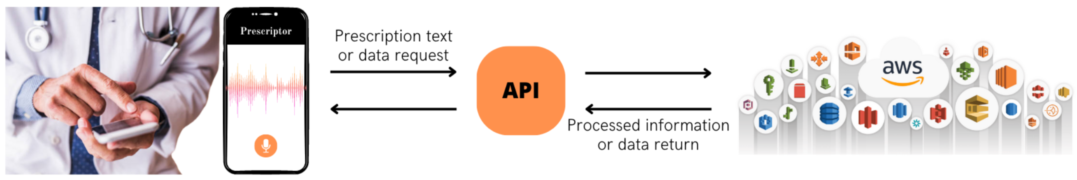


Figure 6.1: Architecture of the developed solution [aws, 2022] [Halfpoint, 2018].

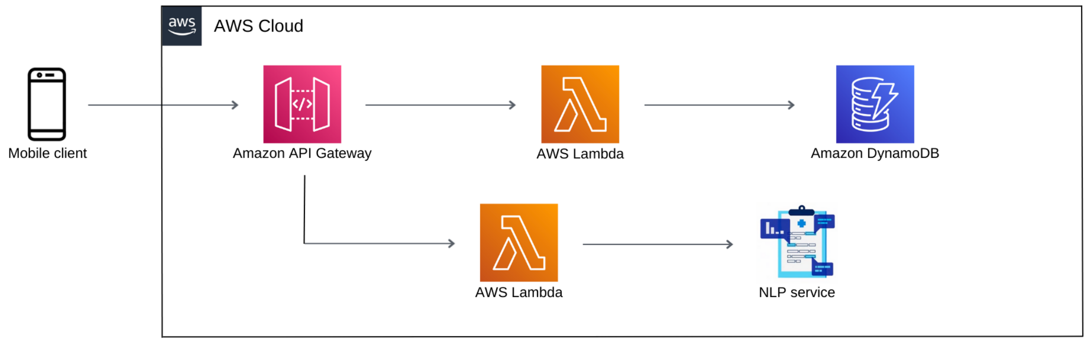


Figure 6.2: Components diagram of the solution to be developed [aws, f] [Microsoft, 2023].

and the cloud itself.

In turn, Figure 6.2 encompasses a component diagram illustrating the components used to develop the solution. In this architecture, the doctor must enter their prescription by dictation or free-text in the mobile application. When dictated, the selected ASR service transcribes it. The text transcribed or entered by the user is then sent to the API that, by invoking a lambda function, should allow the invocation of the selected NLP service. After that, the result of the processing performed by the NLP service based on entity recognition and relationship detection is returned to the application. In addition, the mobile application should be able to make data requests to the API that, by communicating with a different lambda function, should allow the return to the application of data contained in DynamoDB tables.

The system aims that after it interprets the prescription, it is presented to the doctor in the form of structured orders so that they can confirm the success of the interpretation or, on the other hand, change any erroneous information resulting from the processing done.

Figure 6.3 illustrates an example of a multi-International Non-proprietary Name (INN) prescription. The notes illustrate the processing the system should do at the entity detection level on it. In addition to the processing presented in the figure, the system should also allow the detection of relationships between the detected entities. Table 6.1 exemplifies the processing that the system should do at this level.

Entity	Entity	Relation

Omeprazol	20 mg via oral 12 em 12 horas 7 dias	Dosage Route of administration Frequency Duration
Sinvastatina + Ezetimiba	40 mg + 10 mg via vaginal 4 em 4 horas 5 tomas	Dosage Route of administration Frequency Duration
Atorvastatina + Perindopril + Amlodipina	40 mg + 10 mg + 10 mg via anal 1 vez por dia até dia 7 de março	Dosage Route of administration Frequency Ending
Paracetamol + Bromofeniramina + Cafeína + Ácido Ascórbico	250 mg + 3 mg + 10 mg + 36 mg via intramuscular em SOS	Dosage Route of administration Frequency
Xilometazolina + Brometo de ipratrópio	0.5 mg/ml + 0.6 mg/ml 1 unidade por dia via intravenosa do dia 23 de Dezembro até ao dia 27 de Dezembro	Dosage Frequency Route of administration Beginning Ending

Table 6.1: Example of relationships detection on multi-medication prescription.

This example shows the seven distinct types of entities that the system must recognize. These relate to the INN, the dose, the route of administration, the frequency, the beginning, the ending, and the duration. The doctor frequently omits the beginning of the prescription, and when it occurs, the system can assign a default value. This default value corresponds to the day the prescription is done since it is also the current practice adopted for M1. Similarly, in the absence of a specification of an ending or duration, the system understands that this is indefinite until further action by the doctor to cease the medication in question, being assigned the value open-ended to the entity.

It should be noted that the system can fail the processing in multiple ways, namely:

- The transcription can contain errors
- The transcription can miss one or more entities
- The processing may not be capable of identifying an entity
- The processing may identify an entity wrongly
  - In terms of writing
  - In terms of the entity type

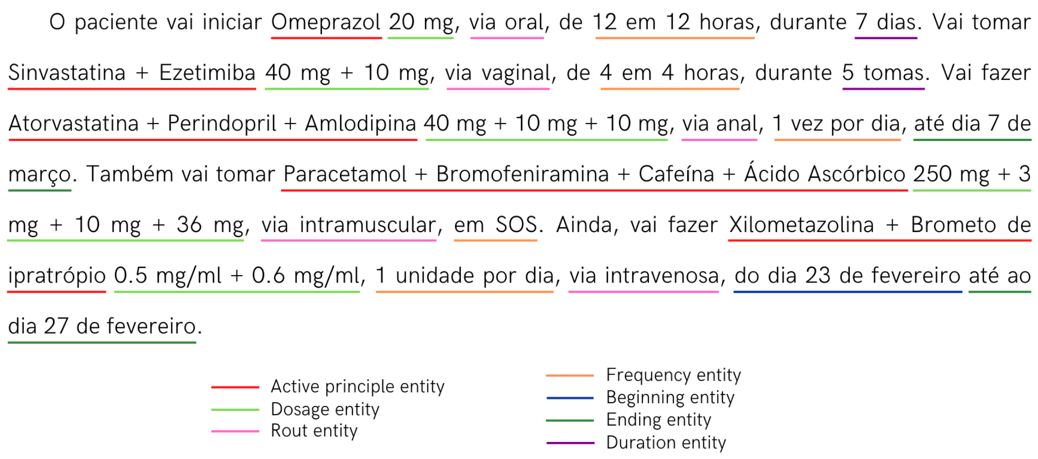

O paciente vai iniciar Omeprazol 20 mg, via oral, de 12 em 12 horas, durante 7 dias. Vai tomar Sinvastatina + Ezetimiba 40 mg + 10 mg, via vaginal, de 4 em 4 horas, durante 5 tomas. Vai fazer Atorvastatina + Perindopril + Amlodipina 40 mg + 10 mg + 10 mg, via anal, 1 vez por dia, até dia 7 de março. Também vai tomar Paracetamol + Bromofeniramina + Cafeína + Ácido Ascórbico 250 mg + 3 mg + 10 mg + 36 mg, via intramuscular, em SOS. Ainda, vai fazer Xilometazolina + Brometo de ipratrópio 0.5 mg/ml + 0.6 mg/ml, 1 unidade por dia, via intravenosa, do dia 23 de fevereiro até ao dia 27 de fevereiro.

Figure 6.3: Example of entity detection on multi-medication prescription.

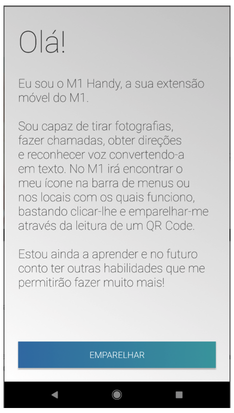
When the transcription fails, the user can edit the transcription or, if preferred, re-record and re-transcribe it. If the processing incorrectly identifies something, the user can edit the information until it reaches its intention.

As important as the prescription itself, the patient's evaluation is relevant and must be registered. This information may justify the choices made by the doctor and keep a history of the patient's health condition. Although, its registration is not contemplated in the current system. This happens because:

- The current system should not be a new, isolated system. It should be an extension of M1 and should be integrated into Handy.
- Handy is already available, allowing the registration of the previously mentioned information.

Handy, in turn, is a mobile application that corresponds to an extension of M1. It is intended to allow the user to perform on the cell phone the tasks it is more well prepared than the computer. A pairing should be made between the mobile application and the user's M1 for the user to use it. After that, the M1 has some fields in which the user can call the mobile application, being a different screen displayed by the application according to the request made by M1. With that, this application has no navigation associated with it, being controlled by M1. Between the functionalities provided by Handy, there are:

- Dictation of information transcribed into text fields on M1, not including prescription information.
- Image capture directly connected with M1.
- Display a patient's information by selecting it in M1.


Olá!

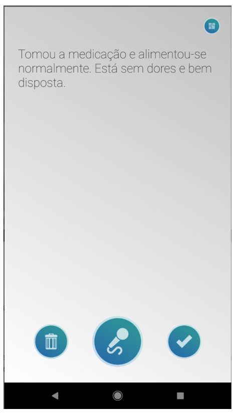
Eu sou o M1 Handy, a sua extensão móvel do M1.

Sou capaz de tirar fotografias, fazer chamadas, obter direções e reconhecer voz convertendo-a em texto. No M1 irá encontrar o meu ícone na barra de menus ou nos locais com os quais funciono, bastando clicar-lhe e emparelhar-me através da leitura de um QR Code.

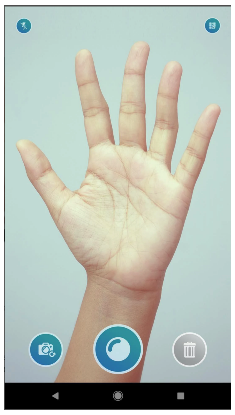
Estou ainda a aprender e no futuro conto ter outras habilidades que me permitirão fazer muito mais!

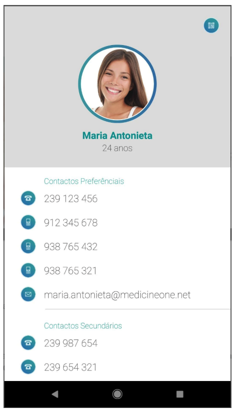
EMPARELHAR

(a)


Tomou a medicação e alimentou-se normalmente. Está sem dores e bem disposta.

(b)

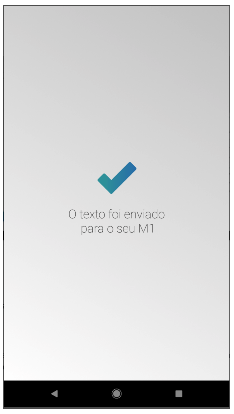

(c)


Maria Antonieta
24 anos

Contactos Preferênciais
239 123 456
912 345 678
938 765 432
938 765 321
maria.antonieta@medicineone.net

Contactos Secundários
239 987 654
239 654 321

(d)


O texto foi enviado para o seu M1

(e)

Figure 6.4: Handy's screens [m1, 2023].

Figure 6.4 illustrates Handy and the previously cited functionalities. In 6.4a, Handy's presentation and request for pairing with M1 are shown. In turn, 6.4b illustrates a case in which the doctor dictated some information about a patient, and the transcribed text is returned. Subfigure 6.4c presents the camera functionality, where the doctor can take pictures that will automatically be sent to M1. Also, 6.4d shows the screen responsible for the displaying of patient's information, and, finally, 6.4e illustrates the confirmation message displayed by handy after the submission of a text to M1.

The integration of the system developed in this project with Handy, which is included in the integration task with M1, is out of the scope of this internship.

6.2 Comparison with the state of the art of medical prescription

Tables 6.2 to 6.12 summarize and compare the state of the art of medical prescription with the present work. This comparison focuses on the type of solution developed, the type of input accepted by the system, the information it stores, if it generates alerts or not, if there is doctor's validation and sending to other systems, the authentication, the ASR component used, the NLP component used, the method used to fill the prescription, the common advantages and the usage of open-source solutions. In these tables, ID 1 refers to the study [Shaikh et al., 2021], 2 to [Kocabiyikoglu et al., 2019] and [Kocabiyikoglu et al., 2020], 3 to [Dhokley et al., 2021], 4 to [Mahatpure et al., 2019] and 5 to the present study.

ID	Application	
	Mobile	Web
1	x	
2	x	
3	x	
4	x	x
5	x	

Table 6.2: Comparison of the state of the art with the current work. (I)

ID	Input	
	Dicate	Free-text
1	x	
2	x	x
3	x	
4	x	
5	x	x

Table 6.3: Comparison of the state of the art with the current work. (II)

ID	Stored information	
	Patient data	Prescriptions
1	x	x
2		
3		
4	x	x
5		

Table 6.4: Comparison of the state of the art with the current work. (III)

ID	Alerts
1	Yes
2	Yes
3	No
4	No
5	No

Table 6.5: Comparison of the state of the art with the current work. (IV)

ID	Doctor's validation

1	First and only
2	Followed by sending to Prescription Management Systems (PMS)
3	First and only
4	First and only
5	Followed by sending to M1

Table 6.6: Comparison of the state of the art with the current work. (V)

ID	**Authentication**
1	Faster because of personal device's usage
2	
3	
4	
5	

Table 6.7: Comparison of the state of the art with the current work. (VI)

ID	**ASR component**	
	One solution	**Several solutions**
1		x
2	x	
3	Unkown	
4	x	
5	x	

Table 6.8: Comparison of the state of the art with the current work. (VII)

ID	**NLP component**
1	None
2	Developed from scratch
3	Intent recognition
4	Developed from scratch
5	Cloud service

Table 6.9: Comparison of the state of the art with the current work. (VIII)

ID	**Prescription filling**
1	String comparison
2	Slot-filling
3	Slot-filling
4	Keywords search
5	Entities detection and relationship extraction

Table 6.10: Comparison of the state of the art with the current work. (IX)

ID	Advantages
1 2 3 4 5	Portability and time saving

Table 6.11: Comparison of the state of the art with the current work. (X)

ID	Open-source solutions
1	Partially
2	Yes
3	Yes
4	Partially
5	No

Table 6.12: Comparison of the state of the art with the current work. (XI)

Besides that, all the projects diverge from the present one in terms of language since none of it targets European Portuguese. Also, they all converge in that all the projects are at least partially focused on time-saving.

The storage of information and the generation of alarms, for example, when adverse interactions exist between INNs, correspond to functionalities conferred to the developed system by M1, then associated with the integration with it. With that, although this first version does not include them, the marketable version will. Besides that, the authentication included in the system is illustrative since the final version should have authentication conferred by M1.

6.3 API

To add NLP capabilities and get resources to the mobile application, it was necessary to create an API to enable communication between the application itself and the cloud services. Here the cloud provider and cloud services used, the programming language, and the tools used are revisited, being the development associated with the API detailed.

6.3.1 Cloud Provider

The cloud provider used corresponds to the one introduced in the previous chapter: Amazon Web Services (AWS). The services used, introduced in the last chapter, correspond to Amazon API Gateway, AWS Lambda, Amazon DynamoDB, and Amazon Cognito.

6.3.2 Programming Language

Since the AWS Lambda functions can be developed in Python, and since Python is the most convenient programming language from the point of view of familiarity, all the functions developed in this project were written in the language.

6.3.3 Tools

Regarding the tools used in the company, particularly in the development team, where this work is included:

- Work management tool: Jira
- IDE: VS Code or Visual Studio
- API testing tool: Postman

6.3.4 Development

Regarding the API development, both the API itself and the services associated with it are presented in the following sections.

Amazon DynamoDB

In the context of this work, it was only necessary to use a database for storing the default values to be loaded into the application. These include the lists of possible values for the INN, the administration route, and the administration frequency. For this reason, three tables were created in DynamoDB, one for the INNs, one for the routes, and another for the frequencies. Their creation followed the flow:

1. Access to DynamoDB console
2. Dashboard access
3. Selection of the "Create table" option
4. Edition of the table name and partition key
5. Table creation
6. Changing of the capacity mode to on-demand
7. Development of a Python script for adding the items to the table. This script invokes commands from the AWS CLI, illustrated at [Amazon Web Services, 2023b].

Once created, the tables were accessed via an AWS Lambda, presented in the next section.

AWS Lambda

As for the actual development of the required Lambda functions, the instructions provided by AWS for generating a Lambda function using Python were taken into account [Amazon Web Services, 2023a]. With this, this generation entailed the following steps of interest:

1. Access to Lambda console
2. Selection of the "Create function" option
3. Entering of the required parameters:
    - The function's name
    - The runtime to be used
    - The architecture to be followed
4. Selection of the "Create function" option

Contrary to what is indicated by AWS, proceeding with the role creation was unnecessary since the service automatically generates it. The previous process allowed the generation of the lambda functions to which were then added the codes that they should execute when triggered.

Regarding the code of the Lambda responsible for accessing the tables in Amazon DynamoDB, the scan example in [Wilinski, 2020] was followed. This lambda function was tested through an event correspondent to a JSON that only included the table's name intended to load. This event invoked the lambda function, and a success code, i.e., code 200, along with the table content, proved the operation's success.

Regarding the AWS Lambda that both invokes Microsoft Azure Text Analytics for Health and processes its result, the code follows the example in [jboback, 2023]. This code uses different Python modules, not all available in the environment provided by AWS. This is the case, among others, of pandas, googletrans, and azure. The modules that, by default, are not available in Lambda functions require that a layer related to them is added to the Lambda function in which they are needed. The layers provided by AWS already include AWSSDKPandas-Python39, version 4, which is the layer that encompasses the pandas module. This module can be directly added to the lambda function, which can already use it. In contrast, AWS does not provide layers for googletrans and azure, so it was necessary to generate custom layers for these modules, including the modules they depend on. Two folders were used for each module, being them inside Python's site-packages folder on the local machine. These folders include the one with the module's name and the one that starts with the module's name and ends with dist-info. These were stored in a new folder that was then compressed. That said, and already in the AWS Lambda layers tab, each of the layers was generated through the following steps [Uncomplicated, 2021]:

1. Selection of the "Create layer" option

2. Definition of the layer name
3. Loading of the compressed folder referring to the desired module and dependencies
4. Selection of the architectures the layer should be compatible with
5. Selection of runtimes the layer should be compatible with
6. Creation of the layer

Changing the timeout in the lambda function settings was necessary for this second function. It is set to be equal to three seconds by default, which generates timeouts in this context. This lambda function was then tested using a prescription for a single INN entity, and an event was used. A JSON was generated with the key prescription and the value of a prescription, being then added to the event. This event was then used to invoke the function, and the return of a 200 code, along with the processed data, proved the operation's success.

API

For the HTTP API creation the following steps were followed [1]:

1. Access to the API Gateway console
2. Selection of the "Create" option
3. Selection of the "Compile" option in the HTTP API box
4. Selection of the "Lambda" option in the Integrations dropdown box
5. Selection of prescriptions_Azure_EN from the Lambda function name dropdown list
6. Selection of version 2.0 from the dropdown box
7. Assignment of the name prescriptions_Azure_EN_API to the HTTP API
8. Confirmation of automatically defined routes. The method used in this API route has been defined as POST
9. Confirmation of the stages, leaving $default as the name of the stage
10. Selection of the Create option
11. Addition of a new route for the get_from_DynamoDB Lambda function with the method ANY

[1]https://docs.aws.amazon.com/apigateway/latest/developerguide/getting-started.html

The API stage refers to a logical reference to a state in the lifecycle of the developed API, identified by the API ID and the stage name [2]. In turn, when it comes to controlling and managing HTTP API accesses, the API Gateway supports several mechanisms [3]:

- Lambda authorizers, which use Lambda functions to manage API accesses. When a call is made to the API by a user, the API Gateway invokes the Lambda function, developed by the client, and its response is used to infer whether the user should be granted access to the API they want to access [4].
- JWT authorizers, which use JSON web token (JWT)s – part of the OIDC and OAuth tools – to restrict access to APIs. Setting up such an authorizer for an API route implies that when a request is made to the API, the API Gateway validates the JWTs that the user submits along with the API requests. Requests are accepted or rejected according to the validation of the token and optionally according to the scopes on it. If scopes are configured for a route, the token must include at least one of the route's scopes [5].
- Standard AWS IAM roles and policies, which provide flexible and robust access controls. These can define who can create and manage client APIs and who can invoke them. With this mechanism, users must sign their requests with AWS credentials, considering Signature Version 4. That said, the API will only be invoked if the user has permission to do so [6].

Because it involves signing API requests with AWS credentials, the last mechanism presented is out of this project's scope. Also, for simplicity reasons, between the Lambda authorizer and the JWT authorizer, the JWT authorizer was chosen. Also, Amazon Cognito user pools were used as an authentication mechanism, which is responsible for the return of tokens for the authenticated user. These tokens allow the user to access the services invoked by the lambda functions.

For testing the developed API, HTTP requests were generated, and Postman was used for the API invocation. Since it has an associated JWT authorizer, adding a user to the user pool generated and associated with the API authorizer was first necessary. This user was assigned a temporary password, which was changed using the set_new_password_challenge() function provided by Python's pycognito module [Vizeli, 2023a]. Then, the user was authenticated using the AWSSRP() and authenticate_user() functions [Vizeli, 2023b] [7]. This authentica-

[2]https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-basic-concept.html

[3]https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-access-control.html

[4]https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-lambda-authorizer.html

[5]https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-jwt-authorizer.html

[6]https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-access-control-iam.html

[7]https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/cognito-idp/client/admin$_i$*nitiate*$_a$*uth.html*

tion allows access, among other things, to the access token, which was used in this case as the value of the request's Authorization header, intended to grant access to the API. Although, in theory, Postman allows both OAuth 2.0 and JWT Bearer authorization, these mechanisms do not work correctly, at least in the context of this API.

The API invokes two different lambda functions based on which route is invoked. Both of the routes require authentication, and both of the functions trigger distinct errors. With this test phase, it should be verified if:

- The access to the API is achieved exclusively by authorized users.
- The errors contemplated in the code are triggered correctly.

Beginning with the route that concerns the invocation of the function responsible for the retrieval of the Amazon DynamoDB tables' information, three errors could be raised from the code:

- Invalid structure.
- Read from table failed.
- No data.

These arise when a request that does not conform with the required by the function is passed, when the read from the database can not be done because there is no table with the name provided or the service is not available, and when the read of the table returns no data, respectively. Since the tables are populated, these tests should not raise the third error.

The tests performed are presented in Table 6.13, and all the results conform to the expected ones, proving the proper functioning of the API's route. The tests correspond to HTTP requests invoking the GET method with the header content presented in each test and with or without the usage of an authorized user to access the API, as expressed by the column Authorized user.

ID	Authorized user	Valid request	Request's header	Result
1	x	x	table_name: Routes	Routes list
2	x	x	table_name: Frequencies	Frequencies list
3		x	table_name: Routes	Unauthorized
4		x	table_name: Frequencies	Unauthorized
5	x		Empty request	Invalid structure.
6	x		table_name:	Read from Table failed.
7	x		tablename: Routes	Invalid structure.
8	x		table_name: Table	Read from Table failed.

Table 6.13: Results obtained by making several requests to the developed API.

Regarding the lambda function that invokes the NLP service and processes its result, three distinct errors can be triggered from the code:

- Invalid structure.
- Azure's service is not responding.
- There's no data to process.

These arise from passing a request without the needed content, the unavailability of the NLP service in use, and the non-identification of entities in the processed text resulting in an empty dataset for post-processing, respectively. Since the second type of error is associated with the availability of a service provided by a third party, it is impossible to test for its occurrence, and only the other types of errors are tested.

The tests presented in Table 6.14 were executed, having all of them the expected result and, therefore, attesting to the proper functioning of the API's route. In this table, the test prescription is "Prescrever Paracetamol 1000 mg, via oral, ao almoço, a começar hoje e durante 3 doses" and also, it should be noted that the lambda function is triggered by an event that should have a key prescription, to which a string value should be associated. The tests correspond to HTTP requests invoking the POST method with the body content presented in each test and with or without the usage of an authorized user to access the API, as expressed by the column Authorized user.

ID	Authorized user	Valid request	Request's body	Result
1	x	x	Test prescription	Correct processing
2		x	Test prescription	Unauthorized
3	x		Empty request	Invalid structure.
4	x		{}	Invalid structure.
5	x		"prescription": ""	Invalid structure.
6	x		"medication": "Paracetamol"	Invalid structure.
7	x		"prescription": "em"	There's no data to process.

Table 6.14: Results obtained by making several requests to the developed API.

Amazon Cognito

The user pools from Amazon Cognito were used as the authentication mechanism for the HTTP API. To do this, the following process was followed, based on [Rajamani et al., 2022]:

1. Creation of the user pool in the Amazon Cognito Console
2. Protection of the HTTP API by adding a JWT authorizer based on the previous user pool

First, the following steps were followed to create the user pool:

1. Access the Amazon Cognito console and selection of the "Create user pool" option
2. Selection of the user pool as a provider only
3. Definition of email login only
4. Definition of password policy with Cognito Standards mode
5. Definition of optional multi-factor authentication with MFA method authenticating applications
6. Enable account recovery with the delivery method for account recovery messages set to email only
7. Disable self-registration
8. Enable permission for Cognito to send messages for verification and confirmation automatically
9. Definition of the attributes to be verified as send email message, verify email address
10. Definition of maintenance of original attribute value when an update is pending and definition of active attribute values when an update is pending, like email address
11. Definition of email as a mandatory attribute
12. Configuration of the sending of emails with Cognito with an email FROM no-reply@verificationemail.com and without a REPLY-TO email
13. Assignment of the name prescriptions_Azure_EN_user_pool to the user pool
14. Disable the use of the Cognito hosted user interface
15. Setting the application client as a public client
16. Assignment of the name prescriptions_app to the application
17. Selection of the option not to generate a client secret
18. Review and creation of the user pool

Next, the generated user pool was added to the HTTP API throw the workflow:

1. Access to the HTTP API
2. Access to the Develop section, to the point related to Authorization
3. Edition of the selection of JWT as the authorizer type
4. Assignment of the name prescriptions_Azure_EN_authorizer

5. Definition of the identity string as $request.header.Authorization
6. Definition of the issuer URL based on https://cognito-idp.us-east-1.amazonaws.com/<your_userpool_id>
7. Addition of the prescriptions_app client ID (Cognito) in the box for the public
8. Creation of the authorizer and appending to the routes

6.4 Mobile application

The final goal of this internship was to have a system accessible through a mobile application that can deal with natural language to optimize internal prescription. To realize this application, a requirements survey was done to understand the market needs. Based on these requirements, the mockups of the solution were developed, being both the requirements and the mockups presented in the following subsections. Besides that, the user interface architecture, the programming language, and the tools are presented. Finally, the mobile application development is detailed.

6.4.1 Requirements

To meet the needs of the market for which the solution developed in this project is intended, a requirements survey was conducted. These requirements should be divided into functional and non-functional requirements, the former being listed in Table 6.15.

As	I want	So that
A doctor	to have access to the patient's current therapy	I understand the adjustments I should make to it
A doctor	to be able to decide whether to prescribe by voice or by writing on my cell phone	I can adapt the way I prescribe to the situation I am in
A doctor	to be able to re-record my prescription	it is possible to correct a prescription if I make a mistake in my dictation
A doctor	to see the transcript of my prescription in real-time	I can check the system's interpretation
A doctor	to check the final transcript of my prescription	I can validate it before submitting it for processing
A doctor	to be able to textually edit the transcript of my prescription before it is submitted for processing	I can correct any errors that may occur in transcription

A doctor	to be able to check the information processed by the system before submitting the result to M1	I can validate it
A doctor	to be able to change the information resulting from the processing done by the system	I can correct any errors arising from the same
A doctor	to be able to send the processed information to M1	it is presented in M1

Table 6.15: Functional requirements of the developed solution.

In turn, the non-functional requirements considered in the context of this project correspond to availability, reliability, security, performance, usability, maintainability, and dependability:

- Availability:
  - Availability refers to the system's ability to be available when its users request. It relates to the expected uptime and the expected downtime, its value being given by the expression $\frac{Expecteduptime}{Expecteduptime+Expecteddowntime}$. In turn, the downtime is related to the failure and the recovery time.
  - In the context of this project, the availability of the application is related to both the availability of the server it runs on and the systems it uses, i.e., Amazon Cognito, Microsoft Azure Cognitive Services for Speech, Microsoft Azure Text Analytics for Health, Amazon DynamoDB, AWS Lambda, and Amazon API Gateway.
    The development of the solution should ensure that its availability is as close to one as possible, and the use of backup systems may be considered to ensure higher availability of the system as a whole.
- Reliability:
  - In turn, reliability corresponds to the system's ability to perform its functions under the given conditions during a given period and is given as a probability.
  - This requirement is also intrinsically related to the system server and the system's solutions. The solution under development should have reliability as close as possible to 100%, and this should be evaluated by performing a set of tests simulating several scenarios of system use.
- Security:
  - Security, on the other hand, comprises three properties relating to confidentiality, integrity, and availability. Confidentiality relates to the absence of unauthorized disclosure of information, while integrity relates to the absence of improper system alteration. This requirement relates to malicious or intentional actions against the system, and systems can

  be compromised when there is a vulnerability, and an attack exploits that vulnerability.

  - The system must handle three types of confidential information: its users' credentials, the tokens granting access to its resources, and the access keys to Microsoft Azure services. Credentials are managed externally by Amazon Cognito, so this service should assure their confidentiality. In turn, the application should use tokens directly, and security mechanisms like tokens' periodic refresh should be used to reinforce their confidentiality. In addition to the periodic regeneration of tokens, brute force tests should be performed to ensure this information is not accessed improperly. Also, access keys to Microsoft Azure services should be available locally and in the cloud. Once their confidentiality is assured in the cloud, brute force tests should also guarantee their confidentiality in the application. Finally, to ensure the integrity of the system, tests should be made to verify the possible existence of vulnerabilities in it.

- Performance:

  - As far as performance is concerned, this is related to the system's operation speed and is measured using two metrics: response time and throughput. The response time defines how quickly the system reacts to user input, while the throughput establishes the number of operations the system can perform in a given period.
  - Both the response time of the application and its throughput are related to the server on which the system is running and the solutions it uses, in particular, the number of simultaneous accesses to them. To determine the system's performance, tests should be made of its operation under different conditions.

- Scalability:

  - Associated with performance, scalability defines the system's ability to deal with an increasing amount of work or its capacity to expand to accommodate this growth.
  - The system must be scalable since it will work with multiple clients that include various doctors and, so, multiple users. Being guaranteed concerning third-party solutions that the system uses, scalability should be measured at its associated server level to understand how it handles the increased workload.

- Usability:

  - On the other hand, usability corresponds to the ease of use and learning of the system and is composed of several attributes: learnability, efficiency, memorability, errors, and satisfaction. Learnability refers to the ease with which a user can accomplish basic tasks for the first time. Efficiency, on the other hand, defines how quickly users perform tasks after they have learned the system. Memorability, in turn, defines how easily a user regains proficiency in the system when they return to it after a period without using it. Also, errors concern both the quantity and severity of errors made by users and the ease with which they recover. Finally, satisfaction determines how enjoyable it is to use the system.

  - The application should be designed to maximize usability. Through a set of methods, the system should convey ease of use to everyone, even to some with less technological knowledge. To evaluate the components of usability, it should be necessary to use tests with end users.

- Maintainability:
  - Maintainability determines the ease with which the system can be changed to respond to various situations, such as correcting defects, installing updates, and adding new features resulting from new requirements.
  - The development of the system should take into account its maintainability. It should follow methodologies and be designed and documented to simplify its change for anyone qualified.

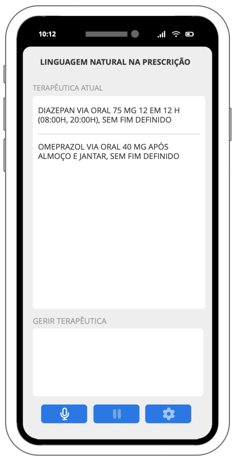
- Dependability:
  - Finally, dependability is about providing a reliable service to avoid unacceptably frequent or severe failures. This requirement involves several attributes that relate to availability, reliability, safety, confidentiality, integrity, and maintainability. Of these, only safety was not presented, which refers to the absence of catastrophic consequences for the user or the environment.
  - The development of the solution should aim to provide a reliable service that is as infallible as possible. Since the system is intended to act as an extension of M1, its failure should not jeopardize its associated task. Although more slowly, internal prescription can also be executed using the M1 or, if it fails, manually. In this way, no catastrophic consequences should be associated with the system's failure, such as inadequate health care provision.

6.4.2 Solution mockups

With the previously defined requirements, the mockups for the mobile application were designed. Figure 6.5 illustrates the mockups designed by the company.

Based on both the mockups developed by the company and the solution's requirements, the mockups present in Figure 6.6 were designed.

The application's first screen is the 6.6a. Here the user should be able to authenticate using Amazon Cognito to access the resources provided by the application. After successful authentication, the screen 6.6b is presented to the user, where they have access to the patient's current therapy and a therapy management section. The user can prescribe there by inserting its prescription in free-text


10:12

LINGUAGEM NATURAL NA PRESCRIÇÃO

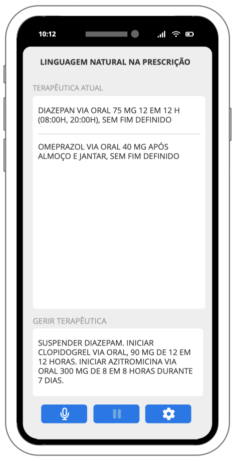
TERAPÊUTICA ATUAL

DIAZEPAN VIA ORAL 75 MG 12 EM 12 H (08:00H, 20:00H), SEM FIM DEFINIDO

OMEPRAZOL VIA ORAL 40 MG APÓS ALMOÇO E JANTAR, SEM FIM DEFINIDO

GERIR TERAPÊUTICA

(a)


10:12

LINGUAGEM NATURAL NA PRESCRIÇÃO

TERAPÊUTICA ATUAL

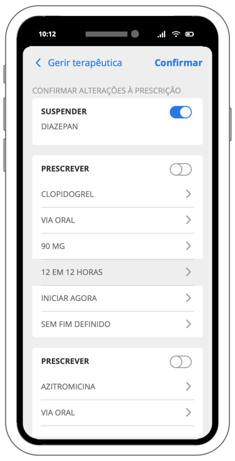
DIAZEPAN VIA ORAL 75 MG 12 EM 12 H (08:00H, 20:00H), SEM FIM DEFINIDO

OMEPRAZOL VIA ORAL 40 MG APÓS ALMOÇO E JANTAR, SEM FIM DEFINIDO

GERIR TERAPÊUTICA

SUSPENDER DIAZEPAM. INICIAR CLOPIDOGREL VIA ORAL, 90 MG DE 12 EM 12 HORAS. INICIAR AZITROMICINA VIA ORAL 300 MG DE 8 EM 8 HORAS DURANTE 7 DIAS.

(b)


10:12

Gerir terapêutica Confirmar

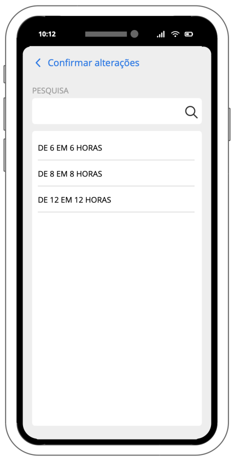
CONFIRMAR ALTERAÇÕES À PRESCRIÇÃO

SUSPENDER
DIAZEPAN

PRESCREVER
CLOPIDOGREL
VIA ORAL
90 MG
12 EM 12 HORAS
INICIAR AGORA
SEM FIM DEFINIDO

PRESCREVER
AZITROMICINA
VIA ORAL

(c)


10:12

Confirmar alterações

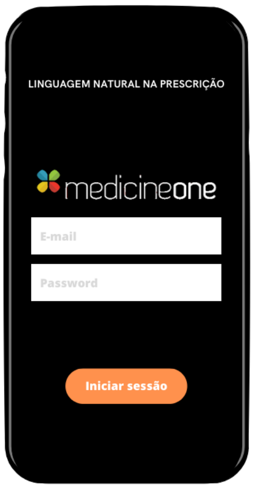
PESQUISA

DE 6 EM 6 HORAS
DE 8 EM 8 HORAS
DE 12 EM 12 HORAS

(d)

Figure 6.5: MedicineOne's mockups for the solution.

on the editable text field or through natural language by clicking the microphone button. When clicked, this button is replaced by the stop button illustrated in Figure 6.6c, which should allow the user to finish recording the prescription. The transcribed text should appear in the "Gerir terapêutica" section as the prescription is recorded, and this text can also be edited after the recording is finished. In addition, this screen allows re-recording and re-transcribing of the recording, with the icon of the record button being replaced by the one in 6.6d. When clicked, this button should generate the 6.6e popup, which, by informing the user that they will lose their current prescription, aims to confirm the intention to re-record the dictated prescription. This screen also displays a forward button, which should be responsible for the transition to the 6.6g. During this transition, the information contained in the prescription text should be processed, and the 6.6f popup will be shown until this processing is finished. After this, the user is redirected to the 6.6g screen, where the processing result is presented in a structured way. Suppose the user is satisfied with the result obtained. In that case, they should activate the toggle associated with the actions they want to include in the prescription and then use the "Confirmar" button to send the information to M1. Otherwise, the user should be able to click on the information that they want to edit, and the screen 6.6i, 6.6j, 6.6k, 6.6l, 6.6m and 6.6n will be displayed if the user wants to edit the INN, the dosage, the route of administration, the frequency, the beginning or the ending, respectively. The INN, the route of administration, and the frequency should be editable by selecting the desired option from the list of possible values, and the desired value can be searched using a search bar. As for the dosage should not be possible to edit its units, but the user can edit the numbers by an editable field. Finally, the start and end dates should be editable by selecting the desired dates in a calendar. That said, and returning to the 6.6g screen, the user can submit their prescription to the M1.


LINGUAGEM NATURAL NA PRESCRIÇÃO

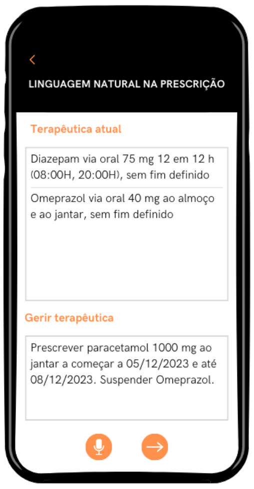
medicineone

E-mail

Password

Iniciar sessão

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

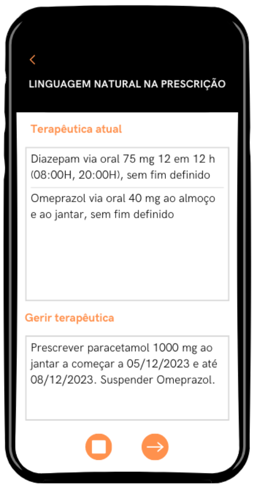
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Prescrever paracetamol 1000 mg ao jantar a começar a 05/12/2023 e até 08/12/2023. Suspender Omeprazol.

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

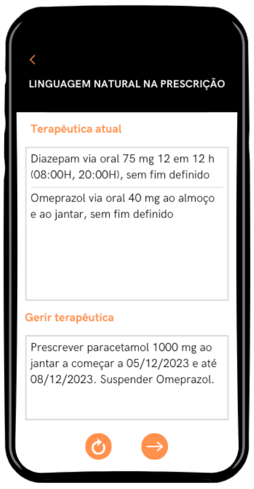
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Prescrever paracetamol 1000 mg ao jantar a começar a 05/12/2023 e até 08/12/2023. Suspender Omeprazol.

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

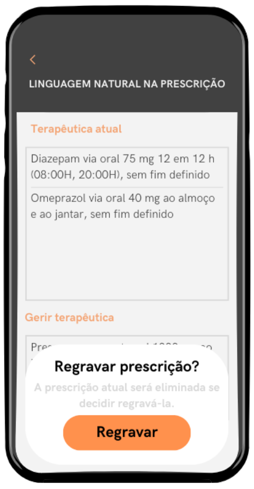
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Prescrever paracetamol 1000 mg ao jantar a começar a 05/12/2023 e até 08/12/2023. Suspender Omeprazol.

(d)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

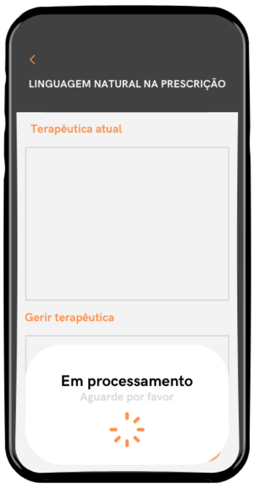
Gerir terapêutica

Regravar prescrição?

A prescrição atual será eliminada se decidir regravá-la.

Regravar

(e)


LINGUAGEM NATURAL NA PRESCRIÇÃO

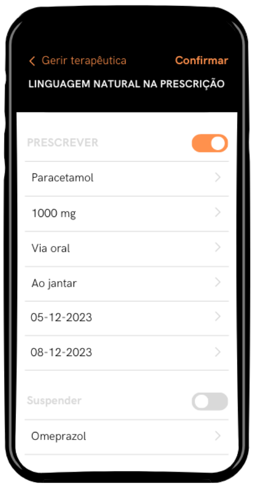
Terapêutica atual

Gerir terapêutica

Em processamento

Aguarde por favor

(f)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

PRESCREVER

Paracetamol

1000 mg

Via oral

Ao jantar

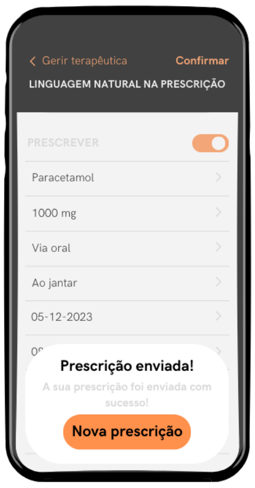
05-12-2023

08-12-2023

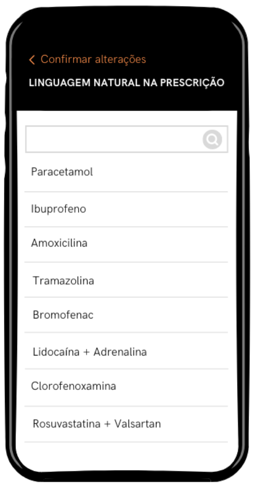
Suspender

Omeprazol

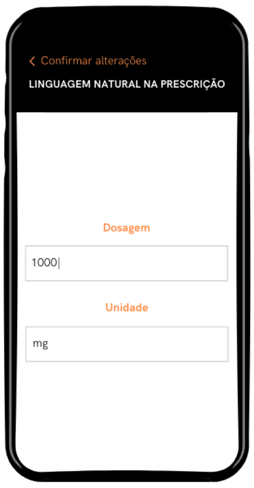
(g)



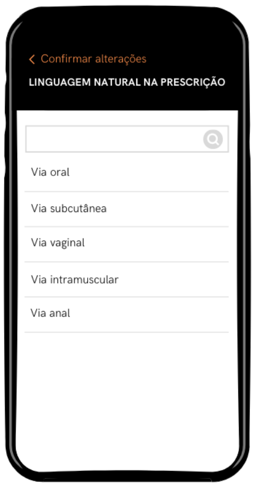
(h)



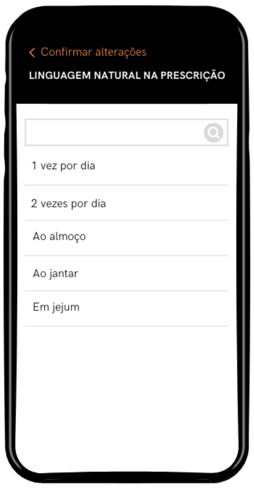
(i)



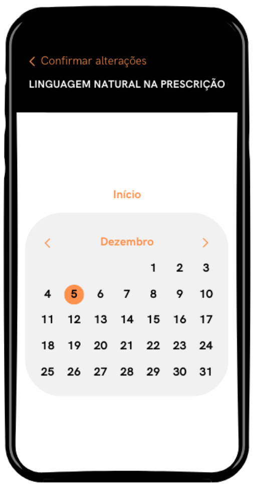
(j)



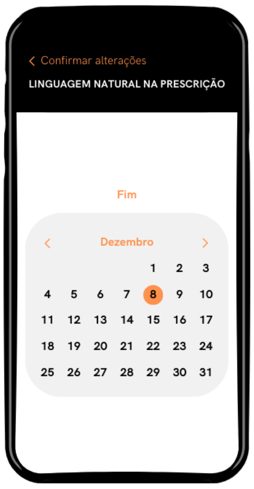
(k)



(l)



(m)



(n)

Figure 6.6: Solution's mockups.

6.4.3 User Interface Architecture

As stated in the previous chapter, the architecture followed corresponded to Model-View-ViewModel (MVVM) since it is the most suitable for the current project.

6.4.4 Programming Language

Since the mobile application was intended to be developed for iOS, the programming language used was Swift, a programming language for, among others, iOS. This programming language is modern, designed for safety, fast, powerful, and open-source [Swi].

6.4.5 Tools

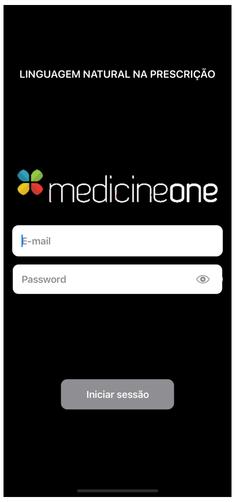
Regarding the tools used in the company, particularly in the development team, where this work is included:

- Work management tool: Jira
- IDE: Xcode

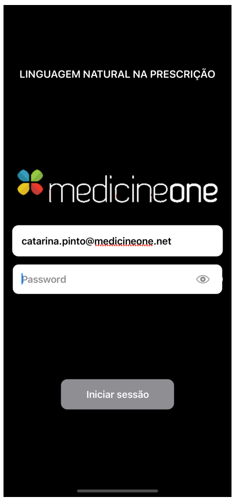
6.4.6 Development

That said, the mobile application was developed. Since the adopted design pattern concerns the MVVM, each screen of the application was developed independently, having its own ViewModel and, when necessary, its Model. Thus, the application development was divided between the Login, PrescriptionOrRecording, Processed, and Changes screens. Each screen is associated with a folder containing, for the Changes screen, the View, the ViewModel, and the Model; for the other screens, only the View and its particular ViewModel. Also, each screen is associated with its development in terms of the graphical interface and its functionalities.

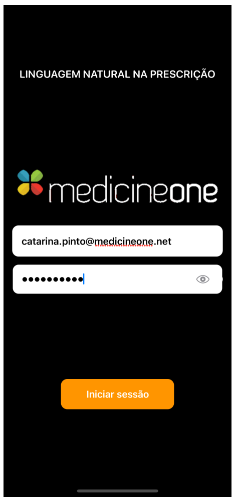
The Login screen, shown in Figure 6.7, allows the user to authenticate in the application. To send an authentication request to the server, the user must fill out both the field related to the email and the field associated with the password, being also possible for the user to verify the password. A request is sent to Amazon Cognito by clicking the login button so the user can sign in. This request is based on Swift's AWSMobileClientXCF package, based on the example shown at [wor, 2022]. A successful login is first associated with collecting the tokens assigned to this user and then passing to the next screen, the PrescriptionOrRecording screen. After the first login, the user will not have to repeat the process for about six months, being directly directed to the PrescriptionOrRecording screen. This is because the system automatically renews their access tokens as long as their refresh token is valid.



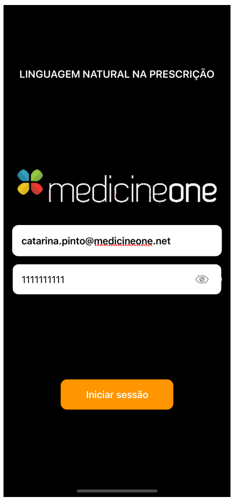
(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO
medicineone
catarina.pinto@medicineone.net
Password
Iniciar sessão

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO
medicineone
catarina.pinto@medicineone.net
••••••••••
Iniciar sessão

(c)

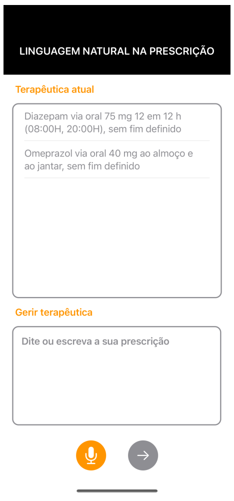

LINGUAGEM NATURAL NA PRESCRIÇÃO
medicineone
catarina.pinto@medicineone.net
1111111111
Iniciar sessão

(d)

Figure 6.7: Login screen.

The PrescriptionOrRecording screen, shown in Figure 6.8, was implemented as described in subsection 6.4.2. By pressing the record button, the user triggers the real-time transcription of Microsoft Azure Cognitive Services for Speech, which allows their dictation to be transcribed and appear on the screen while the doctor speaks. This required downloading and manually adding the Microsoft-CognitiveServicesSpeech package for Swift to the project, as indicated at [eric urban et al., 2022b]. For real-time recognition the recognizeFromMic example in [eric urban et al., 2022a] was followed. In turn, by clicking the follow button, the user triggers the processing of the prescription text, which is done by invoking the developed API, which in turn invokes the lambda function responsible for gathering Microsoft Azure Text Analytics for Health. While this processing occurs, the popup in the subfigure 6.8e is presented, which disappears when the processing stops, giving way to the Processed screen. The current therapy shown on the PrescriptionOrRecording screen has a default value, being information added or removed as the doctor prescribes. Also, it is relevant to highlight that when the therapeutic management order corresponds to changing the current therapeutic, this should be done by the cessation of that INN in the actual therapeutic, followed by its new prescription. This is done to conform with what is done in M1. This current therapy is only illustrative since the adequacy of the current therapy with each patient should be part of the integration with M1 and, so, not included in the scope of this internship.

The Processed screen, illustrated in Figure 6.9, was also implemented as stated in subsection 6.4.2, with some additional popups and an additional text field presenting the prescription made by the user so they always have context on it. Users can use the "Confirmar" button to simulate the prescription submission to M1 when satisfied with their final prescription. If the user does not select any action to include in the prescription, i.e., if they do not activate any toggle, the popup illustrated in 6.9b is displayed. This popup allows the user to return to the PrescriptionOrRecording screen if the "Ok" button is clicked and to dismiss the popup if the "Cancelar" button is clicked. If the user selects to suspend a INN


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

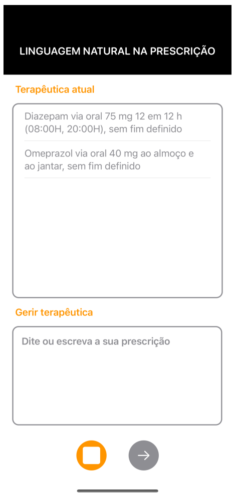
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

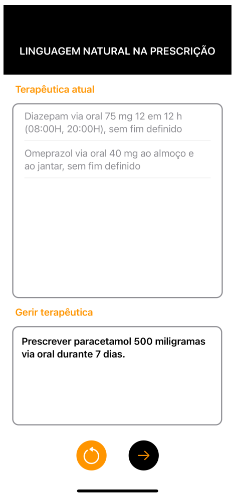
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

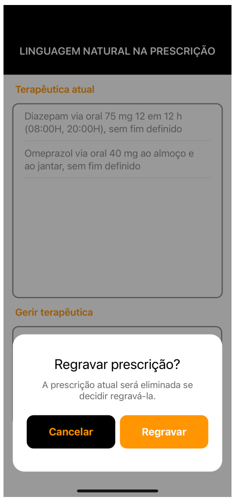
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Prescrever paracetamol 500 miligramas via oral durante 7 dias.

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

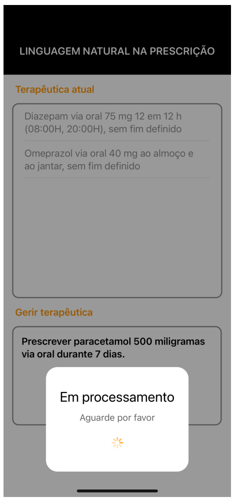
Gerir terapêutica

Regravar prescrição?

A prescrição atual será eliminada se decidir regravá-la.

Cancelar | Regravar

(d)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

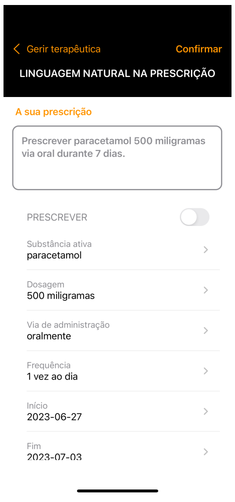
Prescrever paracetamol 500 miligramas via oral durante 7 dias.

Em processamento

Aguarde por favor

(e)

Figure 6.8: PrescriptionOrRecording screen.



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Prescrever paracetamol 500 miligramas via oral durante 7 dias.

PRESCREVER

Substância ativa: paracetamol

Dosagem: 500 miligramas

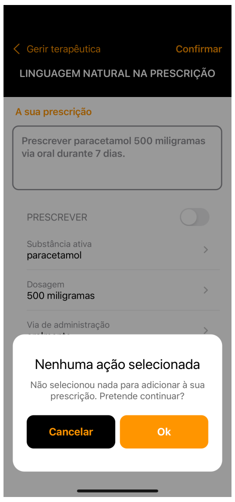
Via de administração: oralmente

Frequência: 1 vez ao dia

Início: 2023-06-27

Fim: 2023-07-03

(a)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Prescrever paracetamol 500 miligramas via oral durante 7 dias.

PRESCREVER

Substância ativa: paracetamol

Dosagem: 500 miligramas

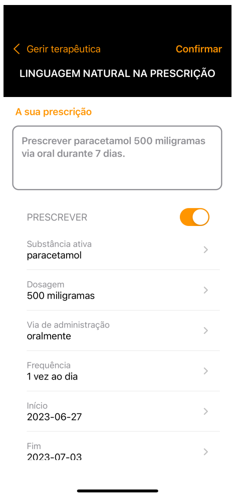
Via de administração

Nenhuma ação selecionada

Não selecionou nada para adicionar à sua prescrição. Pretende continuar?

Cancelar Ok

(b)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Prescrever paracetamol 500 miligramas via oral durante 7 dias.

PRESCREVER

Substância ativa: paracetamol

Dosagem: 500 miligramas

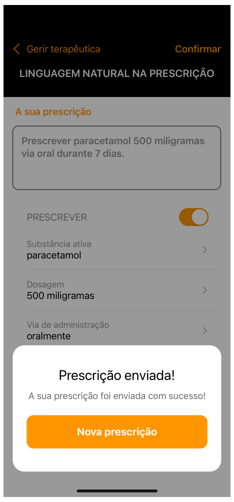
Via de administração: oralmente

Frequência: 1 vez ao dia

Início: 2023-06-27

Fim: 2023-07-03

(c)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Prescrever paracetamol 500 miligramas via oral durante 7 dias.

PRESCREVER

Substância ativa: paracetamol

Dosagem: 500 miligramas

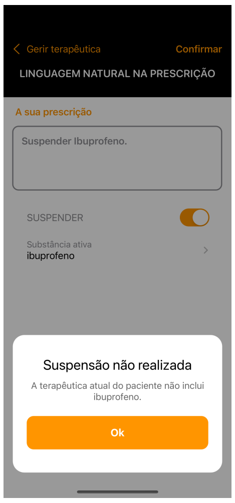
Via de administração: oralmente

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(d)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Ibuprofeno.

SUSPENDER

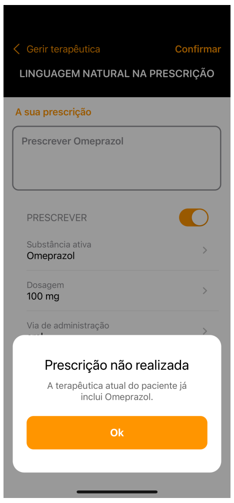
Substância ativa: ibuprofeno

Suspensão não realizada

A terapêutica atual do paciente não inclui ibuprofeno.

Ok

(e)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Prescrever Omeprazol

PRESCREVER

Substância ativa: Omeprazol

Dosagem: 100 mg

Via de administração

Prescrição não realizada

A terapêutica atual do paciente já inclui Omeprazol.

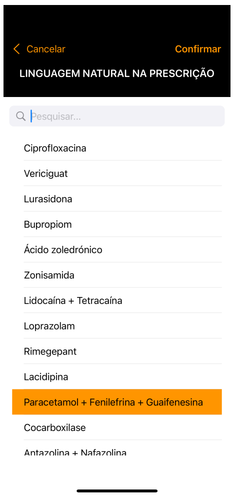
Ok

(f)

Figure 6.9: Processed screen.

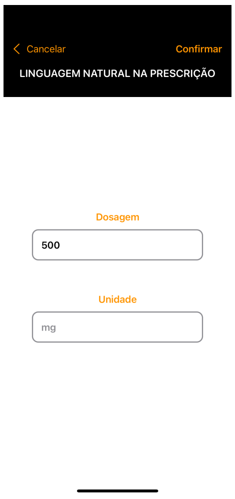
not contemplated in the current therapy, then the popup 6.9e is present. If the user tries to prescribe an INN already present in the current therapy, then the popup 6.9f is shown. Finally, if the user sends a valid prescription, the popup 6.9d is delivered. This popup allows the user to return to the PrescriptionOrRecording screen by tapping the "Nova prescrição" button. Note that the sending to M1 is only a simulation since the integration is not covered in the context of this internship. Besides that, it should be highlighted that the system chooses between a suspension or a prescription according to the verb used. A suspension is assumed if the verb corresponds to Suspender, Parar, Terminar, or Cessar. Otherwise, a prescription is assumed. The verbs used correspond to the most commonly used by doctors.

Finally, the Changes screen, illustrated in Figure 6.10, was implemented as introduced in subsection 6.4.2, with the addition of a "Cancelar" button to undo the changes.



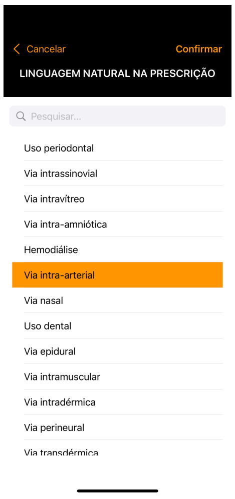
Ciprofloxacina
Vericiguat
Lurasidona
Bupropiom
Ácido zoledrónico
Zonisamida
Lidocaína + Tetracaína
Loprazolam
Rimegepant
Lacidipina
Paracetamol + Fenilefrina + Guaifenesina
Cocarboxilase
Antazolina + Nafazolina

(a)



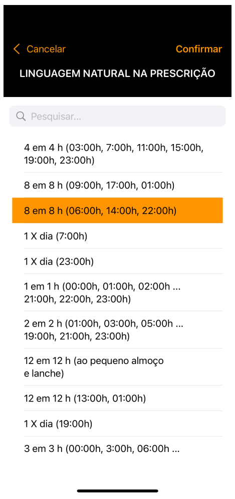
Dosagem
500
Unidade
mg

(b)



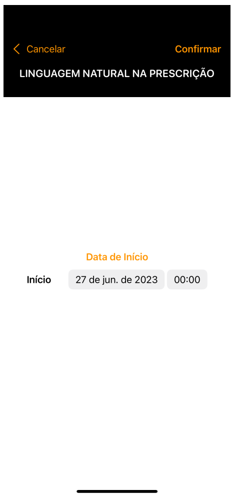
Uso periodontal
Via intrassinovial
Via intravítreo
Via intra-amniótica
Hemodiálise
Via intra-arterial
Via nasal
Uso dental
Via epidural
Via intramuscular
Via intradérmica
Via perineural
Via transdérmica

(c)



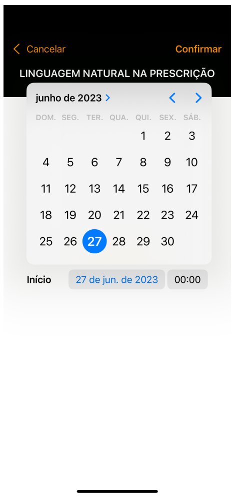
4 em 4 h (03:00h, 7:00h, 11:00h, 15:00h, 19:00h, 23:00h)
8 em 8 h (09:00h, 17:00h, 01:00h)
8 em 8 h (06:00h, 14:00h, 22:00h)
1 X dia (7:00h)
1 X dia (23:00h)
1 em 1 h (00:00h, 01:00h, 02:00h ... 21:00h, 22:00h, 23:00h)
2 em 2 h (01:00h, 03:00h, 05:00h ... 19:00h, 21:00h, 23:00h)
12 em 12 h (ao pequeno almoço e lanche)
12 em 12 h (13:00h, 01:00h)
1 X dia (19:00h)
3 em 3 h (00:00h, 3:00h, 06:00h ...

(d)



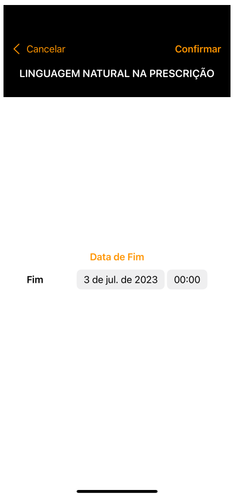
Data de Início
Início 27 de jun. de 2023 00:00

(e)



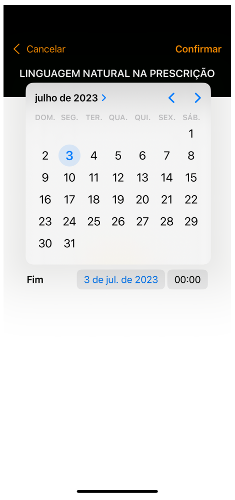
Início 27 de jun. de 2023 00:00

(f)



Data de Fim
Fim 3 de jul. de 2023 00:00

(g)



Fim 3 de jul. de 2023 00:00

(h)

Figure 6.10: Changes screen.

Chapter 7

The System

The present chapter presents the system in action, illustrating its use cases. First, a framework regarding its usage scenarios is given. Then, examples of these use cases are presented.

7.1 Introduction

The system developed can be used for three therapy management actions: prescribing, suspending, and changing. As said in the previous chapters, changing corresponds to a sequence in which a prescribing follows a suspension. Also, using this system can encompass one or more International Non-proprietary Name (INN) in each prescription, and the prescription's information can be partially or fully complete. The following sections illustrate the application workflow in multiple use cases and the times consumed in each.

7.2 Prescription

The prescription order can be used to prescribe one or more INN entities per prescription that can be partial or fully complete. Also, the orders can be dictated or introduced in free-text. Considering this, Table 7.1 summarizes the prescription orders made and the time needed, mapping to the illustrative figure for each use case, present in appendix H.

Prescription	**Input**	**Time consumed**	**Figure**
Iniciar Paracetamol	Free-text	27 seconds	H.1
Iniciar Paracetamol	Dictation	33 seconds	H.2
Iniciar Paracetamol. Fazer Amoxicilina + Ácido Clavulânico	Free-text	36 seconds	H.3
Iniciar Paracetamol. Fazer Amoxicilina + Ácido Clavulânico	Dictation	25 seconds	H.4

Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral	Free-text	53 seconds	H.5
Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral	Dictation	45 seconds	H.6
Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas.	Free-text	50 seconds	H.7
Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas.	Dictated	32 seconds	H.8
Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral de 8 em 8 horas a começar depois de amanhã e até dia 03/07/2023.	Free-text	1 minute and 39 seconds	H.9
Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral de 8 em 8 horas a começar depois de amanhã e até dia 03/07/2023.	Dictated	59 seconds	H.10

Table 7.1: Prescription orders and time consumed to do them.

It can be seen through the referred figures that the system needs additional information to be passed when more than one INN is prescribed so the system does not interpret the multiple INNs as a single one. The user can prescribe each INN individually if preferred. Also, the tests illustrated in figures H.8 and H.9 show examples of where editing the system's transcription is necessary. Besides that, dictation tends to be successively faster as the complexity and size of the prescription increases, which is in concordance with what the authors of [Mahatpure et al., 2019] state.

The previous tests did not encompass any change in the processed data. For comparison, the first test with the edition of all the information processed takes 1 minute and 5 seconds instead of the 27 seconds previously obtained.

7.3 Suspension

The suspension order can suspend one or more INN entities and can be dictated or introduced in free-text. Considering this, Table 7.2 summarizes the suspension orders made and the time needed, mapping to the illustrative figure for each use case, presented in appendix H.

Prescription	Input	Time consumed	Figure
Suspender Omeprazol.	Free-text	24 seconds	H.11
Suspender Omeprazol.	Dictation	24 seconds	H.12
Suspender Omeprazol. Suspender Diazepam	Free-text	30 seconds	H.13
Suspender Omeprazol. Suspender Diazepam	Dictation	41 seconds	H.14
Suspender Omeprazol via oral. Suspender Diazepam	Free-text	29 seconds	H.15
Suspender Omeprazol via oral. Suspender Diazepam	Dictation	44 seconds	H.16

Table 7.2: Suspension orders and time consumed to do them.

In the tests illustrated in figures H.14 and H.16, the Automatic Speech Recognition (ASR) solution was unable to identify the supposed dictation correctly. With that, these tests have a higher consumed time since they required the transcription edition before submission for processing. Also, similar to the prescription tests, these tests show the inability of the Natural Language Processing (NLP) service to distinguish two INN entities as independent when they appear with no more context. The suspending can be done by passing more information for context or, if preferred, suspending one INN entity at a time.

7.4 Changing

The changing order can be used through the suspension followed by prescription of the same INN. Equally, one prescription can include one or more changes, referring to more than one INN and with partial or fully completed prescription orders. Also, the orders can be dictated or introduced in free-text. Considering this, Table 7.3 summarizes the changing orders made and the time needed, mapping to the illustrative figure for each use case, presented in appendix H.

Prescription	Input	Time consumed	Figure
Suspender Omeprazol. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.	Free-text	54 seconds	H.17
Suspender Omeprazol. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.	Dictation	30 seconds	H.18
Suspender Omeprazol via oral. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.	Free-text	1 minute	H.19

Suspender Omeprazol via oral. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.	Dictation	35 seconds	H.20

Table 7.3: Changing orders and time consumed to do them.

Similarly to the previous studies, the system fails when no additional information is provided. Although, the tests proved that with little context, the system is already capable of performing well. If preferred by the user, the change can also be made with two distinct requests: one for suspension and the other for prescription.

7.5 Conclusion

The previous sections illustrate the use cases of the developed solution. Besides that, it reveals that the system not only enables the execution of internal prescription but also enables it to be done in a pratic and structured way, consuming low time. However, tests should still be made on a larger scale and with end users. It also shows some limitations of the system. These should be addressed by specifying how users should use the system. Furthermore, they are expected to be eradicated by improving the technology or adapting the system according to the results of the final tests.

Chapter 8

Conclusion and Future Work

The current chapter summarises the work presented in this document and gives a final reflection. Also, it shows the future work that should be done out of the scope of the internship.

First, the conclusions are presented. Then, the future work is illustrated.

8.1 Conclusion

The current project was developed to optimize internal prescription. This goal was associated with the need to provide the market with a very comfortable way for the user to interact with a prescription system, that is, by natural language dictation. With this, the great challenge was transforming information received in natural language into fully structured information. The existence of overlapping information compounds this difficulty and is related to the fact that International Non-proprietary Name (INN) entities that can be found in isolation are also found in compound INNs. A system such as the one developed in the current internship must distinguish between the two cases and make the correct association of the complementary information in each.

With that, a voice-based prescription system was developed. To date, and as far as we know, no similar system exists for European Portuguese. In tests, it was realized that the Automatic Speech Recognition (ASR) solution to be used in this system should be among those provided by Google and Microsoft Azure since they presented higher performance than the others. The discrepancy between the two solutions, responsible for the decision made, may be due to the greater difficulty presented by Google's solution in the tests made when dealing with poor quality audio or in which the speaker presents an accent. With that, the Microsoft solution, Microsoft Azure Cognitive Services for Speech, was chosen, having 88.02% performance on the tests. In turn, the tests made at the level of the Natural Language Processing (NLP) solution dictated that the one to be integrated into the developed system should be between the two offered by Microsoft Azure since the performance associated with them was superior to those associated with the other solutions. Those solutions presented really close entity detection performance values and differed by less than 1.50% in the macro F1 achieved. Beyond this, these solutions differed mainly in the need for post-processing and in the offer of the relationship detection feature, being that needed and offered by Microsoft Azure Text Analytics for Health. Since the detection of relationships is also relevant, no significant time is spent on post-processing, and the metrics obtained with Microsoft Azure Text Analytics for Health are only slightly lower than the ones achieved with Microsoft Azure Cognitive Service for Language, the former was chosen as the solution used in the current project. This solution presented, through post-processing, a performance of 86.33% for entity recognition and 83.60% for relation detection during the testing phase. The resultant system allows for a prescription to be made quickly that, for more straightforward prescriptions, may not even take up a minute of the doctors' working time. Since there is no available official information regarding model architectures for none of the referred solutions, no comparison could be made at this level to understand the technical reasons for the performance discrepancies.

The developed solution diverged from the state of the art tendencies regarding the prescription filling method. Table 8.1 compares the F1 score obtained by the state of the art systems and the current one at this level. This table makes it possible to conclude that the results obtained in the current project at the level of the NLP component agree with the state of the art, surpassing some of the results obtained by the state of the art systems.

Study	**Prescription filling F1**
[Shaikh et al., 2021]	96.00%
[Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] for doctors	75.00%
[Kocabiyikoglu et al., 2019] [Kocabiyikoglu et al., 2020] for naive users	43.00%
[Dhokley et al., 2021]	91.00%
Current study	86.33%

Table 8.1: Comparison of the results obtained with the state of the art results.

Although the developed system is efficient, it has some limitations associated with its NLP service. They are surmountable by standardization of the prescription form, which makes the system use extremely interesting in the hospital environment. Besides that, those limitations are expected to be surpassed as the system evolves. It will also be essential to recapitulate that integrating this system with M1 should give the doctor the necessary and secure framework to perform internal prescription.

It should be noted that the relationship detection feature was not directly used by now. Although, this feature remains basilar since it is understood that in the future, particularly with the improvement of the NLP system, this should be fundamental for simplifying the post-processing performed and minimizing the time consumed in the task.

Finally, the developed system meets the proposed objectives with performances higher than 85% both in ASR and NLP while still leaving room for improvement.

8.2 Future work

As stated before, final user tests should be made. With that, insights should be retrieved, and it should be understood the system improvements needed to be done. Besides that, the usage of the system should dictate the necessity or absence of backups systems. Also, a system for information retrieval should be developed to get information about how the system fails to improve it at the post-processing level. Finally, at the end of this internship, Microsoft Azure released Custom Text Analytics for Health [aahill and American-Dipper, 2023]. It should be studied and tested to determine whether it is advantageous over the default model and replace it if it is.

References

Features, a. URL https://www.ibm.com/cloud/watson-speech-to-text/features.

Documentation, a. URL https://www.nltk.org/.

Documentation, b. URL https://www.nltk.org/howto/portuguese_en.html.

Swift. URL https://developer.apple.com/swift/.

What is amazon api gateway?, a. URL https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html.

What is amazon dynamodb?, b. URL https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

What is aws lambda?, c. URL https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

Lambda runtimes, d. URL https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html.

Cloud computing with aws, e. URL https://aws.amazon.com/what-is-aws/?nc2=h_ql_le_int.

Aws architecture icons, f. URL https://aws.amazon.com/pt/architecture/icons/.

Mulher fraca na cama do hospital, a. URL https://pt.depositphotos.com/112554506/stock-photo-weak-woman-on-hospital-bed.html.

Mulher médica e paciente masculino no escritório do hospital, b. URL https://pt.depositphotos.com/282097856/stock-photo-woman-doctor-and-male-patient.html.

Facts figures, a. URL https://spacy.io/usage/facts-figures.

Portuguese, b. URL https://spacy.io/models/pt.

Models languages, c. URL https://spacy.io/usage/models.

Language processing pipelines, d. URL https://spacy.io/usage/processing-pipelines.

Linguistic features, e. URL https://spacy.io/usage/linguistic-features.

Training pipelines models, f. URL https://spacy.io/usage/training.

Rule-based matching, g. URL https://spacy.io/usage/rule-based-matching.

Embeddings, transformers and transfer learning, h. URL https://spacy.io/usage/embeddings-transformers.

Layers and model architectures, i. URL https://spacy.io/usage/layers-architectures.

Trained models pipelines, j. URL https://spacy.io/models.

Dependencyparser, k. URL https://spacy.io/api/dependencyparser.

URL https://www.zapsplat.com/music/ward-ambience-inside-an-american-hospital-airy-roo

2010. URL https://nlp.stanford.edu/software/CRF-NER.shtml.

Introduction of finite automata, Jul 2015. URL https://www.geeksforgeeks.org/introduction-of-finite-automata/.

Cmu sphinx files, 2019. URL https://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models/.

Feb 2019. URL https://www.geeksforgeeks.org/nlp-trigramsntags-tnt-tagging/.

2020. URL https://oauth.net/2/.

Oct 2020. URL https://www.geeksforgeeks.org/snowball-stemmer-nlp/.

Dec 2020. URL https://en.wikipedia.org/wiki/Triphone.

Android speech, 2021. URL https://learn.microsoft.com/en-us/xamarin/android/platform/speech.

Tone analytics (classifications), 2021. URL https://cloud.ibm.com/docs/natural-language-understanding?topic=natural-language-understanding-tone_analytics.

What is qna maker?, 2021. URL https://learn.microsoft.com/en-us/azure/cognitive-services/qnamaker/overview/overview.

Oct 2021. URL https://www.geeksforgeeks.org/monotonic-reasoning-vs-non-monotonic-reasoning/.

What is text analytics for health in azure cognitive service for language?, 2022. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/text-analytics-for-health/overview?tabs=ner.

About, 2022a. URL https://cloud.ibm.com/docs/natural-language-understanding?topic=natural-language-understanding-about.

Usage limits, 2022b. URL https://cloud.ibm.com/docs/natural-language-understanding?topic=natural-language-understanding-usage-limits.

Language support - portuguese, 2022c. URL https://cloud.ibm.com/docs/natural-language-understanding?topic=natural-language-understanding-language-support#portuguese.

About speech to text, 2022d. URL https://cloud.ibm.com/docs/speech-to-text?topic=speech-to-text-about#about-languages.

Aws launches region in the united arab emirates, 2022. URL https://ciospeak.com/industry/technology/aws-launches-region-in-the-united-arab-emirates/.

What is azure cognitive service for language?, 2022a. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/overview.

What is language understanding (luis)?, 2022b. URL https://learn.microsoft.com/en-us/azure/cognitive-services/luis/what-is-luis.

Custom named entity recognition service limits, 2022c. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/custom-named-entity-recognition/service-limits.

Supported named entity recognition (ner) entity categories, 2022d. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/named-entity-recognition/concepts/named-entity-categories.

Purchase commitment tier pricing, 2022a. URL https://learn.microsoft.com/pt-pt/azure/cognitive-services/commitment-tier.

Speech to text, 2022b. URL https://azure.microsoft.com/en-us/products/cognitive-services/speech-to-text/#features.

Language and voice support for the speech service, 2022c. URL https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/language-support?tabs=stt-tts.

Speech service quotas and limits, 2022d. URL https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-services-quotas-and-limits.

Speech services pricing, 2022e. URL https://azure.microsoft.com/en-us/pricing/details/cognitive-services/speech-services/.

Real-time speech to text, 2022f. URL https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-to-text#real-time-speech-to-text.

What is speech-to-text?, 2022g. URL https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/speech-to-text.

What is the speech service?, 2022h. URL https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/overview.

2022. URL https://cloud.google.com/endpoints/docs/openapi/openapi-overview.

A pill to ease the symptoms of mild-to-moderate covid-19: Is paxlovid right for you?, 2022. URL https://weillcornell.org/news/a-pill-to-ease-the-symptoms-of-mild-to-moderate-covid-19-is-paxlovid-right-for-you

2022. URL https://github.com/openai/whisper.

Oct 2022. URL https://docs.amplify.aws/sdk/auth/working-with-api/q/platform/ios/#signin.

Supported text analytics for health entity categories, 2023. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/text-analytics-for-health/concepts/health-entity-categories.

Jun 2023. URL https://cloud.google.com/natural-language/docs/reference/rest/v1/Entity#Type.

Jun 2023. URL https://cloud.google.com/healthcare-api/docs/concepts/nlp.

May 2023. URL https://play.google.com/store/apps/details?id=net.medicineone.handyandroid&hl=pt_PT&gl=US&pli=1.

aahill. Data limits for language service features - azure cognitive services, Jun 2023. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/concepts/data-limits.

aahill and American-Dipper. Custom text analytics for health - azure cognitive services, Apr 2023. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/custom-text-analytics-for-health/overview.

aahill, BaherAbdullah, and magrefaat. Quickstart - custom named entity recognition (ner) - azure cognitive services, Feb 2023. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/custom-named-entity-recognition/quickstart?pivots=language-studio.

Rawan Aljamal, Ali El-Mousa, and Fahed Jubair. A user perspective overview of the top infrastructure as a service and high performance computing cloud service providers, Apr 2019. URL https://ieeexplore.ieee.org/document/8717453?arnumber=8717453.

Inc. Amazon Web Services. Languages supported in amazon comprehend, a. URL https://docs.aws.amazon.com/comprehend/latest/dg/supported-languages.html.

Inc. Amazon Web Services. Guidelines and quotas, b. URL https://docs.aws.amazon.com/comprehend/latest/dg/guidelines-and-limits.html.

Inc. Amazon Web Services. What is amazon comprehend medical?, c. URL https://docs.aws.amazon.com/comprehend-medical/latest/dev/comprehendmedical-welcome.html.

Inc. Amazon Web Services. Guidelines and quotas, d. URL https://docs.aws.amazon.com/comprehend-medical/latest/dev/comprehendmedical-quotas.html.

Inc. Amazon Web Services. Pdf annotation files, e. URL https://docs.aws.amazon.com/comprehend/latest/dg/cer-annotation-manifest.html.

Inc. Amazon Web Services. Amazon comprehend pricing, f. URL https://aws.amazon.com/pt/comprehend/pricing/.

Inc. Amazon Web Services. What is amazon comprehend?, g. URL https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html.

Inc. Amazon Web Services. Guidelines and quotas, h.

Inc. Amazon Web Services. Analyzing call center audio with call analytics, 2022a. URL https://docs.aws.amazon.com/transcribe/latest/dg/call-analytics.html.

Inc. Amazon Web Services. Custom vocabularies, 2022b. URL https://docs.aws.amazon.com/transcribe/latest/dg/custom-vocabulary.html.

Inc. Amazon Web Services. Data input and output, 2022c. URL https://docs.aws.amazon.com/transcribe/latest/dg/how-input.html.

Inc. Amazon Web Services. Supported languages and language-specific features, 2022d. URL https://docs.aws.amazon.com/transcribe/latest/dg/supported-languages.html.

Inc. Amazon Web Services. Amazon transcribe pricing, 2022e. URL https://aws.amazon.com/pt/transcribe/pricing/.

Inc. Amazon Web Services. Redacting or identifying personally identifiable information, 2022f. URL https://docs.aws.amazon.com/transcribe/latest/dg/pii-redaction.html.

Inc. Amazon Web Services. What is amazon transcribe?, 2022g. URL https://docs.aws.amazon.com/pt_br/transcribe/latest/dg/what-is.html.

Inc. Amazon Web Services. Building lambda functions with python - aws lambda, 2023a. URL https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html.

Inc. Amazon Web Services. Step 1: Create a table - amazon dynamodb, 2023b. URL https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/getting-started-step-1.html.

Inc. Amazon Web Servoces. Amazon transcribe medical, 2022. URL `https://docs.aws.amazon.com/transcribe/latest/dg/transcribe-medical.html`.

Abraham Bookstein, Vladimir A. Kulyukin, and Timo Raita. Generalized hamming distance. *Information Retrieval*, 5(4):353–375, 2002. doi: https://doi.org/10.1023/a:1020499411651.

Jonny Brooks-Bartlett. Probability concepts explained: Maximum likelihood estimation, Jan 2018. URL `https://towardsdatascience.com/probability-concepts-explained-maximum-likelihood-estimation-c7b4342fdbb1`.

Michael Buckbee. What is saml and how does it work?, Jun 2022. URL `https://www.varonis.com/blog/what-is-saml`.

Oscar Contreras Carrasco. Gaussian mixture models explained, Feb 2020. URL `https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95`.

K. R. Chowdhary. *Chaper 19 Natural Language Processing, Section 19.4 Components of Natural Language Processing*, page 609–612. Springer Nature India, 2020a.

K. R. Chowdhary. *Chaper 19 Natural Language Processing, 19.13 Summary*, page 609–612. Springer Nature India, 2020b.

Google Cloud. Speech-to-text, a. URL `https://cloud.google.com/speech-to-text`.

Google Cloud. Put speech-to-text into action, b. URL `https://cloud.google.com/speech-to-text/#section-2`.

Google Cloud. Speech-to-text (all features), c. URL `https://cloud.google.com/speech-to-text#section-11`.

Google Cloud. Speech-to-text pricing, d. URL `https://cloud.google.com/speech-to-text/pricing`.

Google Cloud. Natural language ai, e. URL `https://cloud.google.com/natural-language#section-6`.

Google Cloud. Using the healthcare natural language api, f. URL `https://cloud.google.com/healthcare-api/docs/how-tos/nlp`.

Google Cloud. Healthcare natural language api, g. URL `https://cloud.google.com/healthcare-api/docs/concepts/nlp`.

Google Cloud. Training entity extraction models for healthcare, h. URL `https://cloud.google.com/natural-language/automl/docs/automl-healthcare`.

Google Cloud. Natural language ai, i. URL `https://cloud.google.com/natural-language`.

Google Cloud. Natural language api basics, j. URL `https://cloud.google.com/natural-language/docs/basics`.

Google Cloud. Language support, k. URL https://cloud.google.com/natural-language/docs/languages.

Google Cloud. Language support, l. URL https://cloud.google.com/natural-language/automl/docs/languages.

Google Cloud. Quotas limits, m. URL https://cloud.google.com/natural-language/quotas.

Google Cloud. Quotas limits, n. URL https://cloud.google.com/natural-language/automl/quotas.

Google Cloud. Cloud natural language pricing, o. URL https://cloud.google.com/natural-language/pricing.

Google Cloud. Cloud healthcare api pricing - healthcare natural language api, p. URL https://cloud.google.com/healthcare-api/pricing#healthcare-natural-language-api.

Google Cloud. Vertex ai pricing - data labeling, q. URL https://cloud.google.com/vertex-ai/pricing#labeling.

Google Cloud. Speech-to-text request construction, 2022a. URL https://cloud.google.com/speech-to-text/docs/basics.

Google Cloud. Speech-to-text supported languages, 2022b. URL https://cloud.google.com/speech-to-text/docs/languages.

Google Cloud. Data logging, 2022c. URL https://cloud.google.com/speech-to-text/docs/data-logging.

Google Cloud. Enable the profanity filter, 2022d. URL https://cloud.google.com/speech-to-text/docs/profanity-filter.

Google Cloud. Enable spoken punctuation and spoken emojis, 2022e. URL https://cloud.google.com/speech-to-text/docs/spoken-punctuation-emojis.

Google Cloud. Enable word-level confidence, 2022f. URL https://cloud.google.com/speech-to-text/docs/word-confidence.

CMUSphinx. Frequently asked questions (faq) (q: What speech feature type does cmusphinx use and what do they represent), a. URL https://cmusphinx.github.io/wiki/faq/#q-what-do-cmn-values-in-pocketsphinx-output-represent.

CMUSphinx. Training an acoustic model for cmusphinx, b. URL https://cmusphinx.github.io/wiki/tutorialam/.

CMUSphinx. Segmentation and diarization using lium tools, c. URL https://cmusphinx.github.io/wiki/speakerdiarization/.

CMUSphinx. Building a language model, d. URL https://cmusphinx.github.io/wiki/tutoriallm/.

CMUSphinx. Overview of the cmusphinx toolkit, e. URL https://cmusphinx.github.io/wiki/tutorialoverview/.

CMUSphinx. Before you start, f. URL https://cmusphinx.github.io/wiki/tutorialbeforestart/.

Daniël De Kok. Neural edit-tree lemmatization for spacy · explosion, Nov 2021. URL https://explosion.ai/blog/edit-tree-lemmatizer.

Waheeda Dhokley, Atif Khan, Danish Ali Shaikh, Amaan Shaikh, and Burhanuddin Fatehi. Speech recognition based prescription generator. *SSRN Electronic Journal*, 2021. doi: https://doi.org/10.2139/ssrn.3867738.

Claire Drumond. Scrum. a guide to scrum: what it is and how it works. URL https://www.atlassian.com/agile/scrum.

eric urban, DavidCBerry13, PatrickFarley, mrbullwinkle, v jaswel, v demjoh, and diberry. Speech to text quickstart - speech service - azure cognitive services, Sep 2022a. URL https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started-speech-to-text?tabs=windows%2Cterminal&pivots=programming-language-swift.

eric urban, DavidCBerry13, mrbullwinkle PatrickFarley, v demjoh, codemillmatt, diberry, wadepickett, kraigb, brandom msft, IEvangelist, rhurey, glecaros, erhopf, and robch. Install the speech sdk - azure cognitive services, Sep 2022b. URL https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/quickstarts/setup-platform?tabs=windows%2Cubuntu%2Cdotnet%2Cjre%2Cmaven%2Cnodejs%2Cios%2Cpypi&pivots=programming-language-swift.

Maël Fabien. Hmm acoustic modeling, Jun 2020. URL https://maelfabien.github.io/machinelearning/speech_reco_1/#introduction-to-hmm-gmm-acoustic-modeling.

Centers for Disease Control and Prevention.

The GraphQL Foundation. Graphql: A query language for apis., 2012. URL https://graphql.org/.

M.J.F. Gales and P.C. Woodland. Mean and variance adaptation within the mllr framework. *Computer Speech Language*, 10(4):249–264, Oct 1996. doi: https://doi.org/10.1006/csla.1996.0013.

Sanchit Gandhi. Fine-tune whisper for multilingual asr with transformers, 2022. URL https://huggingface.co/blog/fine-tune-whisper.

Michael Gilleland. Levenshtein distance, 2020. URL https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm.

Christine Groeneboer. *TABLEAU-BASED THEOREM PROVING FOR A CONDITIONAJA LOGIC*. 1987. URL https://core.ac.uk/download/pdf/56369621.pdf.

Halfpoint. Senior doctor with smartphone working at the office, 2018. URL https://www.istockphoto.com/pt/foto/senior-doctor-with-smartphone-working-at-the-office-gm924090614-253626227.

SuHun Han. Googletrans: Free and unlimited google translate api for python — googletrans 3.0.0 documentation, 2020. URL https://py-googletrans.readthedocs.io/en/latest/.

Hien.Ha. Stanford ner - misc entity?, 2021. URL https://stackoverflow.com/questions/38062203/stanford-ner-misc-entity.

Matthew Honnibal. A good part-of-speech tagger in about 200 lines of python · explosion, Sep 2013. URL https://explosion.ai/blog/part-of-speech-pos-tagger-in-python.

Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing, Sep 2015. URL https://aclanthology.org/D15-1162/.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spacy: Industrial-strength natural language processing in python, 2020. URL https://github.com/explosion/spaCy/blob/master/spacy/ml/models/tagger.py.

https://www.facebook.com/taufiQue74. Text segmentation - approaches, datasets, and metrics, Nov 2021. URL https://www.assemblyai.com/blog/text-segmentation-approaches-datasets-and-evaluation-metrics/.

David Huggins-Daines and Alexander Solovets. Pocketsphinx 5.0.0, 2022a. URL https://github.com/cmusphinx/pocketsphinx/blob/master/README.md.

David Huggins-Daines and Alexander Solovets. Pocketsphinx 5.0.0, 2022b. URL https://github.com/cmusphinx/pocketsphinx.

David Huggins-Daines and Alexander Solovets. Pocketsphinx 5.0.0, 2022c. URL https://github.com/cmusphinx/sphinxtrain.

David Huggins-Daines and Alexander Solovets, 2022d. URL https://github.com/cmusphinx/sphinxtrain/blob/master/python/cmusphinx/gmm.py.

David Huggins-Daines and Alexander Solovets, 2022e. URL https://github.com/cmusphinx/sphinxtrain/blob/master/python/cmusphinx/hmm.py.

David Huggins-Daines and Alexander Solovets, 2022f. URL https://github.com/cmusphinx/sphinxtrain/blob/master/python/cmusphinx/mllr.py.

David Huggins-Daines and Alexander Solovets, 2022g. URL https://github.com/cmusphinx/sphinxtrain/tree/master/python/cmusphinx/feat.

David Huggins-Daines and Alexander Solovets, 2022h. URL https://github.com/cmusphinx/sphinxtrain/tree/master/python/cmusphinx.

Bogdan Iancu. Evaluating google speech-to-text api's performance for romanian e-learning resources. *Informatica Economica*, 23(1):17–25, 2019. doi: 10.12948/issn14531305/23.1.2019.02. URL https://ideas.repec.org/a/aes/infoec/v23y2019i1p17-25.html.

IBM. Pricing, b. URL https://www.ibm.com/cloud/watson-natural-language-understanding/pricing.

IBM. Watson natural language understanding: Pricing, c. URL https://www.ibm.com/uk-en/cloud/watson-natural-language-understanding/pricing.

Kaustumbh Jaiswal. Custom named entity recognition using spacy, 2019. URL https://towardsdatascience.com/custom-named-entity-recognition-using-spacy-7140ebbb3718.

Shivanee Jaiswal. Natural language processing — dependency parsing, Aug 2021. URL https://towardsdatascience.com/natural-language-processing-dependency-parsing-cf094bbbe3f7.

yelsehrawy BaherAbdullah DavidCBerry13 jboback, aahill. Quickstart: Use the text analytics for health rest api and client library - azure cognitive services, Feb 2023. URL https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/text-analytics-for-health/quickstart?pivots=programming-language-python.

Dasaprakash K. Entity extraction and classification using spacy. URL https://www.kaggle.com/code/curiousprogrammer/entity-extraction-and-classification-using-spacy.

Kaldi. About the kaldi project, a. URL https://kaldi-asr.org/doc/about.html.

Kaldi. Models, b. URL https://kaldi-asr.org/models.html.

Kaldi. Kaldi tools, c. URL https://kaldi-asr.org/doc/tools.html.

Uday Kamath, John Liu, and James Whitaker. *Chapter 3 Text and Speech Basics, Section 3.1 Introduction*, page 87–90. Springer Nature Switzerland AG, 2019a.

Uday Kamath, John Liu, and James Whitaker. *Chapter 3 Text and Speech Basics, Section 3.13 Automated Speech Recognition*, pages 120–122. Springer Nature Switzerland AG, 2019b.

Uday Kamath, John Liu, and James Whitaker. *Chapter 1 Introduction, Section 1.2 History*, page 87–90. Springer Nature Switzerland AG, 2019c.

Uday Kamath, John Liu, and James Whitaker. *Chapter 1 Introduction, 1.3 Tools, Libraries, Datasets, and Resources for the Practitioners*, page 87–90. Springer Nature Switzerland AG, 2019d.

Amirhossein Kazemnejad. Transformer architecture: The positional encoding, Sep 2019. URL https://kazemnejad.com/blog/transformer_architecture_positional_encoding/.

Ali Can Kocabiyikoglu, François Portet, Hervé Blanchon, and Jean-Marc Babouchkine. Towards spoken medical prescription understanding. *Archives-ouvertes.fr*, Oct 2019. doi: https://hal.archives-ouvertes.fr/hal-02317503. URL `https://hal.archives-ouvertes.fr/hal-02317503`.

Ali Can Kocabiyikoglu, François Portet, Jean-Marc Babouchkine, and Hervé Blanchon. *Spoken Medical Prescription Acquisition Through a Dialogue System on Smartphone: Perspective of a Healthcare Software Company*. May 2020. URL `http://www.lrec-conf.org/proceedings/lrec2020/industrytrack/pdf/2020.lrec2020industrytrack-1.1.pdf`.

Grzegorz Kondrak. *Algorithms for Language Reconstruction*. 2002. URL `http://webdocs.cs.ualberta.ca/~kondrak/papers/thesisnew.pdf`.

Anupriya Koneru, Nerella Bala Naga Sai Rajani Bhavani, K. Purushottama Rao, Garikipati Sai Prakash, Immadisetty Pavan Kumar, and Velimala Venkat Kumar. Sentiment analysis on top five cloud service providers in the market. *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, May 2018. doi: 10.1109/icoei.2018.8553970.

Raymond Kwok. Viterbi algorithm for prediction with hmm — part 3 of the hmm series, Dec 2019. URL `https://medium.com/analytics-vidhya/viterbi-algorithm-for-prediction-with-hmm-part-3-of-the-hmm-series-6466ce2f5dc`

Video Puppet Limited. Text to speech portuguese. URL `https://www.narakeet.com/languages/text-to-speech-portuguese/`.

Edward Loper, Ewan Klein, and Steven Bird. O'Reilly Media Inc., 2009.

Maulik C. Madhavi and Hemant A. Patil. Vocal tract length normalization using a gaussian mixture model framework for query-by-example spoken term detection. *Computer Speech Language*, 58:175–202, Nov 2019. doi: https://doi.org/10.1016/j.csl.2019.03.005.

Jitendra Mahatpure, Mahesh Motwani, and Kumar Shukla. An electronic prescription system powered by speech recognition, natural language processing and blockchain technology. In *INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH*, volume 8, page 1454–1462, Aug 2019. URL `https://www.ijstr.org/final-print/aug2019/An-Electronic-Prescription-System-Powered-By-Speech-Recognition-Natural-Langua pdf`.

Microsoft. Azure cognitive services, a. URL `https://azure.microsoft.com/en-us/products/cognitive-services/#api`.

Microsoft. Cognitive service for language pricing, b. URL `https://azure.microsoft.com/en-us/pricing/details/cognitive-services/language-service/`.

Microsoft. Cognitive services—apis for ai solutions | microsoft azure, 2023. URL `https://azure.microsoft.com/en-au/products/cognitive-services/`.

Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech Language*, 16(1):69–88, Jan 2002. doi: https://doi.org/10.1006/csla.2001.0184.

Mozilla. Cross-origin resource sharing (cors), 2023. URL `https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS`.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. *Efficient Higher-Order CRFs for Morphological Tagging*. 2013. URL `https://aclanthology.org/D13-1032.pdf`.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. *Joint Lemmatization and Morphological Tagging with LEMMING*. 2015. URL `https://aclanthology.org/D15-1272.pdf`.

Joakim Nivre and Jens Nilsson. Pseudo-projective dependency parsing, Jun 2005. URL `https://aclanthology.org/P05-1013/`.

Naoaki Okazaki. chokkan/crfsuite, May 2023. URL `https://github.com/chokkan/crfsuite`.

Chris D. Paice. Another stemmer. *ACM SIGIR Forum*, 24(3):56–61, Nov 1990. doi: https://doi.org/10.1145/101306.101310.

M. F. Porter. An algorithm for suffix stipping. URL `https://www.cs.toronto.edu/~frank/csc2501/Readings/R2_Porter/Porter-1980.pdf`.

Konstantinos Poulinakis. Gelu, the relu successor? gaussian error linear unit explained, Dec 2022. URL `https://pub.towardsai.net/is-gelu-the-relu-successor-deep-learning-activations-7506cf96724f`.

Daniel Povey, Lukáš Burget, Mohit Agarwal, Pinar Akyazi, Feng Kai, Arnab Ghoshal, Ondřej Glembek, Nagendra Goel, Martin Karafiát, Ariya Rastrow, Richard C. Rose, Petr Schwarz, and Samuel Thomas. The subspace gaussian mixture model—a structured model for speech recognition. *Computer Speech Language*, 25(2):404–439, Apr 2011a. doi: https://doi.org/10.1016/j.csl.2010.06.003. URL `https://www.sciencedirect.com/science/article/pii/S088523081000063X`.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, Jan Silovský, Georg Stemmer, and Karel Veselý. *The Kaldi Speech Recognition Toolkit*. Jan 2011b. URL `https://www.danielpovey.com/files/2011_asru_kaldi.pdf`.

D Prescher, R Scha, K Sima', and A Zollmann. *What Are Treebank Grammars?* URL `https://www.cs.cmu.edu/~zollmann/publications/TreebankGrammars.pdf`.

pressfoto. Mãos da médica irreconhecível, escrevendo no formulário e digitando no teclado do laptop. URL `https://br.freepik.com/fotos-gratis/maos-da-medica-irreconhecivel-escrevendo-no-formulario-e-digitando-no-teclado-do-la`
`5839269.htm#query=medico%20computador&position=0&from_view=keyword`.

Alec Radford, Jong Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever. *Robust Speech Recognition via Large-Scale Weak Supervision*. URL `https://cdn.openai.com/papers/whisper.pdf`.

Alec Radford, Jong Wook Kim, Christine McLeavey Payne, Pamela Mishkin, Tao Xu, Greg Brockman, and Ilya Sutskever. Introducing whisper, 2022. URL `https://openai.com/blog/whisper/`.

Dan Radigan. Product backlog grooming, 2019. URL `https://www.atlassian.com/agile/scrum/backlogs`.

Siva Rajamani, Rajat Mathur, and Sudhanshu Malhotra. How to secure api gateway http endpoints with jwt authorizer | aws security blog, Feb 2022. URL `https://aws.amazon.com/blogs/security/how-to-secure-api-gateway-http-endpoints-with-jwt-authorizer/`.

Sebastian Raschka. Understanding and coding the self-attention mechanism of large language models from scratch, Feb 2023. URL `https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html`.

Max Rehkopf. Sprints | atlassian, 2019. URL `https://www.atlassian.com/agile/scrum/sprints`.

Leland Roberts. Understanding the mel spectrogram, Mar 2020. URL `https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53`.

Rahul Roy. Ml | stochastic gradient descent (sgd), Feb 2019. URL `https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/`.

Krunal Shah. Architecture presentation patterns: Mvc vs mvp vs mvvm, 2020. URL `https://www.thirdrocktechkno.com/blog/architecture-presentation-patterns-mvc-vs-mvp-vs-mvvm/`.

Sara Shaikh, Yousuf Farooqui, Ankita Borade, and Ahmedullah Syed. A voice-based prescription generating system. *International Journal of Advances in Engineering and Management (IJAEM)*, 3:261, 2021. doi: 10.35629/5252-0308261265. URL `https://ijaem.net/issue_dcp/A%20voice%20based%20prescription%20generating%20system.pdf`.

Kazem Taghva, Rania Elkhoury, and Jeffrey Coombs. *Arabic Stemming Without A Root Dictionary*. URL `http://jeffcoombs.com/isri/Taghva2005b.pdf`.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network*. URL `https://nlp.stanford.edu/~manning/papers/tagging.pdf`.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. *Stochastic Gradient Descent Training for L1-regularized Log-linear Models with Cumulative Penalty*. 2009. URL `https://aclanthology.org/P09-1054.pdf`.

DataEng Uncomplicated. How to install pandas on aws lambda function, Sep 2021. URL `https://www.youtube.com/watch?v=1UDEp90S9h8`.

Amey Varangaonkar. What is interactive machine learning?, Jul 2018. URL `https://hub.packtpub.com/what-is-interactive-machine-learning/`.

Pascal Vizeli. pycognito, Jun 2023a. URL `https://github.com/pvizeli/pycognito/blob/master/pycognito/__init__.py`.

Pascal Vizeli. pycognito, Jun 2023b. URL `https://github.com/pvizeli/pycognito#authenticate`.

Lori Wallach. Trade secrets. *Foreign Policy*, (140):70, Jan 2004. doi: https://doi.org/10.2307/4147524.

Rafal Wilinski. Dynamodb python boto3 query cheat sheet [14 examples], Feb 2020. URL `https://dynobase.dev/dynamodb-python-with-boto3/`.

Masashi Yoshikawa, Koji Mineshima, Hiroshi Noji, and Daisuke Bekki. Consistent ccg parsing over multiple sentences for improved logical reasoning, Jun 2018. URL `https://aclanthology.org/N18-2065/`.

Yue Zhang and Stephen Clark. *A Tale of Two Parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search*. Oct 2008. URL `https://aclanthology.org/D08-1059.pdf`.

Appendices

Appendix A

CMUSphinx components' architecture

Although there's no documentation regarding the model architecture of the components provided by this tool, it can be seen through [Huggins-Daines and Solovets, 2022b] and [Huggins-Daines and Solovets, 2022c], the source code of the solution. Regarding the models provided by Pocketshpinx, only a US-English model can be found. Although, as said before, this solution allows the generation of language models. Without some documentation, which type of models are creaTable with PocketSphinx cannot be inferred. Sphinxtrain component gives a little documentation inside code files, namely in [Huggins-Daines and Solovets, 2022h] it can be seen that this component includes distinct Machine Learning (ML) models used for ASR. Those include:

- Gaussian Mixture Model (GMM)s

  Spinxtrain allows its users to train these models throw [Huggins-Daines and Solovets, 2022d]. For that, the user must have audio data to feed the model. These models can be used both for speaker identification and for Vocal Tract Length Normalization (VTLN) which corresponds to the compensation of the spectral variation associated with vocal tract length [Madhavi and Patil, 2019].

  Gaussian Mixture is a soft clustering algorithm in which the Gaussian Mixture corresponds to a function composed of several Gaussian. For each cluster in the dataset, there is a Gaussian in the function, being the parameters of these Gaussian optimized using maximum likelihood so it is ensured that each of them fits all the points that belong to the cluster they represent.

  The maximum likelihood estimation method aims to determine the values of the parameters of a given model, which are selected so that the likelihood of the process described by the model having produced the observed data is maximized [Brooks-Bartlett, 2018].

  With this kind of model, it is possible to extract a probability of how much a data point is associated with a cluster, being that calculated throw the expression present in Figure A.1.

$$p(\mathbf{x}_n) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \mu_k, \Sigma_k)$$

Figure A.1: GMMs formula.

Here, the π parameter corresponds to a mixing probability that defines how big the Gaussian k should be, and the function it multiplies to corresponds to the Gaussian density function, illustrated in Figure A.2. Here, the param-

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Figure A.2: Gaussian density function.

eter μ corresponds to the mean, the center of the distribution, and the parameter $\sum$ corresponds to the covariance, defining the curve's width. By its turn, the D parameter corresponds to the dimensionality of the data points [Carrasco, 2020].

- Hidden Markov Models

By its turn, the training of Hidden Markov Models falls on [Huggins-Daines and Solovets, 2022e]. This provides a basic Hidden Markov Model (HMM) object, providing classes that allow the user to build HMMs based on triphones, i.e., a sequence of three phonemes [tri, 2020], and in the future, should be done based on sentences too.

These models are widely used to assign a correct label sequence to sequential data or to access the probability of a given label and data sequence.

These concern finite state machines characterized by three components:

- The set of states that constitute them
- The probability of the transitions from a given initial state to each final state, which are dependent only on the current state and being time-invariant
- An output probability matrix, which represents the probability of observing each symbol or observation in a given state, which is a probabilistic function of the state it is associated with
- The initial state distribution, which defines the probability of starting in each state
- The output observation alphabet, which concerns the set of symbols that may be observed as the system output

These models correspond to directed graphs in which the arcs are probability weighted. These weights correspond to the probability of a transition between the states that the arc connects. In turn, each vertex emits an output symbol when entered, and this symbol is generated non-deterministically. For this reason, the sequence of output observations does not map directly to a sequence of states, and this corresponds to the hidden component of the model [1].

Besides allowing the training of this type of model, Sphinxtrain also enables the user to adapt acoustic models using maximum-likelihood linear regression [Huggins-Daines and Solovets, 2022f]. This adaptation includes mean, and variance adaptation based on Maximum Likelihood Linear Regression (MLLR), as described in [Gales and Woodland, 1996]. This approach transforms the mean and the variance of a model by the search for a block diagonal matrix that maximizes the likelihood of the adaptation data. Once found, that matrix is multiplied by the mean, resulting in a new mean matrix. Regarding the variance, it is replaced by a diagonal matrix where the non-null values correspond to each parameter variance.

Besides these models, Sphinxtrain has multiple files related to model evaluation and file reading, writing, editing, or formatting. It also encompasses files related to feature extraction [Huggins-Daines and Solovets, 2022g].

[1] https://www.nltk.org/api/nltk.tag.hmm.html

Appendix B

Kaldi components' architecture

Kaldi is based on Finite-State Transducer (FST), which corresponds to a finite automaton in which the state transitions are labeled with the input and the output symbols. Walking through the transducer, an input symbol sequence is encoded through a mapping that produces an output symbol sequence [Mohri et al., 2002]. By its turn, a finite state automaton corresponds to the most simple machine for pattern recognition. It is an abstract machine composed of five elements or tuples. Those include a set of states, a set of input symbols, the initial state, the set of final states, and a set of rules to move between states, depending on the applied input symbol [gee, 2015]. Because of this, any language model that can be represented as a FST can be used in Kaldi.

This tool provides, primarily tools for feature extraction. Regarding acoustic modeling, Kaldi was developed to support only diagonal GMMs and Subspace Gaussian Mixture Model (SGMM)s, but also to be easily extensible to new types of models. The GMMs are supported with diagonal and full covariance structures, being its parameters the means times the inverse covariance and the inverse covariance itself. The acoustic model based on this kind of model corresponds to a collection of context-dependent HMM states, each associated with its GMM. The context-dependent question is responsible for dealing with the fact that the same phoneme can be said differently, depending on the other phonemes it appears with. With that, the phonemes of the audio under study have HMM associated with them, being the acoustic features associated with it generated taking into account the GMM associated with the HMM states of each HMM [Fabien, 2020]. By its turn, the SGMM correspond to a model in which the GMM in all the HMM states have the same structure but the means and weights associated with each normal distribution can vary in a subspace of the entire parameter space, being that controlled. It then differs from the GMM approach in which each HMM state tends to have its own GMM.

Besides that, Kaldi authors also provide phonetic decision trees whose roots can be shared among phonemes and their states, allowing questions to be asked. These questions can be about any phoneme in the context window and the HMM state. Those trees are learned during the training of the model and help in the construction of the acoustic features by association with the HMM-GMM and with the SGMM [Povey et al., 2011a].

Finally, Kaldi allows both model-space and feature-space adaptation. The first can be done using MLLR while the second can be done throw Feature-Space Maximum Likelihood Linear Regression (fMLLR), also known as Constrained Maximum Likelihood Linear Regression (CMLLR). This specific type of MLLR consists of an affine feature transformation in which the features vector is multiplied by a matrix and a bias is summed [1].

Besides that, speaker normalization is also supported using a linear approximation to VTLN [2], conventional feature-level VTLN [3] or gender normalization throw exponential transform [4]. Also, fMLLR and VTLN can both be used for speaker adaptive training of acoustic models. Kaldi's training and decoding algorithms use Weighted Finite State Transducer (WFST)s. Those correspond to a case of FST in which the transducer also assigns weights to the transitions. Those weights may be related to the encoding of probabilities, durations, penalities, or any other cumulative quantity that, accumulating along the path, allows the calculation of the overall weight associated with the mapping of an input sequence to an output sequence [Mohri et al., 2002] [Povey et al., 2011b].

[1]https://kaldi-asr.org/doc/transform.html

[2]https://kaldi-asr.org/doc/transform.htmltransform$_{l}vtln$

[3]https://kaldi-asr.org/doc/feat.htmlfeat$_{v}tln$

[4]https://kaldi-asr.org/doc/transform.htmltransform$_{e}t$

Appendix C

Whisper models' architecture

The model used by Whisper corresponds to a neural network, and its architecture is illustrated in Figure C.1. The input of this system is split into chunks of 30 seconds and converted to a log-Mel spectrogram. This corresponds to a spectrogram where the frequencies are converted into the Mel scale that, in its turn, corresponds to a unit of pitch in which an equal distance in pitch also sounds equally distant to the listener. It is related to the fact that humans can easily detect differences between lower frequencies than between higher ones, even if the distance between the frequencies under comparison is equal [Roberts, 2020].

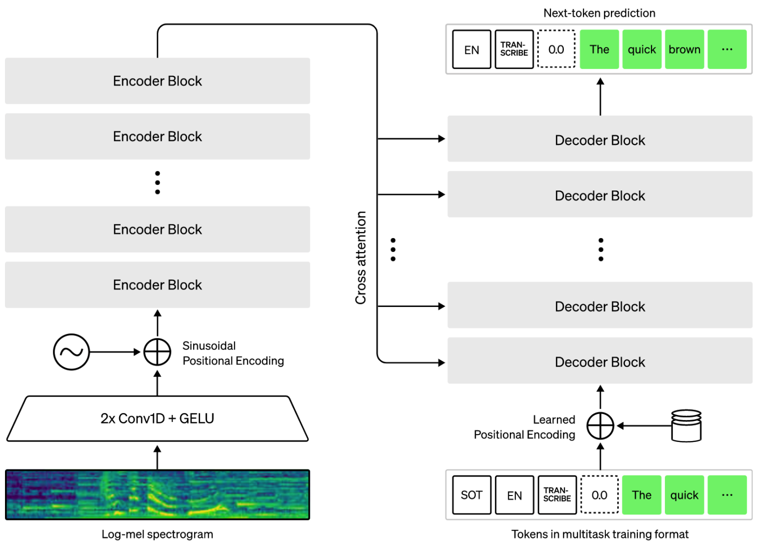


Figure C.1: Whisper's model architecture.

Only before the conversion to a log-Mel spectrogram the input is fed into a neural network composed by two layers. Those layers encompass each a one-dimensional convolutional layer with a GELU activation function. GELU states for Gaussian Error Linear Units, and its motivation corresponds to the association of stochastic regularizers like dropout with non-linearities, that is, activation functions. While the dropout regularization is responsible for the stochastic multiplication of a neuron’s inputs by zero, inactivating them, the ReLU activation function multiplies each input by a value between zero and one, depending on the input value. Associating them, the GELU activation makes the multiplication of each input by a value from zero to one, being this value chosen stochastically but also taking the input’s value into account [Poulinakis, 2022].

After that, the input sequence is submitted to sinusoidal positional encoding, which corresponds to a model component that gives it a sense of order, that is, gives the model information about the order of the words in the sentence [Kazemnejad, 2019]. After that, the input sequence is passed to an encoder block and, with cross-attention, is passed to the decoder block. In the cross-attention mechanism, two input sequences are mixed or combined throw a set of operations illustrated in [Raschka, 2023]. In the case of Whisper’s architecture, the sequence returned from the encoder block should be combined with the input sequence obtained with the learned positional encoding obtained from tokens in multitask training format. The decoder is trained for next-token prediction being the whole model able to perform tasks like language identification, phrase-level timestamps, multilingual speech transcription, and translation of non-English speech to English [Radford et al., 2022].

No information is publicly available regarding the model architecture associated with each of the models available for multi-language, including European Portuguese. About the data used to train the models, it is also only known that it corresponds to 680,000 hours of multilingual and multitask supervised data, and it was collected from the web.

Appendix D

spaCy models components' architecture

The pipelines that spaCy provides for European Portuguese each have eight components. These components are the same for all three pipelines and concern the Tokenizer, Tok2Vec, Morphologizer, DependencyParser, Lemmatizer, SentenceRecognizer, AttributeRuler, and NER. Of these, only SentenceRecognizer is not active in the European Portuguese pipelines [1]:

- Tokenizer
  spaCy provides different models for a variety of languages, all of them having in common their initial component: the Tokenizer. This is the component responsible for tokenization, corresponding to segmenting the input text into words, punctuation, and others. In tokenization, a given text is segmented into tokens, corresponding to the denomination given to the meaningful segments of a text. To perform this task, spaCy applies language-specific rules to the text being processed (which should be a Unicode text) and produces a Doc object consisting of individual tokens. With this object, it is possible to iterate over the tokens, being necessary to have up to three components for its creation: a Vocab instance, a sequence of word strings, and, optionally, a sequence of spaces booleans. These components allow the alignment of the tokens in the original string to be maintained. The spaces booleans or spaces values correspond to a list of boolean values, which indicate whether a space succeeds the token in each position of the word list. By default, it is assumed that all are [2].

  The first step in tokenization concerns splitting the text on spaces characters (similar to text.split(' ')). The tokenizer then processes the text from left to right, making two checks on each of the strings it looks at:

  - If the current substring matches any exception rule;
  - If it is possible to separate any suffix, prefix, or infix.

[1]https://spacy.io/models/pt
[2]https://spacy.io/usage/linguistic-featurescustom-tokenizer

If there is a match in any previous checks, the tokenizer applies the rule in question and then proceeds to iterate over the substrings starting now on the newly split substrings. With this, spaCy can split complex and nested tokens. While the punctuation rules are generally general, i.e., the same for the various languages, the tokenizer exceptions heavily depend on each language's specifics. For this reason, each of the languages that spaCy supports has its subclass, which loads in lists of encoded data and exception rules [3] [4].

With the spaCy Tokenizer, it is possible to:

- Add special case rules to it, that is, rules that only apply to a given field [5]
- Create a tokenizer adapted to the user's needs [6] by specifying:
  * A special case dictionary that handles things like contractions, units of measure, emoticons, and certain abbreviations.
  * A prefix_search function to handle preceding punctuation such as open quotation marks and open parentheses.
  * A suffix_search function to deal with following punctuation such as commas, periods, and closing quotation marks.
  * An infix_finditer function to handle non-whitespace separators such as hyphens.
  * An optional boolean function token_match to match strings that should never be split, which overrides the infix rules.
  * An optional boolean function url_match similar to the previous one, but in this case, prefixes and suffixes are removed before the match is applied
- Change pre-existing rules in a tokenizer [7]
- Use partially annotated text [8]
  * spaCy assumes, by default, that the input text is raw text, but this text may be partially annotated. The most common cause of this partially annotated text concerns pre-existing tokenization. In this case, if the user has a list of strings, he/she can directly generate a Doc object and optionally specify a list of spaces booleans, which should be the same length as the word list.
- Join annotations from different sources that may tokenize differently from spaCy or apply vectors provided by a pre-trained BERT model to spaCy tokens [9]
  * To do this, it is necessary to align the tokenization. For that, spaCy provides the Alignment object, through which it is possible to do

[3]https://spacy.io/usage/linguistic-featurestokenization
[4]https://spacy.io/usage/spacy-101annotations-token
[5]https://spacy.io/usage/linguistic-featuresspecial-cases
[6]https://spacy.io/usage/linguistic-featuresnative-tokenizers
[7]https://spacy.io/usage/linguistic-featuresnative-tokenizer-additions
[8]https://spacy.io/usage/linguistic-featuresown-annotations
[9]https://spacy.io/usage/linguistic-featuresaligning-tokenization

one-to-one mapping of token indexes in both directions and consider indexes where several tokens align to a single one.

* BERT corresponds to a transformer, which refers to one element of a family of neural network architectures that peer to compute representations for document tokens. The representation generated by these models is dense and context-sensitive [10] [11].

The Tokenizer concerns the first component of the pipeline, and it has a distinct signature from the other parts since it receives a text and returns a Doc rather than expecting to receive a tokenized Doc like the other components. It is not replaceable by writing to nlp.pipeline. Instead, it is necessary to overwrite the current tokenizer (nlp.tokenizer) by replacing it with a custom function that receives a text and returns a Doc object [12].

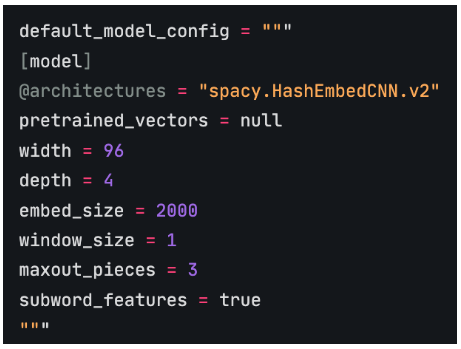
Also, the tokenization performed by spaCy is non-destructive,i.e., it is always possible to reconstruct the original input from the tokenized output. Furthermore, the tokenization makes use of language-specific rules that are optimized for compatibility with treebank annotations, corresponding a treebank to an annotated corpora [13] [14] [Prescher et al.].

- Tok2Vec [15] [16]
  After tokenizing the input text, its processing continues with converting the tokens identified in a vector representation using the Tok2Vec component. This layer uses a token conversion model in its vector representation and defines its results in the Doc.tensor attribute. For its predictions to be used by its successor components in the pipeline, they must use a Tok2VecListener layer as the tok2vec subnetwork of their model. When training, the Tok2Vec component stores its prediction and backpropagation callback for each batch. With this information, the part allows the tok2vec subnetworks of its successor components to backpropagate to achieve the shared weights. By default, the model used in this layer refers to HashEmbedCNN. This is made up of two layers, these being an embedding layer and an encoding layer [17].

  The configuration used by default for this model is shown in Figure D.1.

  From the values mirrored in the configuration of Figure D.1, it is possible to understand that the default HashEmbedCNN model does not encompass pre-trained vectors [18], uses subword features, consists of four convolutional layers with an input and output width equal to 96, uses hash embedding Tables with 2000 rows, considers one token for each side to concatenate at the time of convolutions so the network is sensitive to twelve words at a

[10]https://spacy.io/usage/embeddings-transformers
[11]https://spacy.io/usage/embeddings-transformerstransformers
[12]https://spacy.io/usage/linguistic-featurescustom-tokenizer
[13]https://spacy.io/usage/linguistic-featurestokenization
[14]https://spacy.io/usage/linguistic-featuresaligning-tokenization
[15]https://spacy.io/models/pt
[16]https://spacy.io/api/tok2vec
[17]https://spacy.io/api/architecturesHashEmbedCNN
[18]https://spacy.io/usage/embeddings-transformersstatic-vectors



```
default_model_config = """
[model]
@architectures = "spacy.HashEmbedCNN.v2"
pretrained_vectors = null
width = 96
depth = 4
embed_size = 2000
window_size = 1
maxout_pieces = 3
subword_features = true
"""
```

Figure D.1: Default configuration of spaCy's European Portuguese pipeline Tok2Vec component.

time, and uses three linear functions in the maxout activation function [19], which then corresponds to the function used by default.

By its turn, the embedding layer used by MultiHashEmbedCNN is generated using MultiHashEmbed [20]. This generates an embedding layer that separately embeds some lexical attributes using hash embedding, concatenates the results obtained, and then passes them through a feed-forward subnetwork, yielding a mixed representation of the initial token [21].

In turn, the encoding layer used by HashEMbedCNN is generated using MaxoutWindowEncoder [22]. This generates a template consisting of a CNN and a layer-normalized maxout activation function. This component allows context encoding using convolutions with maxout activation, a normalization layer, and also residual connections [23].

In short, HashEmbedCNN entails two components: an embedding layer - which generates a vector representation of the tokens - and an encoding layer - which encodes the context. MultiHashEmbed, in turn, comprises three stages. The first of these steps corresponds to embedding the attributes with their respective embedding Tables. The second is the concatenation of the result obtained in the first step. If static vectors are used, they are also concatenated at this stage. In the third and last phase, the result of the second step is passed through a subnetwork, culminating in obtaining a mixed representation of the initial token using a maxout layer. Finally, the MaxoutWindowEncoder makes convolutions with maxout activation function, using layer normalization and residual connections in the network.

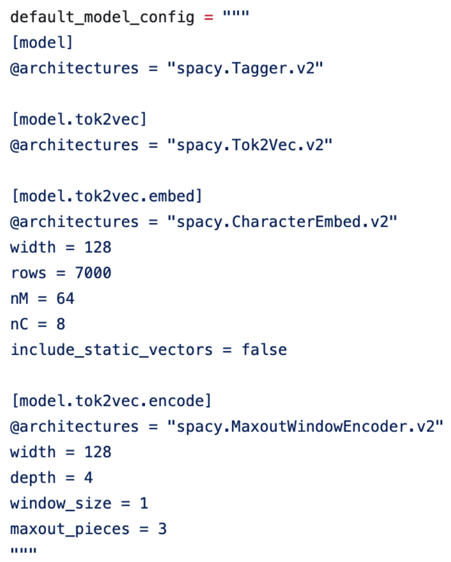
- Morphologizer
  After converting the tokens to the respective vector representation, processing continues with the Morphologizer component [24]. This is a trainable component and is responsible for predicting morphological features and

[19]https://thinc.ai/docs/api-layersmaxout
[20]https://spacy.io/api/tok2vec
[21]https://spacy.io/api/architecturesMultiHashEmbed
[22]https://spacy.io/api/tok2vec
[23]https://spacy.io/api/architecturesMaxoutWindowEncoder
[24]https://spacy.io/models/pt

coarse-grained POS tags following the UPOS and FEATS [25] [26] Universal Dependency annotation guidelines [27].

This component requires some parameters and, between them, the model. It specifies the model to be used in the component for morphological features prediction, and by default, Tagger is used. Tagger comprises a tagger model generated using a token-to-vector component. This model adds a linear layer with a softmax activation function to predict the scores assigned to token vectors [28]. This layer is build using Thinc's Softmax_v2 function [Honnibal et al., 2020] [29].

The default configuration used for the Morphologizer is illustrated in Figure D.2.



```
default_model_config = """
[model]
@architectures = "spacy.Tagger.v2"

[model.tok2vec]
@architectures = "spacy.Tok2Vec.v2"

[model.tok2vec.embed]
@architectures = "spacy.CharacterEmbed.v2"
width = 128
rows = 7000
nM = 64
nC = 8
include_static_vectors = false

[model.tok2vec.encode]
@architectures = "spacy.MaxoutWindowEncoder.v2"
width = 128
depth = 4
window_size = 1
maxout_pieces = 3
"""
```

Figure D.2: Default configuration os spaCy's European Portuguese pipeline Morphologizer component.

From this configuration, it is possible to understand that the token-to-vector model used for the tagger model generation concerns, by default, Tok2Vec. This model has at its base an embedding layer given by the CharacterEmbed model [30] and an encoding layer provided by the MaxoutWindowEncoder model. The function responsible for generating the Tok2Vec model receives two parameters concerning embedding and encoding, generating the token-to-vector layer according to the architecture specified by the values of these parameters. The embed specifies the embedding subnetwork to be used. This component is responsible for embedding the tokens into context-independent word vector representations. Encode, in turn, determines the encode subnetwork to be used by the model. This component is

[25]https://universaldependencies.org/docs/format.html
[26]https://universaldependencies.org/format.htmlmorphological-annotation
[27]https://spacy.io/api/morphologizer
[28]https://spacy.io/api/architecturesTagger
[29]https://thinc.ai/docs/api-layerssoftmax$_v$2
[30]https://spacy.io/api/architecturesCharacterEmbed

responsible for encoding the context into the embedding using an architecture such as a CNN, a BiLSTM, or a transformer.

The encoding layer, in turn, concerns the MaxoutWindowEncoder model. This coincides with the model used by the pipeline component that precedes the Morphologizer, Tok2Vec, already discussed.

- Dependency Parser
  The third component of the pipelines made available by spaCy for Portuguese concerns the DependencyParser, an element for syntactic dependency parsing [31] [32].

  Dependency parsing refers to the process of analyzing the grammatical structure in a sentence and finding both the words that are related and the type of relationship that exists between them. Each of these relations consists of a head and a dependent, which modifies the head. Furthermore, each relationship is labeled according to the nature of the dependency between the head and the dependent, with the labels assigned in Universal Dependency Relations [Jaiswal, 2021].

  The DependencyParser is responsible for this analysis using a transition-based dependency parser [Zhang and Clark, 2008] component.

  The spaCy dependency parser learns sentence segmentation and labeled dependency parsing together and can optionally learn to join tokens over-segmented by the tokenizer. This component uses a variant of the non-monotonic arc-eager transition-system described by [Honnibal and Johnson, 2015] with the addition of a break transition to perform sentence segmentation.

  Also, the authors of spaCy use the pseudo-projective dependency transformation of [Nivre and Nilsson, 2005] to predict non-projective parses.

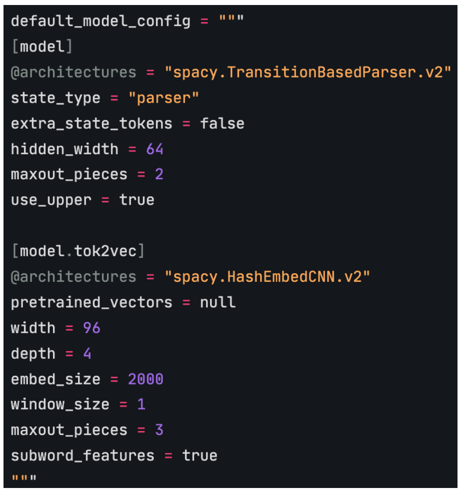
  Finally, the parser used by spaCy is trained using imitation learning objectives. It follows the actions predicted by the current weights and, in each state, determines which actions are compatible with the optimal parse that could be achieved from the current state. With this, the weights are updated to increase the scores given to the set of optimal actions while the scores assigned to the remaining actions are reduced. Note that there can be more than one optimal action for a given state [33].

  This model comprises a transition-based parser that can be applied to NER or dependency parsing, in this case, being applied to dependency parsing. Transition-based parsing refers to an approach to structured prediction in which the structure prediction task is mapped to a series of state transitions. The neuronal network state prediction model consists of two or three subnetworks: the tok2vec, the lower, and the upper. These subnetworks have the function [34]:

[31]https://spacy.io/models/pt
[32]https://spacy.io/api/dependencyparser
[33]https://spacy.io/api/dependencyparser
[34]https://spacy.io/api/architecturesTransitionBasedParser

- tok2vec: responsible for mapping each token into a vector representation, being run once per batch
- lower: responsible for building a feature-specific vector for each pair (token, feature), also run once per batch. The construction of the state representation then corresponds to the sum of the component features and the application of non-linearity
- upper: this is an optional subnetwork and corresponds to a feedforward network that predicts the scores of the state representation. When this layer is not used, the output of the previous one is used directly as action scores. This layer is only recommended for smaller models since it is computed on the CPU, which becomes a bottleneck in larger GPU-based models, for which the layer is also less necessary.

In Figure D.3, it can be seen the architecture used, by default, for the component of the pipeline related to the Dependency Parser. In this, it is possible to understand that the model is used for dependency parsing, that no extra vectors are used to represent the state vectors, that the hidden layer has a size equal to 64, that two linear functions are used in the maxout activation function, and also that the upper layer is used. It is also possible to verify that the tok2vec component used in this model concerns the HashEmbedCNN model. This model has already been presented in the Tok2Vec component and is characterized by the absence of pre-trained vectors, by the fact that it is a network whose input and output have a width equal to 96, by having four convolutional layers, hash embedding Tables with 2000 lines, one token to be selected for each side during convolution, three linear functions to be used in the maxout activation function, and also by the use of subword features.



```
default_model_config = """
[model]
@architectures = "spacy.TransitionBasedParser.v2"
state_type = "parser"
extra_state_tokens = false
hidden_width = 64
maxout_pieces = 2
use_upper = true

[model.tok2vec]
@architectures = "spacy.HashEmbedCNN.v2"
pretrained_vectors = null
width = 96
depth = 4
embed_size = 2000
window_size = 1
maxout_pieces = 3
subword_features = true
"""
```

Figure D.3: Default configuration of spaCy's European Portuguese pipeline Dependency Parser component.

- Lemmatizer
  The Lemmatizer succeeds the Dependency Parser in the Portuguese spaCy

pipelines [35]. These make use of the trainable lemmatizer called EditTreeLemmatizer.

The EditTreeLemmatizer concerns a trainable component to assign tokens to their respective base form. This corresponds to the lemma of a token, that is, the root form of a word, its form without affixes [36] [37]. This component uses a lemmatization model to predict the edit tree that applies to each token.

The edit trees are the output of a rule-finding algorithm that, given a corpus with lemma annotations, automatically infer the lemmatization rules responsible for mapping a word into its lemma. These edit trees correspond to a recursive data structure [De Kok, 2021].

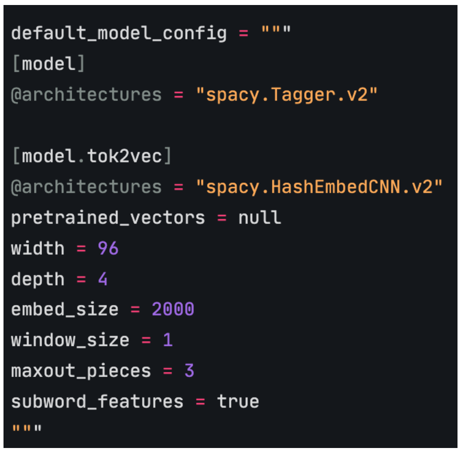
– Inner node: splits the string into prefix, infix, and suffix and returns the concatenation of the three transforms. The infix refers to substrings shared by the token and its lemma
– Leaf node: checks whether its input matches a given value s and, if it does, returns a given value t defined in the node. If the node's input is different from s, then the tree in question cannot be used in the token

The model used by spaCy in this component corresponds to the first log-linear model for morphological analysis and lemmatization that operates at the token level and can lemmatize unknown forms. This solution consists of two components: a lemmatization component which makes use of a log-linear model, and a morphological analysis component which makes use of MARMOT, a high-order CRF [Müller et al., 2013]. These components are combined into a tree-structured CRF [Wallach, 2004], and the parameter estimation of the model as a whole is done using L1 normalized SGD [Roy, 2019] [Tsuruoka et al., 2009] [Müller et al., 2015].

The default configuration of this component can be found in Figure D.4. Here it is possible to see that the HashEmbedCNN model gives the tok2vec component of the default model. This model was presented in the presentation of the Tok2Vec component and, in this context, is characterized by not using pre-trained vectors, having an input and output width equal to 96, having four convolutional layers, using hash embedding Tables with 2000 lines, using one token each side during convolutions, using three linear functions in the maxout activation function and, finally, making use of subword features.

- Sentence Recognizer
  After the Lemmatizer comes the SentenceRecognizer in the European Portuguese pipelines, in these, however, this component is deactivated [38]. The SentenceRecognizer is the component responsible for performing sentence segmentation [39] and has a default configuration illustrated in Figure D.5.

[35]https://spacy.io/models/pt
[36]https://spacy.io/api/edittreelemmatizer
[37]https://spacy.io/usage/linguistic-featuresmorphology
[38]https://spacy.io/models/pt
[39]https://spacy.io/api/sentencerecognizer

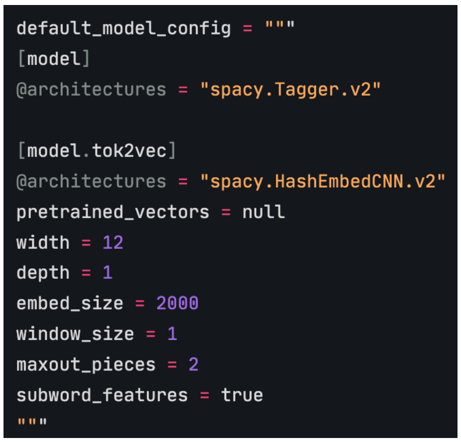


```
default_model_config = """
[model]
@architectures = "spacy.Tagger.v2"

[model.tok2vec]
@architectures = "spacy.HashEmbedCNN.v2"
pretrained_vectors = null
width = 96
depth = 4
embed_size = 2000
window_size = 1
maxout_pieces = 3
subword_features = true
"""
```

Figure D.4: Default configuration of spaCy's European Portuguese pipeline Lemmatizer component.

From this configuration, it is possible to understand that, as with the Lemmatizer component, the tok2vec component used in Tagger is given by the HashEmbedCNN model, already detailed for the Tok2Vec component of the pipeline.



```
default_model_config = """
[model]
@architectures = "spacy.Tagger.v2"

[model.tok2vec]
@architectures = "spacy.HashEmbedCNN.v2"
pretrained_vectors = null
width = 12
depth = 1
embed_size = 2000
window_size = 1
maxout_pieces = 2
subword_features = true
"""
```

Figure D.5: Default configuration of spaCy's European Portuguese pipeline Sentence Recognizer component.

Further, it can be understood that this component does not use pre-trained representations of the tokens, uses an input and output width equal to twelve, has a convolutional layer, uses hash embedding Tables with 2000 rows, considers one token for each side when convoluting, uses two linear functions in the maxout activation function, and uses subword features, specifically the prefix, suffix, and shape of the token.

- AttributeRuler
  The SentenceRecognizer is followed by the AttributeRuler [40]. This component allows the definition of token attributes identified by Matcher patterns. It is typically used to handle exceptions for token attributes and to map

[40]https://spacy.io/models/pt

values between attributes, such as mapping fine-grained POS to coarse-grained POS tags [41].

AtributteRuller uses Matcher patterns to identify tokens and assign them the provided attributes. If necessary, Matcher patterns can include context around the target token [42]. A Matcher is a rule-matching engine that operates on tokens, similar to regular expressions, to perform rule-based matching [43].

- Named Entity Recognition (NER)
  After the AttributeRuler and as the last component of the pipelines provided for European Portuguese, comes NER, a transition-based entity recognition component [44] [45].

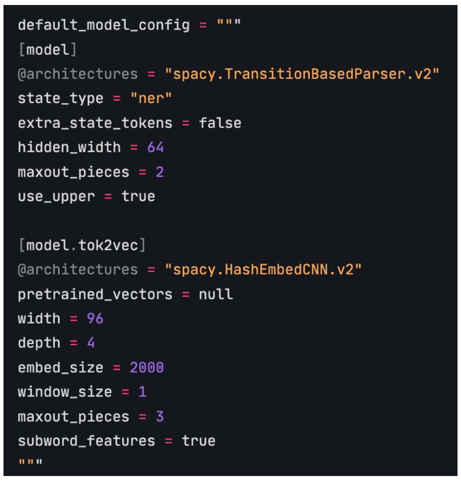
  This component identifies spans of non-overlapping labeled tokens, with several assumptions made by its transition-based algorithm. While these assumptions are effective in so-called "traditional" named entity recognition tasks, they may not be a good fit for all kinds of span identification problems. In particular, it should be noted that the loss function focuses on optimizing the accuracy of the entire entity.

  The transition-based algorithm also assumes that the most decisive information about entities is near their initial tokens. If the problem for which the model is used involves identifying long entities characterized by tokens in their middle, this component will similarly tend to perform poorly on the task.

  Figure D.6 illustrates the configuration used by default for the present pipeline component. From this, it can be seen that the TransitionBasedParser model is used for NER in this component, and no extra tokens are included for the state representation. Furthermore, a hidden layer with a size of 64 is used, two linear functions in the maxout layer and the upper layer is also used. In turn, it is possible to understand that the tok2vec component of this model is performed by the HashEmbedCNN model, exploited in the Tok2Vec component of the pipeline.

  This component, in this context, makes no use of pre-trained representations of the tokens, has an input and output width equal to 96, uses four convolutional layers, has 2000 rows in the hash embed Tables, uses one token each way when convoluting, uses three linear functions in the maxout layer, and also makes use of subword features, specifically the affixes and the token shape.

[41]https://spacy.io/api/attributeruler
[42]https://spacy.io/api/attributeruler
[43]https://spacy.io/usage/rule-based-matchingmatcher
[44]https://spacy.io/models/pt
[45]https://spacy.io/api/entityrecognizer



```
default_model_config = """
[model]
@architectures = "spacy.TransitionBasedParser.v2"
state_type = "ner"
extra_state_tokens = false
hidden_width = 64
maxout_pieces = 2
use_upper = true

[model.tok2vec]
@architectures = "spacy.HashEmbedCNN.v2"
pretrained_vectors = null
width = 96
depth = 4
embed_size = 2000
window_size = 1
maxout_pieces = 3
subword_features = true
"""
```

Figure D.6: Default configuration of spaCy's European Portuguese pipeline NER component.

Appendix E

NLTK components' architecture

Focusing on the features of interest of this system, namely in the architecture of the underlying models:

- Accessing corpora
  For the language processing task called accessing corpora, NLTK provides the corpus module [1].

  This module, in turn, provides functions that can be used to read corpus files in various formats. These can be used to read corpus files distributed in the NLTK corpus package or to read corpus files that are part of external corpora. This module then provides NLTK corpus readers, and the complete list of available NLTK corpora can be found at [2] [3].

  This module comprises 68 separate corpus reader functions, these being contained in the reader submodule [4]. These corpus reader functions are associated with corpus with different formatting, all of which are presented at [5].

- String processing
  In turn, the language processing task related to string processing is mediated by NLTK using the tokenize and stem modules [6].

  The tokenizers are responsible for splitting strings into lists of substrings and, in the present context, can be used for [7]:

  - Find the words and punctuation in a string, requiring Punkt sentence tokenization models [8]
    This tokenizer splits the original string using an unsupervised algorithm to build a model for abbreviation words, collocations, and words

[1]https://www.nltk.org/book/ch00.html
[2]https://www.nltk.org/nltk$_{data}$/
[3]https://www.nltk.org/api/nltk.corpus.htmlmodule-nltk.corpus
[4]https://www.nltk.org/api/nltk.corpus.reader.html
[5]https://www.nltk.org/api/nltk.corpus.reader.html
[6]https://www.nltk.org/book/ch00.html
[7]https://www.nltk.org/api/nltk.tokenize.htmlmodule-nltk.tokenize
[8]https://www.nltk.org/api/nltk.tokenize.punkt.html

that start sentences. This type of models needs training in the target language before being used on it. Besides that, only a pre-trained Punkt tokenizer for English is available.

- Split text on whitespace and punctuation, requiring a regular-expression based tokenizer [9]

  This tokenizer uses regular expressions to split the string into substrings.

- Operate at the sentence level, which requires a sentence tokenizer [10]

  The default sentence recognizer corresponds to PunktSentenceTokenizer [11] and uses an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences. The model is then used to find the boundaries of sentences.

The stem module, in turn, encompasses interfaces that reduce words to their stem, removing morphological affixes from them. The stemming algorithms aim at eliminating these affixes, and due to the existence of irregular words, complicated morphological rules, and part-of-speech and sense ambiguities, the task of reducing words to their stem is a complex problem. The standard interface used in this context is StemmerI, corresponding this to an interface for stemming [12].

This module comprises several stemmers, including:

- ISRI Arabic Stemmer [13]
  This tokenizer is based on the ISRI Arabic Stemmer, which is based on Arabic Stemming without root dictionary [Taghva et al.].

- Lancaster Stemmer [14]
  This corresponds to a word stemmer based on the Lancaster stemming algorithm, also called the Paice/Husk stemming algorithm [Paice, 1990].

- Porter Stemmer [15]
  By its turn, this stemmer is based on the algorithm presented in [Porter], having some optional deviations to it.

- Regexp Stemmer [16]
  The Regexp Stemmer uses regular expressions to identify morphological affixes. Any substring, when matched by regular expressions, is removed.

- RSLP Stemmer [17]
  This stemmer for the Portuguese language allows the removal of af-

[9]https://www.nltk.org/api/nltk.tokenize.regexp.html?highlight=regexpmodule-nltk.tokenize.regexp

[10]https://www.nltk.org/api/nltk.tokenize.htmlnltk.tokenize.sent$_{t}$*okenize*

[11]https://www.nltk.org/api/nltk.tokenize.PunktSentenceTokenizer.htmlnltk.tokenize.PunktSentenceTokenizer

[12]https://www.nltk.org/api/nltk.stem.htmlmodule-nltk.stem

[13]https://www.nltk.org/api/nltk.stem.isri.html

[14]https://www.nltk.org/api/nltk.stem.lancaster.html

[15]https://www.nltk.org/api/nltk.stem.porter.html

[16]https://www.nltk.org/api/nltk.stem.regexp.html

[17]https://www.nltk.org/api/nltk.stem.rslp.html

fixes from tokens by following an eight steps rule-based algorithm, the RSLP algorithm [18].

  - Snowball Stemmers [19]
    This module provides a port for the Snowball stemmer [sno, 2020] developed by Martin Porter comprising stemmers for Arabic, Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Romanian, Russian, Spanish, and Sweeden and the original Porter Stemmer.
  - WordNet Lemmatizer [20]
    Finally, this lemmatizer makes use of WordNet's built-in morphy function [21], which is a corpus reader and Morphy is one of its methods. It makes use of a combination of inflectional ending rules and exception lists to reduce a given form to its stem [22].

- Collocation discovery For the language processing task of collocations discovery, NLTK provides the collocations module [23].

  This module, in turn, provides the tools for identifying collocations within the corpora, where collocations refer to words that appear frequently consecutively. In addition, these tools can also be used to find other associations between word occurrences [24] [25].

  For the discovery of collocations, several steps are necessary:

  1. Calculating the frequencies of words and their appearance in the context of other words
  2. Filtering, when necessary, so that only useful content terms are retained
  3. Scoring of each ngram of words taking into account an association measure

  For that both BigramCollocationFinder and BigramCollocationFinder are provided [26] [27].

- Part-of-Speech (POS) tagging For the task of POS tagging, NLTK provides the tag module, which provides both classes and interfaces for tagging [28] [29].

  A tag is a case-sensitive string that specifies any token property, such as its POS. Tokens with associated tags are encoded as tuples of the type (tag, token).

[18]https://www.inf.ufrgs.br/ viviane/rslp/index.htm
[19]https://www.nltk.org/api/nltk.stem.snowball.html
[20]https://www.nltk.org/api/nltk.stem.wordnet.html
[21]https://www.nltk.org/howto/wordnet.html
[22]https://www.nltk.org/howto/wordnet.htmlmorphy
[23]https://www.nltk.org/book/ch00.html
[24]https://www.nltk.org/api/nltk.corpus.htmlmodule-nltk.corpus
[25]https://www.nltk.org/api/nltk.collocations.htmlmodule-nltk.collocations
[26]https://www.nltk.org/api/nltk.collocations.BigramCollocationFinder.html
[27]nltk.org/api/nltk.collocations.TrigramCollocationFinder.html
[28]https://www.nltk.org/book/ch00.html
[29]https://www.nltk.org/api/nltk.tag.htmlmodule-nltk.tag

NLTK provides an "off-the-shelf tagger" for English and a Russian tagger. The former makes use of the Penn Treebank [30] tag set, while the latter, besides being available by specifying the lang parameter as "rus", makes use of the Russian National Corpus [31].

In addition, this module encompasses several taggers whose principle of operation concerns

1. Receiving a list of tokens
2. Assigning a tag to each token in the list
3. Return of the resulting list, which concerns a list of tagged tokens

NLTK provides different taggers. These taggers relate to:

- Brill Tagger [32], to which the Bill Tagger Trainer [33] is associated

  This tagger concerns Brill's transformational rule-based tagger and, being a Brill tagger acts in two phases [34]:

  1. In the first, an initial tagger is used to assign an initial tag sequence to a text
  2. In the second, an ordered list of transformational rules is applied to correct the tags of individual tokens

- CRF Tagger [35]
  In turn, CRF Tagger concerns a module for POS tagging making use of CRFSuite, [Okazaki, 2023] which concerns software that corresponds to an implementation of CRFs for tagging sequential data [36].

- Hidden Markov Model Tagger [37]
  On the other hand, the hmm module provides both the Hidden Markov Model Tagger and its trainer, this tagger being based on Hidden Markov Models. The HMM make use of the Viterbi algorithm to calculate the optimal path through the graph for a given sequence of word forms [Kwok, 2019] [38].

- Hunpos Tagger [39]
  The hunpos module is an interface to the HunPos [40] open-source POS-tagger, which relies on second-order Markov models [nlp, 2019] [41].

[30]https://www.nltk.org/api/nltk.tokenize.treebank.html?highlight=penn+treebank
[31]https://ruscorpora.ru/en/ tagset
[32]https://www.nltk.org/api/nltk.tag.brill.html
[33]https://www.nltk.org/api/nltk.tag.brill$_{t}$*rainer.html*
[34]https://www.nltk.org/api/nltk.tag.brill.html
[35]https://www.nltk.org/api/nltk.tag.crf.html
[36]https://www.nltk.org/api/nltk.tag.crf.html
[37]https://www.nltk.org/api/nltk.tag.hmm.html
[38]https://www.nltk.org/api/nltk.tag.hmm.html
[39]https://www.nltk.org/api/nltk.tag.hunpos.html
[40]https://code.google.com/archive/p/hunpos/
[41]https://www.techtarget.com/whatis/definition/Markov-model

- Perceptron Tagger [42]
  This tagger concerns a Greedy Averaged Perceptron - a neural network - tagger as implemented by Matthew Honnibal [Honnibal, 2013] [43].
- Senna POS tagger, NER tagger, Chunk tagger [44]
  These taggers are included in the senna module and are based on SENNA, a software tool that allows making predictions about part-of-speech tags, chunking, named entity recognition, semantic role labeling, and also syntactic parsing [45].
- Affix Tagger, Bigram Tagger, Classifier Based POS Tagger, Classifier Based Tagger, Context Tagger, Default Tagger, Ngram Tagger, Regexp Tagger, Sequential Backoff Tagger, Trigram Tagger, Unigram Tagger [46]
  These taggers, in turn, belong to the sequential module and concern classes used for tagging sentences sequentially, from left to right, all based on the abstract class SequentialBackoffTagger. These taggers are trained, so they are able to identify the tag associated with a word based on some context of the word, of its neighbors, or with no context.
- Stanford NER Tagger, Stanford POS Tagger, Stanford Tagger [47]
  The stanford module interfaces with the Stanford taggers. These relate to Stanford NER Tagger [the, 2010], Stanford POS Tagger [Toutanova et al.], and Stanford Tagger.

Besides that, this module encompasses two interfaces that concern:

- API module [48]
  This module provides APIs to tag each token in a sentence with supplementary information such as its POS, and two distinct interfaces are provided:
  * FeaturesetTaggerI This tagger needs tokens to be feature sets, being called features a dictionary that maps from feature names to feature values, being these features descriptive of the token in question. [49]
  * TaggerI
    TaggerI is a processing interface whose task is to assign a tag to each token in a given list.
- Mapping module [50]
  This provides an interface to convert POS tags from different treebanks to the same universal tagset of Petrov, Das, McDonald.

[42]https://www.nltk.org/api/nltk.tag.perceptron.html
[43]https://www.programiz.com/dsa/greedy-algorithm
[44]https://www.nltk.org/api/nltk.tag.senna.html
[45]https://ronan.collobert.com/senna/
[46]https://www.nltk.org/api/nltk.tag.sequential.html
[47]https://www.nltk.org/api/nltk.tag.stanford.html
[48]https://www.nltk.org/api/nltk.tag.api.html
[49]https://www.nltk.org/api/nltk.classify.html?highlight=nltk+classifymodule-nltk.classify
[50]https://www.nltk.org/api/nltk.tag.mapping.html

- Chunking
  For the chunking task, the NLTK provides the chunk [51] method, which, in turn, offers classes and interfaces for identifying non-overlapping linguistic groups in an unrestricted text, i.e., for performing the chunk parsing or chunking task. This is responsible for identifying groups called chunks, and the chunked text is represented as a chunk structure which refers to a shallow tree. This chunk structure corresponds to a tree consisting of tokens and chunks, each of them corresponding to a subtree containing only tokens [52].

  The present module defines:

  - ChunkParserI, a standard interface for chunking texts
  - RegexpChunkParser [53], an implementation of the previous interface that makes use of regular expressions on tags to chunk a text
  - ChunkScore, a utility class for scoring chunk parsers

- Parsing
  For the language processing task called Parsing, NLTK provides both the parse module and the ccg module [54].

  The parse module provides classes and interfaces for performing the parsing task. This task is concerned with producing tree structures, also called parses, representative of the internal structure of a text [55].

  A given text may be considered ambiguous either because it can be represented by more than one tree structure or because it is impossible to determine its correct parse due to a lack of information, and this module does not distinguish these forms of ambiguity. It defines:

  - A standard interface for text parsing, called ParserI
  - A ShiftReduceParser, a simple implementation of the previous interface
  - A RecursiveDescentParser, a simple implementation of the previous interface

  In addition, it provides sub-modules designed for specialized kinds of parsing:

  - the chart, which defines chart parsing, a kind of parsing that makes use of dynamic programming aiming at the efficient parsing of texts
  - the probabilistic, which defines probabilistic parsing, that associates a probability to each parse

  In turn, the ccg module provides a Combinatory Categorial Grammar for parsing [56] [Yoshikawa et al., 2018]. A CCG-based parser is used to parse input premises and hypotheses to obtain their logical formulas.

[51] https://www.nltk.org/book/ch00.html
[52] https://www.nltk.org/api/nltk.chunk.htmlmodule-nltk.chunk
[53] https://www.nltk.org/api/nltk.chunk.regexp.html?highlight=chunkstringnltk.chunk.regexp.ChunkString
[54] https://www.nltk.org/book/ch00.html
[55] https://www.nltk.org/api/nltk.parse.htmlmodule-nltk.parse
[56] https://www.nltk.org/api/nltk.ccg.htmlmodule-nltk.ccg

- Semantic interpretation Also, regarding the Semantic interpretation task, the sem and inference modules are provided [57].

  The sem module concerns the Semantic Interpretation package and contains classes for both the representation of semantic structure in first-order logic formulas and for the evaluation of these same formulas using set-theoretic models [58]. This package consists of two main components that consist of:

  - to logic, which provides support for the analysis of First Order Logic (FOL) expressions
  - to evaluate, which in turn enables recursive truth-in-truth determination in a model for FOL formulas

  The inference module, on its turn, is responsible for providing classes and interfaces for both model building and theorem proving [59]:

  - Discourse [60], which allows:
    * Incremental development of simple discourses
    * Check for semantic ambiguity
    * Consistency check
    * Informativeness check
  - Nonmonotonic [61], which provides nonmonotonic reasoning [mon, 2021].
  - Resolution [62], which is responsible for the proof of theorems based on the resolution technique [63].
  - Tableau [64], which provides the tools for proving theorems based on the Tableau [Groeneboer, 1987].

- Evaluation Metrics
  NLTK also provides a module on Evaluation Metrics [65]. This provides classes and methods for scoring the processing modules and consists of the modules [66]:

  - Agreement [67]
    This module provides implementations of inter-annotator agreement coefficients.

[57]https://www.nltk.org/book/ch00.html
[58]https://www.nltk.org/api/nltk.sem.htmlmodule-nltk.sem
[59]https://www.nltk.org/api/nltk.inference.htmlmodule-nltk.inference
[60]https://www.nltk.org/api/nltk.inference.discourse.html
[61]https://www.nltk.org/api/nltk.inference.nonmonotonic.html
[62]https://www.nltk.org/api/nltk.inference.resolution.html
[63]https://www.javatpoint.com/ai-resolution-in-first-order-logic
[64]https://www.nltk.org/api/nltk.inference.Tableau.html
[65]https://www.nltk.org/book/ch00.html
[66]https://www.nltk.org/api/nltk.metrics.htmlmodule-nltk.metrics
[67]https://www.nltk.org/api/nltk.metrics.agreement.html

- Aline module [68]
  The aline module, in turn, relates to the ALINE algorithm, an algorithm used to align phonetic sequences. This module works as a port of Grzegorz Kondrak's ALINE algorithm [Kondrak, 2002], providing functions for phonetic sequence alignment and similarity analysis.
- Association [69]
  This module provides scoring functions for a variety of association measures through generic and abstract implementations in its BigramAssocMeasures, ContingencyMeasures, NgramAssocMeasures, QuadgramAssocMeasures, and TrigramAssocMeasures classes.
- Distance [70]
  This module allows the calculation of the distance between two items, which typically concern strings.
  The distances included in this module are expressed by the functions:
  * binary_distance
    Which performs a simple equality test returning 0.0 when the labels are identical and 1.0 when they are different.
  * custom_distance
  * edit_distance
    Which is responsible for calculating the Levenshtein edit-distance [Gilleland, 2020] between two strings and allows the specification of the cost associated with substitution edits given the existence of cases where it makes sense to assign higher penalties to substitutions. Optionally, transposition edits can be used - for example, the exchange of "ab" for "ba" - not used by default.
  * edit_distance_align
    This function performs the calculation of the minimum Levenshtein edit-distance based alignment mapping between two strings, this alignment being responsible for determining the mapping that minimizes the edit distance cost.
  * fractional_presence
  * interval_distance
    This function allows the calculation of Krippendorff's interval distance metric, that is, the square of the difference between two labels, which are passed as arguments to the function [71].
  * jaccard_distance
    This function allows the determination of a distance metric that compares the similarity of sets [72].
  * jaro_similarity
    This function calculates the Jaro similarity between two sequences,

[68] https://www.nltk.org/api/nltk.metrics.aline.html
[69] https://www.nltk.org/api/nltk.metrics.association.html
[70] https://www.nltk.org/api/nltk.metrics.distance.html
[71] https://www.nltk.org/$_{m}$*odules/nltk/metrics/distance.htmlinterval*$_{d}$*istance*
[72] https://www.nltk.org/$_{m}$*odules/nltk/metrics/distance.htmljaccard*$_{d}$*istance*

where Jaro distance corresponds to the minimum number of single-character transpositions necessary to transform a given word into another.

  * jaro_winkler_similarity
    This function allows the calculation of the Jaro Winkler distance, an extension of Jaro similarity.
  * masi_distance
    Also, the masi_distance function calculates a distance metric which, in turn, takes partial agreement into account when multiple labels are assigned.
  * presence
    Finally, the presence function refers to a high-order function and is used for testing the presence of a given label, which is passed as input to the function.

- Scores [73]
  This module provides several metrics through the functions:
  * accuracy
  * approxrand
  * f_measure
  * log_likelihood
  * precision
  * recall
- Segmentation [74]
  The present module provides text segmentation metrics that relate to:
  * Windowdiff [https://www.facebook.com/taufiQue74, 2021]
  * Generalized Hamming Distance [Bookstein et al., 2002]
  * Pk text segmentation metric [https://www.facebook.com/taufiQue74, 2021]
- Spearman [75]
  Finally, the Spearman module provides the tools for comparing ranked lists, and these tools refer to the functions:
  * ranks_from_scores
  * ranks_from_sequence
  * spearman_correlation

• Probability and estimation
  Finally, NLTK also provides features related to probability and estimation

[73]https://www.nltk.org/api/nltk.metrics.scores.html
[74]https://www.nltk.org/api/nltk.metrics.segmentation.html
[75]https://www.nltk.org/api/nltk.metrics.spearman.html

through the probability module [76]. This module, in turn, provides classes for processing and representing probabilistic information [77]:

- FreqDist
  This class is used to encode "frequency distributions", which refer to counting the number of occurrences of each outcome in a given experiment.
- ProbDistI
  ProbDistI defines the standard interface for probability distributions. These are responsible for encoding the probability of each outcome for an experiment, and there are two types of probability distribution:
  * "derived probability distributions". These are created based on frequency distributions and aim to model the probability distribution underlying the frequency distribution in question.
  * "analytic probability distributions These are generated based on parameters such as variance, for example.
- ConditionalFreqDist and ConditionalProbDistI
  With the former referring to a class and the latter to an interface, both are used to encode conditional distributions. These distributions can be derived or analytic, and the NLTK only encompasses one implementation of the ConditionalProbDistI interface, called ConditionalProbDist, which corresponds to a derived distribution.

[76]https://www.nltk.org/book/ch00.html
[77]https://www.nltk.org/api/nltk.probability.htmlmodule-nltk.probability

Appendix F

Test dataset with single-INN prescriptions

Prescription
Fazer Carbonato de cálcio + Colecalciferol 1000 mg + 880 U.I., via subcutânea, 1 vez por dia, a começar hoje e durante 3 tomas.
Iniciar Memantina 10 mg, anal, ao jantar, a começar depois de amanhã e sem fim definido.
Iniciar Sildenafil 100 mg, subcutânea, às refeições, a começar amanhã e até dia 12-01-2023.
Tomar Ceftriaxona 1000 mg/3.5 ml, via oral, ao deitar , a começar hoje e até dia 12-01-2023.
Prescrever Valsartan 160 mg, via intravenosa, ao almoço e ao jantar, a começar depois de amanhã e durante 3 tomas.
Fazer Zolpidem 10 mg, anal, ao jantar, a começar hoje e até dia 12-01-2023.
Iniciar Diclofenac 75 mg, intravenosa, ao jantar, a começar hoje e durante 3 tomas.
Fazer Irbesartan + Hidroclorotiazida 150 mg + 12.5 mg, intramuscular, ao jantar, a começar a 03-01-2023 e sem fim definido.
Iniciar Lidocaína 700 mg, vaginal, de 8 em 8 horas, a começar depois de amanhã e sem fim definido.
Iniciar Domperidona 10 mg, via anal, ao almoço, a começar depois de amanhã e durante 3 tomas.
Prescrever Captopril 25 mg, via intramuscular, em jejum, a começar depois de amanhã e durante 3 tomas.
Tomar Valsartan + Hidroclorotiazida 160 mg + 12.5 mg, intramuscular, 1 vez por dia, a começar amanhã e sem fim definido.
Tomar Irbesartan 150 mg, via subcutânea, às refeições, a começar hoje e durante 7 dias.
Tomar Folitropina alfa 75 U.I./0.125 ml, subcutânea, 1 vez por dia, a começar a 03-01-2023 e sem fim definido.
Tomar Enoxaparina sódica 20 mg/0.2 ml, via intramuscular, 1 vez por dia, a começar amanhã e até dia 12-01-2023.

Tomar Rasagilina 1 mg, anal, às refeições, a começar amanhã e até dia 12-01-2023.
Fazer Pravastatina 40 mg, via anal, 1 vez por dia, a começar a 03-01-2023 e sem fim definido.
Iniciar Prasugrel 5 mg, vaginal, ao deitar , a começar depois de amanhã e durante 7 dias.
Tomar Clopidogrel 75 mg, via vaginal, em jejum, a começar a 03-01-2023 e sem fim definido.
Tomar Citrato de potássio 1080 mg, via vaginal, de 12 em 12 horas, a começar depois de amanhã e até dia 12-01-2023.
Fazer Beta-histina 16 mg, vaginal, ao almoço, a começar amanhã e até dia 12-01-2023.
Tomar Apixabano 2.5 mg, intravenosa, ao deitar , a começar hoje e durante 3 tomas.
Tomar Gabapentina 800 mg, oral, ao deitar , a começar a 03-01-2023 e durante 3 tomas.
Tomar Sitagliptina 50 mg, oral, ao deitar , a começar depois de amanhã e durante 7 dias.
Fazer Álcool diclorobenzílico + Amilmetacresol 1.2 mg + 0.6 mg, intramuscular, 3 vezes por dia, a começar amanhã e durante 7 dias.
Iniciar Pioglitazona 15 mg, via anal, em jejum, a começar a 03-01-2023 e sem fim definido.
Tomar Candesartan 16 mg, intramuscular, 1 vez por dia, a começar a 03-01-2023 e até dia 12-01-2023.
Tomar Loratadina 10 mg, via anal, ao jantar, a começar hoje e até dia 12-01-2023.
Tomar Fosfomicina 3000 mg, anal, de 12 em 12 horas, a começar hoje e sem fim definido.
Prescrever Ácido acetilsalicílico 500 mg, intravenosa, 1 vez por dia, a começar depois de amanhã e sem fim definido.
Prescrever Tramadol 100 mg, anal, 4 vezes ao dia, a começar hoje e durante 7 dias.
Prescrever Pantoprazol 20 mg, vaginal, em jejum, a começar a 03-01-2023 e sem fim definido.
Tomar Tramadol 100 mg/ml, intravenosa, de 8 em 8 horas, a começar depois de amanhã e durante 3 tomas.
Fazer Metformina + Dapagliflozina 1000 mg + 5 mg, via intramuscular, em jejum, a começar amanhã e durante 3 tomas.
Iniciar Amlodipina + Atorvastatina 5 mg + 10 mg, intravenosa, 4 vezes ao dia, a começar depois de amanhã e durante 7 dias.
Prescrever Perindopril 4 mg, subcutânea, de 12 em 12 horas, a começar depois de amanhã e durante 3 tomas.
Fazer Pregabalina 300 mg, via intravenosa, ao deitar , a começar amanhã e durante 7 dias.
Iniciar Alginato de sódio + Bicarbonato de sódio + Carbonato de cálcio 250 mg + 133.5 mg + 80 mg, via intramuscular, de 8 em 8 horas, a começar a 03-01-2023 e sem fim definido.

Prescrever Claritromicina 500 mg, anal, 3 vezes por dia, a começar a 03-01-2023 e durante 7 dias.
Fazer Desloratadina 5 mg, via intramuscular, ao deitar , a começar depois de amanhã e durante 3 tomas.
Fazer Iodeto de tibezónio 5 mg, via anal, 2 vezes ao dia, a começar depois de amanhã e sem fim definido.
Prescrever Racecadotril 10 mg, via oral, 3 vezes por dia, a começar depois de amanhã e até dia 12-01-2023.
Fazer Perampanel 0.5 mg/ml, vaginal, ao deitar , a começar a 03-01-2023 e sem fim definido.
Prescrever Tianeptina 12.5 mg, subcutânea, ao almoço e ao jantar, a começar a 03-01-2023 e sem fim definido.
Tomar Montelucaste 5 mg, anal, de 12 em 12 horas, a começar amanhã e durante 7 dias.
Iniciar Enalapril 5 mg, via subcutânea, às refeições, a começar hoje e sem fim definido.
Fazer Rivastigmina 2 mg/ml, intravenosa, ao almoço, a começar a 03-01-2023 e sem fim definido.
Iniciar Mirtazapina 15 mg, via anal, de 12 em 12 horas, a começar amanhã e até dia 12-01-2023.
Fazer Ivabradina 7.5 mg, intramuscular, 1 vez por dia, a começar amanhã e durante 3 tomas.
Prescrever Adrenalina 0.15 mg/0.3 ml, via anal, 1 vez por dia, a começar hoje e durante 7 dias.

Table F.1: Test dataset for the NLP solution's test with single-INN prescriptions.

Appendix G

Confusion matrices

G.1 Amazon Comprehend

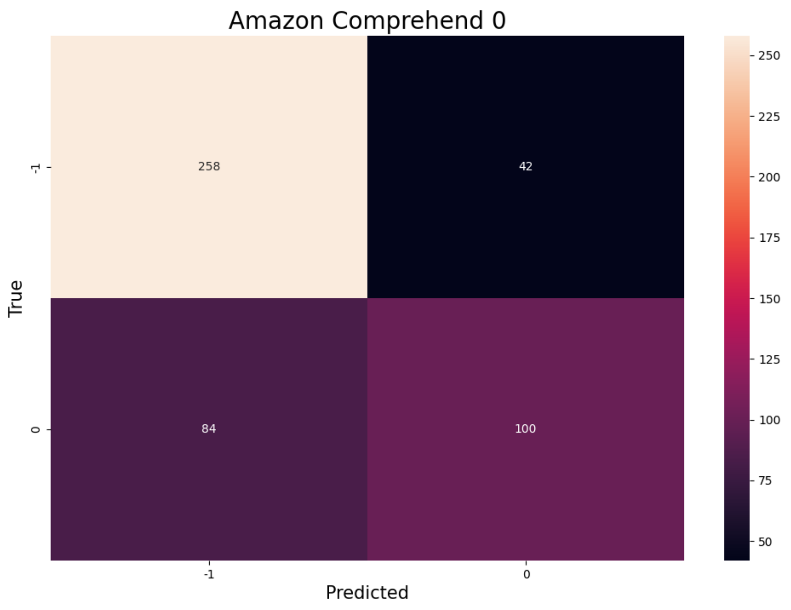


Figure G.1: Confusion matrix obtained for negative entity recognition with Amazon Comprehend.

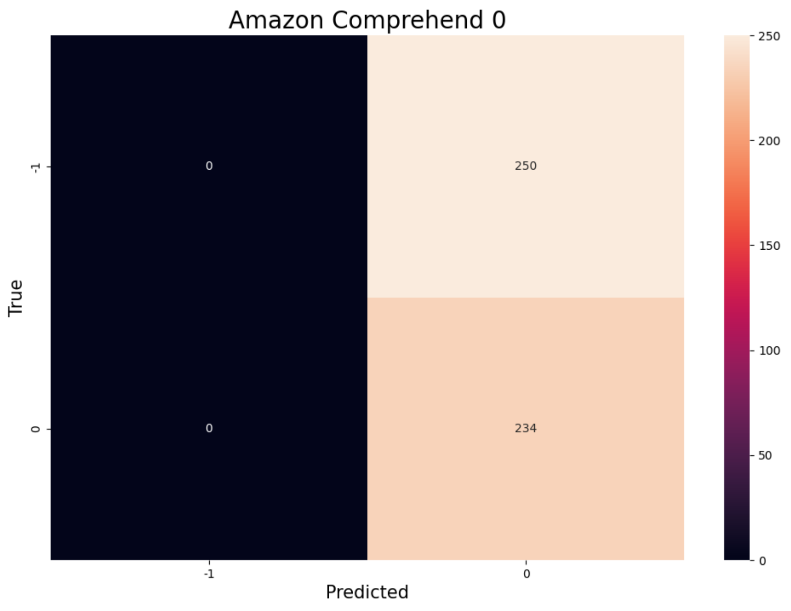


Figure G.2: Confusion matrix obtained for negative relation detection with Amazon Comprehend.

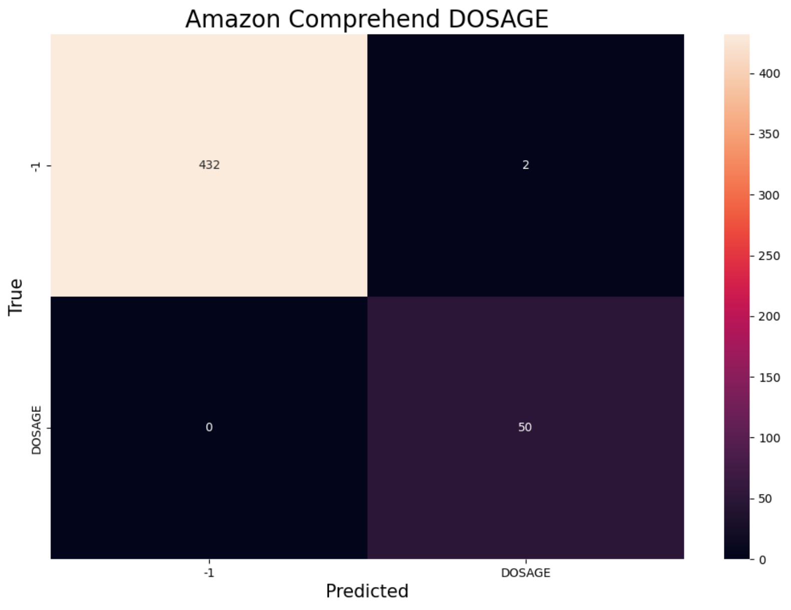


Figure G.3: Confusion matrix obtained for dosage entity recognition with Amazon Comprehend.

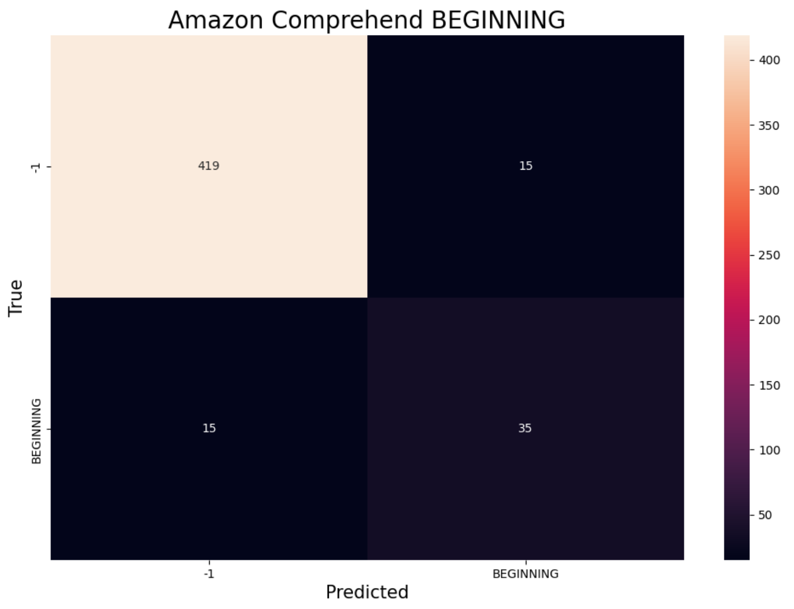


Figure G.4: Confusion matrix obtained for beginning entity recognition with Amazon Comprehend.

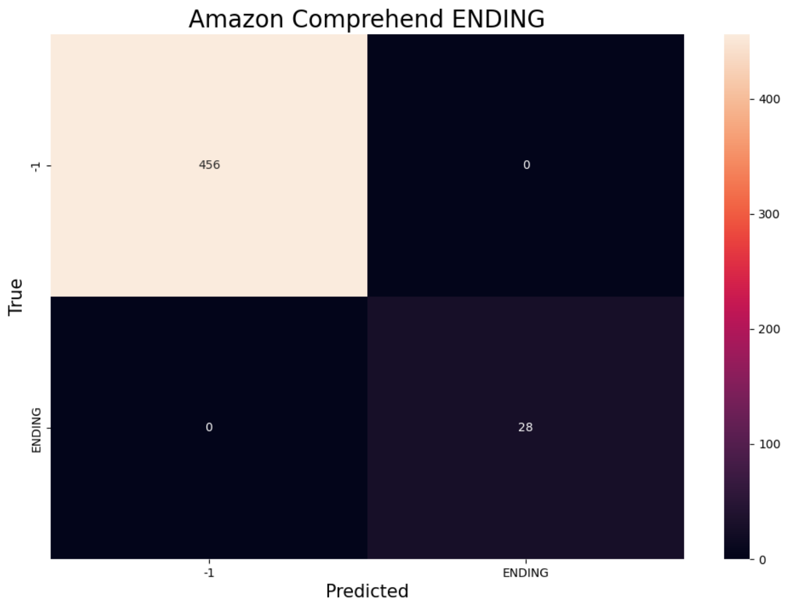


Figure G.5: Confusion matrix obtained for ending entity recognition with Amazon Comprehend.

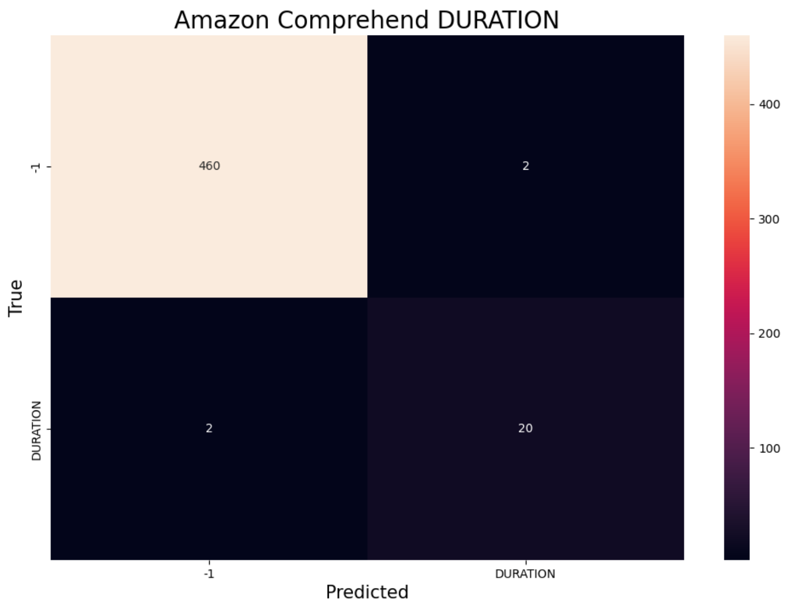


Figure G.6: Confusion matrix obtained for duration entity recognition with Amazon Comprehend.

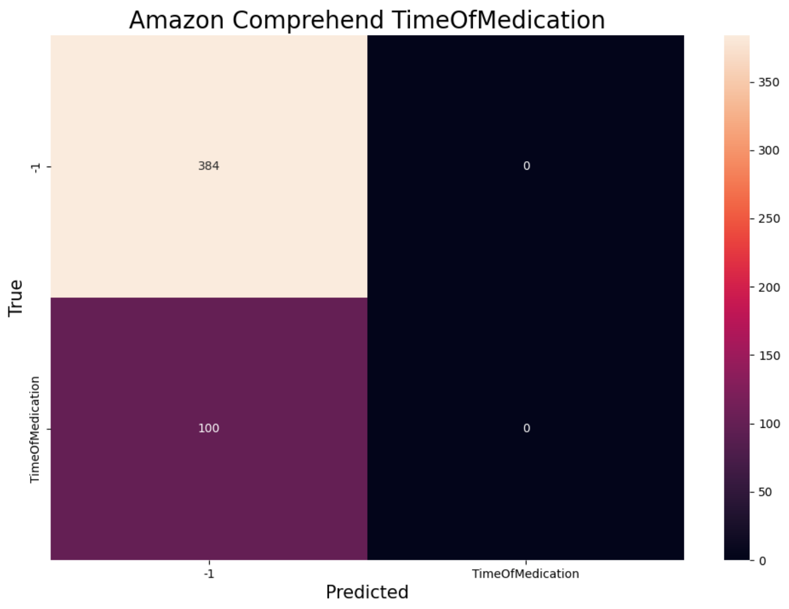


Figure G.7: Confusion matrix obtained for time of medication relation detection with Amazon Comprehend.

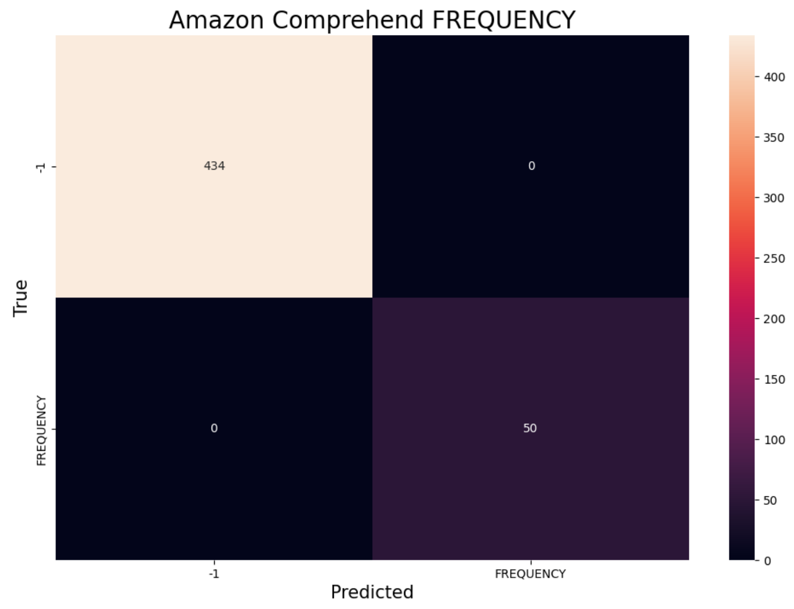


Figure G.8: Confusion matrix obtained for frequency entity recognition with Amazon Comprehend.

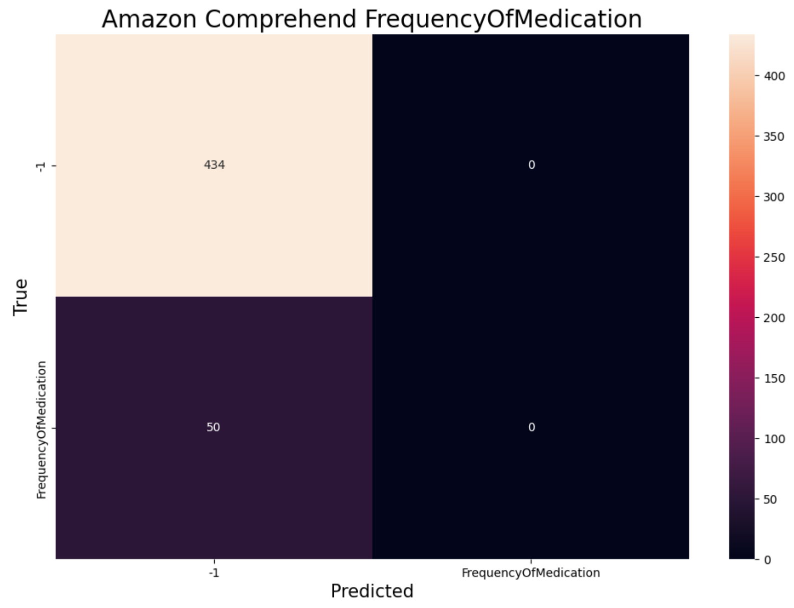


Figure G.9: Confusion matrix obtained for frequency of medication relation detection with Amazon Comprehend.

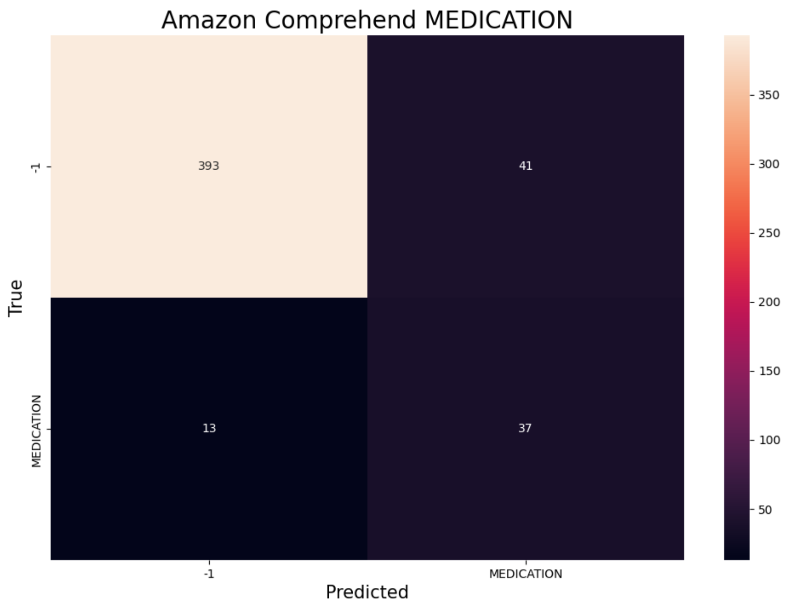


Figure G.10: Confusion matrix obtained for medication entity recognition with Amazon Comprehend.

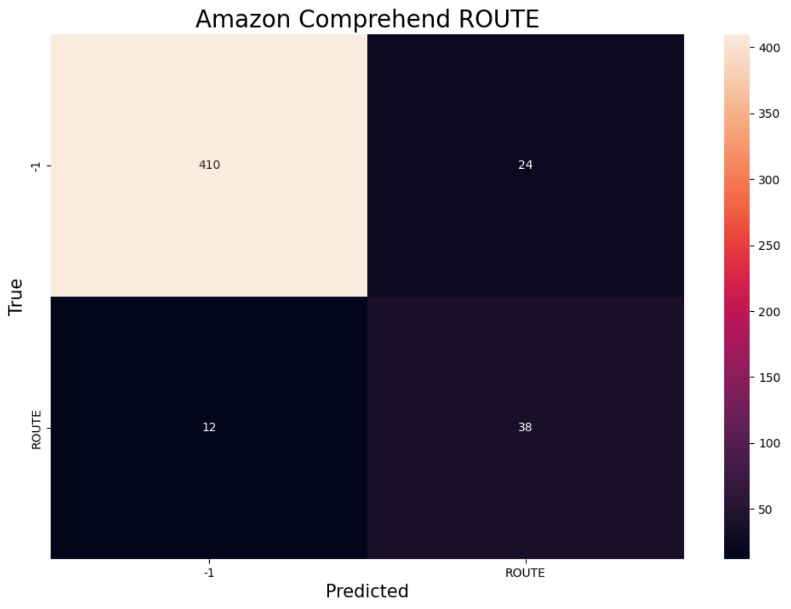


Figure G.11: Confusion matrix obtained for route entity recognition with Amazon Comprehend.

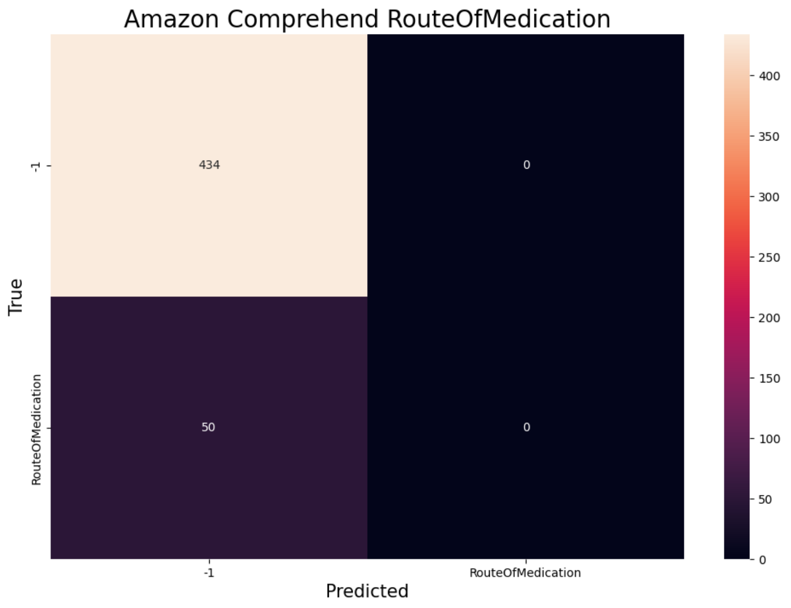


Figure G.12: Confusion matrix obtained for route of medication relation detection with Amazon Comprehend.

G.2 Google Cloud Healthcare Natural Language API

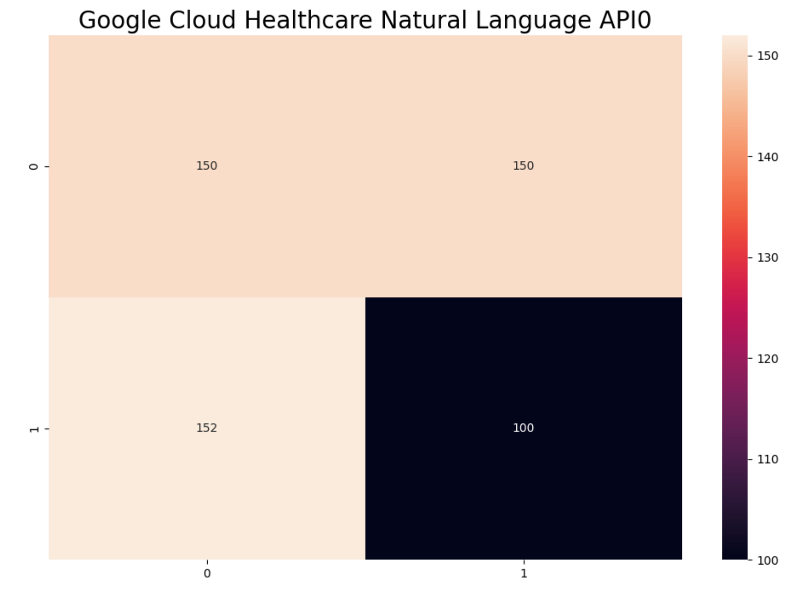


Figure G.13: Confusion matrix obtained for negative entities recognition with Google Cloud Healthcare Natural Language API.

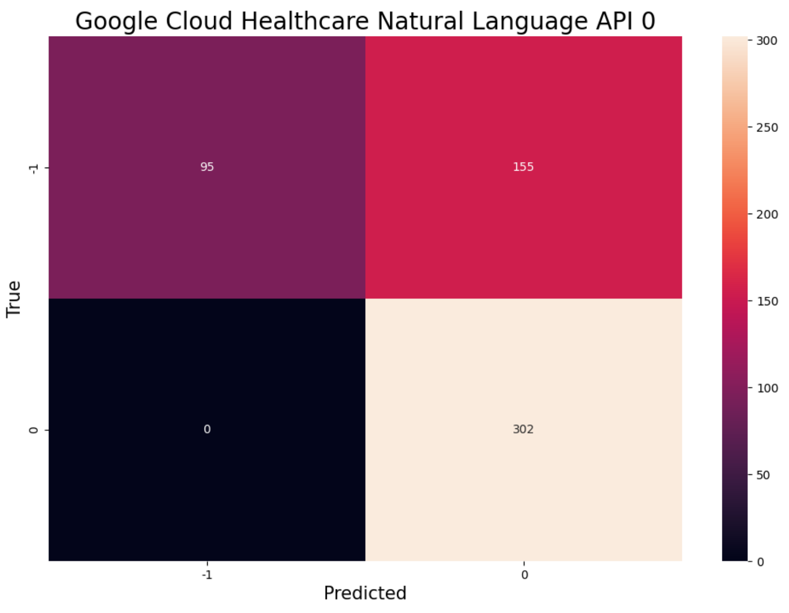


Figure G.14: Confusion matrix obtained for negative relationship detection with Google Cloud Healthcare Natural Language API.

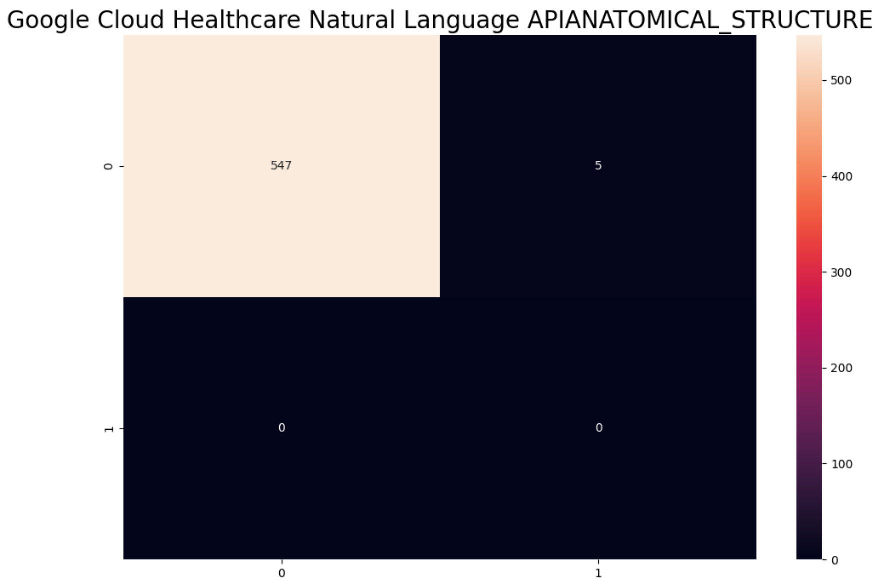


Figure G.15: Confusion matrix obtained for anatomical structure entities recognition with Google Cloud Healthcare Natural Language API.

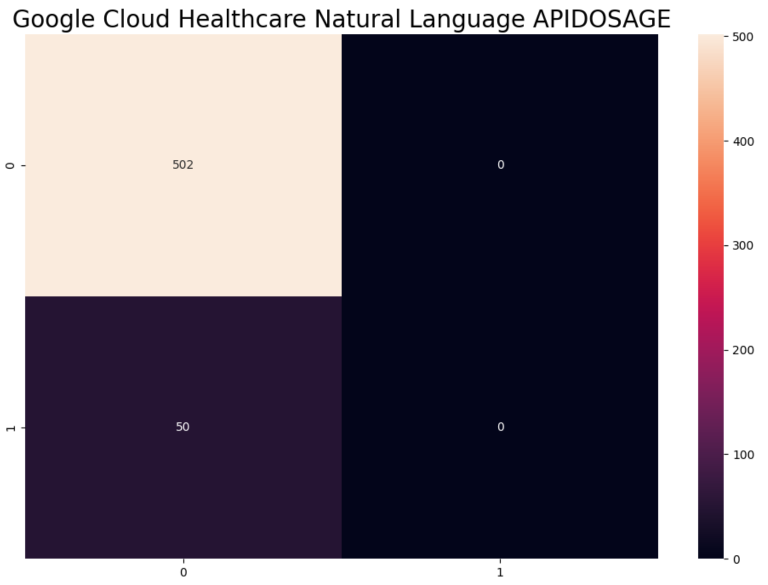


Figure G.16: Confusion matrix obtained for dosage entities recognition with Google Cloud Healthcare Natural Language API.

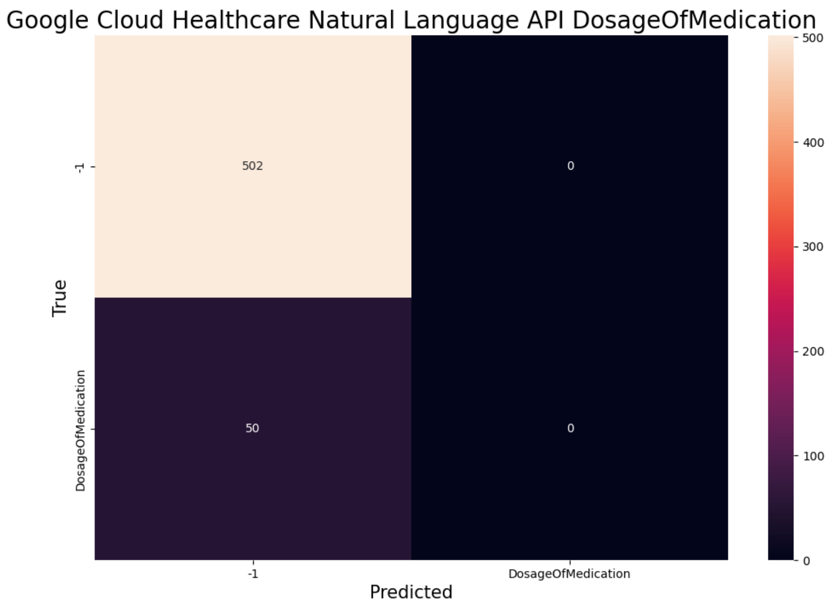


Figure G.17: Confusion matrix obtained for dosage of medication relationship detection with Google Cloud Healthcare Natural Language API.

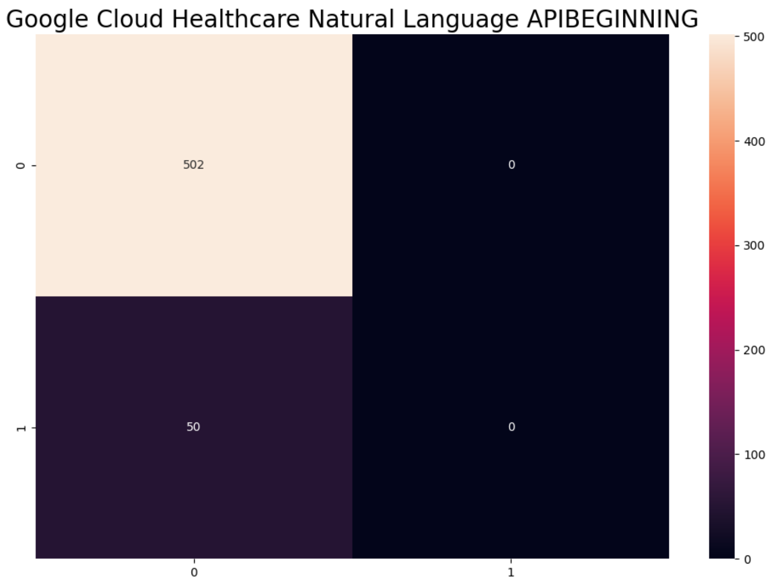


Figure G.18: Confusion matrix obtained for beginning entities recognition with Google Cloud Healthcare Natural Language API.

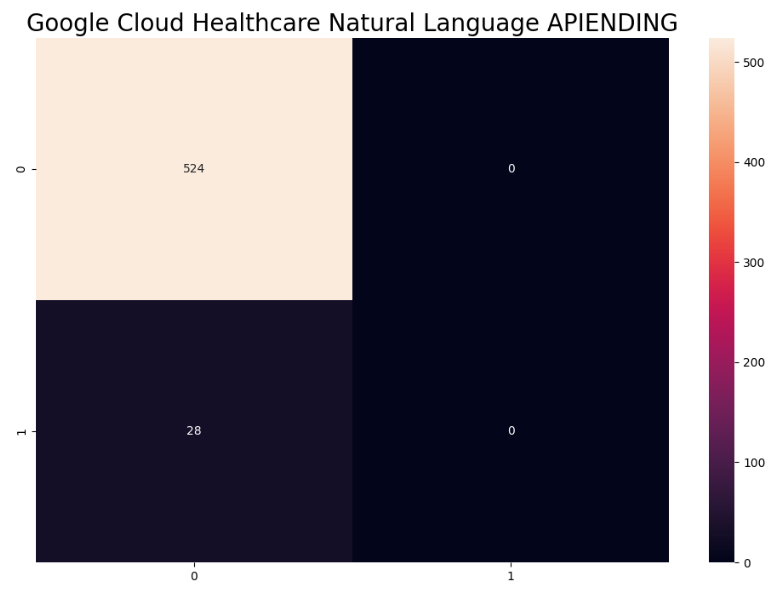


Figure G.19: Confusion matrix obtained for ending entities recognition with Google Cloud Healthcare Natural Language API.

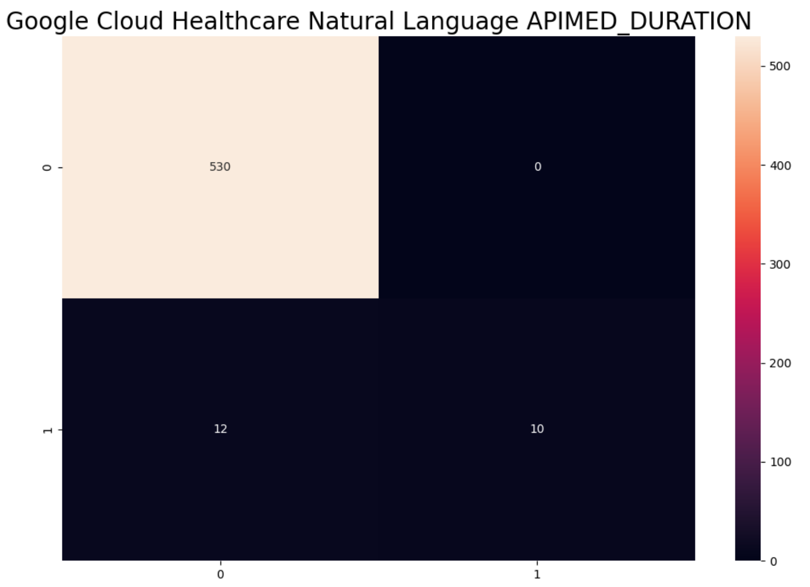


Figure G.20: Confusion matrix obtained for duration entities recognition with Google Cloud Healthcare Natural Language API.

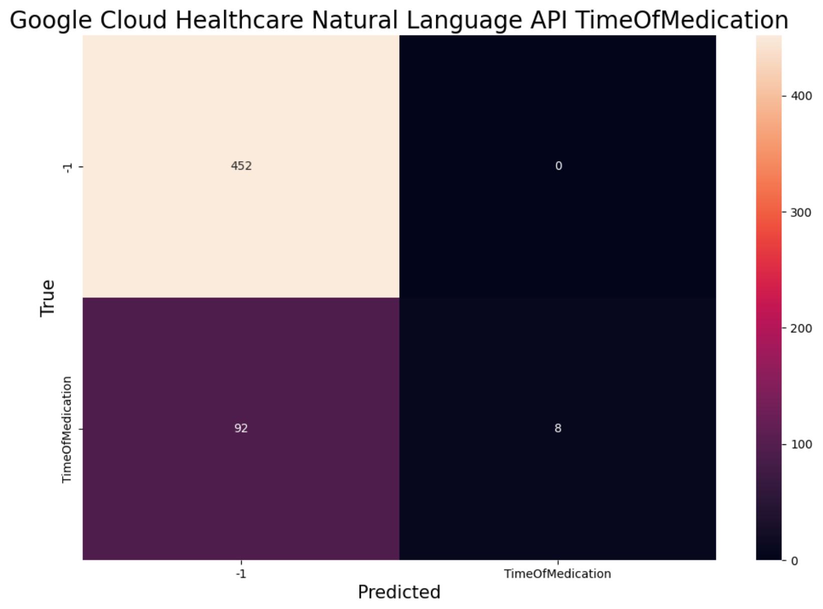


Figure G.21: Confusion matrix obtained for time of medication relationship detection with Google Cloud Healthcare Natural Language API.

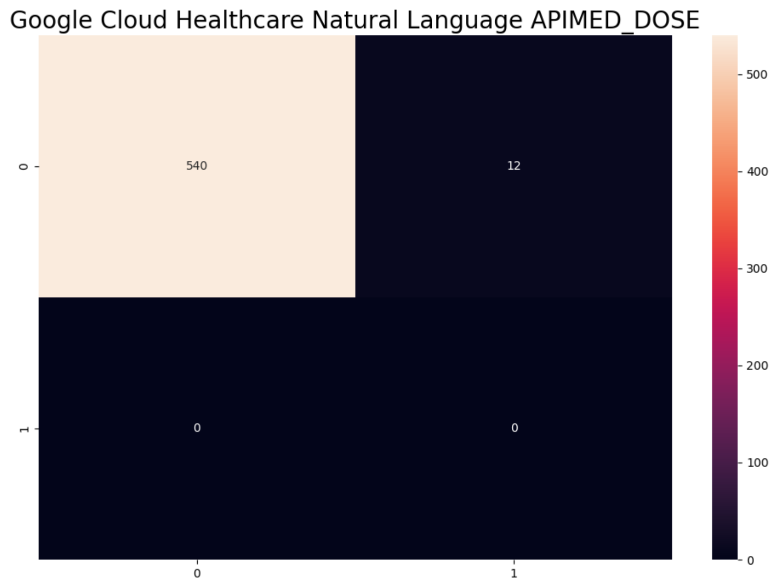


Figure G.22: Confusion matrix obtained for dose entities recognition with Google Cloud Healthcare Natural Language API.

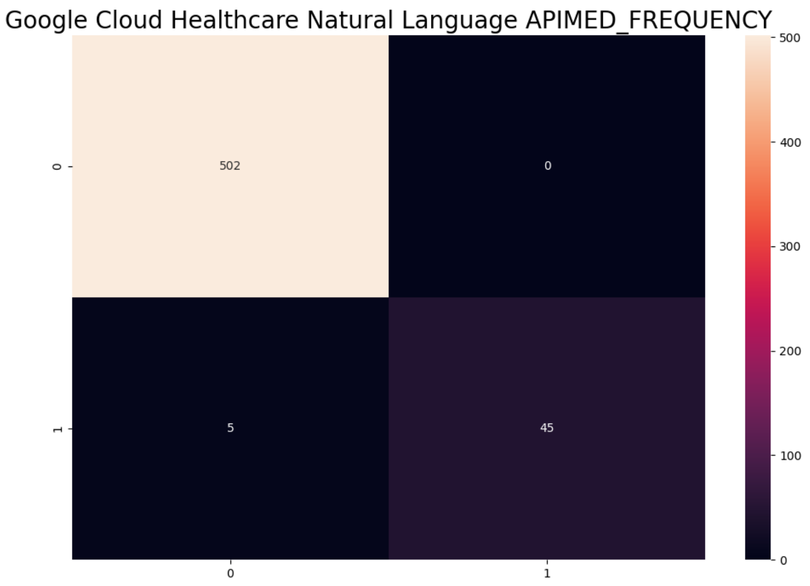


Figure G.23: Confusion matrix obtained for frequency entities recognition with Google Cloud Healthcare Natural Language API.

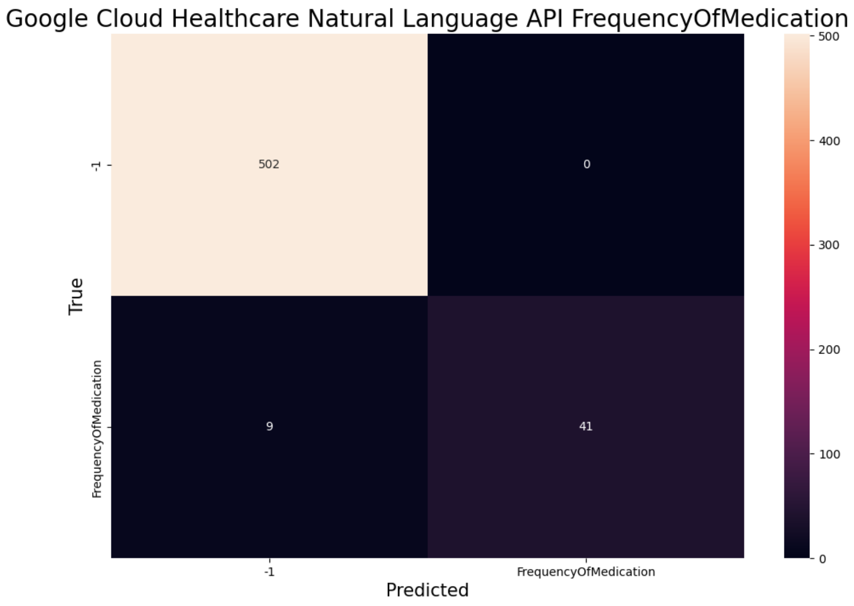


Figure G.24: Confusion matrix obtained for frequency of medication relationship detection with Google Cloud Healthcare Natural Language API.

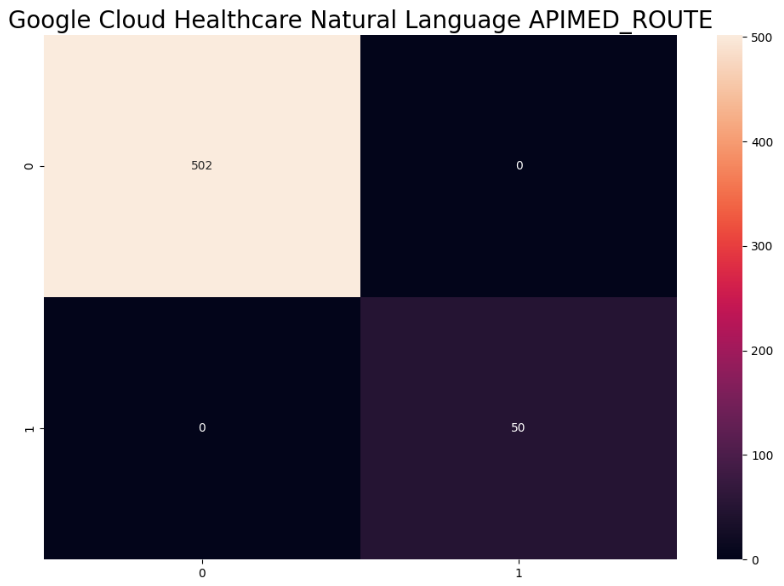


Figure G.25: Confusion matrix obtained for route entities recognition with Google Cloud Healthcare Natural Language API.

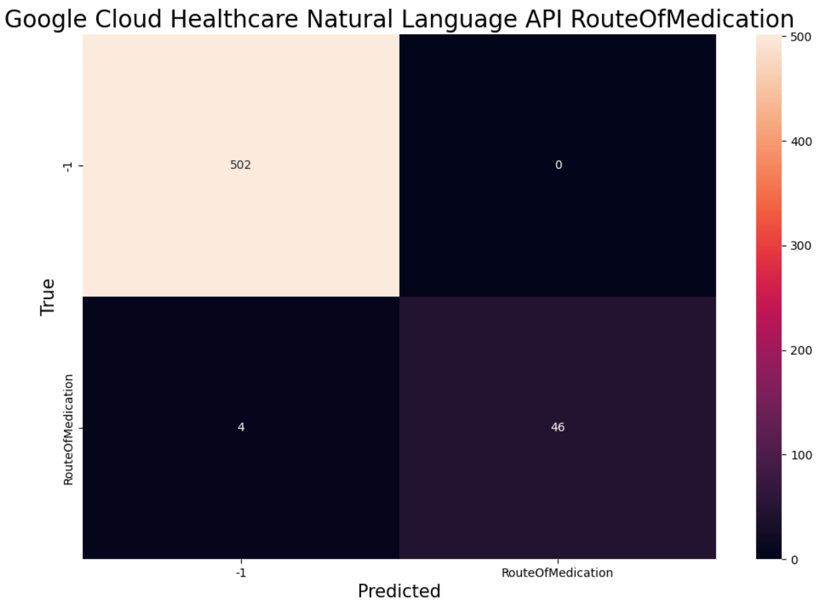


Figure G.26: Confusion matrix obtained for route of medication relationship detection with Google Cloud Healthcare Natural Language API.

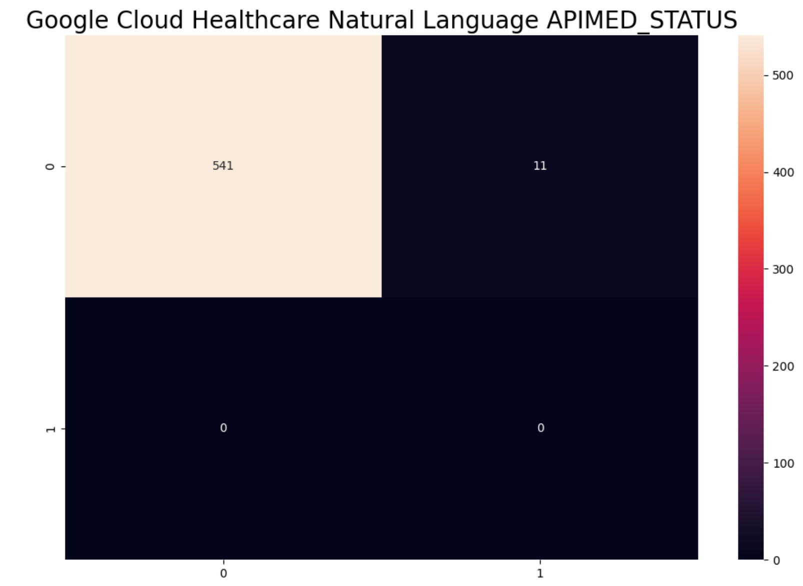


Figure G.27: Confusion matrix obtained for status of medication entities recognition with Google Cloud Healthcare Natural Language API.

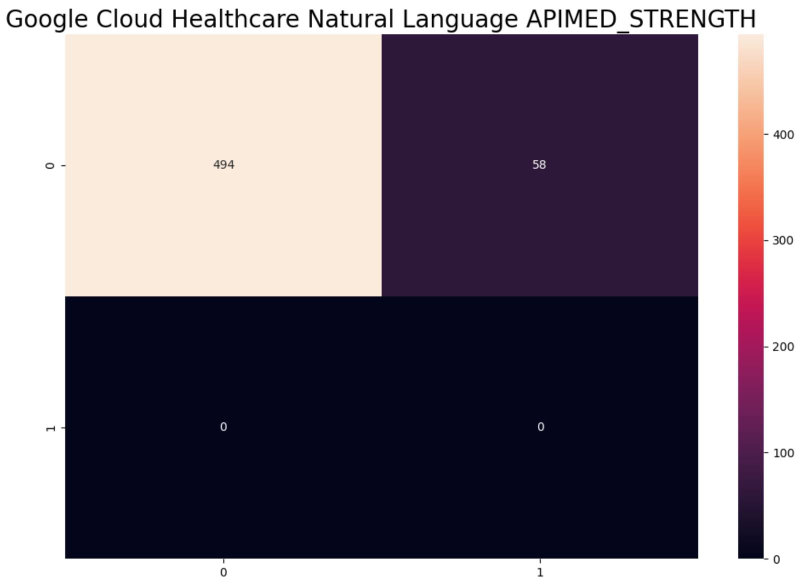


Figure G.28: Confusion matrix obtained for medication strength entities recognition with Google Cloud Healthcare Natural Language API.

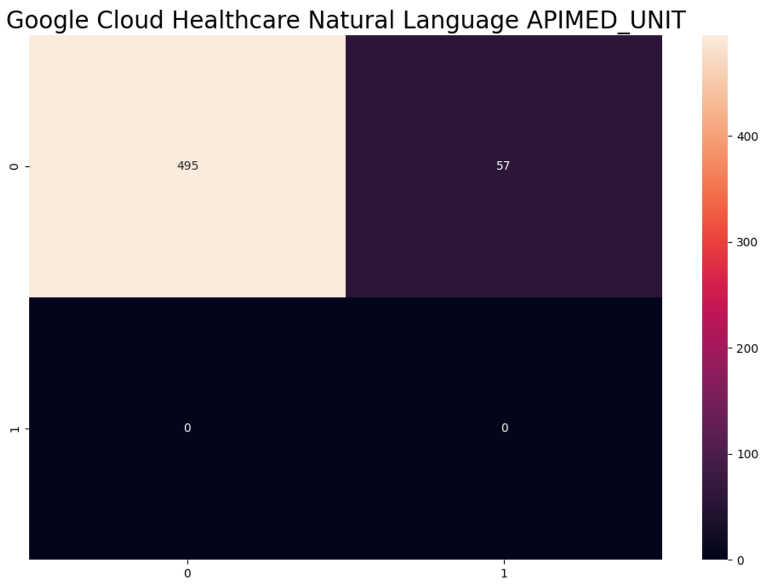


Figure G.29: Confusion matrix obtained for medication unit entities recognition with Google Cloud Healthcare Natural Language API.

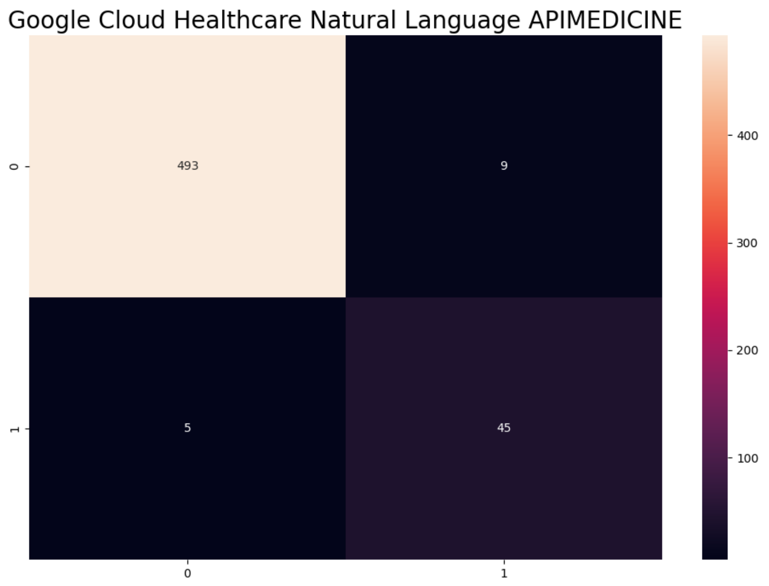


Figure G.30: Confusion matrix obtained for medication entities recognition with Google Cloud Healthcare Natural Language API.

G.2.1 After post-processing

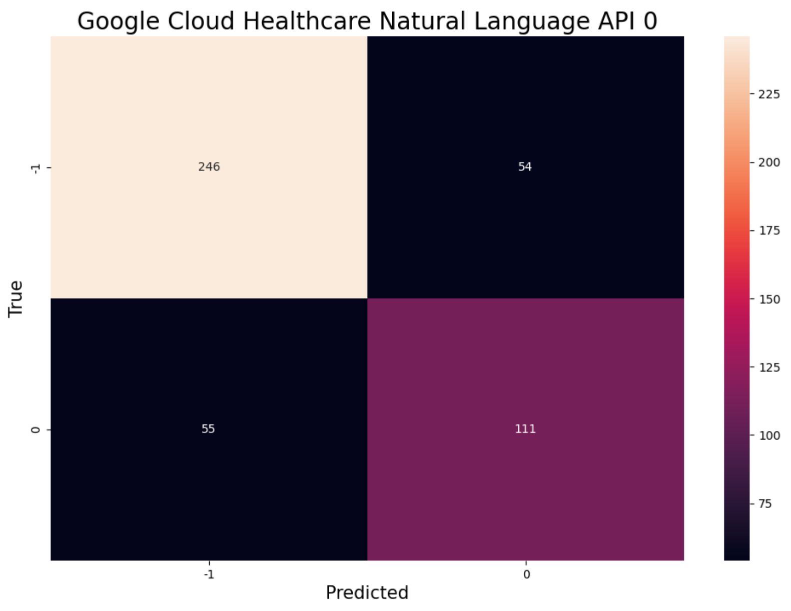


Figure G.31: Confusion matrix obtained for negative entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

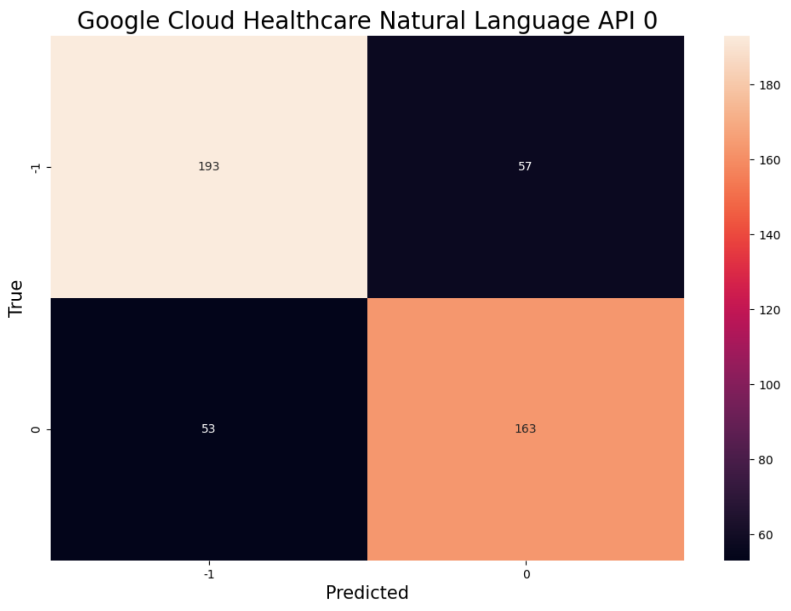


Figure G.32: Confusion matrix obtained for negative relationship detection with Google Cloud Healthcare Natural Language API after post-processing.

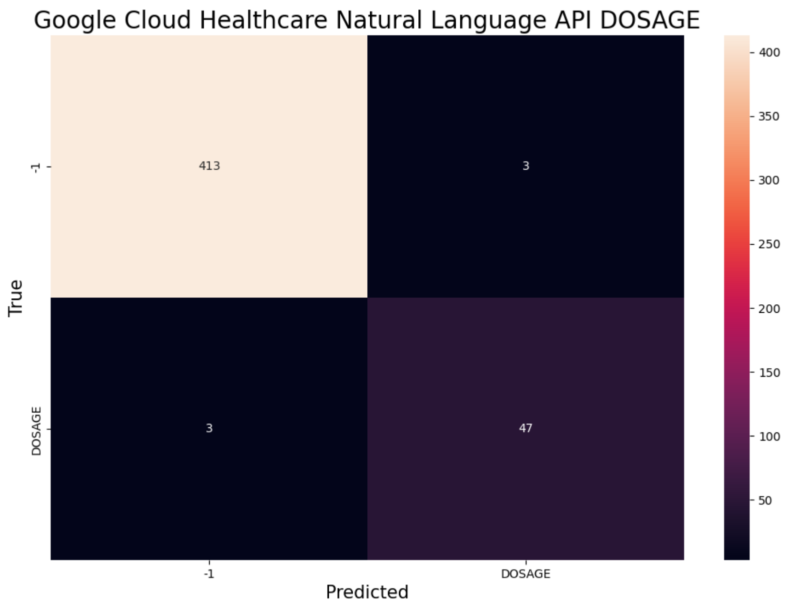


Figure G.33: Confusion matrix obtained for dosage entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

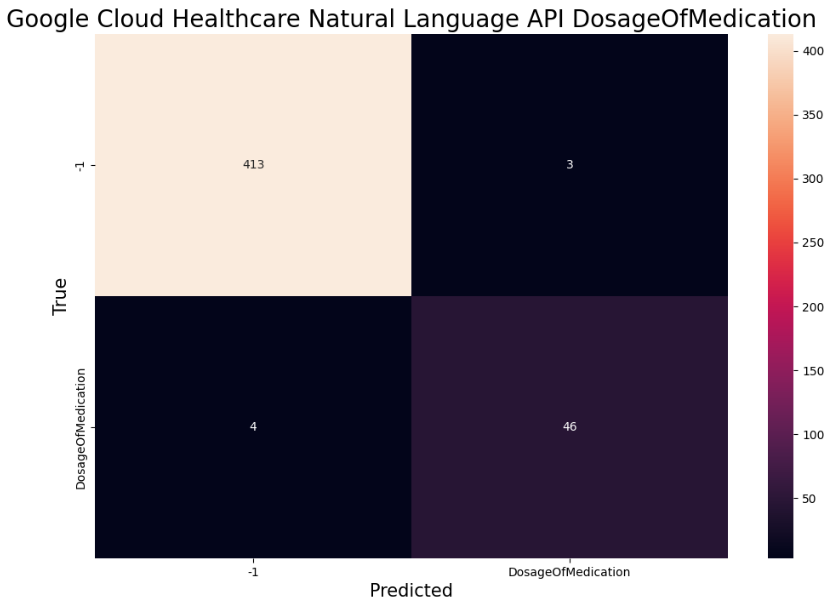


Figure G.34: Confusion matrix obtained for dosage of medication relationship detection with Google Cloud Healthcare Natural Language API after post-processing.

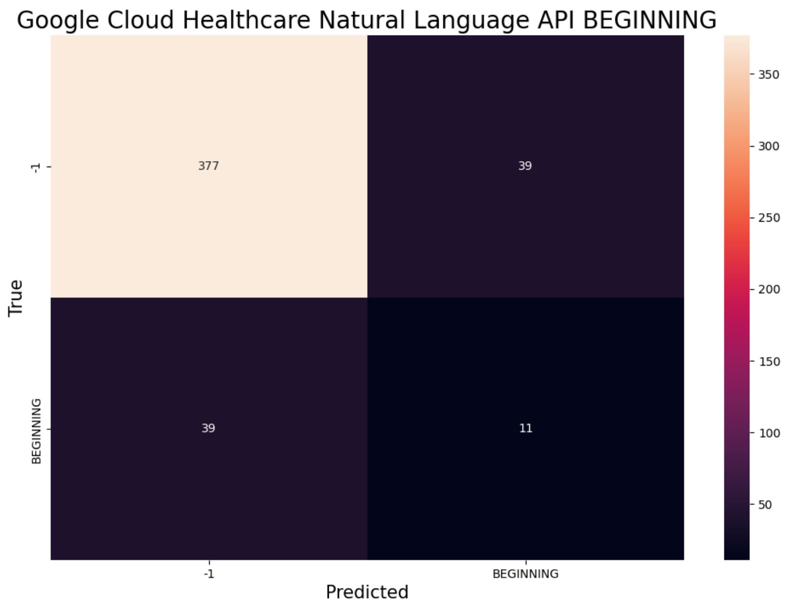


Figure G.35: Confusion matrix obtained for beginning entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

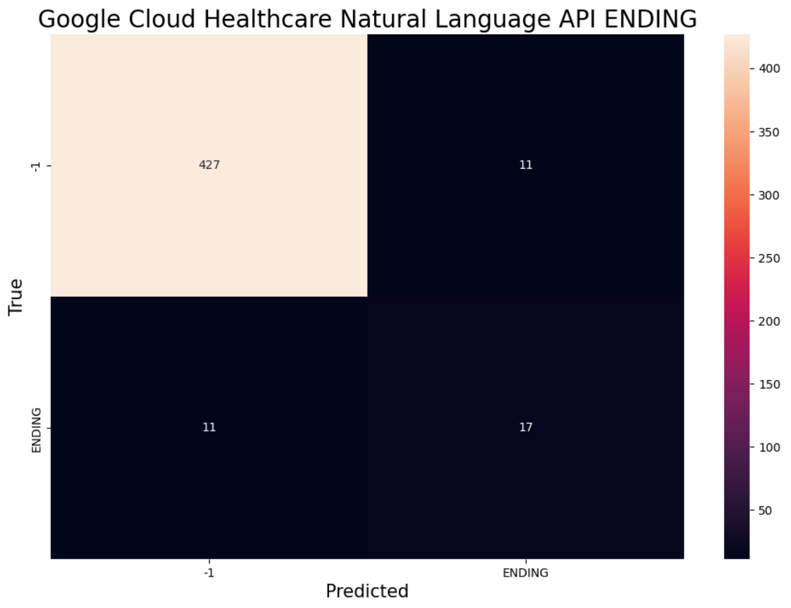


Figure G.36: Confusion matrix obtained for ending entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

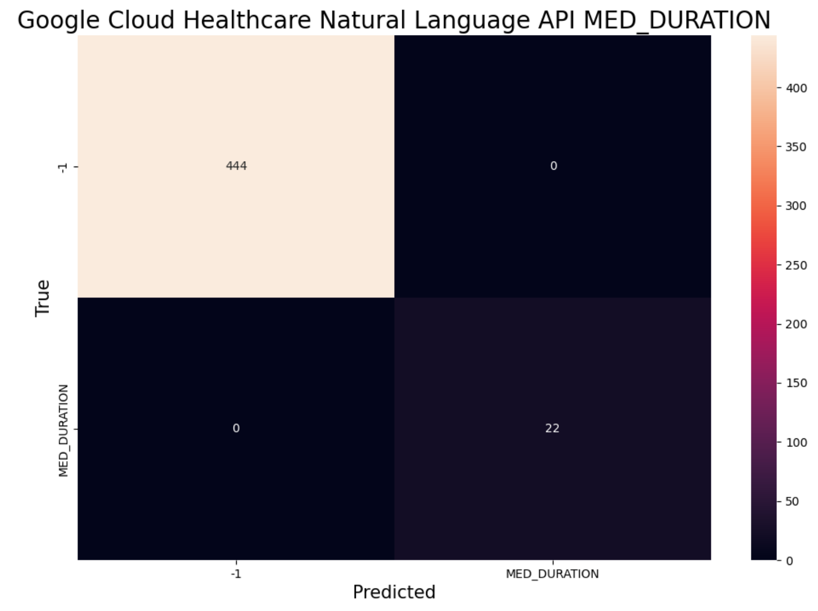


Figure G.37: Confusion matrix obtained for duration entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

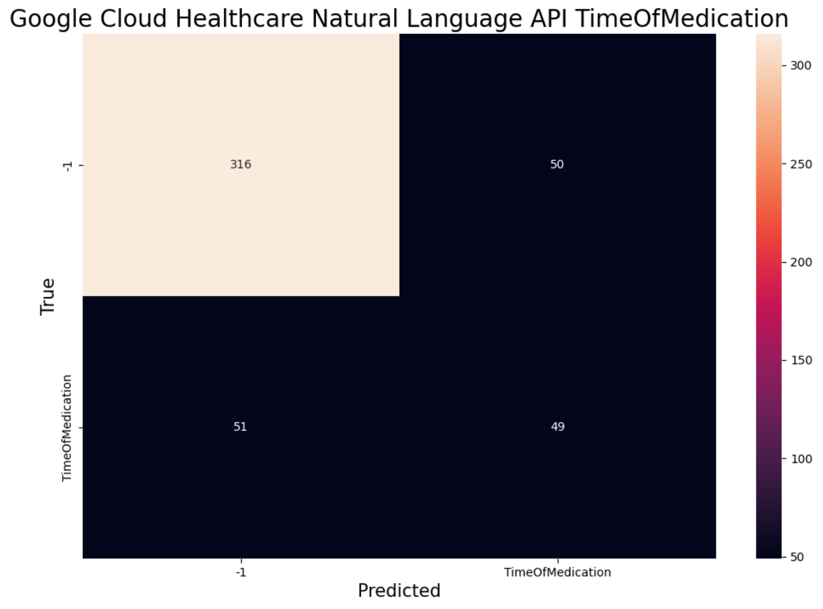


Figure G.38: Confusion matrix obtained for time of medication relationship detection with Google Cloud Healthcare Natural Language API after post-processing.

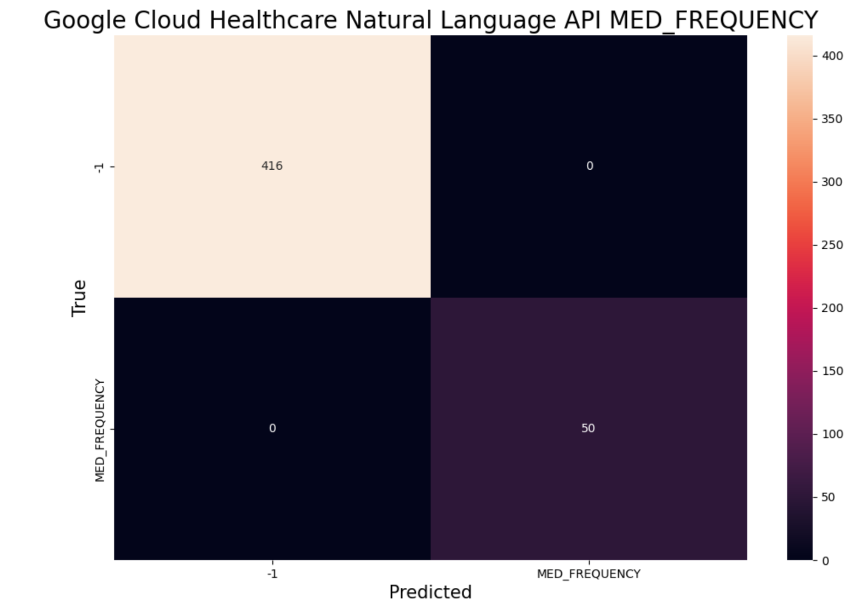


Figure G.39: Confusion matrix obtained for frequency entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

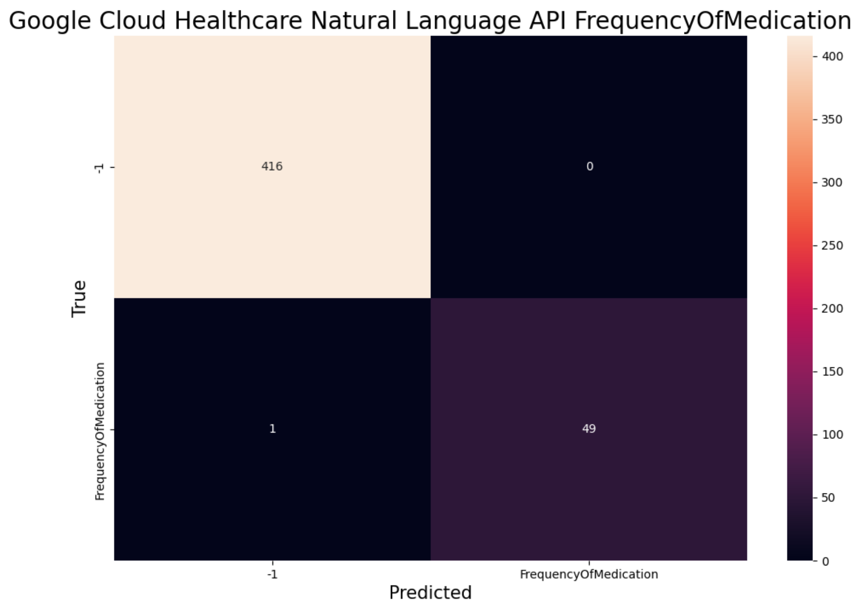


Figure G.40: Confusion matrix obtained for frequency of medication relationship detection with Google Cloud Healthcare Natural Language API after post-processing.

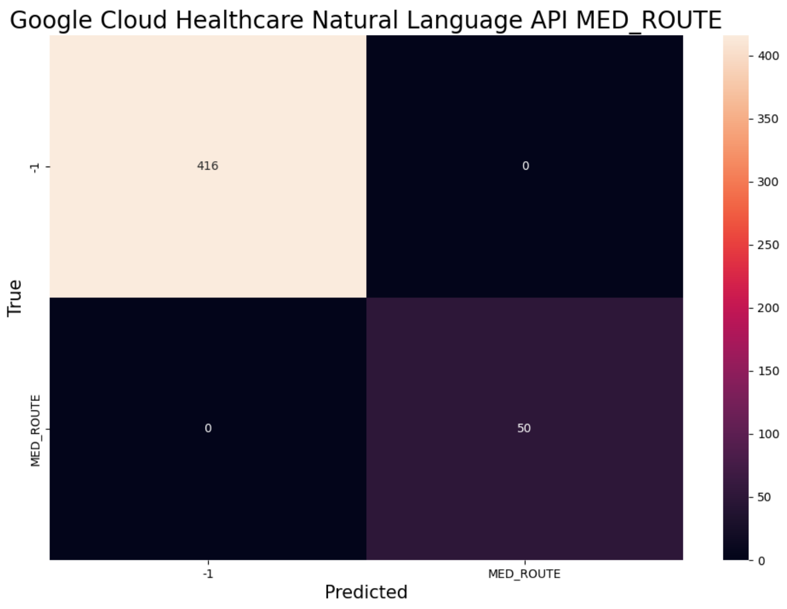


Figure G.41: Confusion matrix obtained for route entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

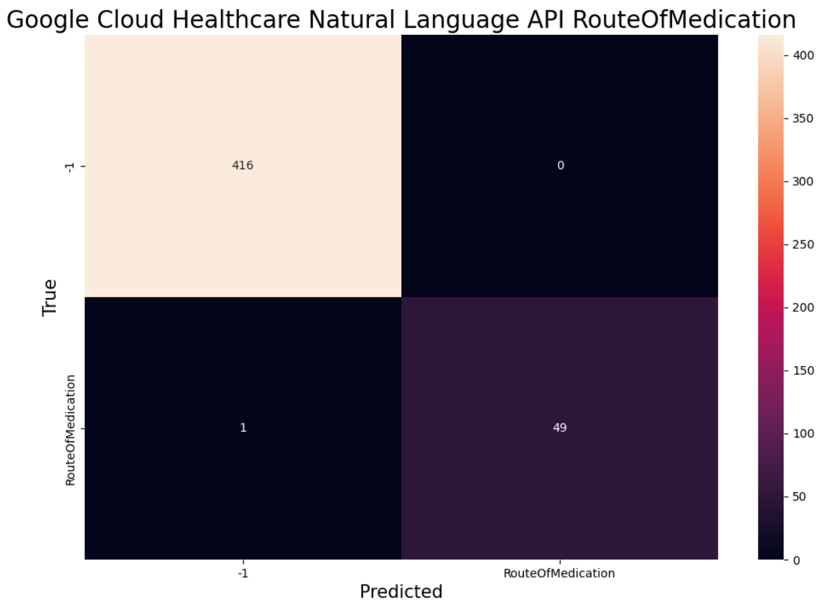


Figure G.42: Confusion matrix obtained for route of medication relationship detection with Google Cloud Healthcare Natural Language API after post-processing.

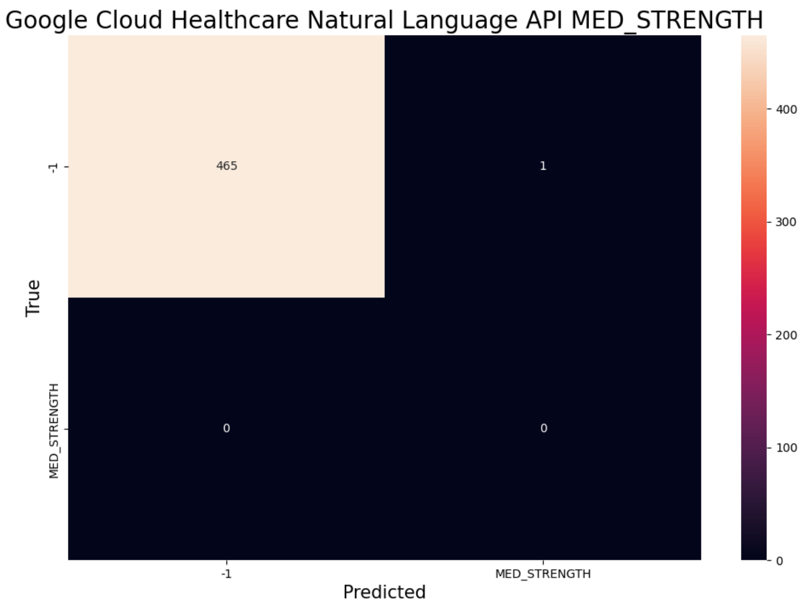


Figure G.43: Confusion matrix obtained for medication strength entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

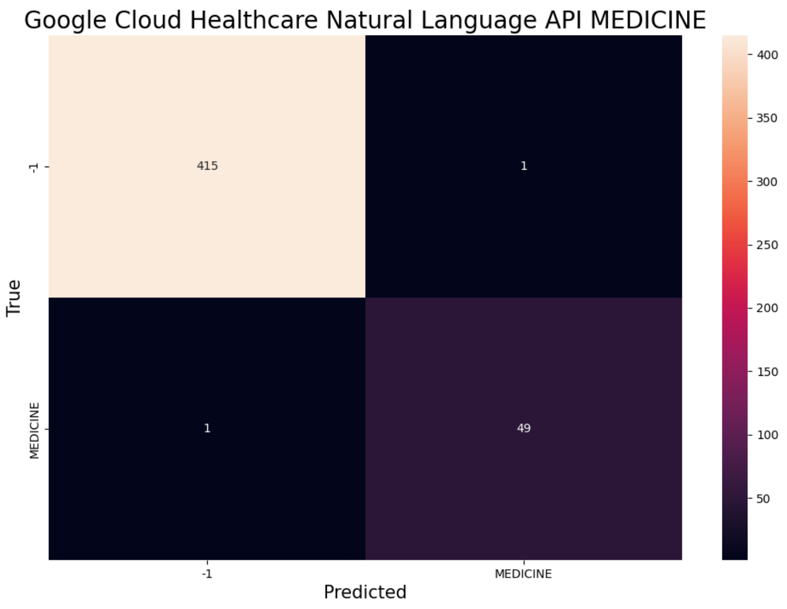


Figure G.44: Confusion matrix obtained for medication entities recognition with Google Cloud Healthcare Natural Language API after post-processing.

G.3 Microsoft Azure Cognitive Service for Language

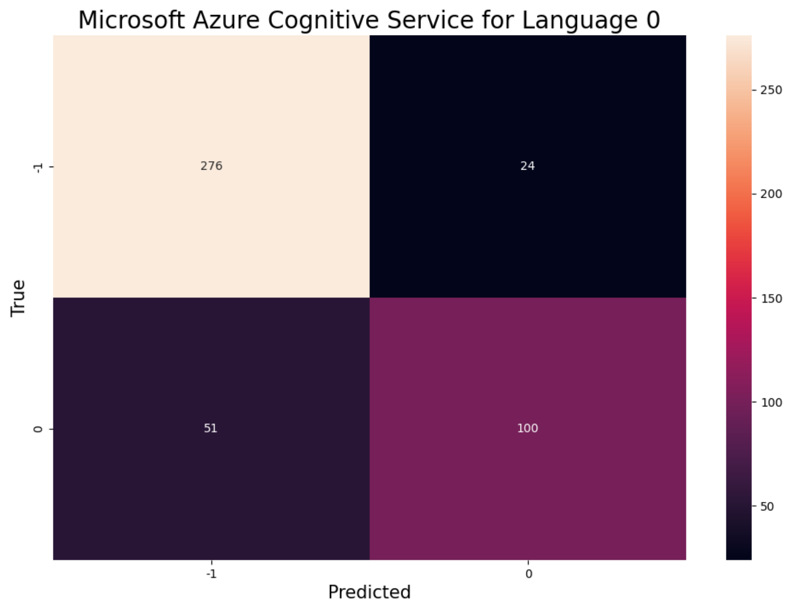


Figure G.45: Confusion matrix obtained for negative entity recognition with Microsoft Azure Cognitive Service for Language.

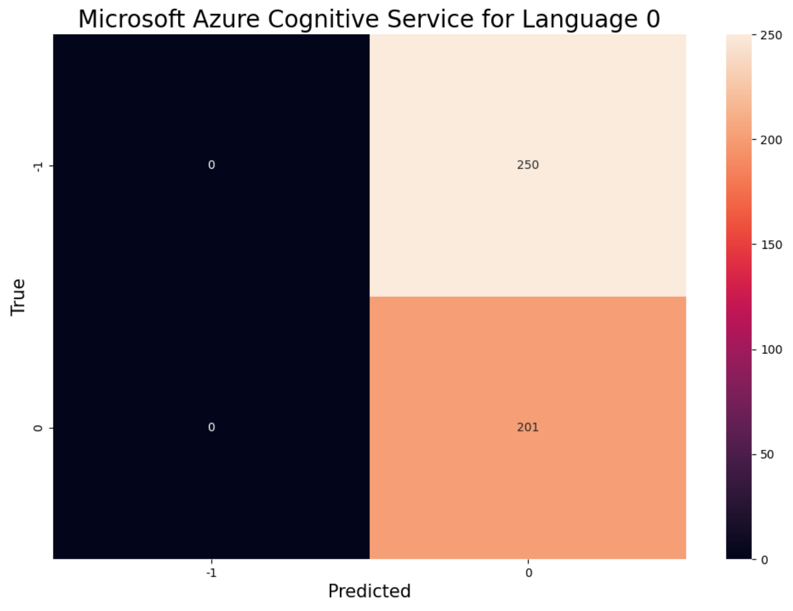


Figure G.46: Confusion matrix obtained for negative relationship detection with Microsoft Azure Cognitive Service for Language.

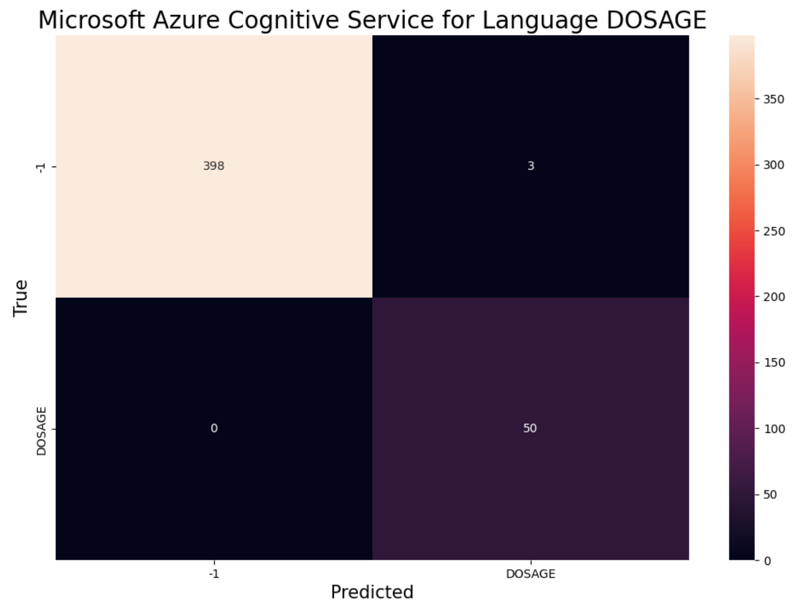


Figure G.47: Confusion matrix obtained for dosage entity recognition with Microsoft Azure Cognitive Service for Language.

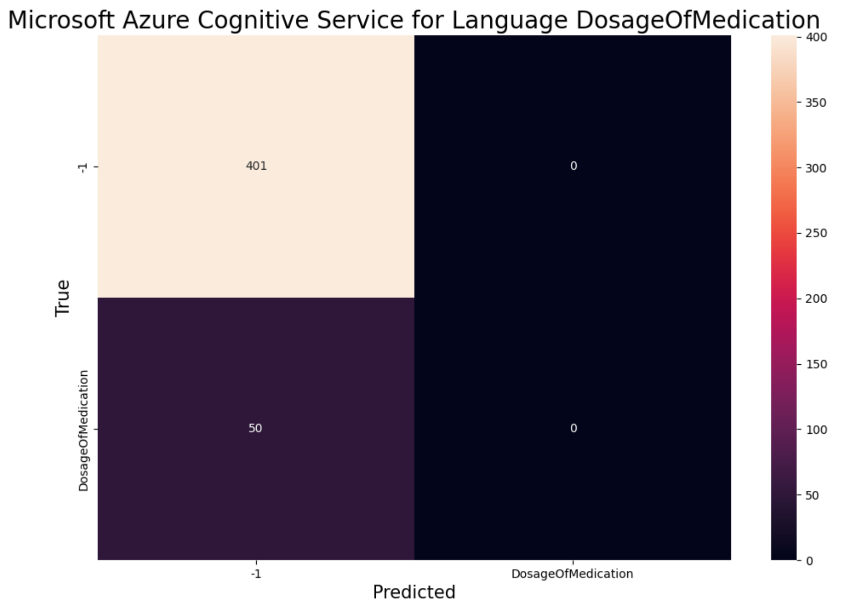


Figure G.48: Confusion matrix obtained for dosage of medication relationship detection with Microsoft Azure Cognitive Service for Language.

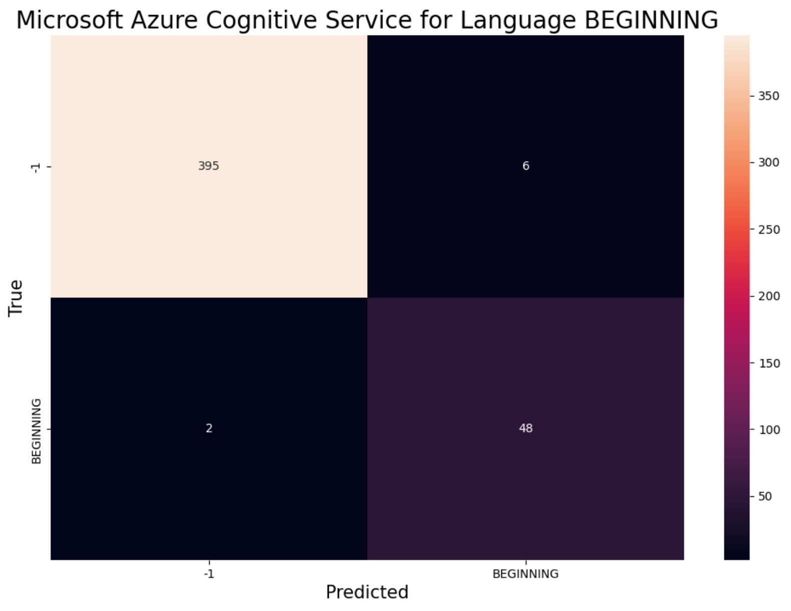


Figure G.49: Confusion matrix obtained for beginning entity recognition with Microsoft Azure Cognitive Service for Language.

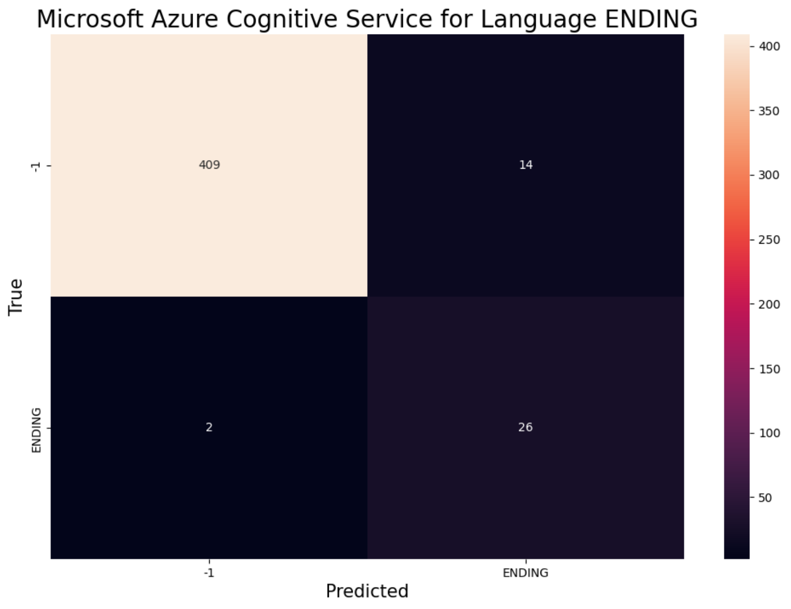


Figure G.50: Confusion matrix obtained for ending entity recognition with Microsoft Azure Cognitive Service for Language.

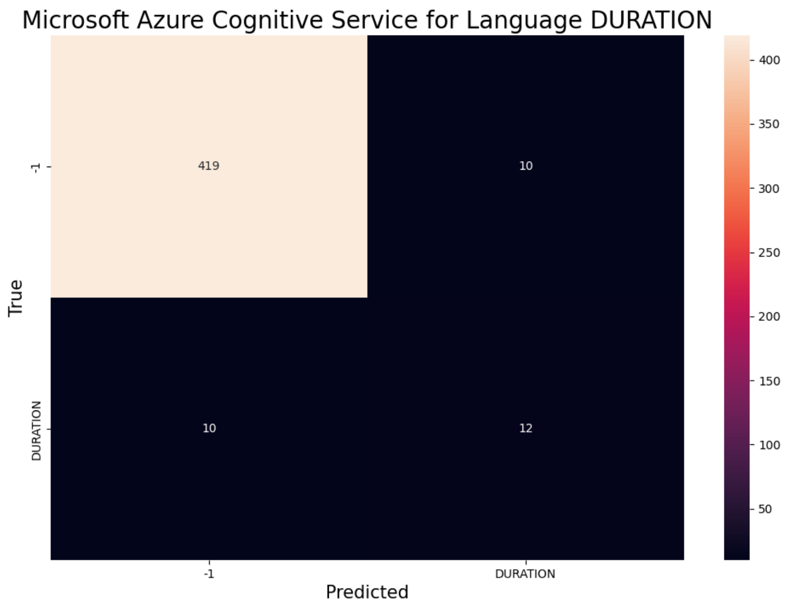


Figure G.51: Confusion matrix obtained for duration entity recognition with Microsoft Azure Cognitive Service for Language.

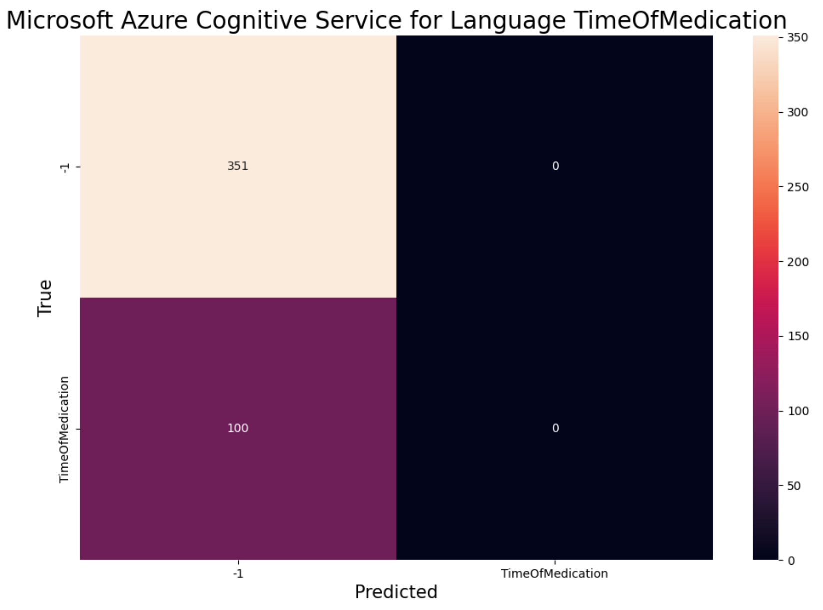


Figure G.52: Confusion matrix obtained for time of medication relationship detection with Microsoft Azure Cognitive Service for Language.

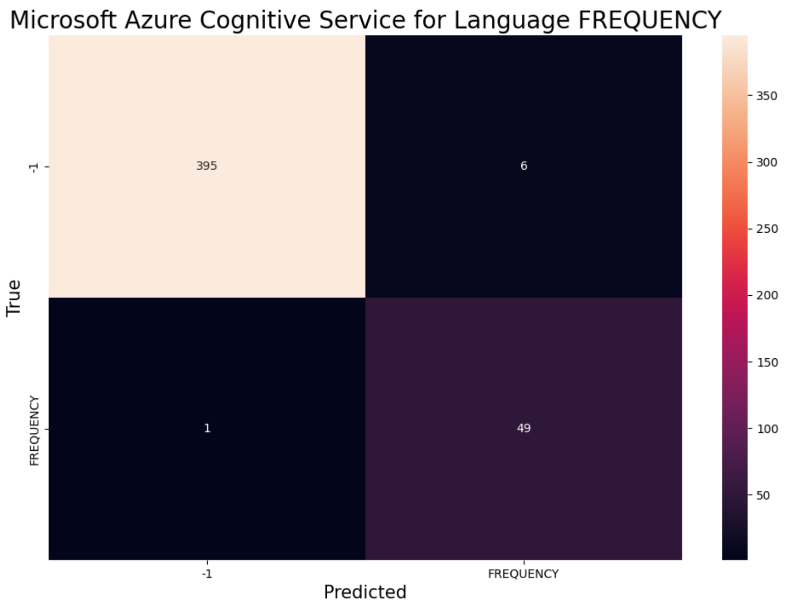


Figure G.53: Confusion matrix obtained for frequency entity recognition with Microsoft Azure Cognitive Service for Language.

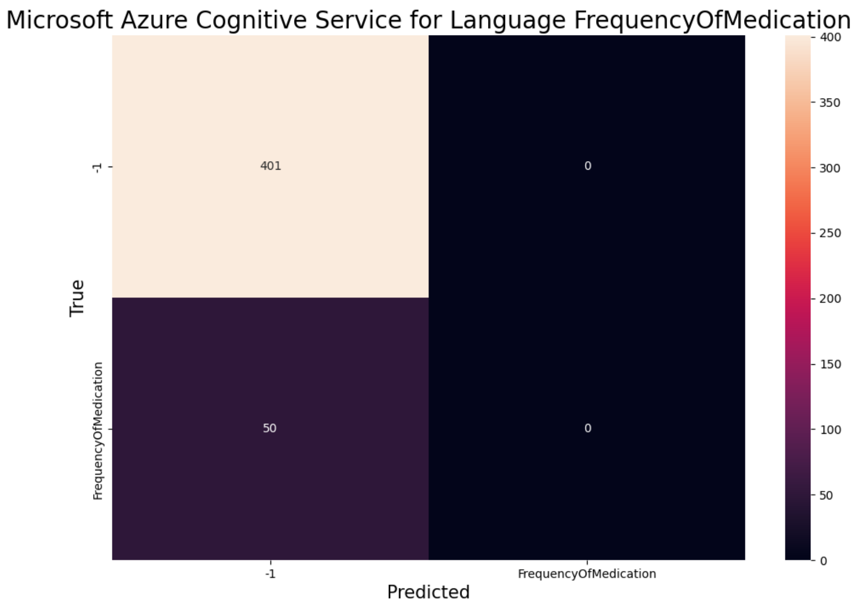


Figure G.54: Confusion matrix obtained for frequency of medication relationship detection with Microsoft Azure Cognitive Service for Language.

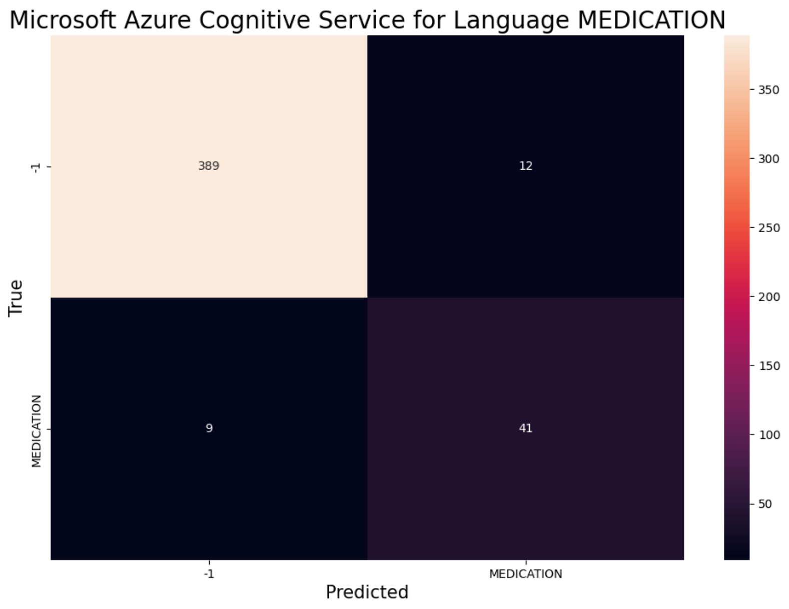


Figure G.55: Confusion matrix obtained for medication entity recognition with Microsoft Azure Cognitive Service for Language.

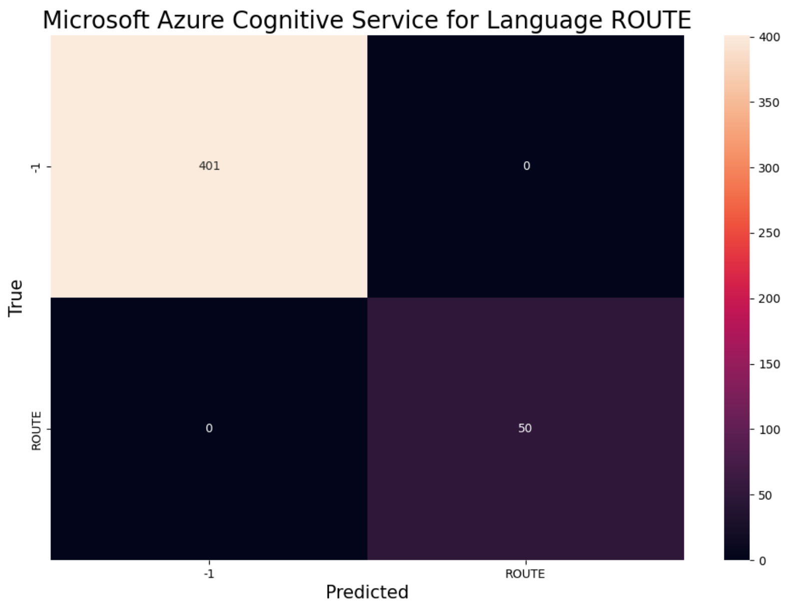


Figure G.56: Confusion matrix obtained for route entity recognition with Microsoft Azure Cognitive Service for Language.

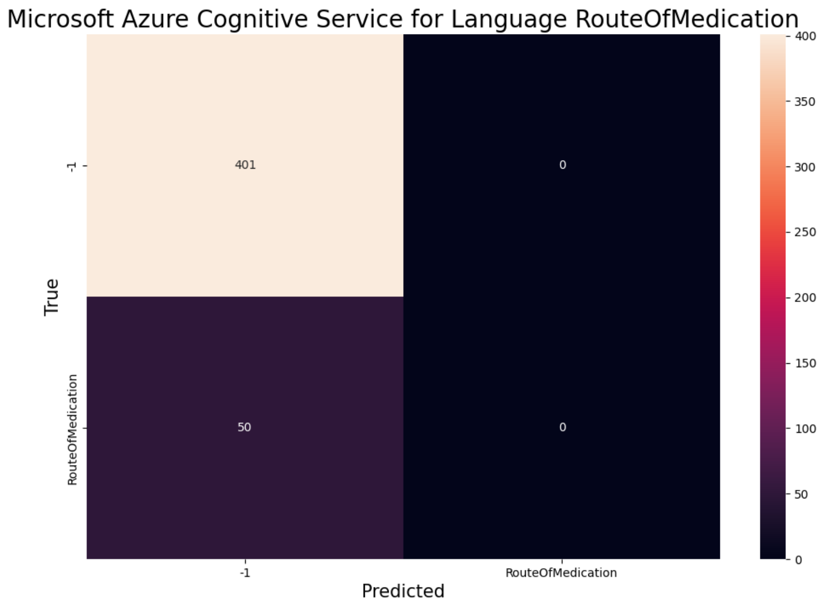


Figure G.57: Confusion matrix obtained for route of medication relationship detection with Microsoft Azure Cognitive Service for Language.

G.4 Microsoft Azure Text Analytics for Health

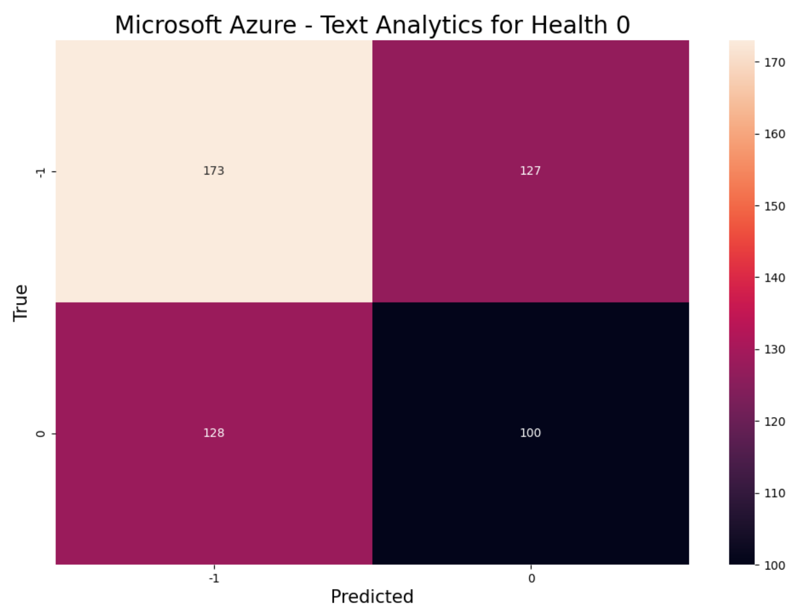


Figure G.58: Confusion matrix obtained for negative entity recognition with Microsoft Azure Text Analytics for Health.

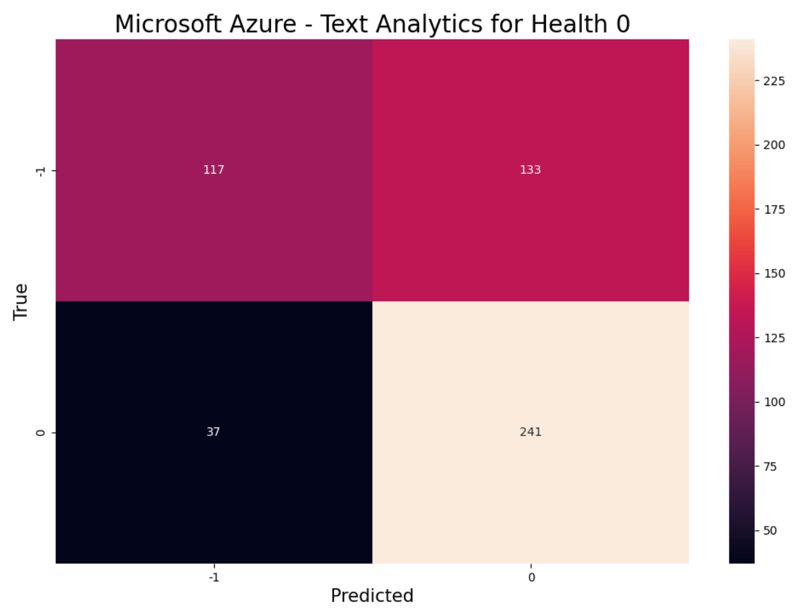


Figure G.59: Confusion matrix obtained for negative relationship detection with Microsoft Azure Text Analytics for Health.

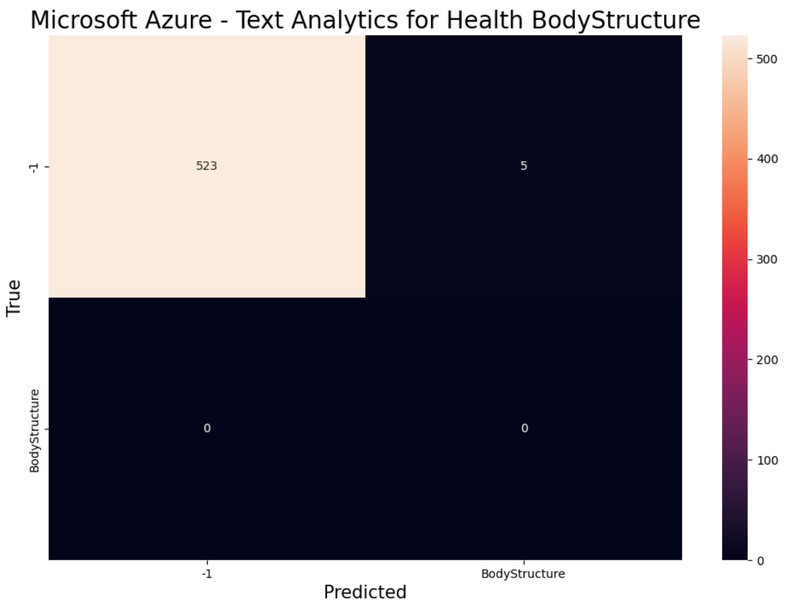


Figure G.60: Confusion matrix obtained for body structure entity recognition with Microsoft Azure Text Analytics for Health.

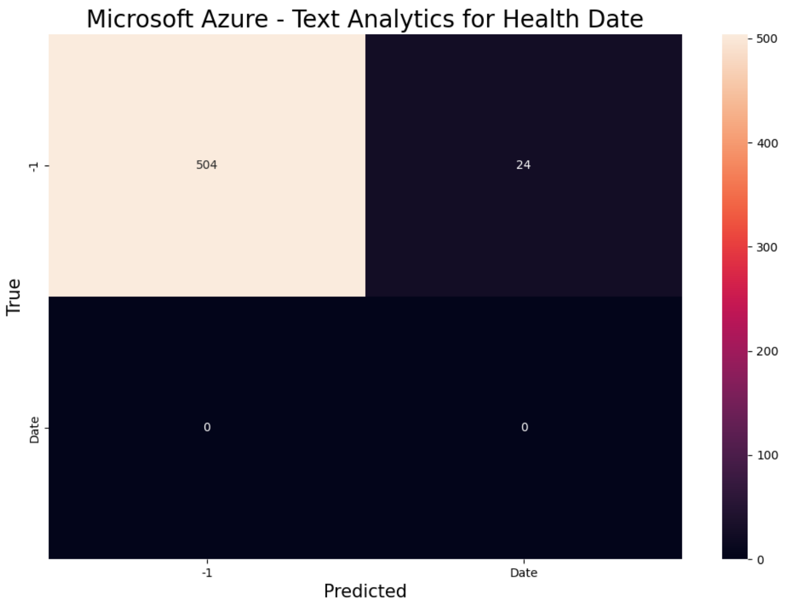


Figure G.61: Confusion matrix obtained for date entity recognition with Microsoft Azure Text Analytics for Health.

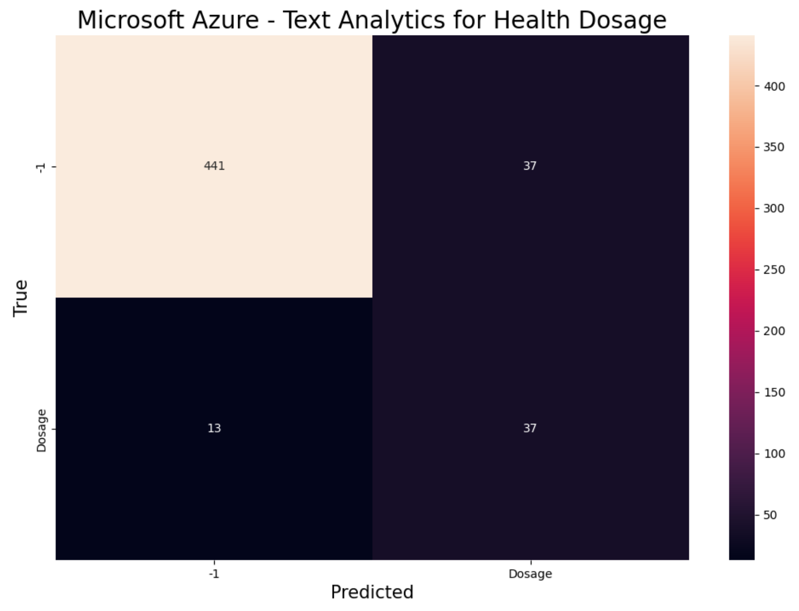


Figure G.62: Confusion matrix obtained for dosage entity recognition with Microsoft Azure Text Analytics for Health.

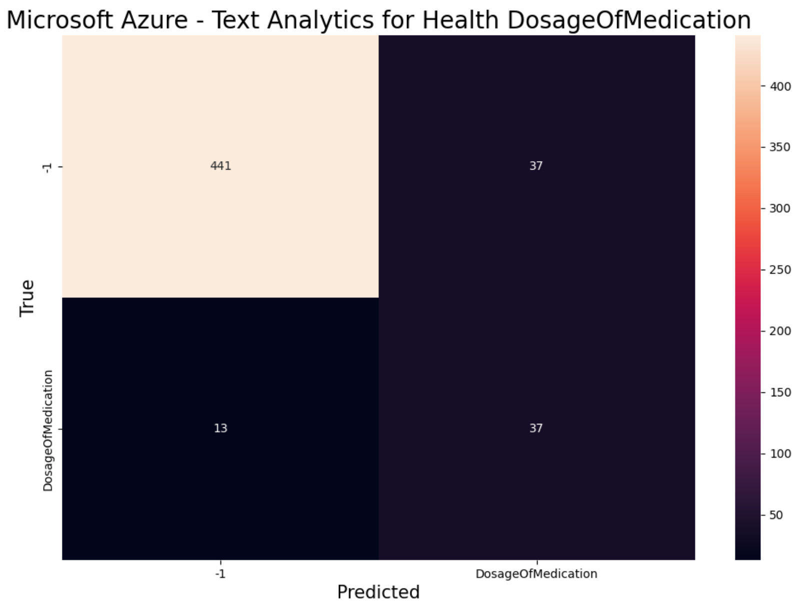


Figure G.63: Confusion matrix obtained for dosage of medication relationship detection with Microsoft Azure Text Analytics for Health.

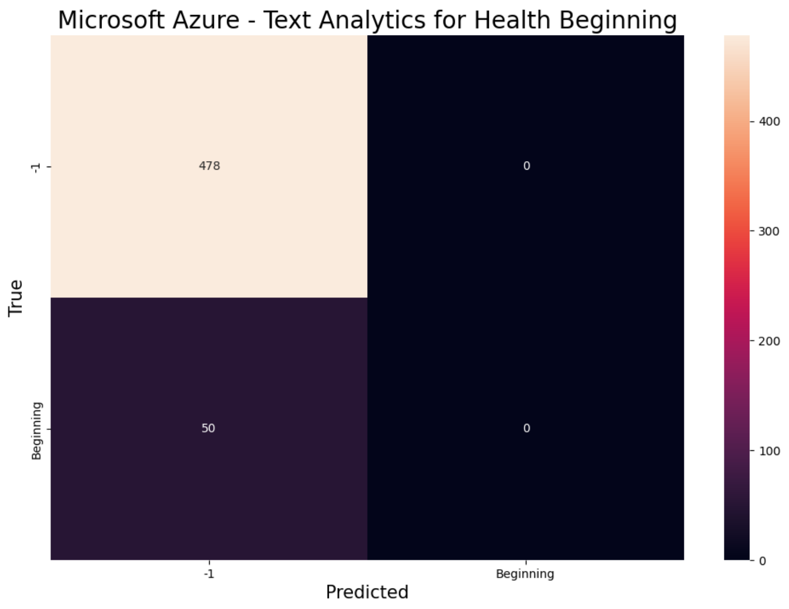


Figure G.64: Confusion matrix obtained for beginning entity recognition with Microsoft Azure Text Analytics for Health.

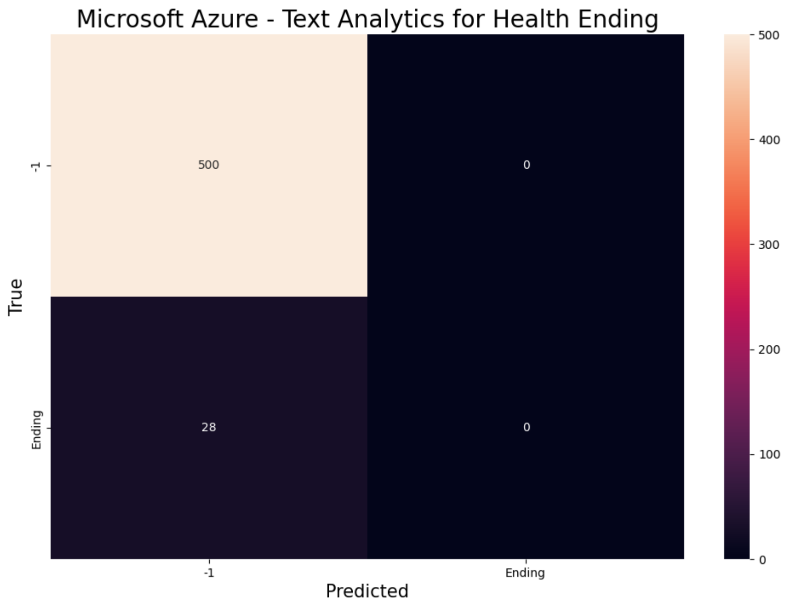


Figure G.65: Confusion matrix obtained for ending entity recognition with Microsoft Azure Text Analytics for Health.

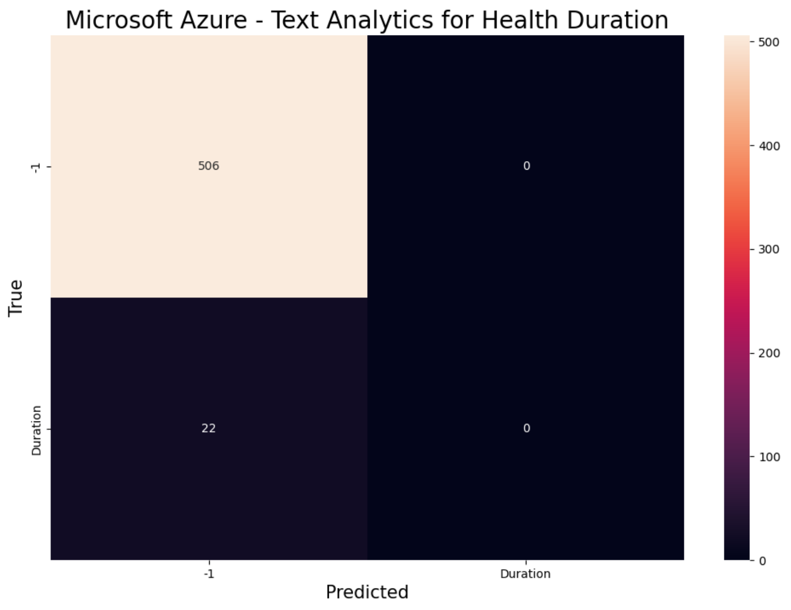


Figure G.66: Caption

Figure G.67: Confusion matrix obtained for duration entity recognition with Microsoft Azure Text Analytics for Health.

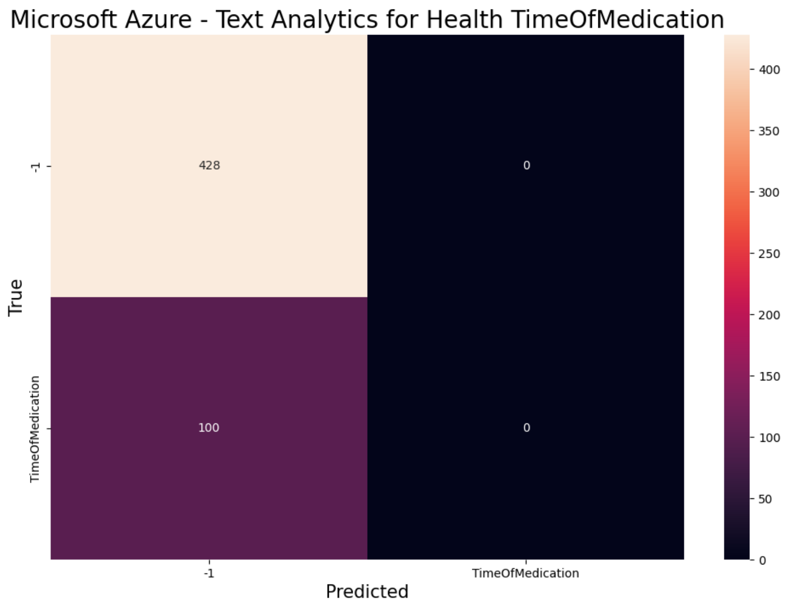


Figure G.68: Confusion matrix obtained for time of medication relationship detection with Microsoft Azure Text Analytics for Health.

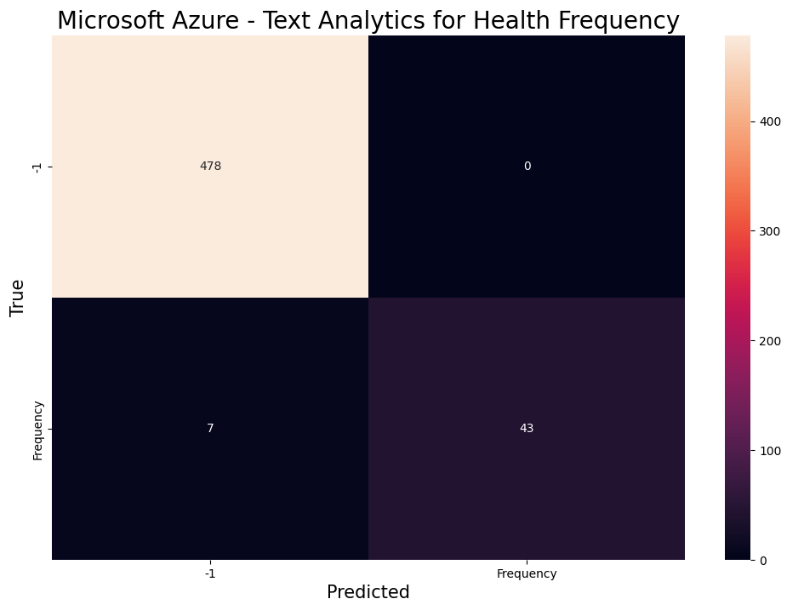


Figure G.69: Confusion matrix obtained for frequency entity recognition with Microsoft Azure Text Analytics for Health.

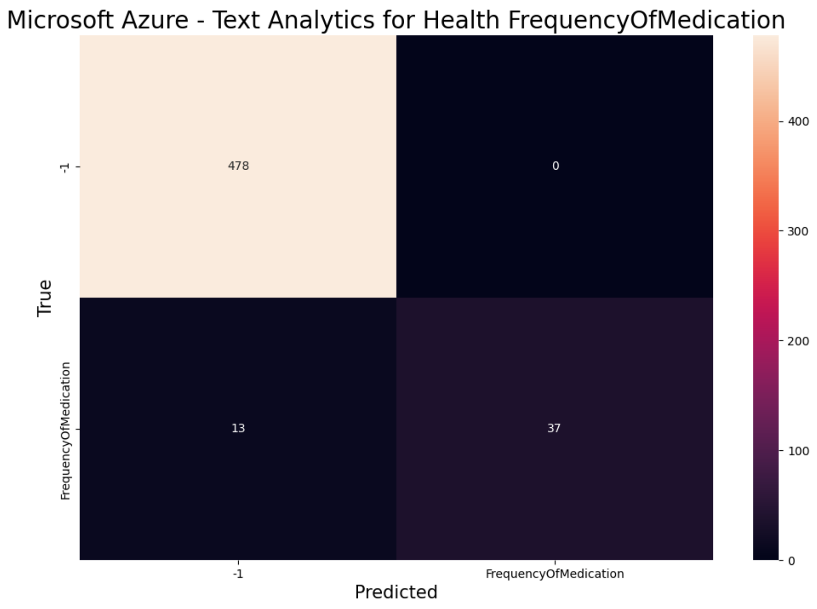


Figure G.70: Confusion matrix obtained for frequency of medication relationship detection with Microsoft Azure Text Analytics for Health.

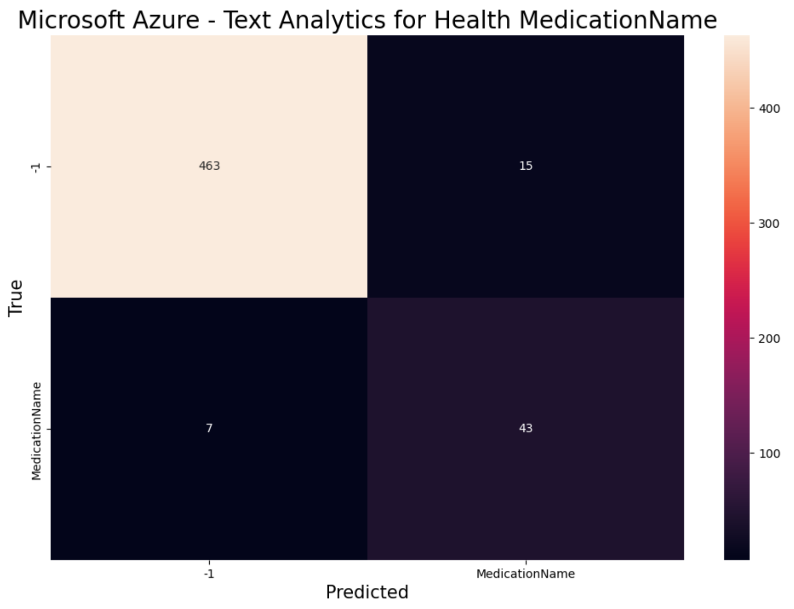


Figure G.71: Confusion matrix obtained for medication entity recognition with Microsoft Azure Text Analytics for Health.

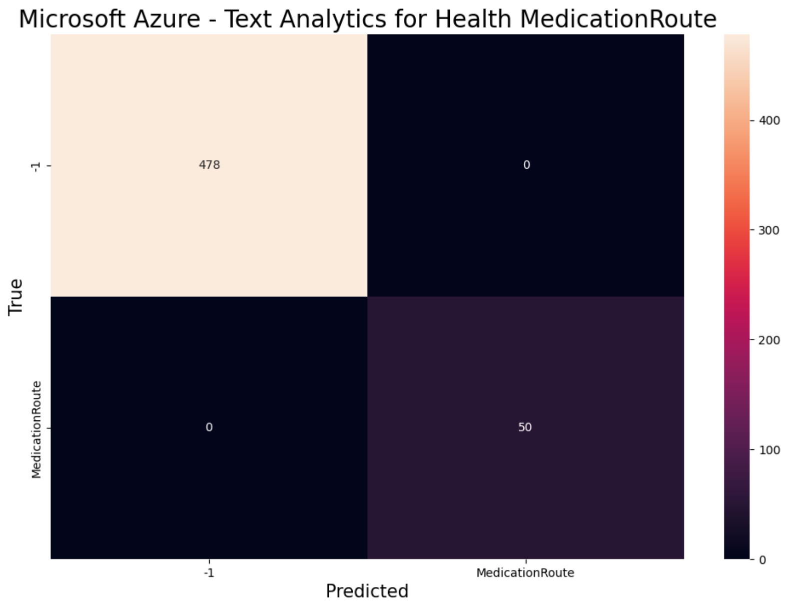


Figure G.72: Confusion matrix obtained for route entity recognition with Microsoft Azure Text Analytics for Health.

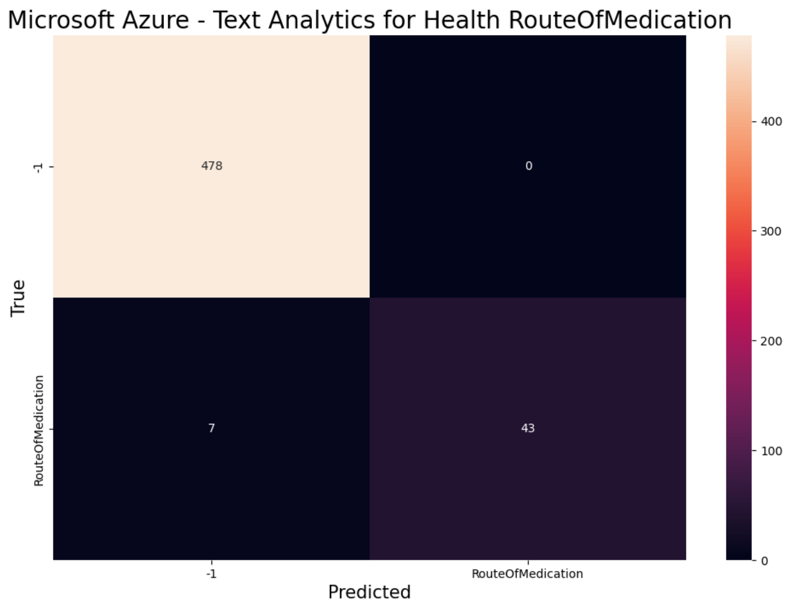


Figure G.73: Confusion matrix obtained for route of medication relationship detection with Microsoft Azure Text Analytics for Health.

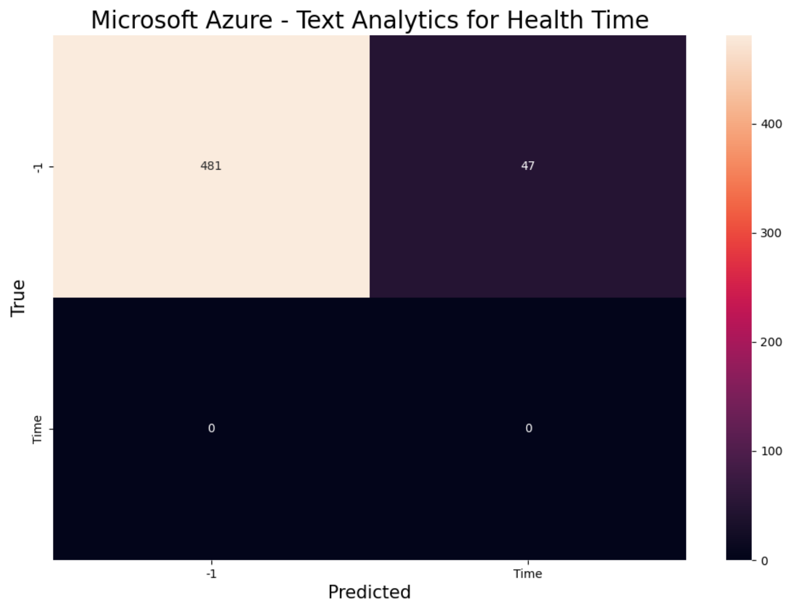


Figure G.74: Confusion matrix obtained for time entity recognition with Microsoft Azure Text Analytics for Health.

G.4.1 After post-processing

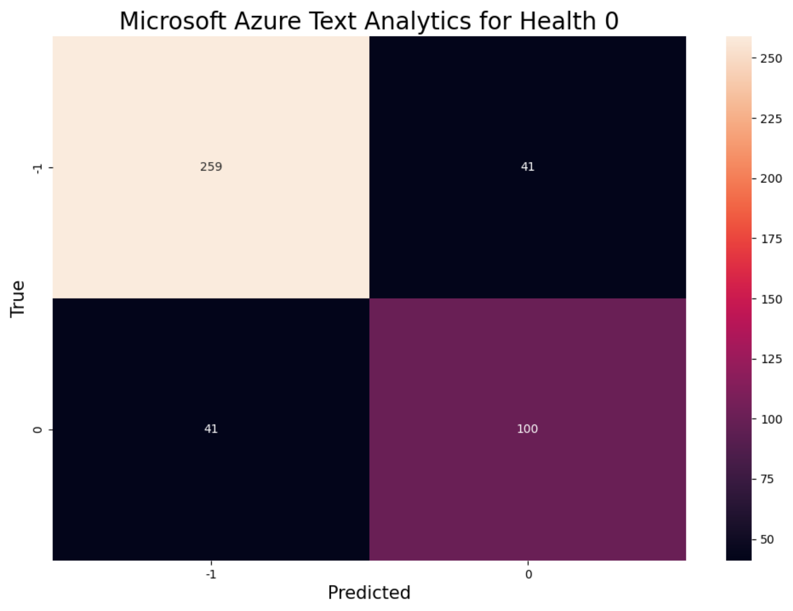


Figure G.75: Confusion matrix obtained for negative entity recognition with Microsoft Azure Text Analytics for Health after post-processing.

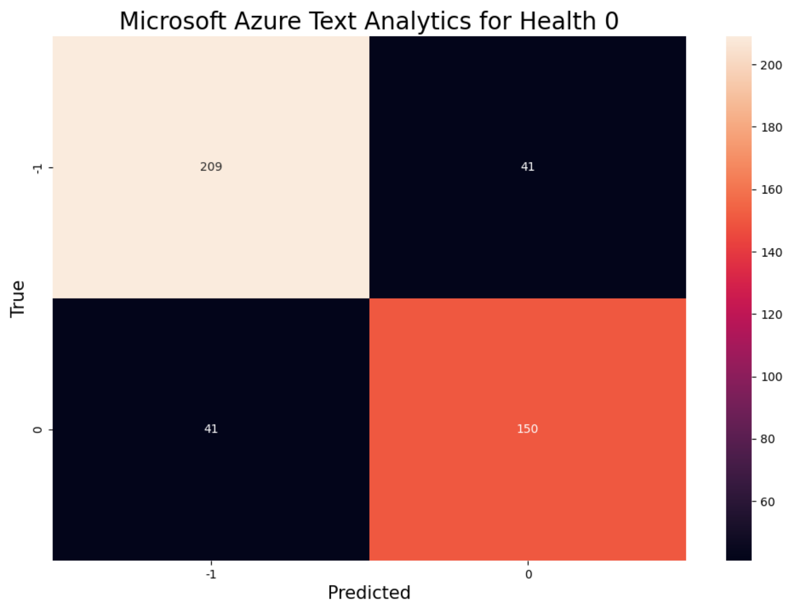


Figure G.76: Confusion matrix obtained for negative relationship detection with Microsoft Azure Text Analytics for Health after post-processing.

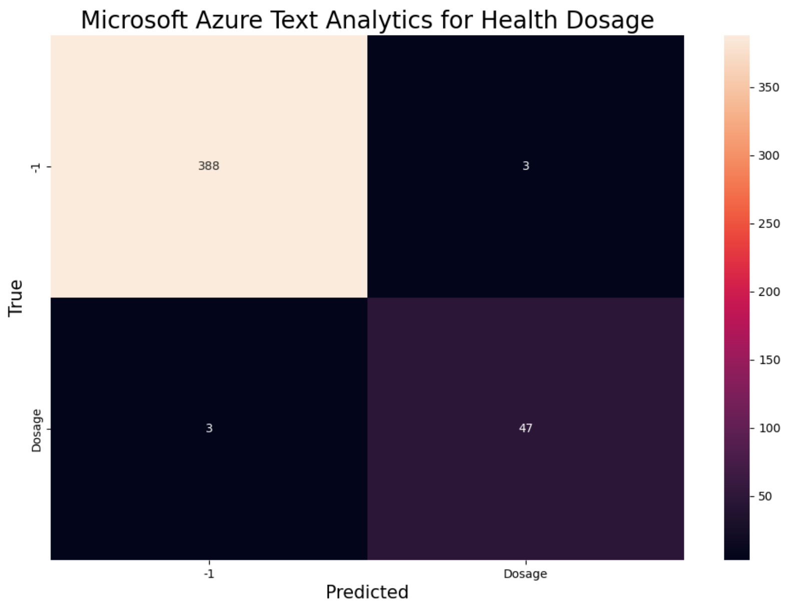


Figure G.77: Confusion matrix obtained for dosage entity recognition with Microsoft Azure Text Analytics for Health after post-processing.

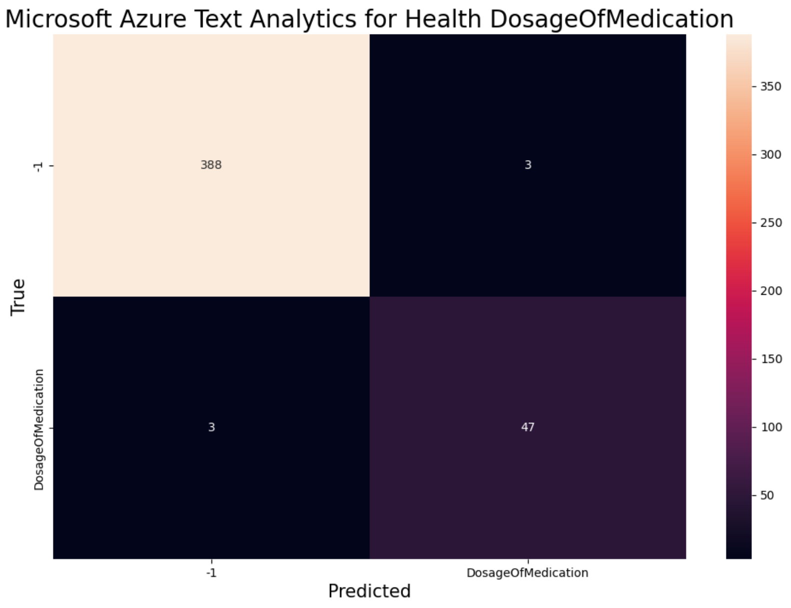


Figure G.78: Confusion matrix obtained for dosage of medication relationship detection with Microsoft Azure Text Analytics for Health after post-processing.

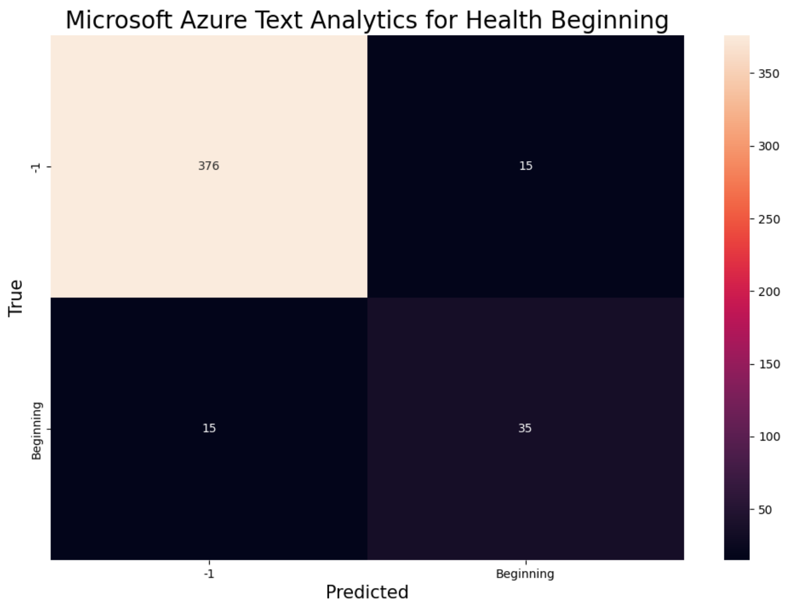


Figure G.79: Confusion matrix obtained for beginning entity recognition with Microsoft Azure Text Analytics for Health after post-processing.

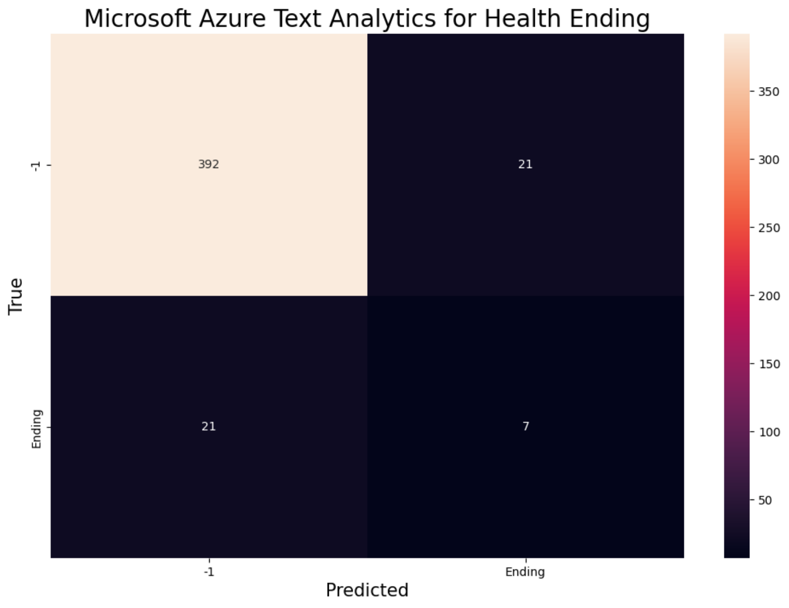


Figure G.80: Confusion matrix obtained for ending entity recognition with Microsoft Azure Text Analytics for Health after post-processing.

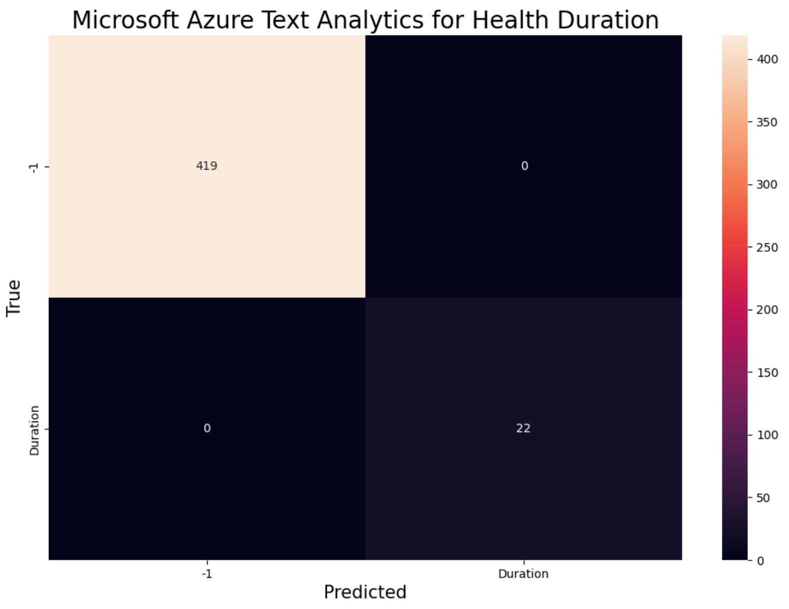


Figure G.81: Confusion matrix obtained for duration entity recognition with Microsoft Azure Text Analytics for Health after post-processing.

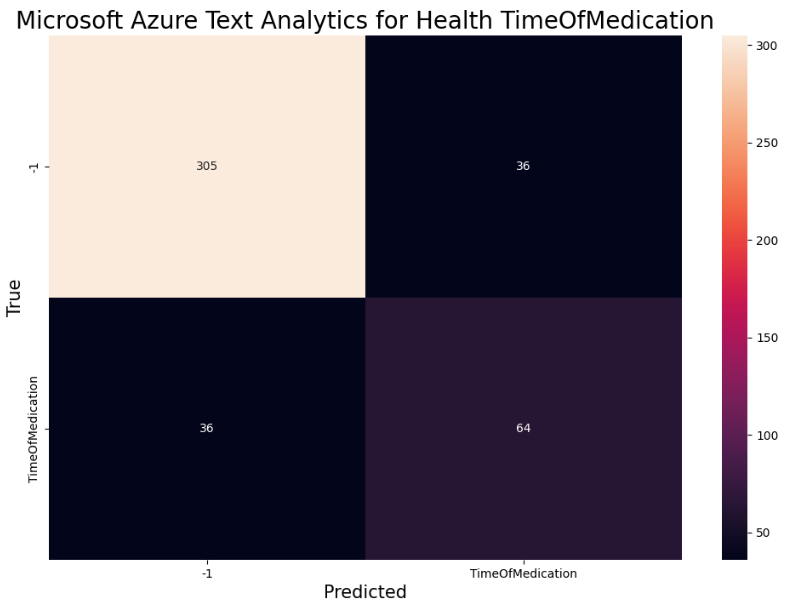


Figure G.82: Confusion matrix obtained for time of medication relationship detection with Microsoft Azure Text Analytics for Health after post-processing.

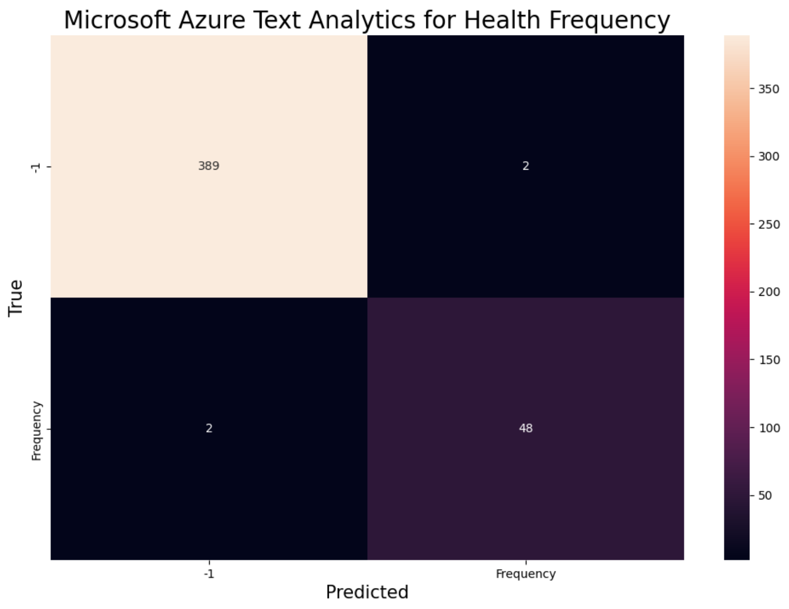


Figure G.83: Confusion matrix obtained for frequency entity recognition with Microsoft Azure Text Analytics for Health after post-processing.

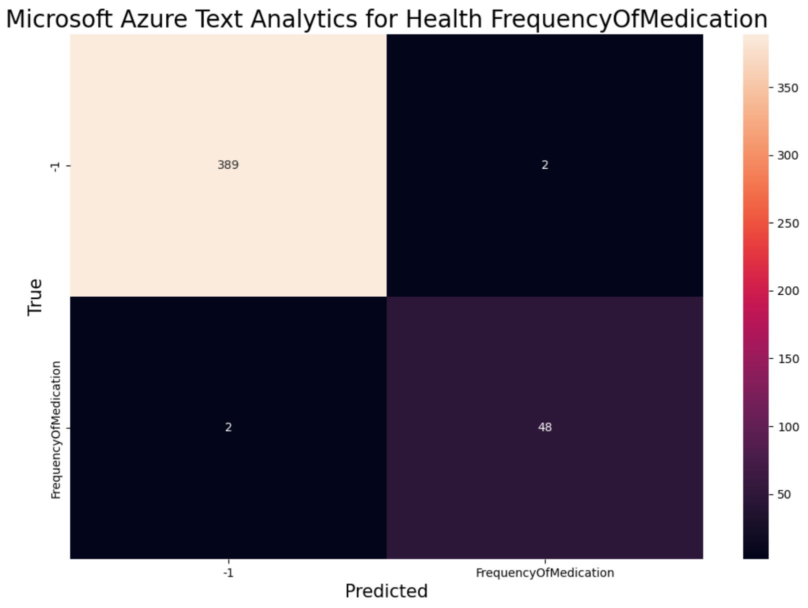


Figure G.84: Confusion matrix obtained for frequency of medication relationship detection with Microsoft Azure Text Analytics for Health after post-processing.

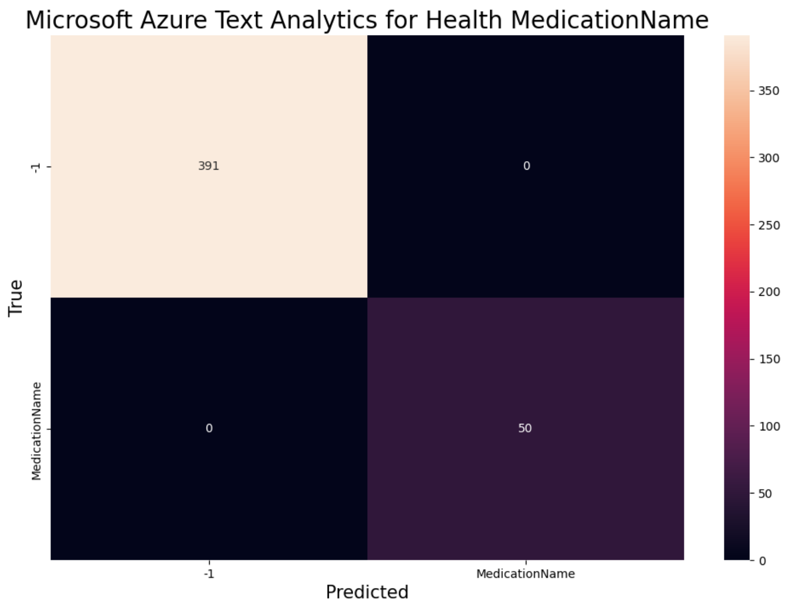


Figure G.85: Confusion matrix obtained for medication entity recognition with Microsoft Azure Text Analytics for Health after post-processing.

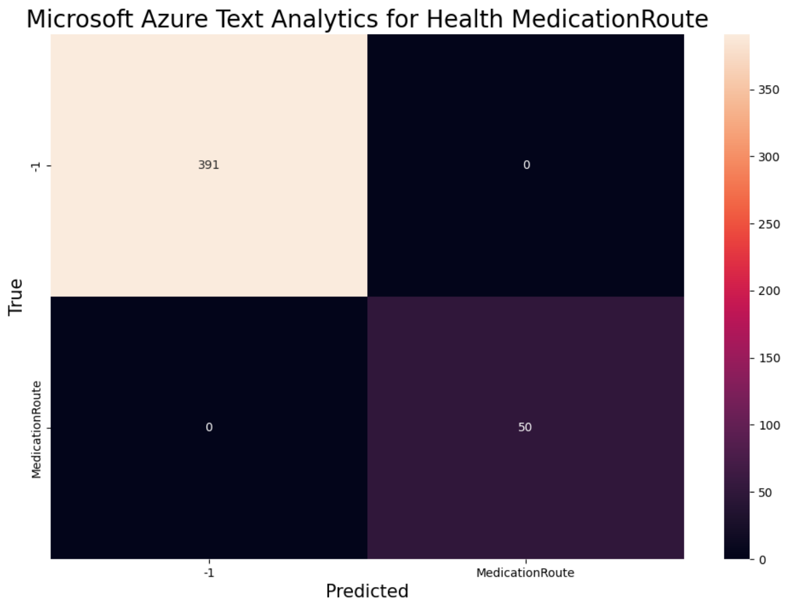


Figure G.86: Confusion matrix obtained for route entity recognition with Microsoft Azure Text Analytics for Health after post-processing.

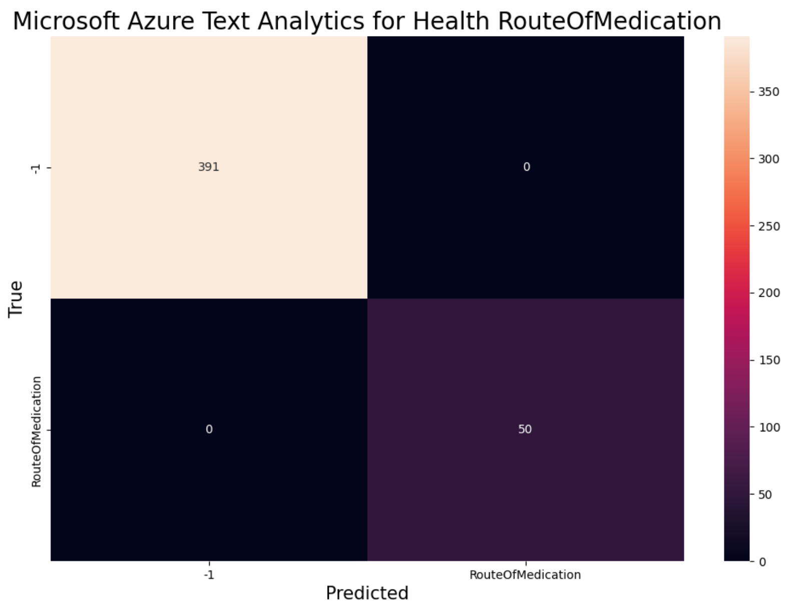
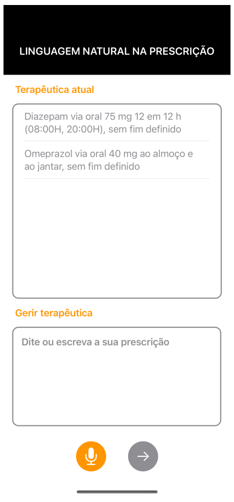


Figure G.87: Confusion matrix obtained for route of medication relationship detection with Microsoft Azure Text Analytics for Health after post-processing.

Appendix H

Application use cases illustration



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

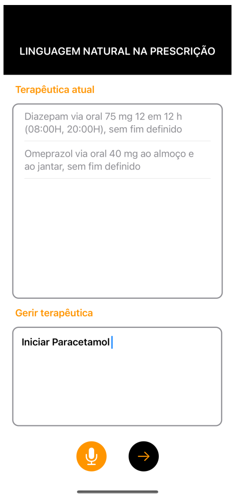
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

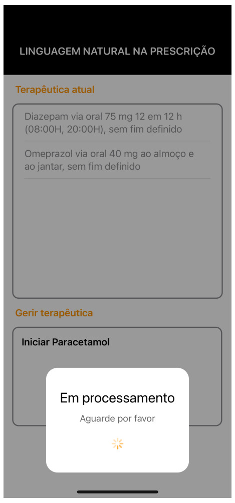
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar Paracetamol

(b)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

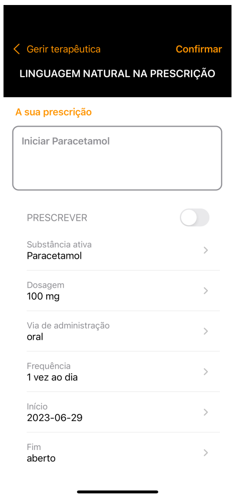
Gerir terapêutica

Iniciar Paracetamol

Em processamento

Aguarde por favor

(c)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol

PRESCREVER

Substância ativa: Paracetamol

Dosagem: 100 mg

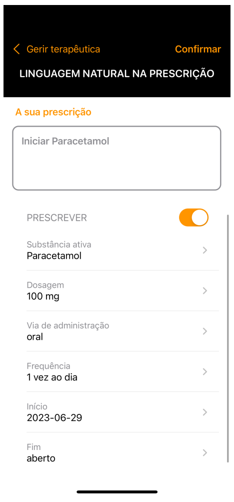
Via de administração: oral

Frequência: 1 vez ao dia

Início: 2023-06-29

Fim: aberto

(d)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol

PRESCREVER

Substância ativa: Paracetamol

Dosagem: 100 mg

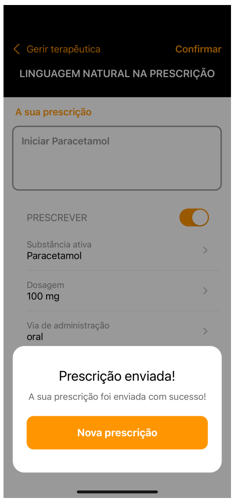
Via de administração: oral

Frequência: 1 vez ao dia

Início: 2023-06-29

Fim: aberto

(e)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol

PRESCREVER

Substância ativa: Paracetamol

Dosagem: 100 mg

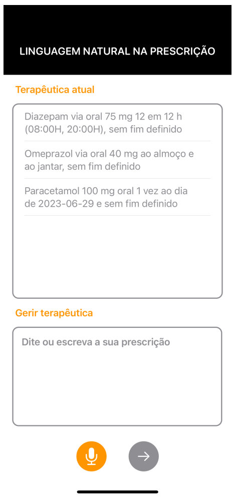
Via de administração: oral

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(f)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

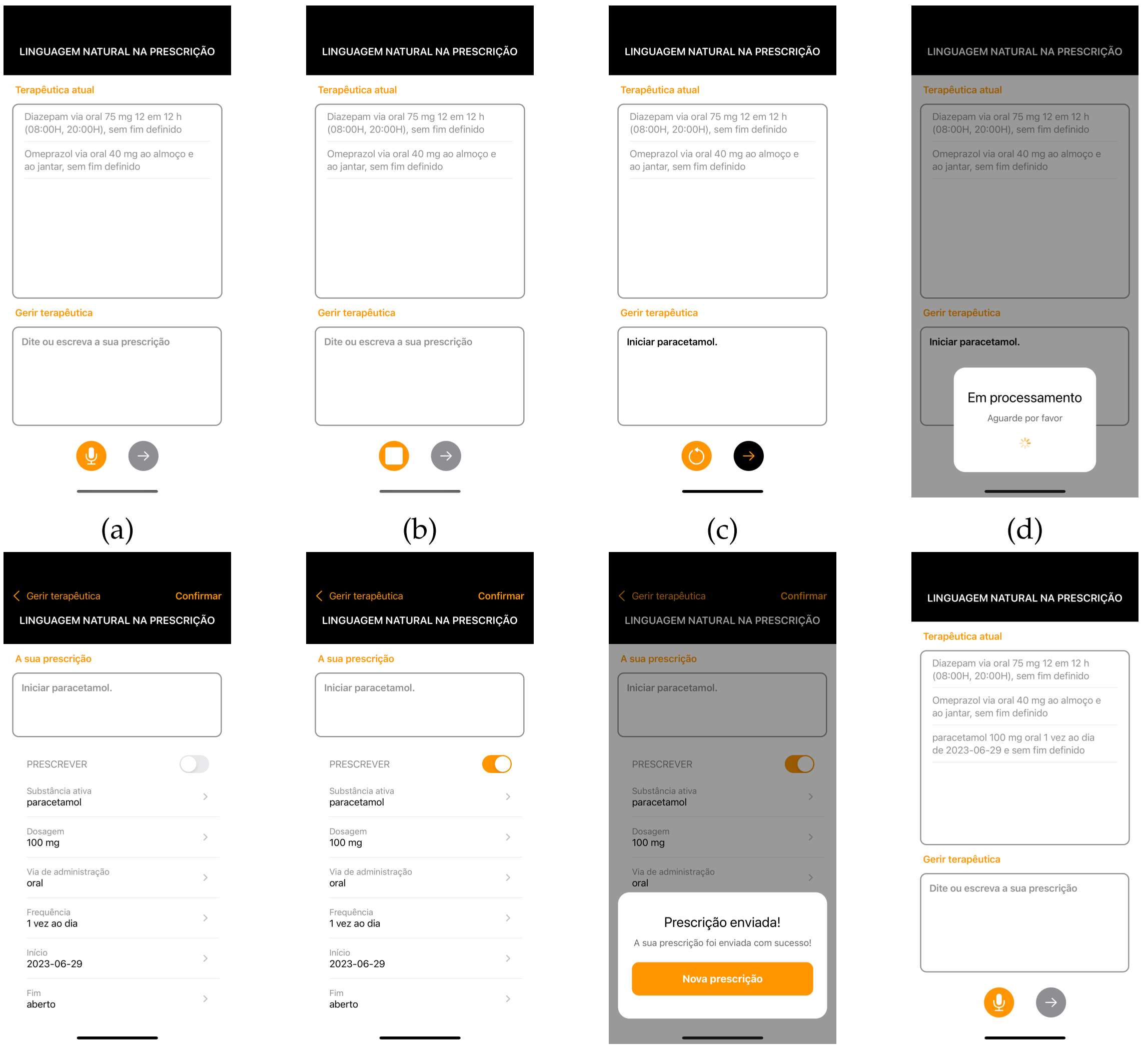
Paracetamol 100 mg oral 1 vez ao dia de 2023-06-29 e sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(g)

Figure H.1: Prescribe use case illustration. (I)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar paracetamol.

(c)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar paracetamol.

Em processamento

Aguarde por favor

(d)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol.

PRESCREVER

Substância ativa
paracetamol

Dosagem
100 mg

Via de administração
oral

Frequência
1 vez ao dia

Início
2023-06-29

Fim
aberto

(e)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol.

PRESCREVER

Substância ativa
paracetamol

Dosagem
100 mg

Via de administração
oral

Frequência
1 vez ao dia

Início
2023-06-29

Fim
aberto

(f)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol.

PRESCREVER

Substância ativa
paracetamol

Dosagem
100 mg

Via de administração
oral

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(g)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

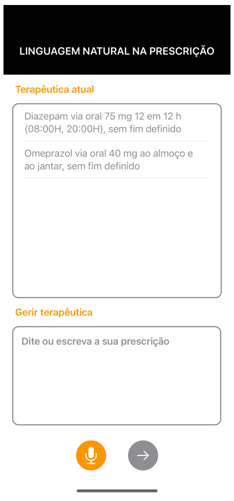
paracetamol 100 mg oral 1 vez ao dia de 2023-06-29 e sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(h)

Figure H.2: Prescribe use case illustration. (II)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

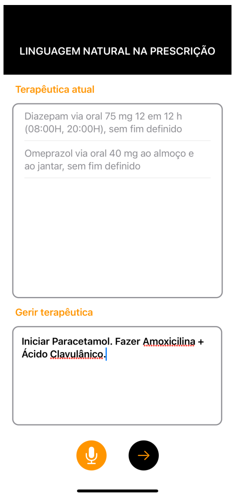
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

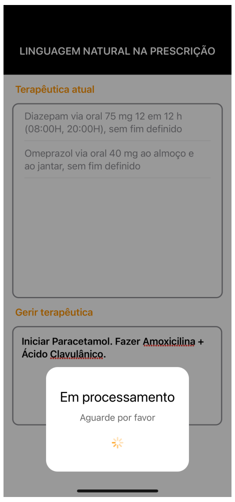
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar Paracetamol. Fazer Amoxicilina + Ácido Clavulânico.

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

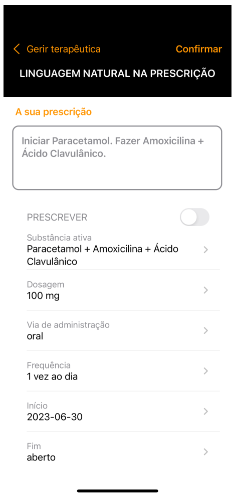
Gerir terapêutica

Iniciar Paracetamol. Fazer Amoxicilina + Ácido Clavulânico.

Em processamento

Aguarde por favor

(c)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol. Fazer Amoxicilina + Ácido Clavulânico.

PRESCREVER

Substância ativa
Paracetamol + Amoxicilina + Ácido Clavulânico

Dosagem
100 mg

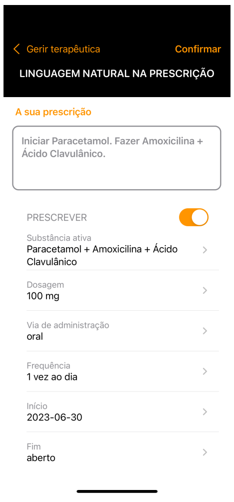
Via de administração
oral

Frequência
1 vez ao dia

Início
2023-06-30

Fim
aberto

(d)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol. Fazer Amoxicilina + Ácido Clavulânico.

PRESCREVER

Substância ativa
Paracetamol + Amoxicilina + Ácido Clavulânico

Dosagem
100 mg

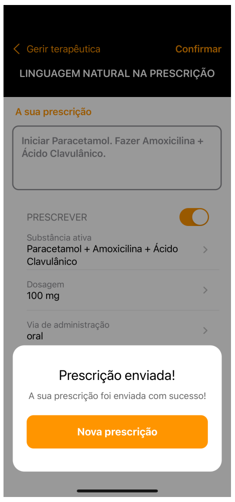
Via de administração
oral

Frequência
1 vez ao dia

Início
2023-06-30

Fim
aberto

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol. Fazer Amoxicilina + Ácido Clavulânico.

PRESCREVER

Substância ativa
Paracetamol + Amoxicilina + Ácido Clavulânico

Dosagem
100 mg

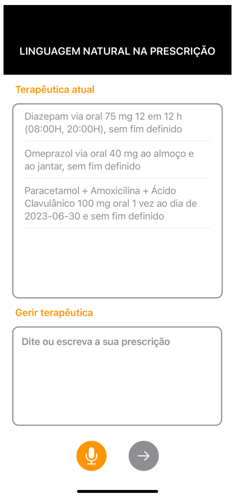
Via de administração
oral

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(f)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

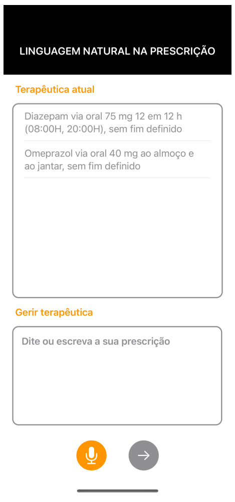
Paracetamol + Amoxicilina + Ácido Clavulânico 100 mg oral 1 vez ao dia de 2023-06-30 e sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(g)

Figure H.3: Prescribe use case illustration. (III)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

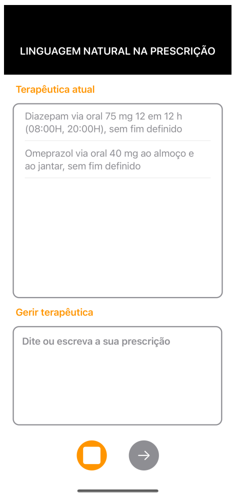
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

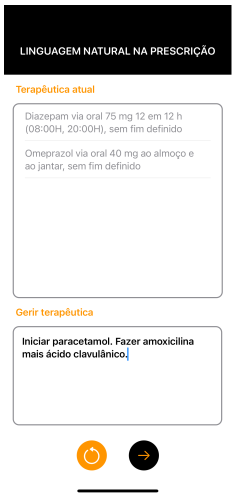
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

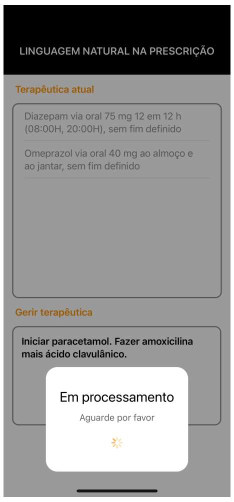
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar paracetamol. Fazer amoxicilina mais ácido clavulânico.

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

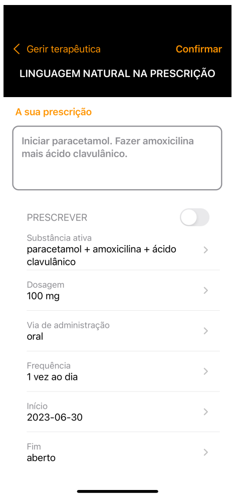
Gerir terapêutica

Iniciar paracetamol. Fazer amoxicilina mais ácido clavulânico.

Em processamento

Aguarde por favor

(d)


Gerir terapêutica — Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol. Fazer amoxicilina mais ácido clavulânico.

PRESCREVER

Substância ativa: paracetamol + amoxicilina + ácido clavulânico

Dosagem: 100 mg

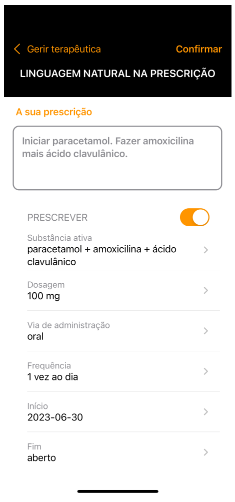
Via de administração: oral

Frequência: 1 vez ao dia

Início: 2023-06-30

Fim: aberto

(e)


Gerir terapêutica — Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol. Fazer amoxicilina mais ácido clavulânico.

PRESCREVER

Substância ativa: paracetamol + amoxicilina + ácido clavulânico

Dosagem: 100 mg

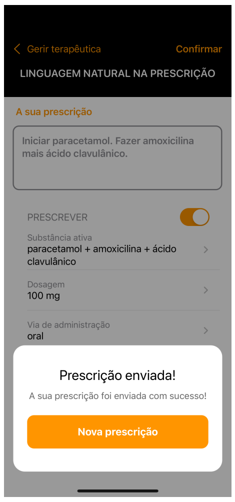
Via de administração: oral

Frequência: 1 vez ao dia

Início: 2023-06-30

Fim: aberto

(f)


Gerir terapêutica — Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol. Fazer amoxicilina mais ácido clavulânico.

PRESCREVER

Substância ativa: paracetamol + amoxicilina + ácido clavulânico

Dosagem: 100 mg

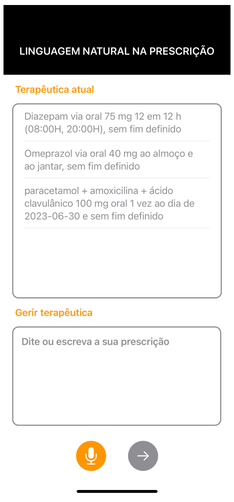
Via de administração: oral

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(g)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

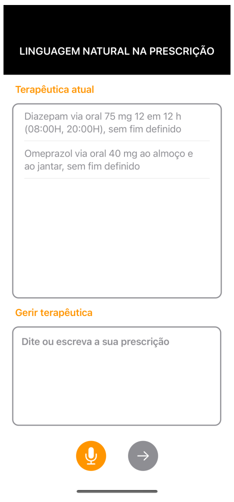
paracetamol + amoxicilina + ácido clavulânico 100 mg oral 1 vez ao dia de 2023-06-30 e sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(h)

Figure H.4: Prescribe use case illustration. (IV)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

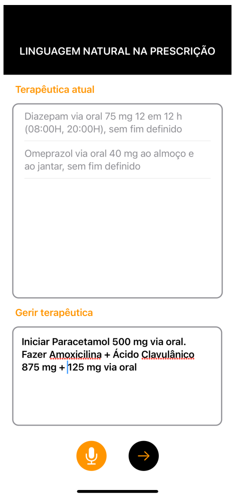
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

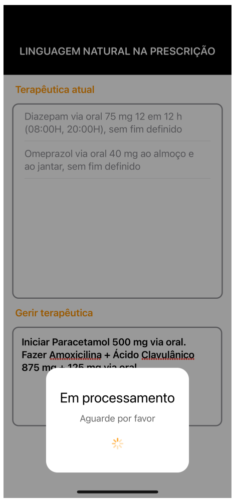
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral

(b)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

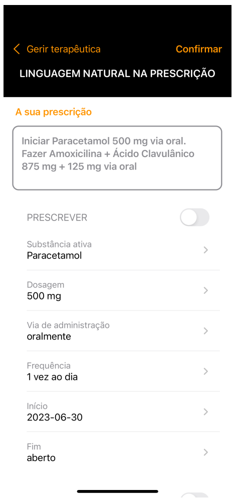
Gerir terapêutica

Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral

Em processamento

Aguarde por favor

(c)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral

PRESCREVER

Substância ativa: Paracetamol

Dosagem: 500 mg

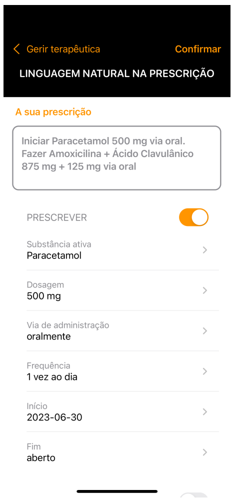
Via de administração: oralmente

Frequência: 1 vez ao dia

Início: 2023-06-30

Fim: aberto

(d)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral

PRESCREVER

Substância ativa: Paracetamol

Dosagem: 500 mg

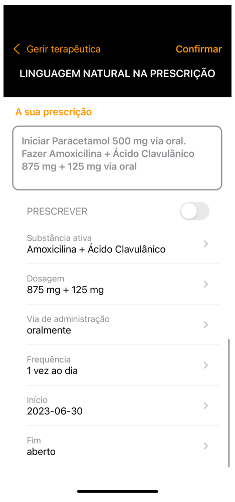
Via de administração: oralmente

Frequência: 1 vez ao dia

Início: 2023-06-30

Fim: aberto

(e)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral

PRESCREVER

Substância ativa: Amoxicilina + Ácido Clavulânico

Dosagem: 875 mg + 125 mg

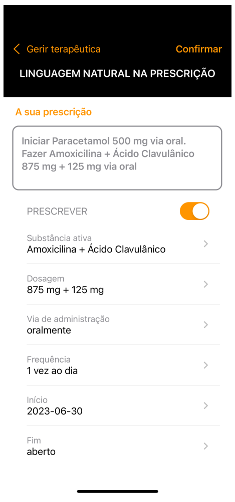
Via de administração: oralmente

Frequência: 1 vez ao dia

Início: 2023-06-30

Fim: aberto

(f)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral

PRESCREVER

Substância ativa: Amoxicilina + Ácido Clavulânico

Dosagem: 875 mg + 125 mg

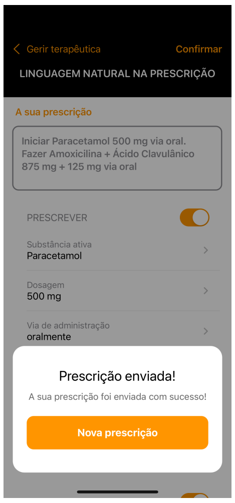
Via de administração: oralmente

Frequência: 1 vez ao dia

Início: 2023-06-30

Fim: aberto

(g)



Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral

PRESCREVER

Substância ativa: Paracetamol

Dosagem: 500 mg

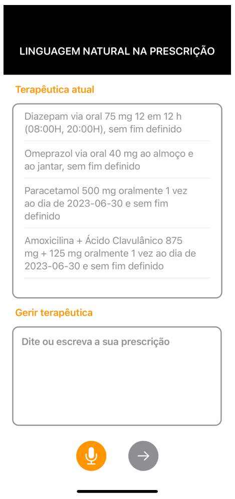
Via de administração: oralmente

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(h)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Paracetamol 500 mg oralmente 1 vez ao dia de 2023-06-30 e sem fim definido

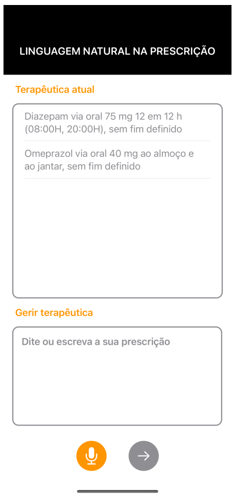
Amoxicilina + Ácido Clavulânico 875 mg + 125 mg oralmente 1 vez ao dia de 2023-06-30 e sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(i)

Figure H.5: Prescribe use case illustration. (V)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

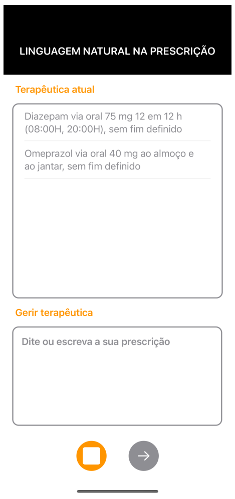
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

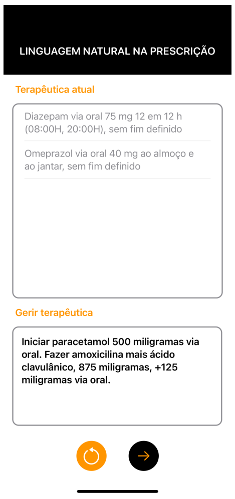
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

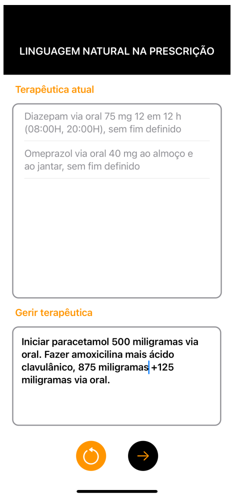
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar paracetamol 500 miligramas via oral. Fazer amoxicilina mais ácido clavulânico, 875 miligramas, +125 miligramas via oral.

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

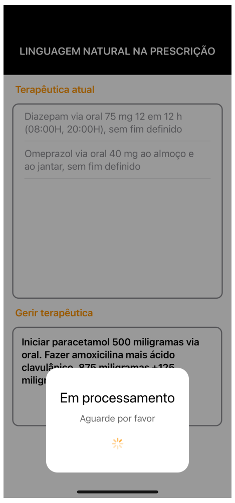
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar paracetamol 500 miligramas via oral. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 miligramas via oral.

(d)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

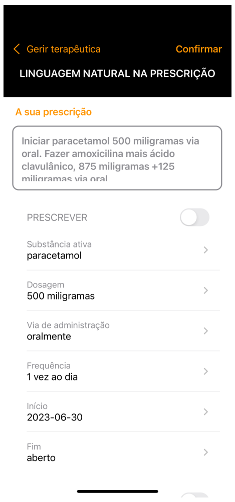
Gerir terapêutica

Iniciar paracetamol 500 miligramas via oral. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 milig

Em processamento

Aguarde por favor

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 miligramas via oral

PRESCREVER

Substância ativa
paracetamol

Dosagem
500 miligramas

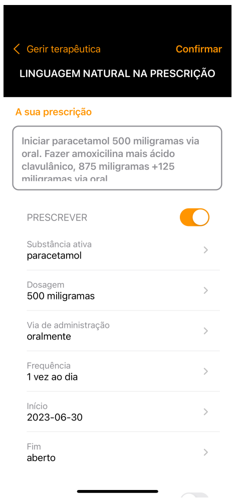
Via de administração
oralmente

Frequência
1 vez ao dia

Início
2023-06-30

Fim
aberto

(f)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 miligramas via oral

PRESCREVER

Substância ativa
paracetamol

Dosagem
500 miligramas

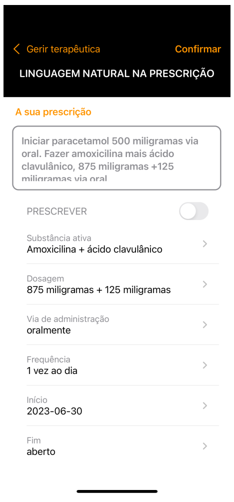
Via de administração
oralmente

Frequência
1 vez ao dia

Início
2023-06-30

Fim
aberto

(g)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 miligramas via oral

PRESCREVER

Substância ativa
Amoxicilina + ácido clavulânico

Dosagem
875 miligramas + 125 miligramas

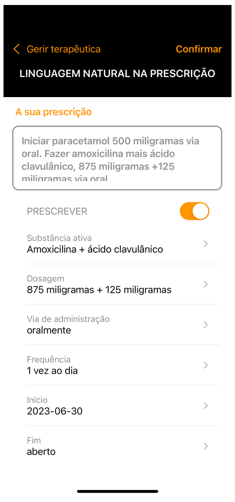
Via de administração
oralmente

Frequência
1 vez ao dia

Início
2023-06-30

Fim
aberto

(h)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 miligramas via oral

PRESCREVER

Substância ativa
Amoxicilina + ácido clavulânico

Dosagem
875 miligramas + 125 miligramas

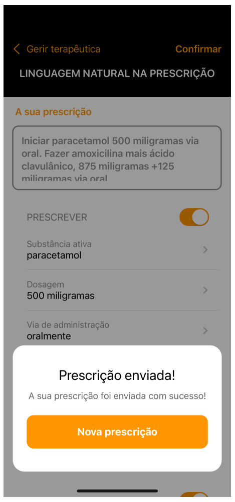
Via de administração
oralmente

Frequência
1 vez ao dia

Início
2023-06-30

Fim
aberto

(i)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 miligramas via oral

PRESCREVER

Substância ativa
paracetamol

Dosagem
500 miligramas

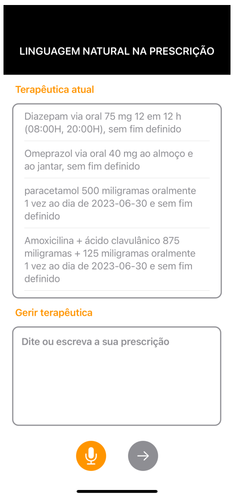
Via de administração
oralmente

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(j)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

paracetamol 500 miligramas oralmente 1 vez ao dia de 2023-06-30 e sem fim definido

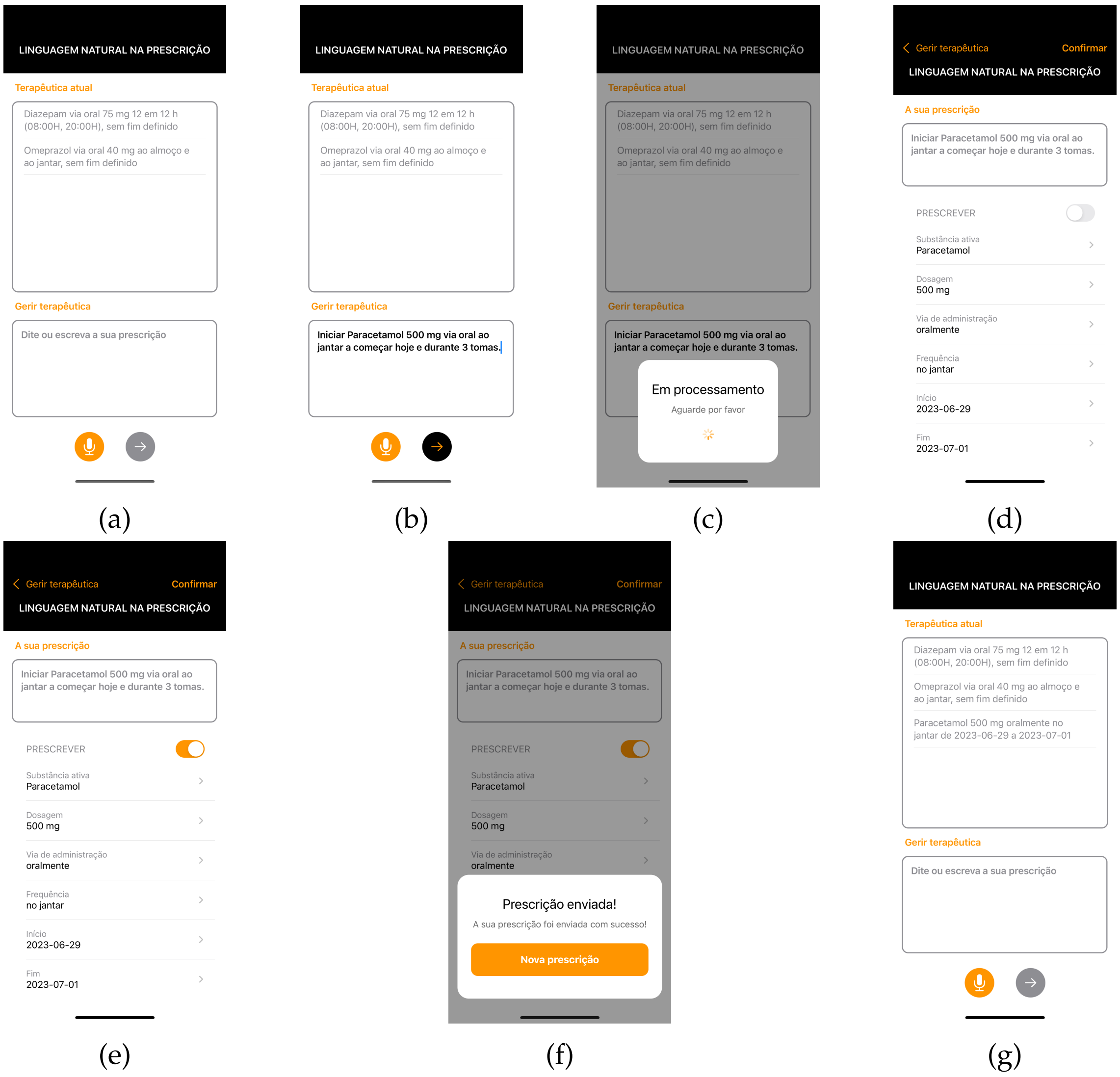
Amoxicilina + ácido clavulânico 875 miligramas + 125 miligramas oralmente 1 vez ao dia de 2023-06-30 e sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(k)

Figure H.6: Prescribe use case illustration. (VI)



(a)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas.

(c)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas.

Em processamento

Aguarde por favor

(d)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas.

PRESCREVER

Substância ativa
Paracetamol

Dosagem
500 mg

Via de administração
oralmente

Frequência
no jantar

Início
2023-06-29

Fim
2023-07-01

(e)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas.

PRESCREVER

Substância ativa
Paracetamol

Dosagem
500 mg

Via de administração
oralmente

Frequência
no jantar

Início
2023-06-29

Fim
2023-07-01

(f)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas.

PRESCREVER

Substância ativa
Paracetamol

Dosagem
500 mg

Via de administração
oralmente

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(g)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

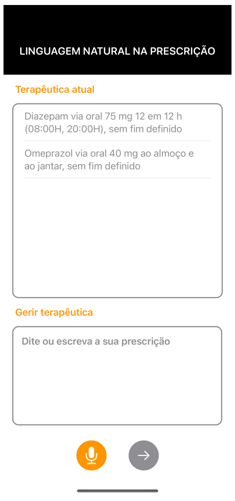
Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Paracetamol 500 mg oralmente no jantar de 2023-06-29 a 2023-07-01

Gerir terapêutica

Dite ou escreva a sua prescrição

Figure H.7: Prescribe use case illustration. (VII)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

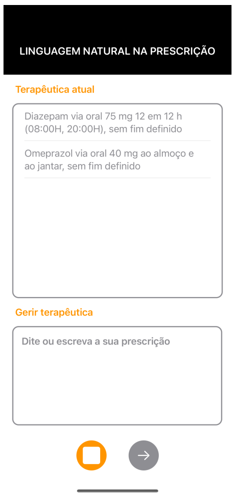
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

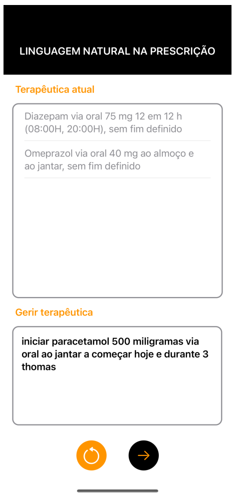
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

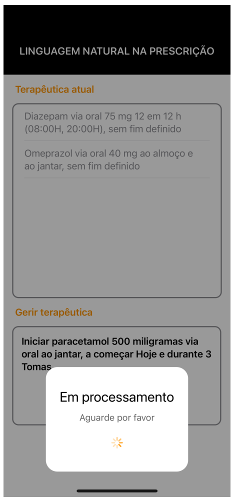
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

iniciar paracetamol 500 miligramas via oral ao jantar a começar hoje e durante 3 thomas

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

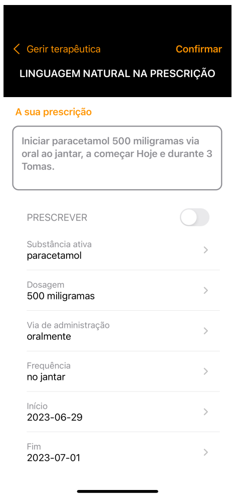
Gerir terapêutica

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 Tomas

Em processamento

Aguarde por favor

(d)


Gerir terapêutica　Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 Tomas.

PRESCREVER

Substância ativa
paracetamol

Dosagem
500 miligramas

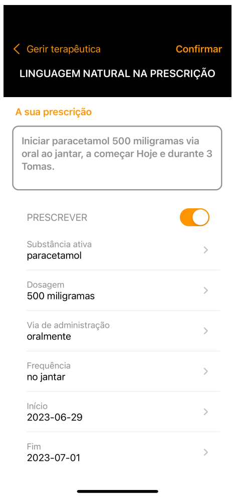
Via de administração
oralmente

Frequência
no jantar

Início
2023-06-29

Fim
2023-07-01

(e)


Gerir terapêutica　Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 Tomas.

PRESCREVER

Substância ativa
paracetamol

Dosagem
500 miligramas

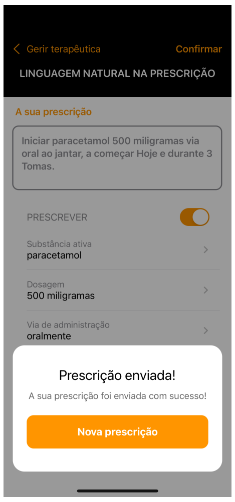
Via de administração
oralmente

Frequência
no jantar

Início
2023-06-29

Fim
2023-07-01

(f)


Gerir terapêutica　Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 Tomas.

PRESCREVER

Substância ativa
paracetamol

Dosagem
500 miligramas

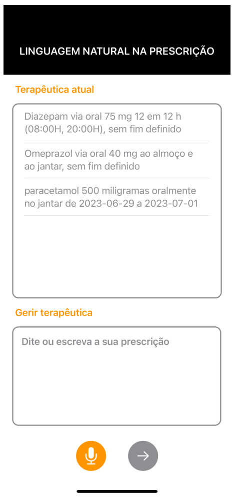
Via de administração
oralmente

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(g)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

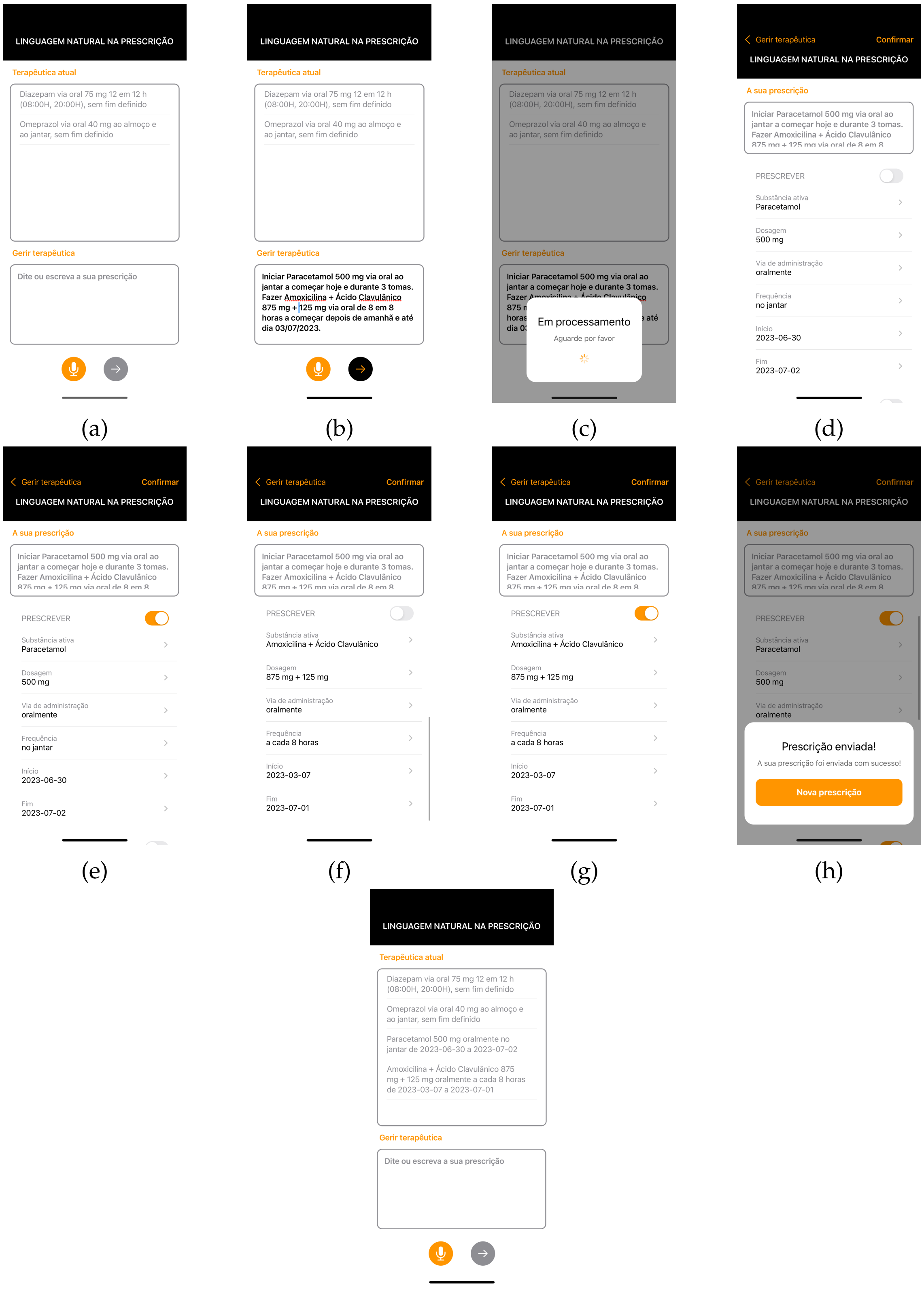
paracetamol 500 miligramas oralmente no jantar de 2023-06-29 a 2023-07-01

Gerir terapêutica

Dite ou escreva a sua prescrição

(h)

Figure H.8: Prescribe use case illustration. (VIII)



LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral de 8 em 8 horas a começar depois de amanhã e até dia 03/07/2023.

(b)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Em processamento

Aguarde por favor

(c)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral de 8 em 8

PRESCREVER

Substância ativa
Paracetamol

Dosagem
500 mg

Via de administração
oralmente

Frequência
no jantar

Início
2023-06-30

Fim
2023-07-02

(d)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral de 8 em 8

PRESCREVER

Substância ativa
Paracetamol

Dosagem
500 mg

Via de administração
oralmente

Frequência
no jantar

Início
2023-06-30

Fim
2023-07-02

(e)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral de 8 em 8

PRESCREVER

Substância ativa
Amoxicilina + Ácido Clavulânico

Dosagem
875 mg + 125 mg

Via de administração
oralmente

Frequência
a cada 8 horas

Início
2023-03-07

Fim
2023-07-01

(f)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral de 8 em 8

PRESCREVER

Substância ativa
Amoxicilina + Ácido Clavulânico

Dosagem
875 mg + 125 mg

Via de administração
oralmente

Frequência
a cada 8 horas

Início
2023-03-07

Fim
2023-07-01

(g)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar Paracetamol 500 mg via oral ao jantar a começar hoje e durante 3 tomas. Fazer Amoxicilina + Ácido Clavulânico 875 mg + 125 mg via oral de 8 em 8

PRESCREVER

Substância ativa
Paracetamol

Dosagem
500 mg

Via de administração
oralmente

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(h)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Paracetamol 500 mg oralmente no jantar de 2023-06-30 a 2023-07-02

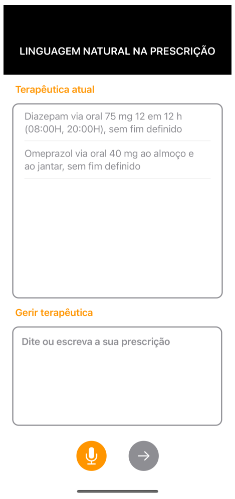
Amoxicilina + Ácido Clavulânico 875 mg + 125 mg oralmente a cada 8 horas de 2023-03-07 a 2023-07-01

Gerir terapêutica

Dite ou escreva a sua prescrição

(i)

Figure H.9: Prescribe use case illustration. (IX)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

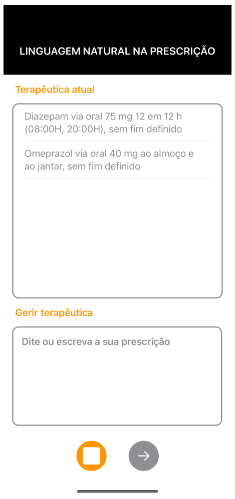
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

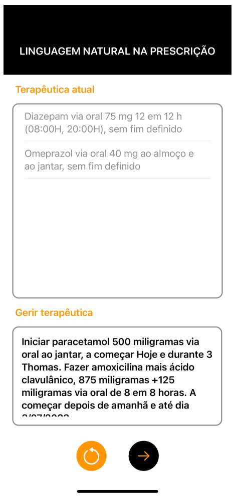
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

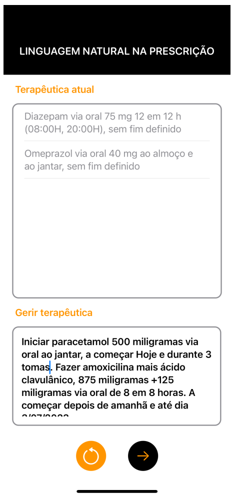
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 Thomas. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 miligramas via oral de 8 em 8 horas. A começar depois de amanhã e até dia

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

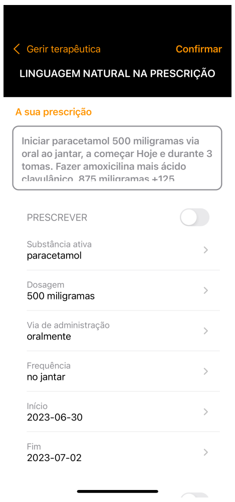
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 tomas. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125 miligramas via oral de 8 em 8 horas. A começar depois de amanhã e até dia

(d)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

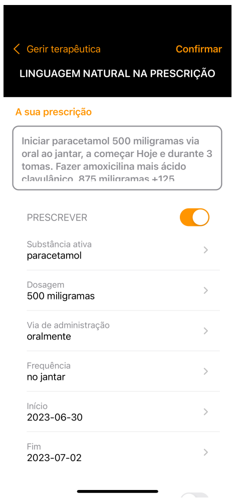
A sua prescrição

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 tomas. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125

PRESCREVER

Substância ativa	paracetamol
Dosagem	500 miligramas
Via de administração	oralmente
Frequência	no jantar
Início	2023-06-30
Fim	2023-07-02

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

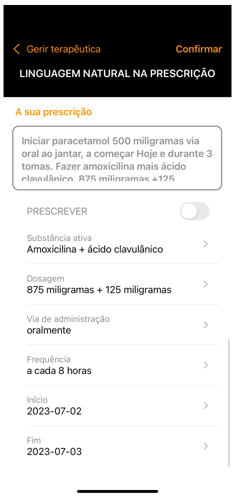
A sua prescrição

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 tomas. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125

PRESCREVER

Substância ativa	paracetamol
Dosagem	500 miligramas
Via de administração	oralmente
Frequência	no jantar
Início	2023-06-30
Fim	2023-07-02

(f)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

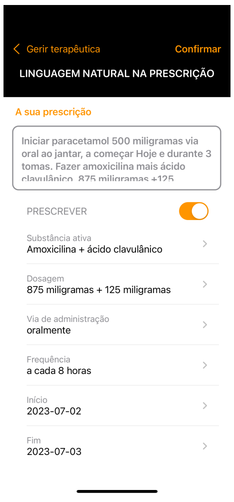
A sua prescrição

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 tomas. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125

PRESCREVER

Substância ativa	Amoxicilina + ácido clavulânico
Dosagem	875 miligramas + 125 miligramas
Via de administração	oralmente
Frequência	a cada 8 horas
Início	2023-07-02
Fim	2023-07-03

(g)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

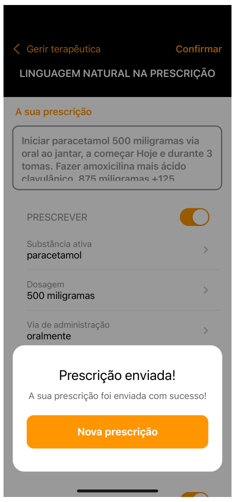
A sua prescrição

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 tomas. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125

PRESCREVER

Substância ativa	Amoxicilina + ácido clavulânico
Dosagem	875 miligramas + 125 miligramas
Via de administração	oralmente
Frequência	a cada 8 horas
Início	2023-07-02
Fim	2023-07-03

(h)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Iniciar paracetamol 500 miligramas via oral ao jantar, a começar Hoje e durante 3 tomas. Fazer amoxicilina mais ácido clavulânico, 875 miligramas +125

PRESCREVER

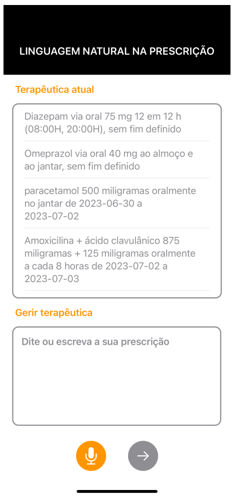
Substância ativa	paracetamol
Dosagem	500 miligramas
Via de administração	oralmente

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(i)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

paracetamol 500 miligramas oralmente no jantar de 2023-06-30 a 2023-07-02

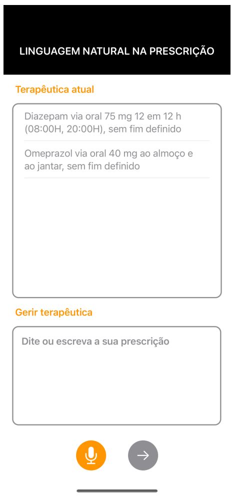
Amoxicilina + ácido clavulânico 875 miligramas + 125 miligramas oralmente a cada 8 horas de 2023-07-02 a 2023-07-03

Gerir terapêutica

Dite ou escreva a sua prescrição

(j)

Figure H.10: Prescribe use case illustration. (X)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

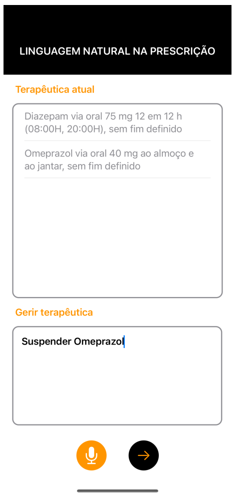
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

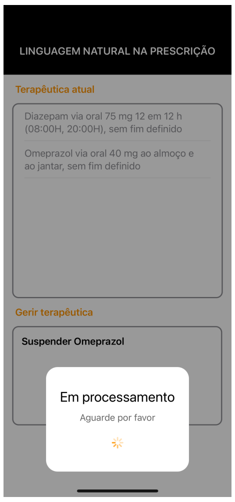
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender Omeprazol

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

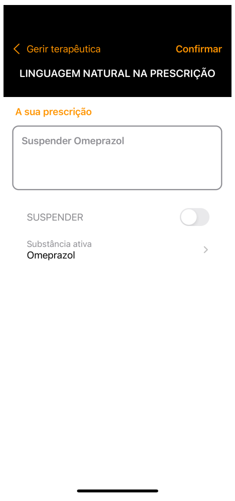
Gerir terapêutica

Suspender Omeprazol

Em processamento

Aguarde por favor

(c)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

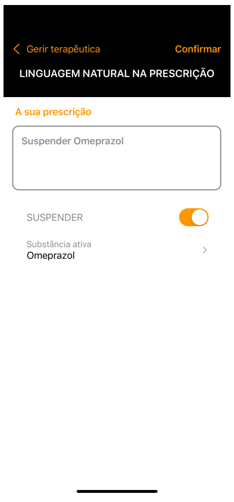
A sua prescrição

Suspender Omeprazol

SUSPENDER

Substância ativa
Omeprazol

(d)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

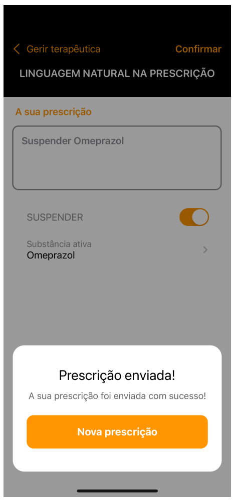
A sua prescrição

Suspender Omeprazol

SUSPENDER

Substância ativa
Omeprazol

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol

SUSPENDER

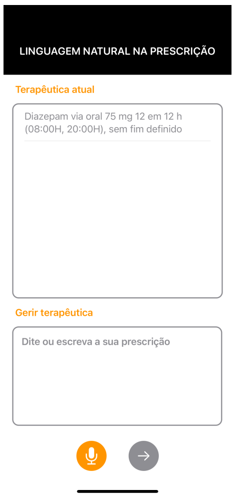
Substância ativa
Omeprazol

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(f)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

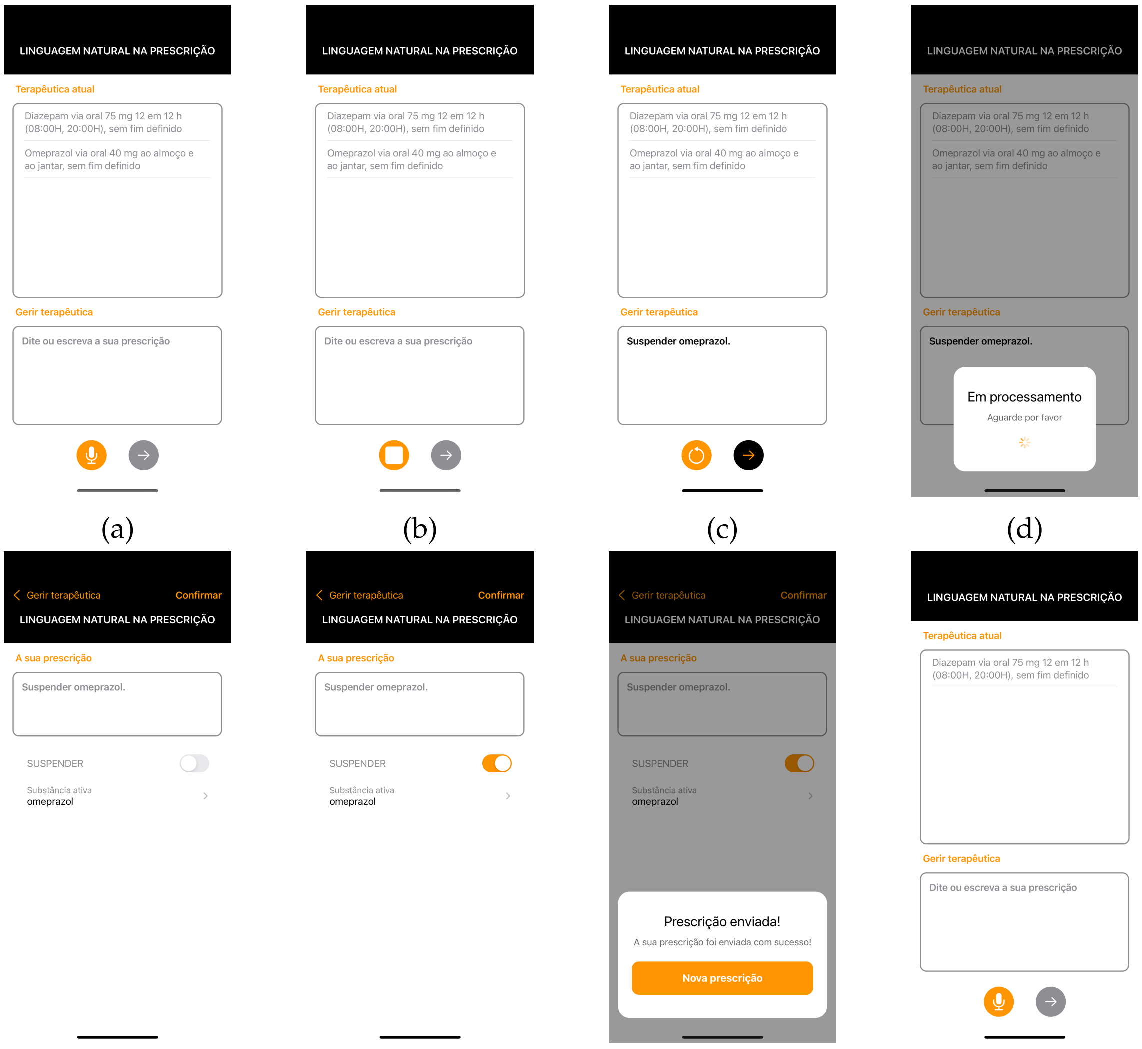
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(g)

Figure H.11: Suspend use case illustration. (I)



(a)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(c)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender omeprazol.

(d)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender omeprazol.

Em processamento

Aguarde por favor

(e)

Gerir terapêutica — Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol.

SUSPENDER

Substância ativa
omeprazol

(f)

Gerir terapêutica — Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol.

SUSPENDER

Substância ativa
omeprazol

(g)

Gerir terapêutica — Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol.

SUSPENDER

Substância ativa
omeprazol

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(h)

LINGUAGEM NATURAL NA PRESCRIÇÃO

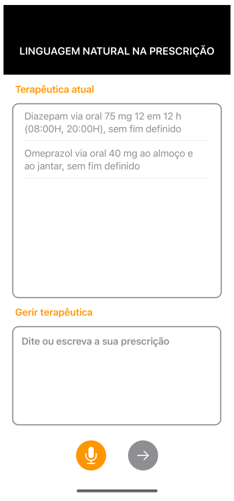
Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

Figure H.12: Suspend use case illustration. (II)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

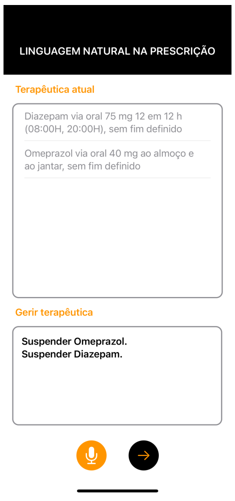
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

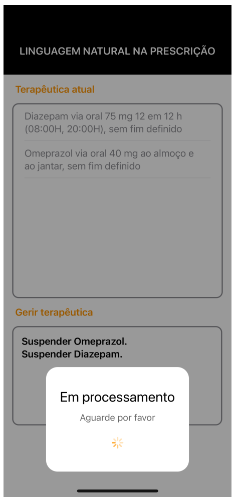
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender Omeprazol.
Suspender Diazepam.

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

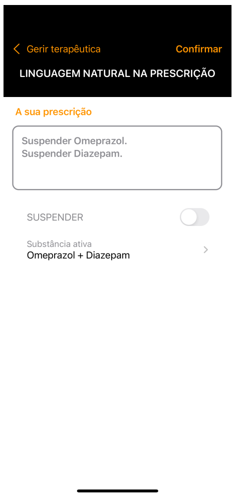
Gerir terapêutica

Suspender Omeprazol.
Suspender Diazepam.

Em processamento

Aguarde por favor

(c)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

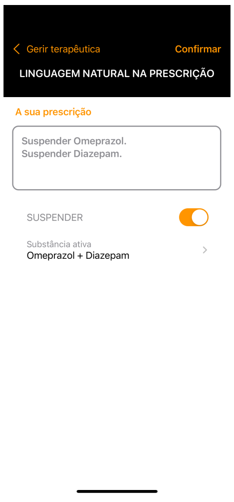
A sua prescrição

Suspender Omeprazol.
Suspender Diazepam.

SUSPENDER

Substância ativa
Omeprazol + Diazepam

(d)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

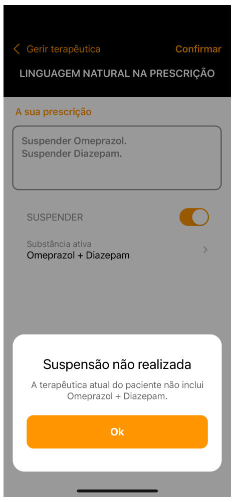
A sua prescrição

Suspender Omeprazol.
Suspender Diazepam.

SUSPENDER

Substância ativa
Omeprazol + Diazepam

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol.
Suspender Diazepam.

SUSPENDER

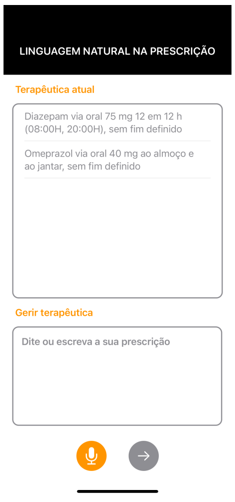
Substância ativa
Omeprazol + Diazepam

Suspensão não realizada

A terapêutica atual do paciente não inclui Omeprazol + Diazepam.

Ok

(f)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

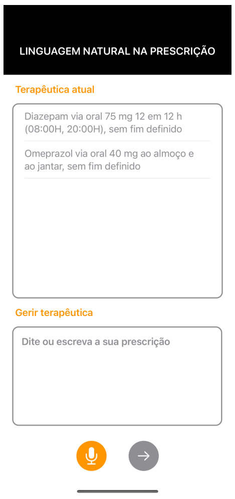
Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(g)

Figure H.13: Suspend use case illustration. (III)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

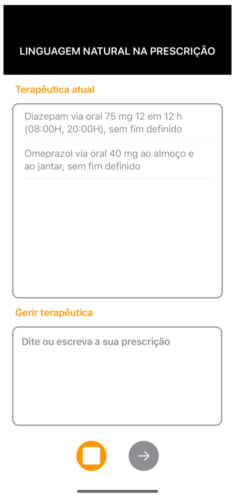
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

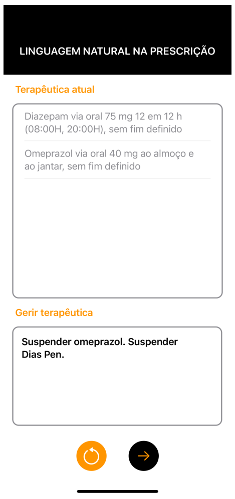
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

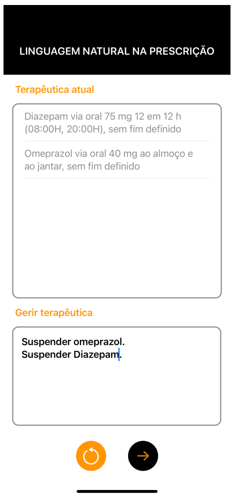
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender omeprazol. Suspender Dias Pen.

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

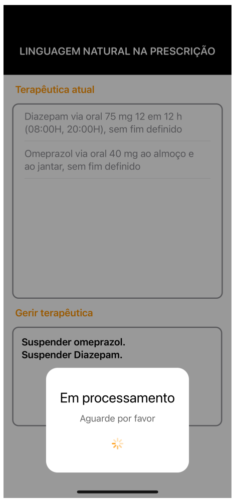
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender omeprazol. Suspender Diazepam.

(d)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

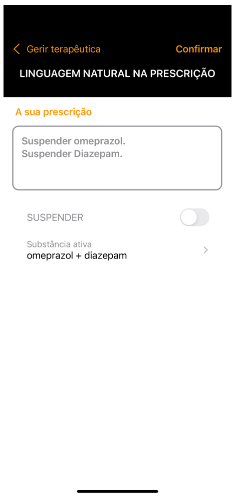
Gerir terapêutica

Suspender omeprazol. Suspender Diazepam.

Em processamento

Aguarde por favor

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

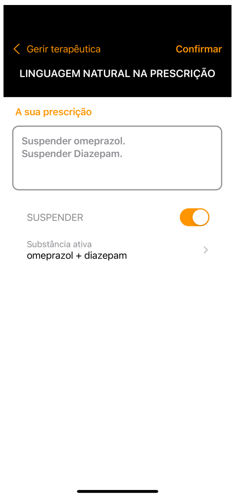
A sua prescrição

Suspender omeprazol. Suspender Diazepam.

SUSPENDER

Substância ativa
omeprazol + diazepam

(f)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

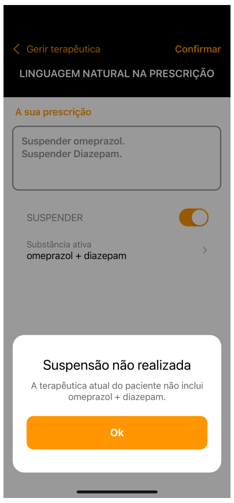
A sua prescrição

Suspender omeprazol. Suspender Diazepam.

SUSPENDER

Substância ativa
omeprazol + diazepam

(g)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol. Suspender Diazepam.

SUSPENDER

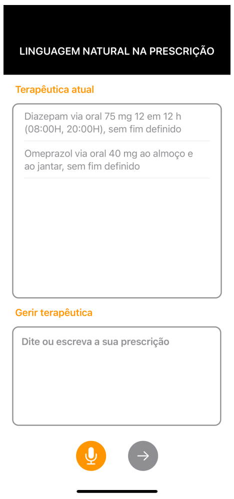
Substância ativa
omeprazol + diazepam

Suspensão não realizada

A terapêutica atual do paciente não inclui omeprazol + diazepam.

Ok

(h)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

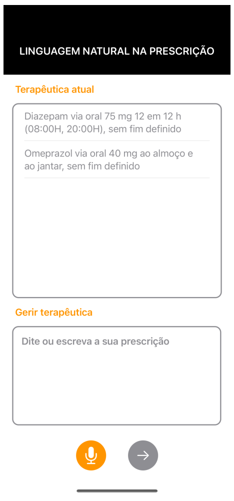
Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(i)

Figure H.14: Suspend use case illustration. (IV)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

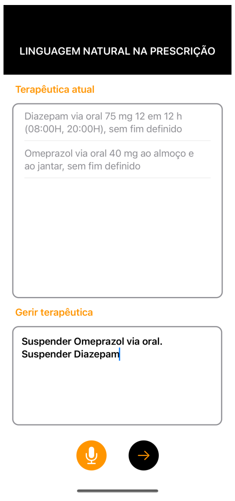
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

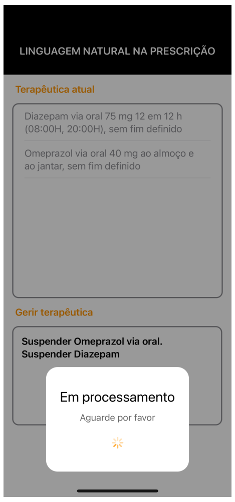
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender Omeprazol via oral.
Suspender Diazepam

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

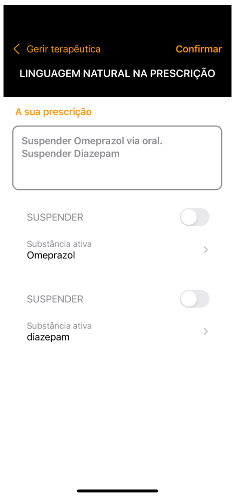
Gerir terapêutica

Suspender Omeprazol via oral.
Suspender Diazepam

Em processamento

Aguarde por favor

(c)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol via oral.
Suspender Diazepam

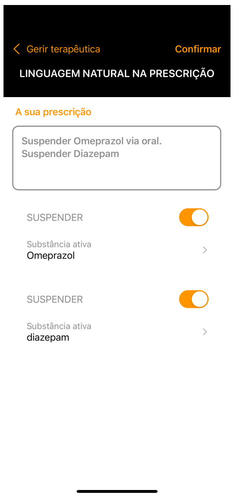
SUSPENDER

Substância ativa
Omeprazol

SUSPENDER

Substância ativa
diazepam

(d)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol via oral.
Suspender Diazepam

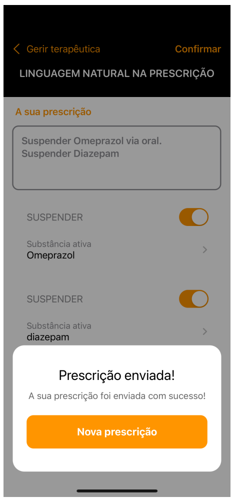
SUSPENDER

Substância ativa
Omeprazol

SUSPENDER

Substância ativa
diazepam

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol via oral.
Suspender Diazepam

SUSPENDER

Substância ativa
Omeprazol

SUSPENDER

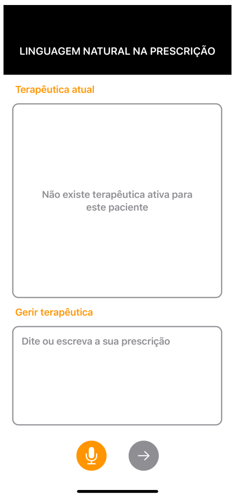
Substância ativa
diazepam

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(f)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

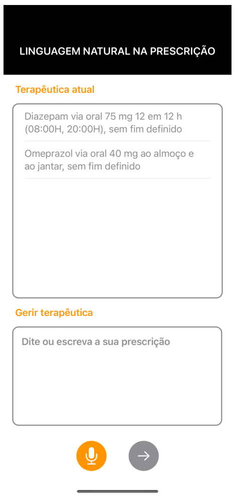
Não existe terapêutica ativa para este paciente

Gerir terapêutica

Dite ou escreva a sua prescrição

(g)

Figure H.15: Suspend use case illustration. (V)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

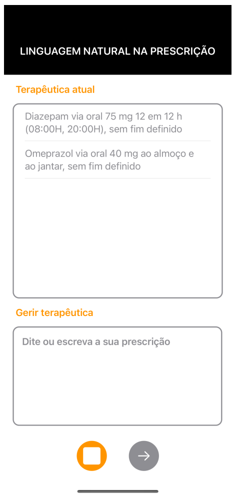
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

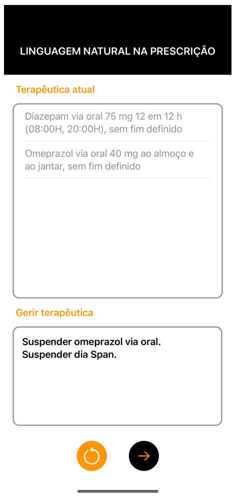
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

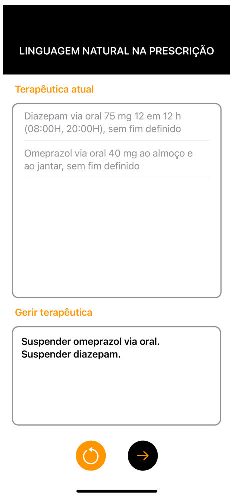
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender omeprazol via oral. Suspender dia Span.

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

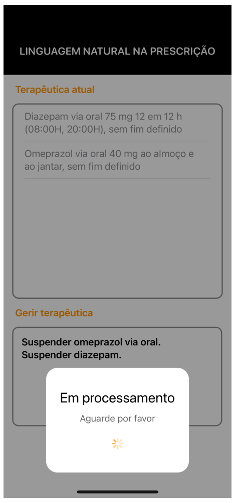
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender omeprazol via oral. Suspender diazepam.

(d)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

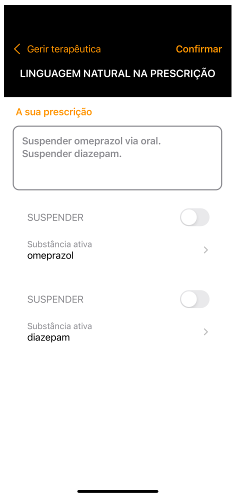
Gerir terapêutica

Suspender omeprazol via oral. Suspender diazepam.

Em processamento

Aguarde por favor

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol via oral. Suspender diazepam.

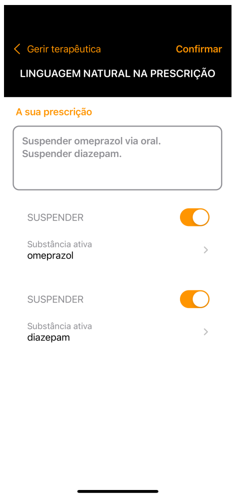
SUSPENDER

Substância ativa
omeprazol

SUSPENDER

Substância ativa
diazepam

(f)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol via oral. Suspender diazepam.

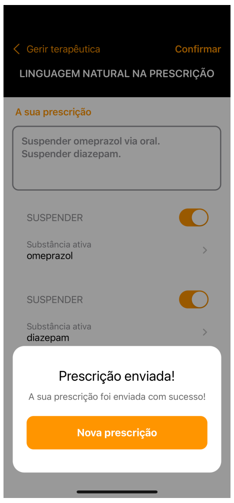
SUSPENDER

Substância ativa
omeprazol

SUSPENDER

Substância ativa
diazepam

(g)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol via oral. Suspender diazepam.

SUSPENDER

Substância ativa
omeprazol

SUSPENDER

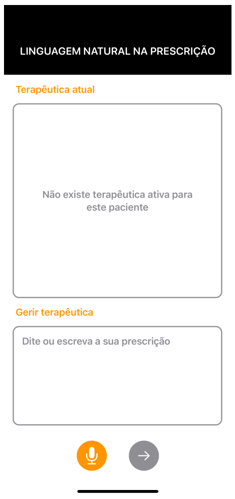
Substância ativa
diazepam

Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

(h)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

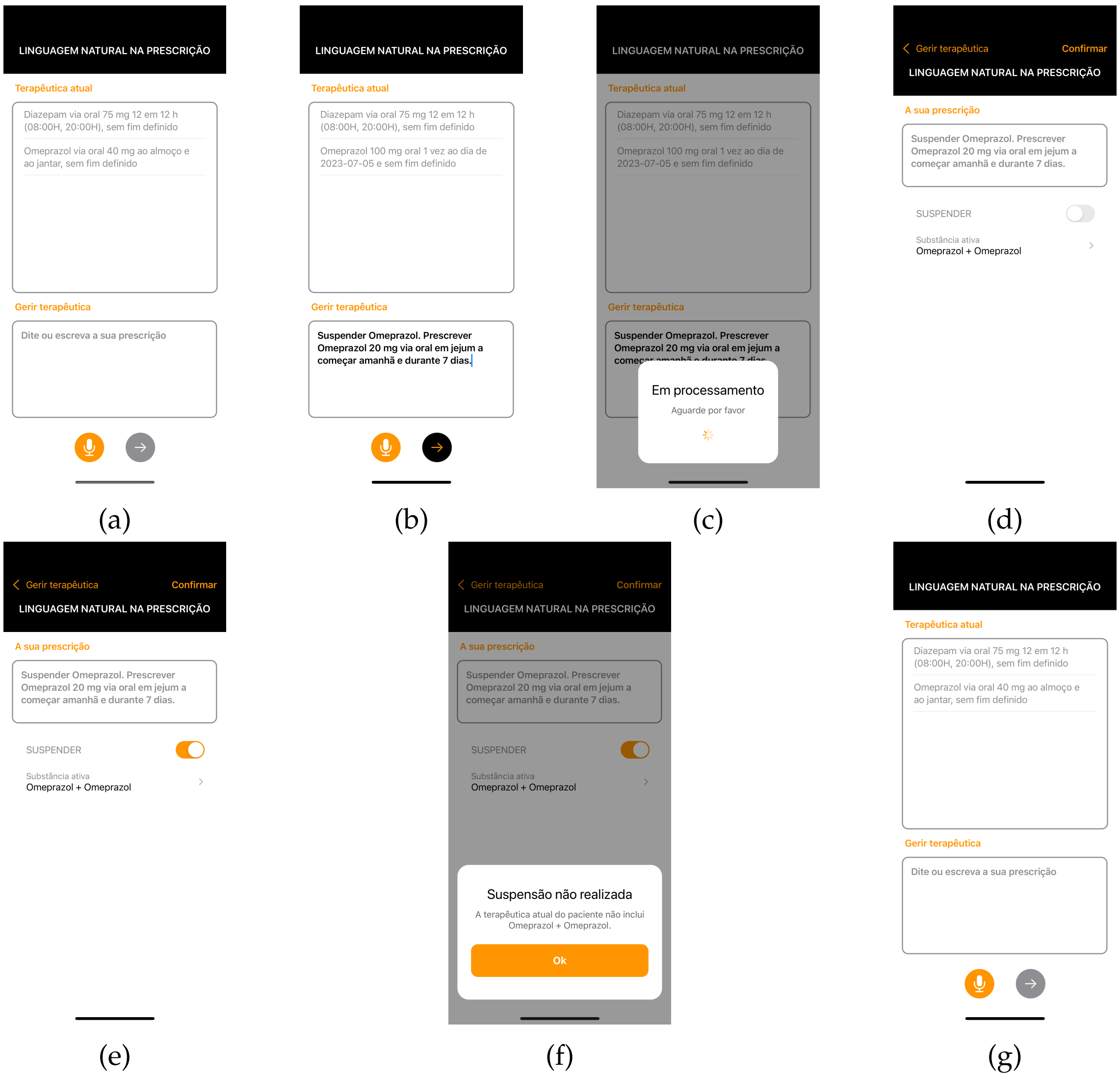
Não existe terapêutica ativa para este paciente

Gerir terapêutica

Dite ou escreva a sua prescrição

(i)

Figure H.16: Suspend use case illustration. (VI)



(a)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol 100 mg oral 1 vez ao dia de 2023-07-05 e sem fim definido

Gerir terapêutica

Suspender Omeprazol. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.

(c)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol 100 mg oral 1 vez ao dia de 2023-07-05 e sem fim definido

Gerir terapêutica

Suspender Omeprazol. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.

Em processamento

Aguarde por favor

(d)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.

SUSPENDER

Substância ativa
Omeprazol + Omeprazol

(e)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.

SUSPENDER

Substância ativa
Omeprazol + Omeprazol

(f)

Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.

SUSPENDER

Substância ativa
Omeprazol + Omeprazol

Suspensão não realizada

A terapêutica atual do paciente não inclui Omeprazol + Omeprazol.

Ok

(g)

LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

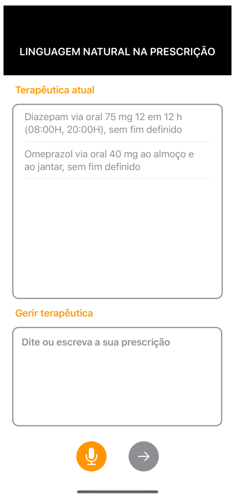
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

Figure H.17: Change use case illustration. (I)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

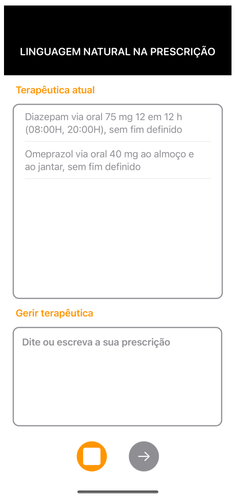
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

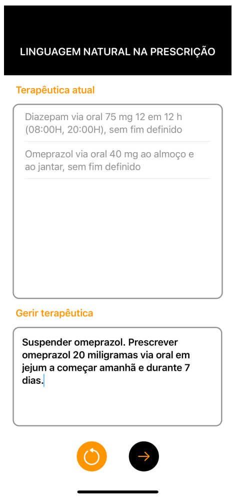
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

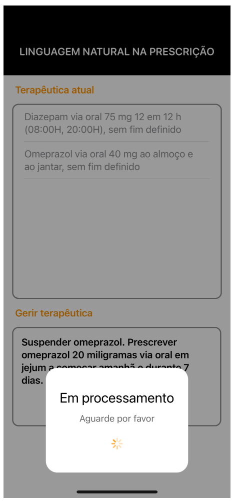
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender omeprazol. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias.

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

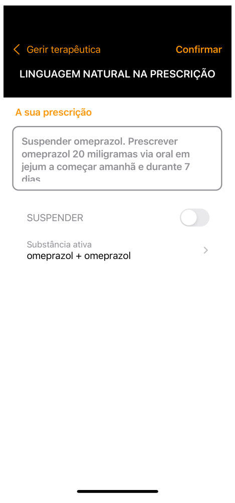
Gerir terapêutica

Suspender omeprazol. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias.

Em processamento

Aguarde por favor

(d)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

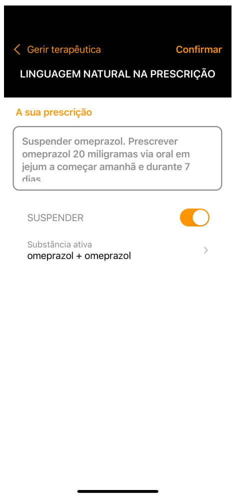
A sua prescrição

Suspender omeprazol. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

Substância ativa
omeprazol + omeprazol

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

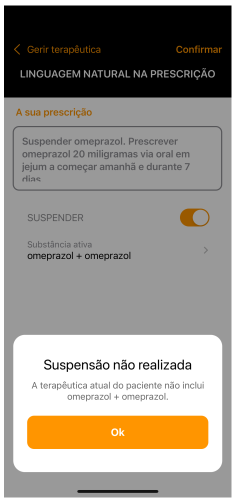
A sua prescrição

Suspender omeprazol. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

Substância ativa
omeprazol + omeprazol

(f)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

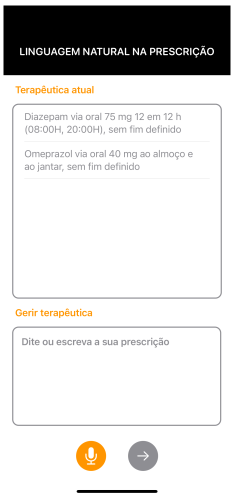
Substância ativa
omeprazol + omeprazol

Suspensão não realizada

A terapêutica atual do paciente não inclui omeprazol + omeprazol.

Ok

(g)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

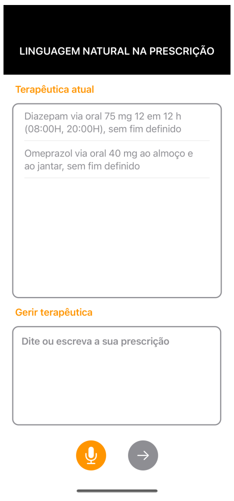
Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(h)

Figure H.18: Change use case illustration. (II)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

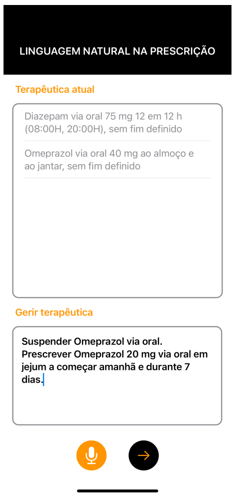
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

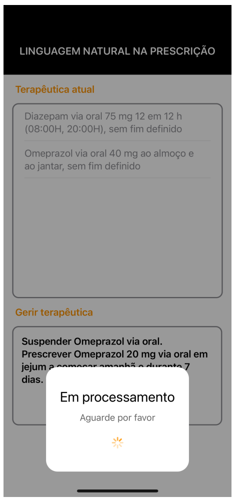
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender Omeprazol via oral. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

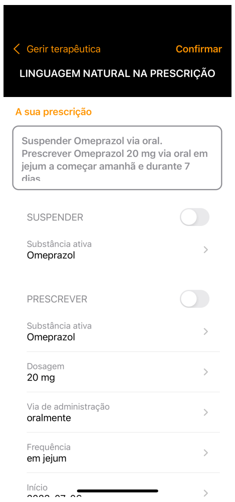
Gerir terapêutica

Suspender Omeprazol via oral. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias.

Em processamento

Aguarde por favor

(c)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol via oral. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

Substância ativa
Omeprazol

PRESCREVER

Substância ativa
Omeprazol

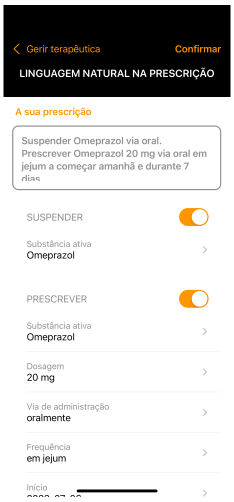
Dosagem
20 mg

Via de administração
oralmente

Frequência
em jejum

Início

(d)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol via oral. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

Substância ativa
Omeprazol

PRESCREVER

Substância ativa
Omeprazol

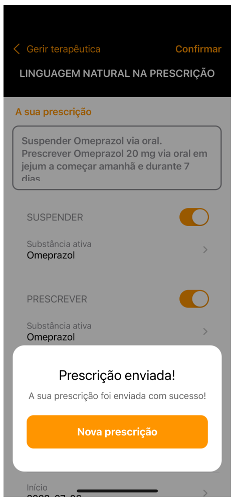
Dosagem
20 mg

Via de administração
oralmente

Frequência
em jejum

Início

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender Omeprazol via oral. Prescrever Omeprazol 20 mg via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

Substância ativa
Omeprazol

PRESCREVER

Substância ativa
Omeprazol

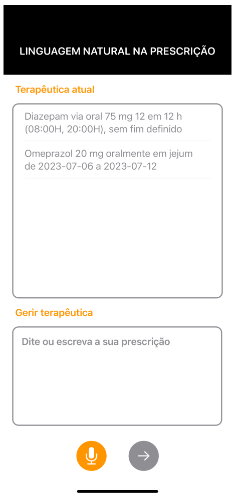
Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

Início

(f)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

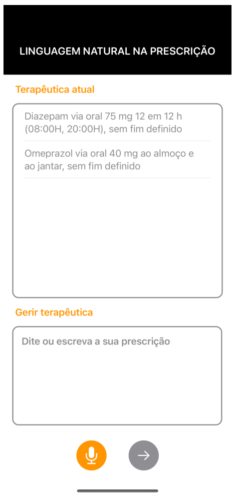
Omeprazol 20 mg oralmente em jejum de 2023-07-06 a 2023-07-12

Gerir terapêutica

Dite ou escreva a sua prescrição

(g)

Figure H.19: Change use case illustration. (III)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

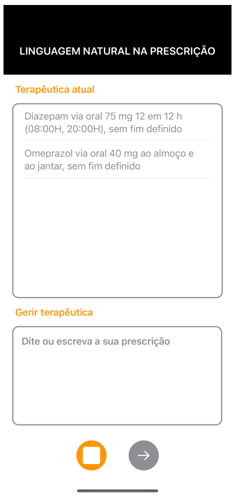
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(a)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

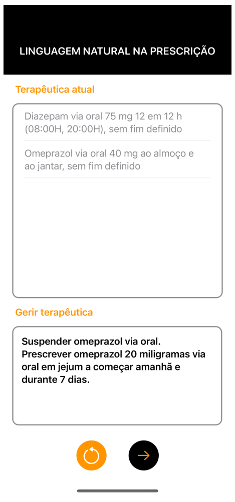
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Dite ou escreva a sua prescrição

(b)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

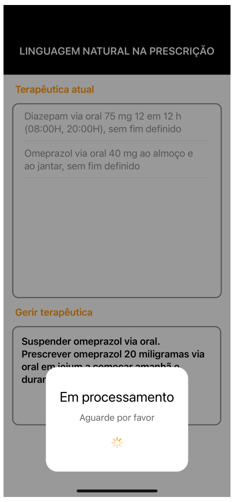
Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

Suspender omeprazol via oral. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias.

(c)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

Omeprazol via oral 40 mg ao almoço e ao jantar, sem fim definido

Gerir terapêutica

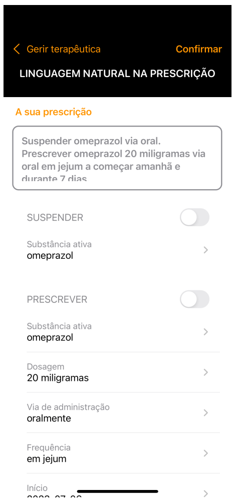
Suspender omeprazol via oral. Prescrever omeprazol 20 miligramas via oral em

durar

Em processamento

Aguarde por favor

(d)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol via oral. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

Substância ativa
omeprazol

PRESCREVER

Substância ativa
omeprazol

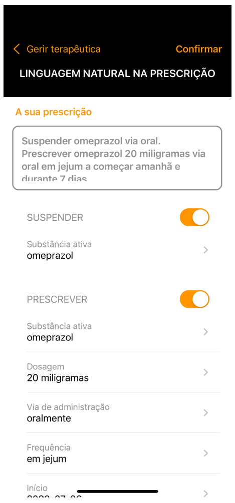
Dosagem
20 miligramas

Via de administração
oralmente

Frequência
em jejum

Início

(e)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol via oral. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

Substância ativa
omeprazol

PRESCREVER

Substância ativa
omeprazol

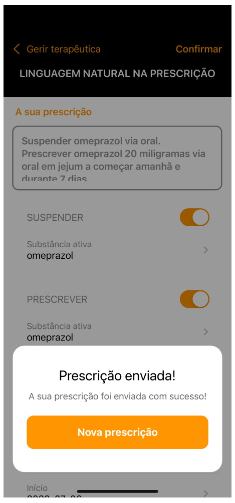
Dosagem
20 miligramas

Via de administração
oralmente

Frequência
em jejum

Início

(f)


Gerir terapêutica Confirmar

LINGUAGEM NATURAL NA PRESCRIÇÃO

A sua prescrição

Suspender omeprazol via oral. Prescrever omeprazol 20 miligramas via oral em jejum a começar amanhã e durante 7 dias

SUSPENDER

Substância ativa
omeprazol

PRESCREVER

Substância ativa
omeprazol

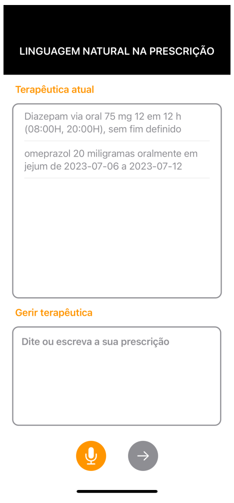
Prescrição enviada!

A sua prescrição foi enviada com sucesso!

Nova prescrição

Início

(g)


LINGUAGEM NATURAL NA PRESCRIÇÃO

Terapêutica atual

Diazepam via oral 75 mg 12 em 12 h (08:00H, 20:00H), sem fim definido

omeprazol 20 miligramas oralmente em jejum de 2023-07-06 a 2023-07-12

Gerir terapêutica

Dite ou escreva a sua prescrição

(h)

Figure H.20: Change use case illustration. (IV)