

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automatic Categorization of Health-related Messages in Online Health Communities

João Paulo Gomes Torres Abelha



Mestrado em Engenharia Informática e Computação

Supervisor: Carla Teixeira Lopes

July 29, 2022

Automatic Categorization of Health-related Messages in Online Health Communities

João Paulo Gomes Torres Abelha

Mestrado em Engenharia Informática e Computação

July 29, 2022

Resumo

A ubiquidade da Internet levou ao crescimento das Comunidades de Saúde Online (OHC - *Online Health Communities*, em inglês) ao longo dos anos. As pessoas utilizam-nas para procurar aconselhamento e apoio emocional e partilhar as suas experiências. Nas OHCs, as discussões são realizadas sob *threads* em que as mensagens são vistas como uma representação das intenções e sentimentos dos utilizadores. Investigadores e programadores podem aproveitar os dados utilizando métodos distintos para prever ou melhorar a experiência dos utilizadores. A categorização de mensagens pode ser útil em vários cenários, podendo ser utilizada para sinalizar mensagens urgentes, fornecer informação adicional aos sistemas de recuperação de informação, e ajudar a compreender as motivações e comportamentos dos utilizadores. Contudo, a classificação manual de mensagens pode ser demorada e propensa à subjetividade de cada indivíduo.

O principal objetivo da dissertação é categorizar automaticamente as mensagens de fóruns de saúde. Esta segue um esquema de classificação baseado num trabalho anterior, cujo objetivo foi desenvolvê-lo considerando a análise do conteúdo de OHCs. Dito isto, criámos três classificadores diferentes: uma categorização de primeiro nível (*offering support, seeking support e group interactions*), uma categorização de segundo nível (*informational, emotional, esteem e network support, personal experience, e interpersonal*), e outra baseada nas polaridades emocionais das mensagens.

Após a recolha de instâncias de dados da MedHelp, um fórum de saúde dividido em diferentes comunidades, expandimos um *dataset* através da construção de uma plataforma de *crowdsourcing*. Criámos várias *pipelines*, fazendo experiências com métodos de extração de características baseados na frequência, *word embeddings* e *Universal sentence embeddings* pré-treinadas. Foram experimentados modelos clássicos de *Machine Learning* (ML) e redes neuronais. O modelo *Support Vector Machine* (SVM) e a Regressão Logística, juntamente com *sentence embeddings*, destacaram-se tanto nos classificadores de primeiro como de segundo nível. Inspirado por vários trabalhos que executam agrupamento de *word embeddings* para representar um documento fazendo a média, tentámos a mesma abordagem usando *sentence embeddings*, obtendo os nossos melhores resultados. Conseguimos desempenhos que conseguem competir com classificadores já conhecidos da literatura, obtendo 82% e 62% de *weighted F1-score* para a categorização de primeiro e segundo nível, respetivamente. Apenas quando se utilizou a extração de características com base na frequência é que *feature engineering*, nomeadamente Part-Of-Speech (PoS), e a técnica de *data augmentation* denominada *Multi-label Synthetic Minority Oversampling Technique* (MLSMOTE), contribuíram para o aumento do desempenho. No entanto, os seus melhores resultados não ultrapassaram os nossos. Conduzimos experiências com várias ferramentas baseadas no léxico para a análise de polaridade, que tiveram um desempenho não tão bom. Por outro lado, uma ferramenta *out-of-the-box* mais complexa usada foi *BERTweet*, que alcançou uma *accuracy* de 66% ao distinguir entre polaridade neutra, positiva, e negativa.

Definimos ainda vários modelos *baseline* que utilizam a vetorização de texto com base na frequência e técnicas clássicas de ML. O nosso melhor obteve uma *accuracy* de 72%, no entanto, em

comparação com *BERTweet*, não foi tão notável ao lidar com mensagens de polaridade positiva. Se apenas considerarmos instâncias positivas e negativas, ao utilizar as redes neurais, *transfer learning*, e as *Universal Sentence Embedding*, foi possível alcançar uma *accuracy* de 90%.

Palavras-Chave: Classificação de texto, Comunidades de Saúde Online, Processamento Natural de Linguagem, Machine Learning

Abstract

The ubiquity of the Internet has led to the growth of Online Health Communities (OHC) over the years. People use them to seek advice and emotional support and share their experiences. In OHCs, the discussions are performed under threads where messages are seen as a proxy of the users' intentions and sentiments. Researchers and developers can leverage the data using distinct methods to predict or enhance user experience. Message tagging can be helpful in several scenarios, as it could be used to flag urgent messages, provide additional information to retrieval systems, and help comprehend users' motivations and behaviors. However, manual message labeling can be time-consuming and prone to the labeler's subjectivity.

The project's primary goal is to categorize online health messages in forums automatically. Our dissertation follows a classification scheme based on a previous work, whose objective was to develop it considering the content analysis of OHCs. That said, we created three different classifiers: a first-level categorization (offering and seeking support and group interactions), a second-level categorization (informational, emotional, esteem and network support, personal experience, and interpersonal), and another based on the emotional polarities of messages.

After collecting data instances from MedHelp, a health-related network divided into different communities, we expanded a dataset tailored for OHC content by building a crowdsourcing platform. We created several pipelines, experimenting with frequency-based feature extraction methods, pre-trained words and sentence embeddings. We tried both classical and neural network-based models. Support Vector Machine (SVM) and Logistic Regression, together with Universal Sentence Encoders, stood out in both first- and second-level classifiers. Inspired by several works performing word embedding grouping to represent a document, we tried out the same approach using sentence embeddings, getting our best results. We got performances that can compete with classifiers from the literature by obtaining 82% and 62% of the weighted F1-score for the first-level and second-level categorization, respectively. Only when using frequency-based feature extraction were feature engineering, namely Part-Of-Speech (PoS), tagging, and the augmentation technique called Multi-label Synthetic Minority Oversampling Technique (MLSMOTE), a beneficial boosting performance. Nevertheless, their best pipeline did not surpass the capabilities of our best one. We experimented with several lexicon-based tools for sentiment analysis, which performed poorly. The best out-of-the-box tool was *BERTweet*, which achieved an accuracy of 66% when distinguishing between neutral, positive, and negative polarity.

We defined several baselines that used frequency-based text vectorization and classical machine learning techniques. Our best one achieved a 72% accuracy, yet, compared to *BERTweet*, it was not as remarkable when dealing with positive instances. If only dealing with positive and negative instances, by leveraging Neural networks, transfer learning, and Universal Sentence embedding, we achieved an accuracy of 90%.

Keywords: Text Classification, Online Health Communities, Natural Language Processing, Machine Learning

Acknowledgements

This endeavor would not have been possible without my supervisor, professor Carla Teixeira Lopes for her engagement, knowledge, support, and continuous feedback. I am also extremely grateful to those who contributed to the annotation of messages for their patience and demonstrated availability.

I am also grateful to my friends, especially Diana Cardoso, André Madureira, Margarida Cruz, and Rafael Matias for the companionship, support, and motivation they gave me throughout the months of work.

Lastly, I would be remiss not to mention my family, especially my parents. Their belief in me has kept my spirits and motivation high during this process.

*“Nothing in life is to be feared, it is only to be understood. Now is the time to understand more,
so that we may fear less.”*

Marie Curie

Contents

1	Introduction	1
1.1	Context and Motivations	1
1.2	Challenges	4
1.3	Contributions	4
1.4	Document Outline	5
2	Background: Automatic Text Classification	6
2.1	Data Collection and Annotation	6
2.2	Pre-processing	7
2.2.1	Feature Selection	10
2.2.2	Feature Extraction	12
2.2.3	Feature Engineering	14
2.3	Text Classification	14
2.3.1	Lexicon-based Approaches	14
2.3.2	SentiWordNet	16
2.3.3	Machine Learning Models	16
2.3.4	Evaluation	23
2.4	NLP Libraries	27
3	Automatic Classification of Messages in Online Health Communities	29
3.1	Common Tasks	29
3.1.1	Intent Classification	29
3.1.2	Depression and Schizophrenia Detection	31
3.1.3	Suicide Detection	34
3.1.4	Surveillance	36
3.1.5	Sentiment Analysis	38
3.1.6	Topic Classification	39
3.2	Literature Review Summary	41
3.3	Relevant Datasets and Schemes	43
3.4	NLP Tools	45
3.5	Existing Online Health Communities	46
3.6	Ethical Concerns	46
4	Problem Statement and Methodology	49
4.1	Previous Content-based Hierarchical Scheme	49
4.2	Adapted Hierarchical Scheme	49
4.2.1	High-level Classifier	50
4.2.2	Second-level Classifier	51

4.2.3	Polarity Analysis	53
4.3	Research Questions	54
4.4	Methodology	54
4.4.1	First and Second Level Categorization of Messages	54
4.4.2	Polarity Analysis	54
4.4.3	Evaluation Metrics	56
5	Dataset Building and Analysis	57
5.1	Data Collection	57
5.2	Data Labelling and Crowdsourcing Platform	58
5.2.1	Relabel Existing Messages for Polarity Analysis	63
5.2.2	Level of Agreement of the New Instances	63
5.3	Dataset Characterization	64
5.3.1	High-level Categories	64
5.3.2	Second-level Categories	65
5.3.3	Polarity Analysis	65
5.3.4	Text Statistics	65
6	Data Preprocessing and Text Vectorization	69
6.1	Data Preprocessing	69
6.1.1	Remove HTML Tags	69
6.1.2	Dealing with Certain Patterns	69
6.1.3	Slang Expansion	70
6.1.4	Dealing with Contractions	70
6.1.5	Spell Correction	71
6.1.6	Tokenization	71
6.1.7	Removal of Stop Words	71
6.1.8	Lowercasing	72
6.1.9	Stemming/Lemmatizing	72
6.1.10	Punctuation	72
6.2	Text Vectorization	72
6.2.1	Frequency-based Feature Extraction	73
6.2.2	Word Embeddings	73
6.2.3	Sentence Embeddings	74
6.2.4	Feature Space Size	75
6.2.5	Relation Between Preprocessing, Text Vectorization and Experiments . .	75
7	First-level Categorization	78
7.1	Procedure Steps	78
7.1.1	Dataset Splitting	78
7.1.2	Adversarial Validation	79
7.1.3	Machine Learning Models	80
7.1.4	MLSMOTE	83
7.1.5	Feature Engineering	84
7.2	Results and Discussions	84
7.2.1	Preprocessing	85
7.2.2	Adversarial Validation	86
7.2.3	Comparison Between Pipelines	87
7.2.4	Feature Engineering	92

7.2.5	MLSMOTE	94
8	Second-level Categorization	96
8.1	Results and Discussions	96
8.1.1	Preprocessing	96
8.1.2	Adversarial Validation	96
8.1.3	Comparison Between Pipelines	98
8.1.4	Feature Engineering	103
8.1.5	MLSMOTE	104
9	Sentiment Polarity	106
9.1	Lexicon-based Approaches	107
9.1.1	ANEW	107
9.1.2	VADER	107
9.1.3	TextBlob	107
9.1.4	AFINN	108
9.1.5	SentiWordNet	108
9.2	Machine learning-based Approaches	108
9.2.1	Dataset Splitting	108
9.2.2	Baseline Models	108
9.2.3	Neural Networks and Transfer Learning	109
9.3	Transformers and Deep Learning	110
9.4	Results and Discussions	111
9.4.1	ANEW	111
9.4.2	AFINN	113
9.4.3	VADER	114
9.4.4	TextBlob	115
9.4.5	SentiWordNet	116
9.4.6	Baseline Models	118
9.4.7	Neural Network and Transfer Learning	119
9.4.8	Transformers and Deep Learning	120
10	Conclusions and Future Work	122
	References	125
A	Appendix	135
A.1	First-level Categorization	135
A.2	Second-level Categorization	139
A.3	Polarity Analysis	143

List of Figures

2.1	Data points with 2 features in a 2D space	18
2.2	Support Vector Machine concepts	19
2.3	Simple decision Tree for animal classification	20
2.4	Fully connected network with 3 output classes	21
2.5	Unrolled recurrent neural network	22
2.6	Confusion matrix for a binary classifier	24
2.7	Error of training and validation set over time	25
2.8	AUC	26
3.1	Health fields and corresponding approaches	42
3.2	Target Languages	42
3.3	Data sources	43
4.1	High-level classification classes	51
4.2	Hierarchical scheme with second-level categories	52
4.3	Emotion classification schema	53
4.4	Pipeline for classical machine learning	55
4.5	Pipeline to test different polarity analysis tools	55
4.6	Neural network with transfer learning pipeline	56
5.1	Website stack	58
5.2	Site Map	59
5.3	Instruction Page	59
5.4	Participation Page	60
5.5	Thanks Page	60
5.6	Labeling strategy using a majority vote to decide the presence of a label	60
5.7	Help button in each label during annotation	61
5.8	Help button Pop up	62
5.9	Entire process of message annotation	62
5.10	Multilabel classifier votes' distribution	63
5.11	Sentiment polarity votes' distribution	64
5.12	Upset diagram of first-level classes	64
5.13	Upset diagram of the second-level classes	65
5.14	Classes distribution of the second-level of the hierarchical scheme	66
5.15	Polarity distribution	66
5.16	Length distribution of the textual data	66
5.17	Number of words distribution of the textual data	67
5.18	Number of sentences distribution	67

6.1	PoS-tagging when doing lemmatization	72
6.2	Grouping word embeddings to create document embeddings	74
6.3	Pre-processing steps done for all the tasks	76
6.4	Pre-processing example	76
7.1	Adversarial Validation	79
7.2	Polarity analysis pipeline	80
7.3	Tasks different ML algorithms support	81
7.4	Neural network pipeline	83
7.5	BILSTM pipeline	83
7.6	Loss of the training and test data	92
9.1	Polarity analysis approaches explored over the chapter	106
9.2	Neural architecture used for transfer learning	109
9.3	Sentiment140 sample word length distribution	110
9.4	Valence distribution of the dataset	111
9.5	Valence distribution of the dataset	113
9.6	Accuracy and weighted F1-score for different threshold values	114
9.7	Accuracy and weighted F1-score for different threshold values of VADER	114
9.8	Accuracy and weighted F1-score for different threshold values of TextBlob	116
9.9	Accuracy and weighted F1-score for different threshold values of SentiWordNet	117

List of Tables

2.1	Open source NLP libraries and their features	27
2.2	Open source NLP libraries and their features	27
3.1	Intention classification in OHC	30
3.2	Datasets for intent classification	31
3.3	Depression and Schizophrenia detection in OHC	33
3.4	Datasets for Depression and Schizophrenia detection	33
3.5	Suicide detection in OHC	35
3.6	Datasets for suicide detection in OHC	35
3.7	Surveillance in OHC	37
3.8	Datasets for surveillance in OHC	38
3.9	Sentiment analysis in OHC	39
3.10	Datasets for emotional analysis in OHC	40
3.11	Topic classification in OHCs	41
3.12	Datasets of topic classification in OHCs	41
3.13	Scheme proposed by Balakrishnan, Athira et. al (2021) [13]	44
3.14	Scheme proposed by Balakrishnan, Zhang et. al (2019) [121]	44
3.15	Tools used in the literature	45
3.16	Popular websites that contain OHC	47
4.1	Scheme proposed by Lopes & Guimarães (2019) [76]	50
6.1	Parameter set for Bow and TF-IDF vectorization	73
6.2	Pre-trained embeddings	75
6.3	Impact of preprocessing steps on some textual statistics	77
7.1	Classical machine learning and their varying parameters	81
7.2	Neural network settings	82
7.3	Features to be used as feature engineering	85
7.4	Average performance and standard deviation of some noticeably interesting combinations for the first-level message categorization	86
7.5	Adversarial Task on BOW	86
7.6	Adversarial Task with TF-IDF vectorization	86
7.7	Adversarial Task with Universal Sentence Encoders	87
7.8	Best configuration parameters for SVM (binary relevance) and frequency based vectorization technique	87
7.9	Classification report for SVM using binary relevance	88
7.10	Best configuration parameters for Logistic Regression (classifier chain) and frequency based vectorization technique	88

7.11	Classification report for Logistic Regression using classifier chains	88
7.12	Classification report for SVM with word embeddings	89
7.13	Classification report for Logistic Regression with word embeddings	89
7.14	Best classical machine learning models using universal sentence encoders	91
7.15	Classification report for neural networks with pre-trained embeddings	91
7.16	BiLSTM evaluation metrics	92
7.17	Feature engineering on SVM (binary relevance)	93
7.18	Feature engineering on Logistic Regression (Classifier Chain)	93
7.19	Feature engineering on SVM (Classifier Chain) with Universal Sentence Encoders	94
7.20	Best Logistic Regression model performance with MLSMOTE	94
7.21	SVM with universal sentence encoders performance with MLSMOTE	95
8.1	Average performance and standard deviation of some noticeably interesting combinations for the second-level classifier	97
8.2	Adversarial Task on BOW	97
8.3	Adversarial Task with TF-IDF vectorization	97
8.4	Adversarial Task with Universal Sentence Encoders	97
8.5	Gaussian NB best configuration for the third task	98
8.6	Classification report for Gaussian NB with the best parameters	98
8.7	SVM best configuration for the third task	99
8.8	Classification report for SVM with the best parameters	99
8.9	Logistic Regression best configuration for the third task	100
8.10	Classification report for Logistic Regression with the best parameters	100
8.11	Classification report for SVM with word embeddings	101
8.12	Classification report for Logistic Regression with word embeddings	101
8.13	Best classical machine learning models using universal sentence encoders	102
8.14	Classification report for neural networks with pre-trained embeddings	103
8.15	Feature engineering for SVM with TF-IDF vectorization	104
8.16	Feature engineering for Gaussian NB with Universal Sentence Encoder	104
8.17	SVM with TF-IDF using MLSMOTE	105
8.18	NB with universal sentence encoders with SMLSMOTE	105
9.1	Training configuration	109
9.2	ANEW classification report using a mean strategy	112
9.3	ANEW classification report using a median strategy	112
9.4	ANEW lemmas frequency and their polarity	112
9.5	AFINN classification report for a threshold of zero	113
9.6	VADER classification report for a threshold of 0.65	115
9.7	VADER classification report for a threshold of 0.3	115
9.8	TextBlob classification report for a threshold of 0.25	116
9.9	Classification report for positive and negative samples using TextBlob	116
9.10	Classification report of SentiWordNet	117
9.11	Classification report of SentiWordNet for positive and negative classes	117
9.12	Adversarial Task on BOW for polarity analysis	118
9.13	Adversarial Task with TF-IDF vectorization for polarity analysis	118
9.14	SVM classification report with preprocessing	119
9.15	Neural network classification report on negative and positive samples of our dataset	119
9.16	Neural network classification report after fine-tuning	120
9.17	Classification report of <i>BerTweet</i>	120

9.18	Classification report of transformer-based model <i>en – sentiment</i>	121
9.19	Classification report of RNN-based model <i>en-fast</i>	121
A.1	Best configuration parameters for Decision Tree and frequency based vectoriza- tion technique	135
A.2	Classification report for Decision Tree using binary relevance	135
A.3	Best configuration parameters for Gaussian NB (binary relevance) and frequency based vectorization technique	136
A.4	Classification report for Gaussian NB using binary relevance	136
A.5	Best configuration parameters for Random Forest and frequency based vectoriza- tion technique	136
A.6	Classification report for Random Forest	136
A.7	Best configuration parameters for KNN and frequency based vectorization technique	137
A.8	Classification report for Random Forest	137
A.9	Classification report for Decision Tree with word embeddings	137
A.10	Classification report for Gaussian NB with word embeddings	138
A.11	Classification report for Random Forest with word embeddings	138
A.12	Classification report for KNN with word embeddings	138
A.13	Decision Tree best configuration for the third task	139
A.14	Classification report for DT with the best parameters	139
A.15	Random Forest best configuration for the third task	139
A.16	Classification report for Random Forest with the best parameters	140
A.17	KNN best configuration for the third task	140
A.18	Classification report for KNN with the best parameters	140
A.19	Classification report for Decision Tree	141
A.20	Classification report for Random Forest with word embeddings	141
A.21	Classification report for Gaussian NB with word embeddings	141
A.22	Classification report for KNN with word embeddings	142
A.23	Small summary of the metrics related to frequency-based vectorization methods and classical machine learning	142
A.24	Pipeline best parameters	143
A.25	Multinomial NB classification report without preprocessing	144
A.26	Multinomial NB classification report with preprocessing	144
A.27	Decision Tree classification report without preprocessing	144
A.28	Decision Tree classification report with preprocessing	144
A.29	Gaussian NB classification report without preprocessing	144
A.30	Gaussian NB classification report with preprocessing	145
A.31	KNN classification report without preprocessing	145
A.32	KNN classification report with preprocessing	145
A.33	SVM classification report without preprocessing	145
A.34	Logistic Regression classification report without preprocessing	145
A.35	Logistic Regression classification report with preprocessing	146
A.36	Random Forest classification report without preprocessing	146
A.37	Random Forest classification report with preprocessing	146

Abbreviations

ADR	Adverse drug reaction
API	Application Programming Interface
AUC	Area under the curve
BOW	Bag of words
BPE	Byte Pair Encoding
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
CV	Cross Validation
ELU	Exponential linear unit
FN	False Negative
FP	False Positive
IRPL	Imbalance ratio per label
LR	Logistic Regression
LDA	Linear Discriminant Analysis
LIWC	Linguist Inquiry and Word Count
LSTM	Long short-term memory
MARS	Multivariate Adaptative Regression Spline
MIR	Mean Imbalance ratio
MERN	MongoDB, Express, React, Node
ML	Machine Learning
MLSMOTE	Multi-label Synthetic Minority Oversampling Technique
NB	Naive Bayes
NER	Name-Entity Recognition
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NN	Neural network
OHC	Online Health Community
OOV	Out of Vocabulary
PCA	Principal component analysis
POS	Part-of-speech
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SG	Skip-gram
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
tanh	Hyperbolic tangent
TF-IDF	Term Frequency–Inverse Document Frequency
TN	True Negative
TP	True Positive
UMLS	Unified Medical Language System
Vader	Valence Aware Dictionary and Sentiment Reasoner

Chapter 1

Introduction

The current chapter introduces the context and motivations for the dissertation. It describes the main challenges and contributions and, lastly, the document outline.

1.1 Context and Motivations

Nowadays, billions of people use the internet as a tool for information retrieval, communication, and entertainment. According to a recent Pew Internet Research report [26] conducted in 2021, not only did 72% of the American population but also 84% of Americans between 18 and 29 years old utilize social media regularly. According to another Pew Internet Research [49] in 2014, 72% of adults have searched for health-related content online. Additionally, around a fourth of adult individuals stated having read or watched someone's experience regarding health issues on the internet over the year before. Moreover, Google Health Vice President David Feinberg divulged in 2019 that an estimated 7% of Google's daily searches are health-related. There is a wide range of factors behind the incremental usage of social media for health-related purposes.

New posts are constantly arising not only because making them is quick, convenient, and economical, but it also aids users to voice their problems and concerns according to their state of affairs [33]. Online conversations are capable of breaking the time zone and geographical barriers, enabling people all over the world with different backgrounds to participate in them. In addition to these advantages, what motivates the usage of online medical forums is to gather information, partake in discussions, ask questions, share personal experiences, or assist others while maintaining anonymity [1, 19]. Patients reveal that it is meaningful for them to connect with peers and use the internet as a source of information, as was shown in a cancer patient study [19]. The ubiquity of social media and the relentless generation of data make them vast information sources with many hidden potentials.

Leveraging health-related information from online posts allows for the exploration of patient data from sources other than clinical or specialized reports [75]. Apart from this, it has been exhibited the preference of individuals to share clinical information online to the detriment of their clinical research and physicians [38], especially when it comes to users with stigmatizing illnesses

[17]. Some works even show that there is knowledge in online forums that differ from that of physicians [47, 57]. This can be regarded as a double-edged sword. Although it caters differentiated knowledge often grounded by successful personal experiences, it can also be dangerous. So as long as people take information with a grain of salt, there is much to be gained. For the most part, people can focus on suggestions with a fair amount of engagement, such as likes, dislikes, and comments. People can use them to assess the veracity and trustworthiness of what is transmitted.

The user's motivations stated before, and the event of the web 2.0 prompted the thrive of patient-centered health communities, such as WebMD and DailyStrength. These constitute two popular health forums. In addition, more general-purpose social networks such as Reddit and Meta are organized in content-specific communities to which people can subscribe or adhere. Users can interact with others by posting or engaging in a discussion board. In addition, other interactions may be possible through downvoting and upvoting. This feature often serves as a guide to assess the quality of what is being shared. With this evident proactive and support-driven attitude, a new shift has been observed in the role of the "online patient" from consumer towards both the consumer and producer of health information [13]. Besides this, there are users in some communities whose role is the moderator. They are specifically relevant in communities where controversial or sensitive topics are discussed. Moderators typically have extra privileges as they can ban users and delete posts that disrupt the forum's goal or hinder other participants' well-being. This creates opportunities to address some issues so as to possibly assist moderators as the role they play in communities where depression or suicide is the focal topic is paramount. Furthermore, some health forums also have users that are health professionals. Their role may be occasionally decisive mainly because they are knowledgeable. Moreover, if experts are present, people may feel more confident about sharing their medical conditions and experiences on the internet.

The literature already showcased approaches with the potential to tackle health problems such as surveillance, outbreak predictions, health communication, sentiment, and behavior analysis. Not only is the extraction of information from health communities for each individual relevant, but also as a group since they can be utilized as a surveillance system, able to predict outbreaks of diseases. For instance, the tool Flu Trends, a tool for outbreak detection of Influenza activity in over 25 countries, was able to predict the outbreak of influenza 7 to 10 days before the Centers for Disease Control and prevention surveillance systems [24]. This reveals the potential of using the available information in health forums to promote individual and collective well-being. Outbreak detection would also enable the timely creation of preventive measures and policies. Another example of a surveillance task deals with the detection of substance abuse as there has been an increased usage of E-cigarettes over the last few years [10].

Another common task existent in the literature is sentiment analysis which is widely applied in the context of Natural Language Processing (NLP). Sentiment analysis has been frequently used for interdisciplinary tasks, i.e., as a proxy to give voice to the users concerning different domains and subjects, such as treatments [69]. There are works concerned with the topics discussed in social media, which are usually related to symptoms, diagnoses, and non-medical information

[13].

All the works have a motivation behind them, either to save lives, namely those related to suicide prevention, or to better understand and predict human behavior in order to act accordingly. This can, in the long-term, lead to a society where health is monitored to a point where we would be able to prevent disastrous scenarios or events.

Literature often performs post-level binary or multiclass classification [5, 9, 12, 65, 79] rather than multilabel classification [13, 121], which is unfortunate as posts are information-rich and often convey several aspects of data. The former problem is conditioned to choose a single label among several ones, whereas the latter allows for the selection of every possible subset. Moreover, the classification schemes exhibited in the literature tend to be tailor-made for specific health communities. Hence, there is a lack of research regarding a more holistic classification of messages that considers any health-related community's sentiments, intentions, and motivations. This leaves a gap in the literature that can be filled, leading to a more insightful analysis of the messages exchanged. In addition, this work can make provision for a more complex study of forums' dynamics.

Online conversations have hidden patterns about the behaviors, state of mind, opinions of users, discussed topics, and much more that can be leveraged to create more user-centric products and services. Likewise, performing user profiling and segmenting the target audience facilitates the creation of personalized services according to the users' preferences and psyche perceived through message categorization. Together with similarity and ranking measures, it can be drawn on searches for fitting information already available on the web. Take as an example two questions encoded as a vector of numbers with a small distance measure. This may lead to questions already answered. Advantages can be listed immediately: we can find an answer quickly, preventing the generation of redundant information by making another post. On top of that, it can be used to pore over the sentiments and behaviors of users and their evolution over time.

With the quantity of information ceaselessly being generated online, there is a call for automated methods. It is pretty common to formulate a problem as a classification problem to make them more tractable. Consider a multilabel dataset with n samples:

$$\mathcal{D} = \{ (x_i, y_i) \mid i = 1 \dots n \} \quad (1.1)$$

where:

x_i represents a health-related post

y_i is the subset of corresponding predefined categories

Being T_i the vectorial space representation of the text, x_i , giving it as input, we want to predict y_i . By doing so, we are also creating a scalable and cost-effective process of automatically structuring the data. Furthermore, it can perform real-time analysis, and its discretization is consistent, contrary to humans who are affected by external factors. Individuals are prone to lack consistency and suffer from boredom and fatigue while manually labeling instances.

All in all, to the best of our knowledge, no previous work focuses on automatically characterizing the different types of messages circulating on health-related forums in such a holistic manner. This dissertation aims to automate that process.

1.2 Challenges

The study's novelty, the nature of the information used, and existing limitations in currently available solutions impose obstacles that need to be addressed.

First, having a different objective compared to other researches requires the expansion of a new dataset whose set of labels is novel. Hence, a new dataset needs to be constructed and expanded by labeling online posts manually from health communities.

Secondly, since the language used in social media uses colloquialisms, abbreviations and is non-error-free [13], the pre-processing phase where the text is adequately treated, regularized, and normalized is pivotal to achieving remarkable results. The amount of labeled data used to construct classifiers is conservative. So, making the use of techniques that can alleviate that downside is important. Textual data is represented in a high feature space, making it bound to suffer from the curse of dimensionality. Additionally, data is bound to be imbalanced, requiring extra carefulness.

There are many different classifiers and text representations, so finding the best combinations will require extensive empirical testing. It is essential to choose the most suited one for the problem at hand. Hartmann et. al (2019) compared different automated classification methods, but the tested datasets did not include health-related posts.

1.3 Contributions

In this dissertation, we try to tackle the challenges mentioned in the previous section.

First, we expanded the dataset whose classification scheme was created in a previous work [76], adding 285 new annotations from Medhelp ¹.

We thoroughly tested different approaches and techniques for the first level of the scheme mentioned above. Similar to the first-level categorization, we created several pipelines for the second-level classifier and highlighted the major performance bottleneck and possible ways to combat it. We explored how popular polarity analysis tools behaved with our dataset (OHC domain) and defined baseline models that use frequency-based feature extraction techniques and classical machine learning models. All the code is available at Github ².

Finally, a scientific article was submitted at The 13th International Workshop on Health Text Mining and Information Analysis.

¹<https://rdm.inesctec.pt/dataset/cs-2022-008>

²<https://github.com/JoaoAbelha/health-messages-categorization>

1.4 Document Outline

The current chapter introduced the context of the study, motivation, and objectives. The following chapter describes essential concepts that should act as base knowledge. In the next chapter, we review the existing literature to expose the key research gaps, techniques, methodologies, limitations, and developments. In Chapter 4, the problem statement and methodology, including the research questions are shown. Then, in Chapter 5, the construction and analysis of the dataset are performed and, subsequently, Chapter 6 presents the data preprocessing, and text vectorization approaches we took. Afterwards, a chapter on the first-level classifier, second-level classifier, and polarity analysis will showcase the experiments and their discussions. To conclude, in the final chapter, the conclusion and future work is presented.

Chapter 2

Background: Automatic Text Classification

The current chapter includes an overall review of relevant background knowledge. It includes background information, namely essential concepts, and ideas regarding text classification tasks and its typical NLP pipeline, including different classification models and evaluation metrics. Furthermore, it also contains a quick summary of popular NLP libraries and their main features that are relevant for the task at hand.

2.1 Data Collection and Annotation

Data collection is the process of gathering data from one or different sources. There are mainly three ways to collect data to build a dataset. It can be by using web Application Programming Interfaces (APIs), web crawling or leveraging existing datasets. A combination of them is also possible. Moreover, merging different data sources can lead to gains of performance [98]. Following that phase, the datasets are annotated. In case they are manually annotated, that activity can be time-consuming, may require some training and knowledge, is prone to subjectivity, boredom, and fatigue of the annotators [108]. To measure the level of agreement when more than one labeler annotates the same message, which is recommended due to the downsides mentioned above, the kappa score is often calculated [75, 13, 63, 51]. It measures to what extent raters (annotators) assign the same score to the same variable [81]. Its value ranges between -1 and 1. However, concerns about that measure usage in health research have drawn attention as it might not be the most appropriate. The same authors suggest using that variable together with the percentage agreement (the number of agreement scores divided by the total number of scores). Besides calculating that score, some studies go even further and consult an expert to validate the labels [13] or perform meetings to develop guidelines or solve incongruences [51]. Furthermore, for the annotation, it has been pretty common to resort to Amazon's Mechanical Turk as a way to crowdsource [90, 2].

This phase relevancy should not be left unnoticed since the models constructed are as good as the data collected, as they are trained on it. So, data collection practices should be done in a way so as the further steps are not negatively affected. Data needs to be error-free, and it should contain relevant information for the task at hand. For NLP tasks, data collected via APIs or web crawling commonly needs to be filtered to remove unimportant instances. For example, a filtering process of the tweets can be based on the presence of specific keywords, such as a disease [90]. Moreover, there should be an extra effort to make sure the data collected is representative enough to minimize the bias introduced in the model. Lastly, when dealing with sensitive and personal information, the secure storage and anonymization of information should not be overlooked.

2.2 Pre-processing

Preprocessing text means turning the text into a known format and easier to analyze for an NLP task. The preprocessing phase is a crucial step in Text Mining, NLP, and Information Retrieval [54]. It transforms text into a more standard form before processing the data. Moreover, the preprocessing phase is essential and impactful since it will influence all further phases of the NLP task. There is a myriad of techniques used with different functionalities and objectives. The most common techniques will be briefly described next.

2.2.0.1 Lowercasing

Lowercasing is one of the most common preprocessing steps. Without it, words like "phone" and "Phone" would be treated as two different words when they mean the same. As illustrated, it helps with inconsistencies and mitigates sparsity issues, especially when the dataset is relatively small. Nevertheless, this technique should not be applied to all problems, as it can lead, in some cases, to the loss of relevant information. For instance, the upper case can reveal a person's excitement, anger, or bitterness when performing sentiment analysis. In addition, a word in its uppercase form may correspond to an acronym while in lowercase to a name.

2.2.0.2 Tokenization

Tokenization is the process of breaking a corpus into smaller discrete elements - tokens - such as words, phrases, or characters. A vocabulary is the set of unique tokens or the k-most frequent ones. Hence, tokens can be seen as the building blocks of natural language. The technique is both used in traditional and modern NLP methods. Traditional NLP approaches, such as Count Vectorizer and TF-IDF, use vocabulary as features where each token represents a different feature. Additionally, pre-trained word embeddings such as Word2vec and GloVe are also subjected to tokenization. Even deep learning architectures for text processing, namely Recurrent Neural Network (RNN) and Long short-term memory (LSTM), use tokens as inputs where each token is processed sequentially.

There are different levels of tokenization - words, character, and subword (n-gram) tokenization. The most common way to perform tokenization is using words as token units. A disadvantage of it is to deal with out of vocabulary (OOV) words. They represent words encountered during the testing phase which did not appear in the vocabulary of the training set. A possible solution is to use the k-most frequent tokens and an unknown token (<UNK>) to act as a placeholder for the rest of them. This way, the representation of the least frequent tokens will be learned. Nevertheless, all those words are represented by the same token, which leads to a loss of possibly meaningful information. Another issue is the vocabulary size, which tends to be vast for large datasets.

A possible solution to this might be to use character tokenization. The size of the vocabulary (number of different characters) is small and can naturally handle OOV words. Unfortunately, the input length increases very quickly, becoming challenging to learn relationships among characters that form meaningful words. Then, lastly, we have subword tokenization that splits the text into subwords or n-gram characters. An example of a popular subword tokenization algorithm is Byte Pair Encoding (BPE). Not only does it tackle the OOV problem, but the length of the input is smaller than character tokenizations. The algorithm works by segmenting the most frequently occurring sequence of characters. Popular tools used for tokenization are NLTK, TextBlob, spaCy, Gensim, and Keras. Sadly, due to the nature of some languages, tokenization cannot be applied to them. An example of such language is Mandarin Chinese [54].

2.2.0.3 Stop Word Removal

Stop words refer to words that are frequently used in the human vocabulary. They account for 20 to 30% of words in text documents [54]. Even though they occur a lot, they carry little to no meaning. Removing those words gives more focus to the rest of the words. Additionally, it reduces the dataset size, with consequently the training time being reduced due to the smaller number of tokens involved during training. However, performing this technique is dependent on the task to perform. For example, when performing sentiment analysis, removing some stop words that completely change the polarity of a sentence, which is not intended. On the other hand, classification tasks generally do not need to perform stop-word removal as the focus should be on the words that better characterize and discriminate a specific class.

In terms of algorithms to perform this task, the paper [113] identified four: The classical method whose removal is dependent on a pre-compiled list of words; The Zipf's Law that besides having a stop-word list it also deletes frequent words and singleton words (i.e., that appear only once in the text); the mutual information algorithm that measures how much information does a term have in relation to a class, and deletes those whose values are low; and, finally, the based random sampling algorithm that randomly selects chunks of data and their terms are ranked according to the Kullback-Leibler divergence measure. Examples of tools used for stop word removal are NLTK, spaCy, and Gensim. These tools enable adding and deleting words from their predefined list of stop words.

2.2.0.4 Stemming and Lemmatization

Stemming and lemmatization are used to conjugate words with the same semantic meaning in a single term. Both have the same objective: converting morphological variants of a word into its root. For instance, loving, loved, and lovable match their root form of love. Performing these techniques leads to reduction of the vocabulary up to 50% [107, 54]. Moreover, in Information Retrieval Systems, they are used to enhance recall, and, in complex morphological language, they also enhance precision [107].

While stemming uses the word's root, lemmatization uses the context in which the word is inserted. Stemming cuts off the prefix or suffix of the word by using a list of common suffixes and prefixes. On the other hand, lemmatization morphologically analyses words. For that, using dictionaries to map the form/word to the corresponding lemma is essential.

A lemma is the base form of all inflectional forms, but a stem is not. Unlike a lemma, a stem may not represent an existing word. For example, "studies" is converted into "studi" using a stemmer. Unlike lemmatization, the technique is fast, but, in some cases, it may be as performant as lemmatization. In addition, lemmatization may not have a noticeable impact on the accuracy when using neural networks for text classification and for non-domain-specific datasets [23].

As examples to highlight, we have the Porter stemmer, the most famous one for English, and the WordNet dictionary, commonly used to perform the mapping mentioned above needed in lemmatization. Both Natural Language Toolkit (NLTK) and spaCy have the techniques implemented and are often used.

2.2.0.5 Normalization

Normalization refers to the process of transforming a word into its canonical form. For example, from "coool" to "cool". Another example is the mapping of almost identical words such as "social-media" and "social media". This technique is important for noisy texts such as social media posts since they present abbreviations, misspellings, inexistent words, and improper grammar. A study has revealed an improvement of 4% on the accuracy by performing this technique to the text of Tweets [101]. Unfortunately, there is no generic established solution regarding text normalization. The way to do it is highly dependent on the task. For example, normalizing clinical texts or email messages is presumably quite distinct.

Typical methods incorporate dictionary mappings, statistical machine translation, and spelling-correction-based approaches. The last method has been shown to be more flexible and to outperform the first one [101].

2.2.0.6 Noise Removal

Noise removal deals with removing characters and chunks of text, which can negatively interfere with the text analysis. Examples of noisy data would be "1.go", "go.", "go!", "<p>go</p>". By looking at the previous example, we can see that this step may also affect other steps, such as

stemming, so it should be performed before it. Nevertheless, we should be aware that it is a domain-dependent task that depends on what is considered noise.

2.2.0.7 Others

Apart from those above, other steps can be performed. Some include URL removal, rare words removal, frequent words removal, spelling check, and so forth. The preprocessing steps done must take into consideration the characteristics of the data being explored.

2.2.1 Feature Selection

Features selection can improve performance, reduce the feature space, and gain a deeper insight into the data processes [97]. It is also known as variable selection or attribute selection, being the process of selecting a subset of features considered relevant to the task at hand.

The intuition behind using this method is that data can contain irrelevant and redundant features. Consequently, removing them will result in minimal loss of information while gaining advantages. Interestingly, it has been noted that, normally, a subset of features are optimal. For example, using the two best performing features can be suboptimal [48]. On the other hand, some features are only significant when used together with others [43]. Redundant and irrelevant features are two possible different characteristics of data. A feature can be redundant and relevant. This happens when two relevant features are highly correlated, and hence one can be removed. On the other hand, it is also possible to have irrelevant and non-redundant features. However, either way, unrelated features ought to be removed as they seamlessly contribute to the model's performance. If features are independent from the classes they are deemed as irrelevant [68].

Feature selection is particularly relevant when a dataset is highly dimensional, and the number of samples is reduced. By doing so, we steer clear of the curse of dimensionality problem. Examples of fields where this technique is especially relevant are text analysis and DNA microarray analysis. Additionally, the number of features is positively correlated with the training time. The higher the number of features, the slower the training is. Moreover, correlated features in training make the loss ill-conditioned since the feature space becomes ellipsoid and gradient descent takes more time to converge. Other advantages of making feature selection are that models with fewer features are easier to interpret and construct. It can even enhance generalization, and models with fewer features are less prone to errors.

A feature selection algorithm has two essential components: a searching technique to propose a new subset of features and an evaluation measure to score a subset of features, so that different subsets can be compared among themselves [58]. The optimal and most straightforward algorithm is a brute-force one with an exhaustive search of the space, but that usually is computationally intractable, unless the number of features is significantly reduced. The evaluation metric allows feature selection algorithms to be categorized into three groups: filter, wrapper, and embedded techniques [55].

The filter method selects the features independently of the machine learning model, as this method is done before feeding the data to the model. They are based solely on the features, and several measures have been used to evaluate the subset of features. A rank is given to the features, and they are sorted. Its selection excludes those whose value is below a threshold or by selecting the k best ones. Examples of standard filter methods are Principal component analysis (PCA), mutual information, information gain, pointwise mutual information, relief-based methods, chi-square, correlation-based, fisher score, ANOVA, Fast Correlation based filter algorithm, and Linear discriminant analysis (LDA). The advantage of this method is the shorter computational time and robustness to overfitting. Sadly, the feature set is not specifically engineered for the ongoing task, which can be translated into lower prediction performance when compared to other methods [125].

On the other hand, wrapper techniques consider the models, and hence it is possible to assess the interaction between variables [95]. They use iterative search procedures and model performance to guide the selection of the following subset to evaluate. This method will converge into a set of predictors with at least as good of a performance as the original predictor. They can have a greedy or non-greedy approach. A greedy one is based on the selection of a subset of features that leads to the best improvement at that moment. This generally leads to local minima. On the other hand, non-greedy techniques evaluate previous feature sets that were not initially favorable, but appeared beneficial at the current iteration. This last method can help to escape local minima. This technique provides good performance when trained on a particular problem. However, they are more likely to overfit when data samples are small, and if the number of variables is large, the computational time can be heavy. Popular greedy wrapper methods are forward selection and backward selection. With forward selection, we begin with no feature and iteratively add the feature that best increases the performance until no further enhancement can be made. Backward elimination starts with all the features and removes the least improved features until no removal is deemed advantageous. When there is no improvement in removing some features, the method converges. Examples of non-greedy wrapper methods are hill climbing and genetic algorithms.

Embedded methods are fast since the selection process is within the model training, so no external feature selection tool is needed. There is a connection between the feature selection and the objective function: the objective function is the statistic that the model is optimizing (classification and feature selection are performed simultaneously). So, contrary to the previous models, a separated search procedure is not needed to identify the feature to be used. They try to merge the advantages of the techniques mentioned above. This method controls the value of the parameters by giving less important parameters weights close to zero. This is a way to perform regularization. Examples of these techniques are lasso (L1), ridge (L2), and elastic regularization (combination of L1 and L2). There is also tree- and rule-based methods such as LightGBM and XGBoost, and Multivariate Adaptive Regression Spline (MARS).

Considering the input features of a classifier, training can be an intensive activity. It can even be the most time-consuming task while building a successful machine learning pipeline [43].

2.2.2 Feature Extraction

Feature extraction refers to the process of converting raw data into numerical values (structured format) so that they can be used as input of machine learning models. Text vectorization is the technique's name used to transform the text into a vector of numbers.

The simplest way to encode the textual data is using one-hot encoding. There are two ways to do it: encode a whole document or encode each document token and construct a matrix of tokens. The former, the input vector, is the same size of the vocabulary with n bits set to 1, being n the number of unique tokens in the dictionary. The latter is a matrix with n vectors of the vocabulary size, but the order is considered. Some machine learning algorithms can leverage the order of items in a sequence. For improved classification results, recurrent neural networks and LSTM layers, for example, can make use of the sequence order. To analyze the space needed, take the following phrase "The sun is shining" and consider a vocabulary of 50,000 words: the first method uses 50,000 bits whereas the second one $4 * 50,000$ bits. Besides taking all that memory, useful data is sparse as most of the matrix is taken up by zeros since there are only 4 bits set to one. In addition, relations between words are discarded. It only captures which words are present in a document. This leads to the system reasoning that the difference between the words "car" and "job" to be the same as "car" and "cars".

There is a method that further improves the ones above by incorporating the frequency each word has in a document. It uses a vector of frequencies rather than a vector of Boolean values. This model is called Bag of Words (BOW). However, it considers the order of words in textual data to be arbitrary even though some words tend to appear in similar contexts, which makes this technique still flawed. Take the following as an example: "I have no money" and "No, I have money". They mean completely different things, but both would be encoded the same way in this method. Nevertheless, BOW can bring acceptable results if the problems are simple. It can also work as a good baseline estimator. The feature space of a document is the length of the vocabulary.

A problem with scoring word count is that more frequent words get higher importance (score) in a document even if they are stop words. A possible solution is for each word to use the metric Term Frequency – Inverse Document Frequency (TF-IDF instead of its frequency). The weight associated with each word depends on word frequency in a document and how frequent that particular word appears in the whole corpus. The weight associated with a word is the highest when the word is rare in the entire corpus, but frequent in the respective document. Unfortunately, the order is still not considered and is very corpus-dependent as different corpus generate different word weights. This technique does not consider words' meaning and context as well. On the other hand, the technique is particularly successful in document classification and language modeling tasks.

Bag of n -grams is another extension of the Bag of words. An n -gram represents a sequence of n tokens that occurs in the corpus. Consequently, contrary to the BOW and its variants presented above, it captures some information about the context around words, since the vocabulary corresponds to groups of n words. For instance, creating a vocabulary of pairs of two words is called a

bigram, where all two-word tokens are counted. In the phrase "I am perplexed", the vocabulary of a bigram model whose tokens are words would be "I am", "am perplexed". However, the feature set created can be more sparse and larger than BOW: data dimension exponentially increases with n 's increase. This model can also find some unseen samples in the testing set. Apart from the n -gram model, these algorithms assume that the word occurrence is independent of each other. All of them do not capture the semantics of the words and suffer from high sparsity.

Neural networks have been recurrently used to model words and their interaction as input of an NLP task. They use different neurons to represent a word, and each neuron can be a part of different words. Consequently, they learn shared concepts of the language instead of modeling each word independently. The embedding size is flexible, representing a hyperparameter, and it is generally way smaller than the vocabulary size. Additionally, it is an unsupervised method, so there is no need for previous manual annotation. Even though word embedding is often grouped with deep learning approaches, creating them does not require deep learning. The architecture of the network has a single hidden layer. The weights of the hidden layer are the embeddings of the word the task is trying to find, and the weights updates are done via backpropagation. This method captures the semantics of words as it can be used to find similarities between words using mathematical similarity measures, such as cosine similarity. It can also be used to perform simple arithmetic to obtain logical answers. For example, we can do King - Man + Woman to obtain Queen. Word embeddings models have seen an utter evolution over the last years.

A popular method to train word embeddings published by Google is word2vec [83] in 2013. There are two architectures, namely the Continuous Bag of Words (CBOW) and the (Continuous) Skip-gram (SG). CBOW predicts the middle word in a sentence, while SG predicts the context words from one target word. The former is more efficient in smaller datasets and trains faster than skip-gram. The latter performs better in large datasets, but training requires more resources. Facebook furthered this concept, and fastText was released [62]. This last model dealt with out-of-vocabulary words, representing words as bags of character n -grams. Additionally, the Stanford NLP group published another model named GloVe [92]. GloVe proposes to combine the benefits of local context window techniques with global matrix factorization models. In 2018, ELMO was released by the Allen Institute [94]. It uses an architecture based on bidirectional LSTM layers. BERT is a further improvement of ELMO, also published in 2018 [41]. It is based on word2vec because it includes the context of a sentence in evaluating the vector. Further models have been launched, such as TransformerXL [36], and ALBERT [70].

There are two ways to train these last models: using data solely related to the task at hand or pre-trained models on massive corpora to capture common relationships among words in a specific language as models are language-dependent. The latter is often used to accelerate the training process and achieve better results. Currently, most state-of-the-art techniques use word embeddings.

2.2.3 Feature Engineering

Designing new features that target a specific task can significantly improve text classification. It is common to extract extra features from the original data before turning the text into a numerical array.

Regarding feature engineering, we can start with a simple analysis of the length of the data. For example, maybe a category is consistently longer than others, and that feature would be simply the only needed. Even though this happening in real-world data is uncommon, using length measures such as word count, sentence count, average word length, and average sentence length may lead to improvements.

Sentiment analysis (opinion mining) is a technique used to discover the polarity of text. Doing so is an arduous task due to ambiguity in data. When there is insufficient data, pre-trained models can be used. One of the most popular is Textblob, which can assign valence to words and assess the emotion of the entire text. On the other hand, Vader (Valence aware dictionary and sentiment reasoner) is a rule-based model that excels at social media data. We can check if there are patterns between the sentiments and categories.

Name-Entity Recognition (NER) is the process of labeling entities that appear in the text with a pre-defined category. This task is time-consuming and requires a vast dataset. We can associate a counter of tags to each sample and see if more tags are more prominent in some classes than others. NER can be helpful to apply before removing stop words. For example, the removal of the word “will” in the context “Will Smith is a famous actor”. We could recognize the personality and change it to another placeholder by performing name detection.

Another feature we can calculate is the n-gram frequency. If n-grams appear more in a class than others, that can be a helpful feature. When we vectorize the entire corpus and use all the n-grams as features, we perform the bag of words approach. Topic modeling using LDA is a statistical model that can discover the topics in a document. Each document is represented as a set of latent topics, which can be used as discriminant features.

In a nutshell, there are different types of features we can try to use to improve the task further; however, the ultimate test to check which combination of features is the best encompasses empirical testing.

2.3 Text Classification

This section shows different approaches we can use for text classification. The lexicon-based approaches we will exhibit consist of lexicons used for polarity analysis. On the other hand, for machine learning models, we display standard models used with textual data.

2.3.1 Lexicon-based Approaches

Each lexicon approach is often associated with different dictionaries of words to characterize their polarity. Usually, these dictionaries have words linked with values that correspond to positive and

negative valence. Notably, these approaches have lowercase words and do not use punctuation to extract the sentence polarity. This is quite unfortunate as those can be good indicators and emphasize what is being said.

An important finding to bear in mind is that those dictionaries are not ready to deal with negation in phrases. Not only this but the sentence's valence is limited by the number of words available in the dictionaries.

An interesting observation regarding lexicon classifiers is that if in an instance where there are only OOV tokens, they will be classified as neutral since they were not found in the vocabulary. In these approaches, it is also not needed to remove stop words as they only account for relevant words, i.e., that are present in the wordlist.

2.3.1.1 ANEW

The ANEW (Affective Norms for English Words) is a dictionary with 1,034 words that were manually labeled on three different measures: pleasure (valence), arousal (excitement), and dominance (level of control) [18]. Later, in 2013, the dictionary was expanded to 13,915 lemmas [115]. The scores for each metric range between 1 and 9. In terms of sentiment analysis, the valence metric is the one we are interested in.

2.3.1.2 VADER

VADER is an open-source rule-based model used to perform polarity analysis that generally performs exceptionally well in social media posts but can also be generalized for other domains. Following its construction process [60], it can be summarized as follows: first, a gold-standard list of features was built using features from several widely used lexicons, such as ANEW and LIWC (Linguistic Inquiry and Word Count), which currently is a paid software. Moreover, they even added lexical features used to express sentiments, such as emoticons, acronyms, and slang. They used an approach called wisdom-of-the-crowd to acquire point estimations of sentiment valence and intensity ratings from Amazon Mechanical Turk workers. Additionally, they even established five heuristics for sentiment analysis of text for punctuation, capitalization, intensifiers, the use of the contrastive conjunction "but," and negation. These heuristics are meant to overcome the obstacles already mentioned above. VADER is implemented in the NLTK's library, with its scores normalized on a scale between -1 and 1.

2.3.1.3 TextBlob

TextBlob is a python library that can be used for several NLP tasks. One of them is the analysis of the polarity of a sentence, outputting a value between -1 and 1, where negative values correspond to negative polarity and positive values to a positive one. Its implementation is based on the Pattern library, which outputs the polarity and subjectivity of a sentence. It uses a lexicon for English adjectives, the wordnet3 lexical database for English, and, to get the pos tagging, it uses the Penn Treebank. When calculating the sentiment for a specific word, it uses averaging according to the

occurrence of a certain word in the lexicon. It also handles negation by multiplying the polarity by -0.5, and it ignores one-letter words.

2.3.1.4 AFINN

AFINN, as described in paper [87], is a new ANEW, specifically constructed with microblogs in mind. The paper shows that it can perform better than ANEW but not better than more elaborated approaches. Twitter messages were used for both construction and validation of the set. Contrary to ANEW, only valence was considered. They compared several word lists, namely ANEW, General Inquirer, OpinionFinder, and SentiStrength, for 1000 tweets labeled resorting to the amazon mechanical Turk. Each tweet was labeled ten times with a rate from 1 (negative) to 9 (positive). In the end, the average is taken to get the valence. However, they show that SentiStrength has the overall best performance among them with a correlation to the human labels of 0.610. In terms of the General Inquirer and OpinionFinder, even though they have the largest word lists, they could not perform better than the SentiStrength lexicon, the proposed new list, and ANEW for sentiment strength detection in a microblog posting. Yang et al. even modified the lexical tool AFINN by removing some medical conditions that should not have polarities in this context [118]. Examples of those are "disorder" and "tumor".

Even though SentiStrength seemed to be the best approach, another paper [60] revealed that VADER (Valence Aware Dictionary for sEntiment Reasoning) showed a higher Pearson product Moment Correlation of 0.881 compared to human raters who scored 0.888 at matching group truth. This revealed that VADER could accomplish results practically as good as humans.

2.3.2 SentiWordNet

SentiWordNet [46] is a lexical resource used for sentiment mining. It assigns three different scores for each synset of WordNet (synset instances are groups of synonyms words that express the same concept), describing the objectiveness, positiveness, and negation. The method to develop Sentiwordnet was based on a quantitative analysis of the glosses associated with the synsets and semi-supervised synset classification. Some words may have one synset, while others may have several. The scores result from eight ternary classifiers with similar accuracy but behave differently. The lexical resource is present in the NLTK library.

2.3.3 Machine Learning Models

Text classification, also called text categorization or text tagging, refers to mapping text to a pre-defined set of labels. For this current subsection, X represents the set of features $\{x_1, x_2, \dots, x_n\}$ and Y is the probability of being y , the target class. The incentive to have systems that execute automatic text classification is to tone down the disadvantages of executing it manually. It is a time-consuming process, prone to subjectivity, may require qualified and trained individuals, and fatigue may bottle up annotators' discernment.

A considerable quantity of labeled data is generally required to build machine learning models able to perform, and it has been revealed that simpler machine learning models can beat more promising ones that use fewer data [43]. This is to show how relevant data is.

Supervised learning uses solely labeled data to build models. It would be pertinent to leverage data without labels, especially if there is a good amount of unlabelled data available, which happens to be the case of online forums [75, 99, 98]. Several studies have tackled the lack of annotated data by using adequate machine learning methods and by using semi-supervised machine learning methods as they leverage non-annotated data.

2.3.3.1 Logistic Regression

Logistic regression is a binary classification model, mapping a data sample to two possible classes. First of all, the model makes some assumptions. It assumes that samples are independent of each other. It requires little to no multicollinearity among the independent variables (they should not be highly correlated). It assumes linearity between independent variables and logs odds; in other words, independent variables are linearly related to the log odds.

$$g(x) = \frac{1}{(1 + e^{-x})} \quad (2.1)$$

$$y = g(B_0 + B_1 * X_1 + \dots + B_n * X_n) \quad (2.2)$$

The model is commonly used as a baseline to compare with more complex classification approaches. In terms of text classification, words are represented as feature vectors, and weights are given to them according to the importance each one has when discriminating between classes. Finally, the sigmoid activation function (2.1) is used to output a probability of score (2.2). The loss function used is called Logistic Loss.

A general guideline for the dataset size is to have at least ten cases with the least frequent outcome for each independent variable. For example, if there are four independent variables and the expected probability of the least frequent outcome is 0.2, the sample size should be 200 ($10 * 4 / 0.2$).

2.3.3.2 K-nearest Neighbors (KNN)

K-nearest neighbors is a classification method based on the fact that similar data samples live near each other. It works by finding the k objects in the training set closest to the test sample. So, the chosen label is based on the predominance of a particular class in the neighborhood of that point.

Measuring the distance is done by choosing a distance metric such as Euclidean or Manhattan. Depending on the classification task, a distance measure may also be more appropriate than others. For example, when classifying documents, it is probably better to use the cosine distance rather than the Euclidian distance as the distance of this last one becomes more indiscriminate as the number of features increases. Moreover, since computing the distance is scale-dependent to

prevent some features from dominating, there should be a rescaling so that all features fit under the same range of values.

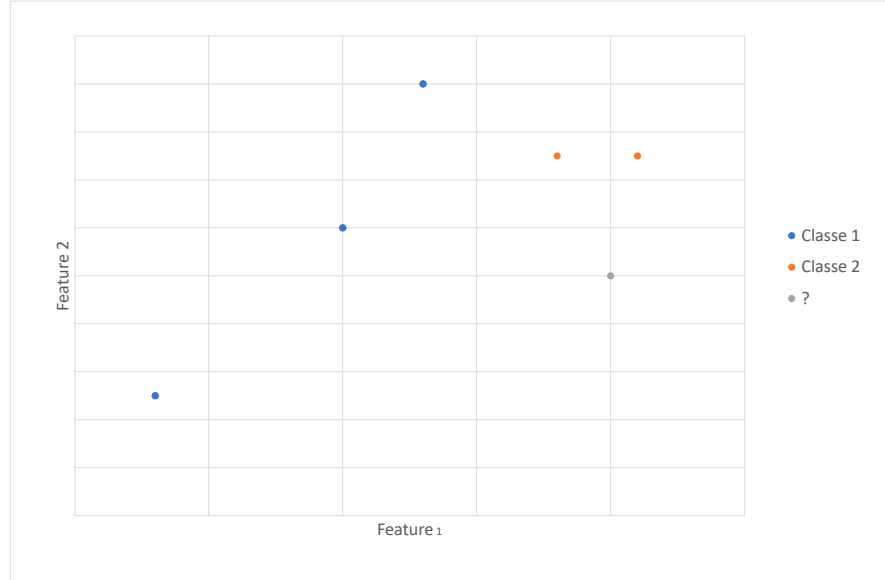


Figure 2.1: Data points with 2 features in a 2D space

A particular important hyperparameter is the choice of K . If it is too small, it can be sensitive to noisy points; if it is too large, it may include far too many points from other classes. In the Figure 2.1, the test sample point (grey) for $k = 3$ would belong to class 2, but for $k = 5$ would have class 1. An improvement that can alleviate the sensitivity of K is giving weights to the voting process by giving a higher score to the closest points.

Since there is no training phase, building the model is cheap; however, classifying is expensive as we need to calculate the k nearest neighbors of a point. Fortunately, methods have been developed that leverage the data structure, preventing the exploration of all the training set's samples.

2.3.3.3 Naive Bayes (NB)

Naive Bayes is a classification algorithm based on Bayes' theorem. The formula is based on the calculation of probabilities when we know others. With the corpora, the probabilities $P(X)$, $P(X|y)$, and $P(y)$ can be calculated. Given the features X , the probability we want to calculate is the posterior probability of $P(y|X)$, the probability of being class y . First, we create a table with the token frequencies for each feature against the target class. With that, we can calculate $P(x|y)$, $P(X)$, and $P(y)$. After that, the Bayesian formula is applied:

$$P(y|X) \propto \prod_{i=1}^{|X|} P(x_i|y) \quad (2.3)$$

There are cases where the classification can be multivariate. Thus, finding which class has the highest posterior probability is the answer:

$$y = \underset{y}{\operatorname{argmax}} P(y) \prod_{i=1}^{|X|} P(x_i|y) \quad (2.4)$$

There are two assumptions about the features X . They are independent and contribute equally to the final result. The name naive comes from the fact that real-world data is never independent. Since it assumes the independence of features, it only needs some data samples in the same order of magnitude as the number of features. However, it can only make simpler mappings compared to algorithms that consider feature interaction (but need more data).

A disadvantage of the method is that the probability is zero when no word occurrences exist in a particular class label. This means that frequency-based probability is null. Consequently, as seen in Equation (2.3), the zero cancels all other probabilities. An approach to overcome the downside is to add a unit for every value-class combination.

There are three types of NB classifiers: Multinomial, Bernoulli, and Gaussian Naive Bayes, that make extra assumptions about the features.

2.3.3.4 Support Vector Machine

Support vector machines (SVM) offer a robust and accurate method to perform classification amongst all the classical machine learning algorithms. It does not require a large dataset to learn and is insensitive to the number of dimensions, rendering a must-try solution for problems with a high number of features.

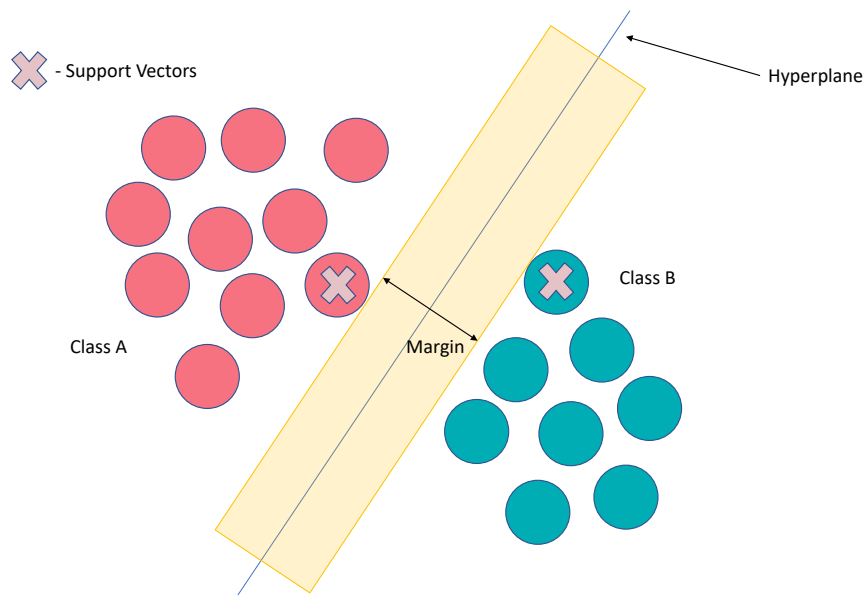


Figure 2.2: Support Vector Machine concepts

Considering a binary classification task and a linearly separable dataset, the algorithm solution is geometrically equivalent to finding the best hyperplane that separates both. The best hyperplane is the one that maximizes the margin between both classes, being the margin the shortest distance between a data sample to a point in the hyperplane. The reason for maximizing the margin is to provide a better generalization capability.

The problem can also be extended for nonlinear-separable classes. The solution encompasses the application of kernel functions, also referred to as the kernel trick. It involves transforming the non-linear input data into linear separable ones. It is applied to each data sample to map the observations into a higher-dimension feature space in which they become separable. Moreover, we can also extend the formulation to a multiclassification problem. It entails creating n classifiers when there are n different labels. Each one is responsible for discriminating between belonging or not to a class.

The original definition of SVM is sensitive to noisy data. Therefore, the soft margin concept was introduced so that the hyperplane allows a few noisy data points to existing, i.e., on the wrong side of the hyperplane. Another initial drawback of SVM was the performance when dealing with large datasets. However, advancements have been made, and some solutions can solve the problem rapidly.

2.3.3.5 Decision Trees

Decision Tree algorithms are supervised machine learning algorithms that can be used for regression and classification problems. Continuous variables are discretized prior to applying the algorithm. Building an optimal decision tree is an NP-complete problem [71]. Hence, heuristics usually are employed, usually using a greedy approach.

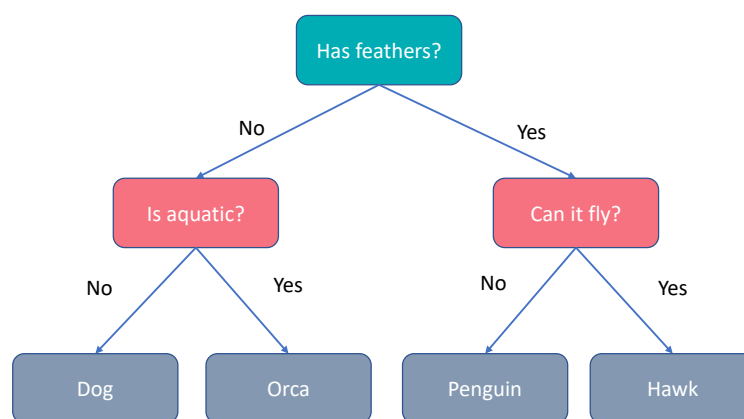


Figure 2.3: Simple decision Tree for animal classification

A decision tree is a tree in which the internal nodes are associated with an input feature. The arcs that come from that node are labeled with each of the possible target values of that feature.

They lead to a nether decision node on a different feature. Finally, each leaf represents a class or a probability distribution over the classes.

The splits are based on rules that depend on the features. This process is repeated, starting with the whole dataset on the root node and recursively creating subset partitions on each split. The stop criteria are when the subset elements at a node have the same value or when splitting no longer adds significance to the predictions. Different algorithms use different metrics to create the best splits. They often measure the subsets' homogeneity to quantify the split's quality (the ones that create more homogeneous subsets are better). Those metrics include Gini impurity, information gain, and variance reduction.

The main advantages of this method are its human interpretability, good performance in big datasets, and robustness against co-linearity. In addition, it has a built-in feature selection since only the splits that better discretize the classes happen across the different features.

On the other hand, if no limit is placed on a decision tree, it may create an overcomplex model that overfits the training data. There are two standard solutions to prevent this: either perform pruning or apply the random forest ML method. The pruning consists of cutting off the tree's branches if they do not affect the overall accuracy. Random Forest is an example of ensemble learning, in which multiple decision tree models are built at training time to obtain better predictive performance.

2.3.3.6 Neural Networks

Feed-forward neural networks are the simplest artificial network architecture. The number of hidden layers and their number of nodes correspond to hyper-parameters, and they are often correlated with the complexity of the task. For a classification problem, the output layers should have a number of nodes equal to the number of possible classes, except for binary problems where only one output node is needed.

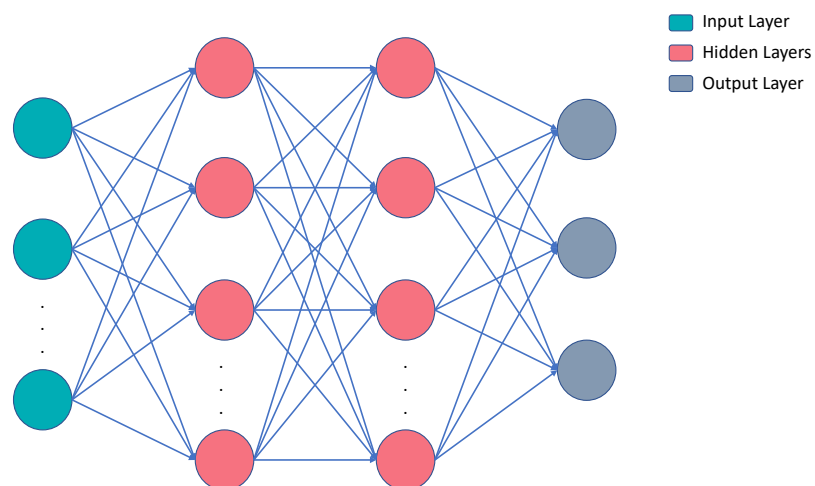


Figure 2.4: Fully connected network with 3 output classes

Generally, the activation function used for the hidden layer nodes is non-linear since the combination of linear functions results in a linear function. As most problems are non-linearly separable, it is more convenient to use ReLU (Rectified Linear Unit), tanh (Hyperbolic tangent), or ELU (Exponential linear unit) activation functions. The activation function used for the output layer depends on the task. We use sigmoid for binary classification, while for multiclass classification, softmax. Sigmoid squashes the output value into a range within zero and one corresponding to the probability. Softmax allows normalizing the network's output to a probability distribution over predicted classes. Finally, the class chosen is the one with the highest probability.

Regarding multilabel classification, the number of output nodes equals the number of possible labels. For each output node, the sigmoid activation function is used, and hence a likelihood of a sample belonging to all the different labels is computed. A threshold value is used to distinguish whose labels should be considered from those which should not. All use the cross-entropy cost function to calculate the error. The weights are updated using backpropagation.

Feed-Forward networks have some disadvantages when dealing with textual data: they cannot handle sequential data, only consider the current input, and do not memorize previous inputs.

Recurrent neural networks overcome the problems mentioned above. They are interesting because they save the output of a layer and feed it back to the next one. Consequently, a layer receives the input vector of the current timestep, $x(t)$ and the state of the previous layer $h(t-1)$. They can memorize previous input since they have internal memory.

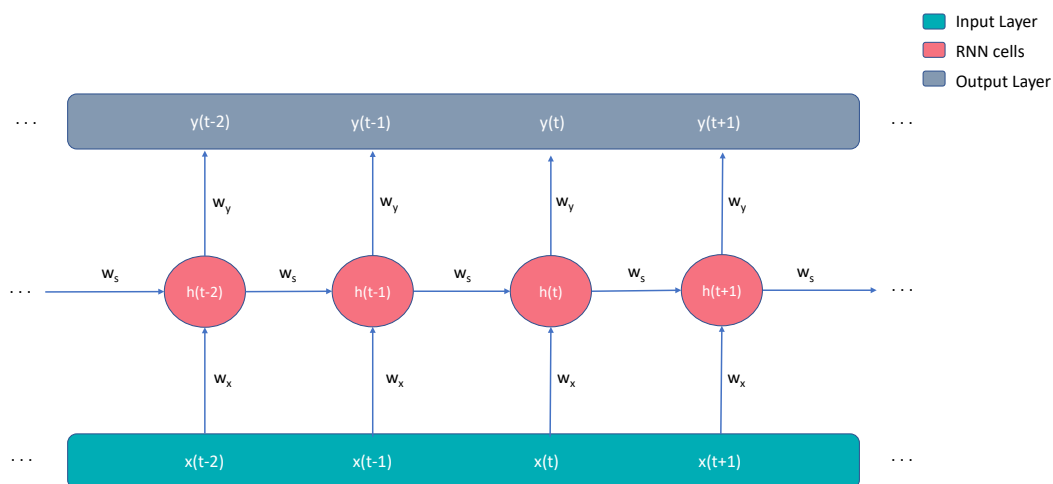


Figure 2.5: Unrolled recurrent neural network

There are four types of recurrent neural networks: one to one, one to many, many to one, and many to many, based on the input size and output size. For example, sentiment analysis is an example of many to one and language translation of many to many. Both receive as input many tokens that are fed at different timestamps. For sentiment analysis, the output is only one label corresponding, for example, to the polarity. For the translation, the output may be several words that correspond to the translation of the input into another language.

Recurrent Neural Networks are capable of modeling time-dependent and sequential data. However, they suffer from the vanishing gradient problem. Due to the weight update rules, more multiplications will be progressively used to calculate the gradients during the backpropagation phase. If the gradient values are small, that can lead to tiny values in the first layers. Thus, the parameter updates become insignificant, rendering learning long data sequences challenging. Exploding gradients can occur if huge values are constantly multiplied instead of small values. This leads to huge updates to the neural network model during training. The consequence of the problem is long training time and poor performance. A solution to that problem is the use of LSTMs. They are capable of learning long-term dependencies by remembering information for long periods. They have long and short memory states and can filter which information they should maintain and discard. Updates are done via backpropagation over time.

Even though convolution neural networks (CNN) are a class of neural networks most applied in visual tasks, they also have found prevalence in addressing NLP tasks. Like images, text can also be represented by an array of vectors, with each word corresponding to a vector. That array can be processed using a CNN. Since the text is sequential data, 1D-convolutions are applied. It also works by finding increasingly complex patterns as data goes through the layers. Before passing the data into a fully connected layer for text classification, it is common to have a global pooling layer that downsamples its input. Yoav Goldberg [53] argues that neural networks generally perform better than classical machine learning methods, mainly when using pre-trained word embeddings. Yoon Kim [67] adds that using pre-trained word embeddings may offer good universal features to use in NLP. Ye Zhang and Byron Wallace [124] studied the sensitivity of each hyper-parameter on sentence classification using CNN. Their objective was to provide an excellent general setting to perform that task. They established a good starting point based: (1) the choice of pre-trained word2vec and GloVe depends on the problem at hand, but they outperform one-hot encoded word vectors, (2) the number of kernels and their size are relevant and should be tuned. (3) the 1-max pooling layers generally outperform other poolings types, and (4) the dropout has little impact on performance. However, their finding is based on empirical results from binary classification.

The use of deeper CNN networks for text classification has been investigated, too [123]. They achieve state-of-the-art classification by using a new architecture called VDCNN that operates on the character level, uses small kernels and pooling layers, shortcut connections, and uses up to 29 layers. An important finding is that the architecture works well on small and large datasets.

2.3.4 Evaluation

Before splitting the dataset into three partitions used for training, testing, and validation, it should be shuffled so that each set is deemed a good representation of the target distribution. The testing dataset is used to evaluate the model's prediction capabilities. After training, the testing samples are fed to the model, and its predictions are compared to the corresponding labels. There are four different possible outcomes:

		Predicted label	
		Positive	Negative
Actual label	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Figure 2.6: Confusion matrix for a binary classifier

The confusion matrix visible in Figure 2.6 can be expanded to consider multiple target labels. Each label should appear in both a row and a column, and the number of samples that match correctly correspond to the true positives. Should the order of the labels be the same in the columns and rows, then the sum of the main diagonal would correspond to the true positive values. With these values, it is possible to calculate several metrics:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total} \quad (2.5)$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2.6)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ negatives} \quad (2.7)$$

$$F_{score} = (1 + \beta^2) * \frac{Precision * Recall}{(\beta^2 * Precision) + Recall} \quad (2.8)$$

The Equation 2.5 represents the percentage of correct predictions the model made. The Equation 2.6 is the percentage of relevant examples (true positives) among all predicted to belong to a specific class. The Equation 2.7 is the percentage of relevant examples (true positives) among all the examples that belong to a certain class.

Precision and recall are appropriate when classes are imbalanced. For example, if only a tiny percentage of samples corresponds to a class, say 1%, the classifier could guess the other class all the time and have an accuracy of 99%. Nevertheless, by using recall in this classifier, it would be clear that something is wrong, assuring that the small percentage of a class would not

be overlooked. Precision would guarantee that there are not too many misclassifications of true positives. Usually, a good balance between both is often required. Nevertheless, if we wish to represent the quality of an ML model employing a single metric, it would make sense to combine both precision and recall. That is what the Equation 2.8 does. The β parameter allows controlling the tradeoff between them, giving a bigger weight to one to the detriment of the other. When $\beta < 1$, it focuses more on precision; otherwise, it focuses more on recall. For example, if we are dealing with a suicide detection model, it would seem sensible to put a higher weight on the recall as it would be better to have an intervention with individuals that do not consider suicide than to miss someone that is considering suicide. Nevertheless, we still do not want the model to have too many false positives.

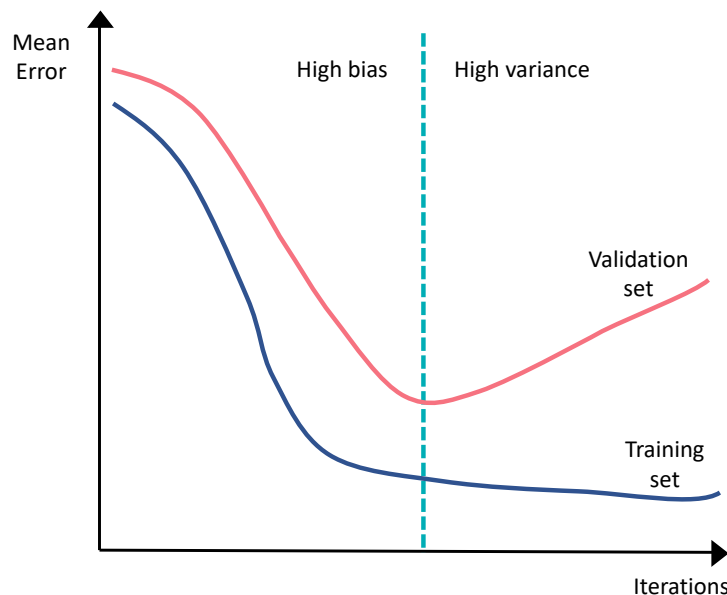


Figure 2.7: Error of training and validation set over time

On top of this, it is possible to evaluate the model during training, as seen in Figure 2.7. For that, a validation set is required. The benefit of doing so is that it is possible to locate the turning point between underfitting and overfitting. We can plot the training and validation error of the model to diagnose if the model is suffering from high bias or variance. When both errors decrease, the model is still underfitting, but when the validation error augments while the training error diminishes, the model starts overfitting. Normally, there needs to be a bias-variance tradeoff. A model with high bias pays little regard to the data and usually suffers from underfitting as the model is too simple to model the data. On the other hand, when the model fits the training data too much, it will also capture the noise of the data, causing the model to suffer from high variance. It does not generalize well as it overfits the training data. Ideally, the model is able to balance between the two extremes.

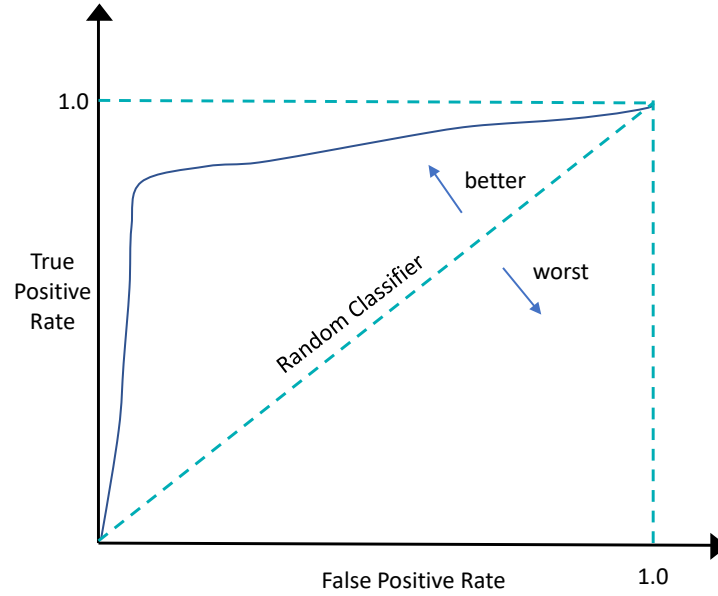


Figure 2.8: AUC

The ROC (receiver operating characteristic) curve depicts the performance of a classifier at different thresholds. The axis corresponds to the true-positives and false-positive rates at different threshold values. Lowering the threshold leads to more positives, increasing both false and true positives. To compute the points in a ROC curve, we could evaluate a model many times using different thresholds. Fortunately, we can analyze the AUC (area under the ROC curve), exemplified in Figure 2.8, which depicts the performance of a classification model at all classification thresholds. A model whose prediction is always wrong has an AUC of zero and a value of one if 100% correct.

Moreover, for multilabel classification, it is quite common to use the weighted Jaccard Index and the Hamming Loss.

The Jaccard Index can be computed by the Formula 2.9.

$$\text{Jaccard Score}(\text{True}, \text{Predicted}) = \frac{\text{True} \cap \text{Predicted}}{\text{True} \cup \text{Predicted}} \quad (2.9)$$

The Hamming Loss is defined in the equation 2.10 with L being the number of labels and N the number of samples.

$$HL = \frac{1}{NL} \sum_{i=1}^N \sum_{j=1}^L \text{xor}(y_{i,j}, z_{i,j}) \quad (2.10)$$

Table 2.1: Open source NLP libraries and their features

Tool	polyglot	spaCy	NLTK	GenSim	CoreNLP	TextBlob	AllenNLP	OpenNLP	Pattern
Tokenization	x	x	x	x	x	x	x	x	x
Wordstop removal		x	x	x	x	x	x	x	
Lemmatization / Stemming		x	x	x	x		x	x	x
NER	x	x	x		x	x	x	x	
POS tagging	x	x	x		x	x	x	x	x
Parsing		x	x	x	x	x	x	x	x
Text classification		x	x		x	x	x	x	x
Evaluation		x	x		x	x	x	x	x
Sentiment Analysis	x		x		x	x	x	x	x
Topic modeling				x					
N-grams		x	x	x	x	x		x	x
Language detection	x	x				x		x	
Spelling check		x			x	x			x

Table 2.2: Open source NLP libraries and their features

2.4 NLP Libraries

In this section, we will look at the most used NLP libraries, emphasizing the ones that can be used with Python. First, a quick summary of their features is presented in a tabular form. Then a comparison between them is made according to what they are more often used for, highlighting their advantages and downsides.

The NLTK library is the most well-known and can build an entire NLP pipeline. It supports the highest number of languages of all existing libraries. Similarly, Polyglot also supports many languages (between 16 and 196 depending on the task). However, this last one is not very well-known, leading to a lack of support from the community. NLTK presents several approaches for each NLP task which can be a double-edged sword. On the one hand, we have a very flexible library; on the other, it can be rendered more complicated to learn. Unfortunately, NLTK is known to be, in general, pretty slow. On the contrary, spaCy is the fastest NLP library. Compared to NLTK, only sentence tokenization is slower. Another advantage is the fact that it has built-in word vectors. Gensim was initially built for topic modeling but now supports various other NLP tasks. It has been frequently used to work with word embeddings, providing access to fastText, word2vec, and doc2vec for training. It even allows loading pre-trained word embedding models downloaded from the internet. On the other hand, spaCy already comes with pre-trained word embeddings models. Nevertheless, there are many of them available online, so they should be

carefully selected considering the type of task to perform. Another disadvantage of Gensim is that by itself, it is not possible to build a whole NLP pipeline.

TextBlob is based on NLTK and another library called Pattern. Sadly, it inherits the low performance of the former. So for large-scale projects is not the best solution. However, it is a convenient library and provides language translation and detection powered by Google. Pattern is a multipurpose library since it includes machine learning, data mining, visualization, and network analysis capabilities apart from handling NLP tasks. It also performs Fact and Opinion detection, making it stand out from the others.

Scikit-learn is not specifically for NLP but is still very useful. First, it has various classical machine learning models, such as SVM, Naive Bayes, and Random Forest. Moreover, it has an array of functionalities to perform classification, ranging from the metrics used to different types of cross-validation.

There are also frameworks focused on deep learning. The most well-known are Tensorflow and PyTorch. Tensorflow is easy to use, while Pytorch allows adding more details at the cost of coding at a lower level. Pytorch debugger is also often preferred (no eager execution).

Chapter 3

Automatic Classification of Messages in Online Health Communities

This chapter describes work based on previous research concerning automatic text classification of health-related messages in online health communities. Initially, a brief section describes common problems in the medical field in which text classification is used. Following that, the most common themes of classification tasks are given, emphasizing the methodology undertaken, the data sources, and corresponding datasets built. Afterward, a subsection describes the classes of the most relevant datasets. Finally, a subsection with ethical considerations regarding Natural Processing Language (NLP) tasks is presented.

3.1 Common Tasks

The most common tasks found in the literature are topic classification, adverse drug reaction, suicide detection, depression detection, user intent, surveillance, and sentiment analysis. Each task will be discussed next, and the methodologies and datasets used will also be analyzed. The best performance will be presented, namely the best F1-score. Nevertheless, another metric that they present, such as accuracy or AUC, shall be indicated if it is not stated.

3.1.1 Intent Classification

Intent classification is the task of labeling textual data based on the writer's aim behind the message. Users' intent is highly dependent on the domain [110] and, consequently, defining a set of labels requires research and analysis.

Historically, user intent focused on identifying intention categories based on pre-defined keywords or personalized data [78]. In general, there have been different approaches to performing intent analysis using clustering techniques [31], graph-based methods with an information propagation technique [114] and using semantic parsing [73]. In addition, this problem is frequently devised as a classification problem.

In terms of data sources, the application of intent analysis has been seen in different types of social media, such as underground forums, i.e., forums related to cybercrime [22], for company feedback [93], tourism, real estate, and transportation [110], health forums [21], cosmetic, and beauty forums [77]. Besides, there is research whose target is the intent analysis of short texts in social media [96] rather than a specific type of forum.

For automated intent analysis, classical machine learning has often been used. Still, in recent years there is evidence of work conducted using deep learning models [56, 116, 21, 77]. Besides using an LSTM variant, a study showed that we could use ensemble neural networks to enhance the intent extraction task further [78].

Regarding OHCs, some works focus on the general intent of the posts, and others specifically target queries. For instance, a study used 1101 medical questions to create a question taxonomy based on 23 different intents [21]. They made a CNN-LSTM attention model for understanding user query intention. Similarly, another work that analyzed the intention behind questions found that including semantic types from the Unified Medical Language System (UMLS) as additional features to BOW can improve the classifier performance [120]. Furthermore, combining NER with supervised learning can lead to gains in performance [116].

On the other hand, some works focus on all types of textual publications and comments. While a work focused on a more general and extensive intent classification [45], another performed a more specific but less extensive classification [122]. The former distinguishes 17 different classes that could be applied to any forum. Examples of such classes are "assert", "sustain", and "wish". The latter distinguishes five different classes (manage, cause, adverse, combo, story) related to OHCs.

A systematic summary of the literature described above is presented in the Table 3.1 and in the Table 3.2.

Paper	Year	Feature extraction	Model	Evaluation
[21]	2017	Word embedding, POS, NER	CNN-LSTM	95% top-3 recall
[116]	2020	Word embedding	Multi-task attention (with BiLSTM) for NER and intent classification	86,52% (micro F-score)
[122]	2014	Pattern-based features	SVM	75% (F-score)
[45]	2018	n-grams, discourse, conversation features	SVM	71.1% (F-score)
[120]	2005	BOW with UMLS features	SVM	59.5% (F-score)

Table 3.1: Intention classification in OHC

According to Table 3.1, there are three statistical methods and two deep learning methods. All the statistical ones use SVM, and the deep learning ones use an LSTM-based model. We can also verify that, in general, the latter is more recent.

By analyzing Table 3.2, we can see that the data sources for this task are pretty diverse, and both English and Chinese are the most recurrent languages.

Paper	Data source	Classes	Dataset size	Language
[21]	xywy.com	29 different pairs of intent categories	11,964	Chinese
[116]	ask.familydoctor.com.cn.	Diagnosis, Prognosis, Syndrome, Concept, Treatment, Reason, and Complication	12,535	Chinese
[122]	Medhelp	manage, cause, adverse, combo, story	1,192	English
[45]	Reddit + Bhatia + SWBD	assert, sustain, guess, agree, disagree, rejoice, complain, wish, apologize, thank, greet, suggest, direct, engage, accept, refuse, other	2,279 + 461 + 94,950	English
[120]	physicians' questions [44]	Non-clinical, Specific, No Evidence, Intervention, and No Intervention	200	English

Table 3.2: Datasets for intent classification

By observing both Table 3.1 and Table 3.2, there is a positive correlation between the dataset size and performance. We can also conclude that deep learning methods confer the best performance. The small dataset size may be the reason for the poor performance of [120].

The most common motivations to perform intent analysis are the analysis of thread conversations, the identification of key participants in the forums, and the possibility of designing an efficient cooperative information system between participants and organizations. Besides this, knowing the motives behind feedback and providing valuable information to help decision-making and enhance user services are also motivators.

3.1.2 Depression and Schizophrenia Detection

Mental health issues are one of the topics more extensively covered in the literature regarding the classification of online health-related messages. This fact is somewhat expected as mental issues are an urgent topic. According to the World Health Organization, globally, the number of depressed people surpassed 300 million in 2015, and there were 264 million that suffered from anxiety [91]. Moreover, depression is the leading contributor to suicide, with a number reaching nearly 800,000 per year. With the covid outbreak, the situation even got worse since the changing factors can even worsen stress, anxiety, and depression feelings [117].

Depression influences the ability of users to perform their daily tasks, typically affecting users' mood, sleep, eating habits, and social interactions [84, 5]. In several countries, only a tenth of depressed people seek treatment mainly because of the social stigma, and 70% of people would not reach a psychologist during the infant stages of the illness [50]. Another reason for people not getting treated is lack of awareness [84, 106]. So, identifying people who need help in prevention and detection is a core motivation. Undoubtedly, this leads to the degradation of mental well-being. Performing automated detection of depression using social media is a possible path to help fight against the problem. This is also motivated by the fact that depressed people are more prone to sharing their feelings online [39].

Before experimenting with machine learning for automatic categorization, there have been studies that firstly extract LIWC features to try to discriminate between the different classes [106, 14].

Studies have been developed regarding the behavior analysis of depressed people. They found that they are more likely to tweet more at night and often use first-person singular, except for posts with Schizophrenia signals that tend to use the third-person plural [14]. Moreover, they rarely use emojis, and when they do, it is to convey negative emotions [84]. Also, when talking about their symptoms, a negative tone is associated.

There is a study that uses various features, namely emotional, topical, behavioral, user level, and depression-related n-gram, together with an LSTM classifier to predict depression intensities [50]. Moreover, they developed a relabel technique to create the different levels of depression scale. Other studies focused on the early prediction of depression using statistical methods, such as Logistic Regression (LR) [111], and others even showed the possibility of transferring depression labels across domains from Twitter to Weibo [104].

Regarding which model is the best, it seems to be highly dependent on the task prediction and dataset. For depression detection with three different classes, TF-IDF, which includes n-gram features with Random Forest, was the best classifier outperforming frequently used algorithms, such as SVM and NB with an 82.39% [84]. TF-IDF with n-grams beats a simpler Bag of words variants. Nevertheless, for example, the authors [85] showed that using an extensive corpus with 2.5M of tweets with a combination of a simple text representation, such as Bag of words with NB, outperformed other classifiers, reaching an accuracy of 86%. In the research for binary classification of Schizophrenia with TF-IDF with random forest, they obtained an accuracy of 96% using more than 26,000 posts. Unfortunately, the feature selection reasoning was only based on popularity, giving more opportunities for improvement. However, this classification was most likely biased since words such as "schizophrenia" or "schizo" were not masked during training.

The word embeddings used in the literature were passed normally to deep learning models, so there is no clear evidence that they would outperform simpler text representation when using classical machine learning methods, especially with the amount of data available usually not being gigantic.

According to Table 3.3, the number of methods that used statistical methods outnumbers the deep learning methods. While there are five different studies for the latter, there are only two for

Paper	Year	Feature extraction	Model	Evaluation
[111]	2018	Word embeddings	LR	73% (F-score)
[5]	2017	-	NB	73.4% (F-score)
[50]	2021	emotional, topical, behavioral, user level, and depression-related, n-gram features	LSTM	87.14% (F-score)
[14]	2021	LIWC, topic distributions	RF	96% (F-score)
[104]	2018	textual features, visual features, use profile & post behaviors features, social interaction features	Deep Neural Network model with Feature Adaptive Transformation & Combination strategy (DNN-FATC)	78.5 +- 1.2 % (F-score)
[85]	2016	BOW	LR	73% (F-score)
[84]	2022	unigram, negation features	RF	82.53% (F-score)

Table 3.3: Depression and Schizophrenia detection in OHC

the former. Random Forest and Logistic Regression are the methods most used.

Paper	Data source	Classes	Dataset size	Language
[111]	eRisk 2017 Dataset (constructed using reddit)	depressed / non-depressed	30,851 (TP) + 264,172 (TN) + 18,706 (TP) + 217,665 (TN)	English
[5]	LiveJournal, Twitter, Facebook	depressed / non-depressed	2,073 (TP) + 2,073 (TN)	English
[50]	Twitter	no / mild / moderate / severe	4,245,747 tweets from depressed and non-depressed users	English
[14]	subreddit r/schizophrenia	with / without Schizophrenia	13,156 (TP) + 34,413 (TN)	-
[104]	Twitter, Weibo	depressed / non-depressed	3,948	Chinese
[85]	Twitter	depressed / non-depressed	almost 2M tweets from depressed and non-depressed users	English
[84]	Twitter	depressed / neutral / non-depressed	4,452	Arabic

Table 3.4: Datasets for Depression and Schizophrenia detection

According to Table 3.4, several publications have been made regarding detecting mental diseases, especially using Twitter and the English language. However, this matter has been developed

in other languages, such as Arabic and Chinese. Besides Twitter, Reddit has been a popular social media for data collection regarding mental health.

The majority of studies also focused on binary classification. The best model used LR (statistical method). This can be justified by the fact that some works [104] may not have enough annotated data samples to converge to a maximum. Moreover, comparing two classifiers that act in different languages may not be reasonable as they must overcome different challenges.

Machine learning has successfully detected symptoms and depression and assisted in characterizing those groups. Acquiring more data can lead to better improvements even if the algorithm and text representation is simpler when compared to other tasks that are useless data.

3.1.3 Suicide Detection

According to the World Health Organization, in 2015, suicide was the second leading cause of death in individuals between 15 and 29 years [25]. Suicide detection on social media can lead to its prevention, so it has been a topic of interest over the years. This is possible because people are sharing their social thoughts on social media [3].

There has been work that focused on binary classification. A work revealed that Random Forest was the best model with an accuracy of 85.8% [102]. Even though they developed an LSTM, it outperformed, except on Recall, where it had the best performance among the classifiers. Being outperformed could be because it was trained on a small dataset with only 822 tweets annotated as "suicidal" (true positives). In terms of feature selection, it was disclosed that by including substantial different types of features such as statistical features (number of words and chars), LIWC features, part of speech counts, TF-IDF, and topics probabilities (using LDA), there were performance gains. Similarly, another binary detection classifier demonstrated that using LIWC and sentiment matrices for both the title and body of the messages with TF-IDF brings performance gains [3]. This last paper discusses the importance of a good balance between precision and Recall and what this type of task implies.

A multiclass classifier was developed using Random Forest [32] had an accuracy of almost 88%. Besides the "suicide" and "non-suicide" labels, they added the "flippant" tag that comprises facetious remarks about suicide. Another work also performed multiclass classification by distinguishing different levels of concern regarding suicide [89], reaching an accuracy of 76% using an SVM.

There have been developments in this research using neural networks [12]. They built a binary classifier using neural networks to correctly predict 65% of the test set. Moreover, interestingly, they also tested the impact on the performance of maintaining and removing the emojis from messages, and the latter is always preferred.

A deep learning model that performed noticeably well [35] used a pre-train GloVe embedding to reduce the risk of overfitting. The word embeddings served as input to a bidirectional LSTM model that achieved an 85% of the true-positive rate. They analyzed, on average, 473 messages per user of a total of 418 different users. They also show that analyzing messages months prior leads to a higher AUC.

Paper	Year	Feature extraction	Model	Evaluation
[3]	2018	TF-IDF vectorization, LIWC, sentiment score	SVM	92% (F-score)
[30]	2017	SC-LIWC, language features	SVM	0.48-0.75 AUC
[12]	2018	unigrams and bigrams	Neural network	78.36% (accuracy)
[102]	2018	statistical features, LIWC features, POS, TF-IDF, topics	Random Forest (LSTM in recall); Rule based not good enough	84.4% (accuracy)
[32]	2018	bag of words and TF-IDF vectorization	Decision Tree	87.9% (F-score)
[89]	2015	TF-IDF vectorization	SVM	76% (accuracy)
[35]	2018	word embedding	LSTM	0.89-0.94 (AUC)

Table 3.5: Suicide detection in OHC

Table 3.5 reveals that SVM and tree-based models are more used. Moreover, there are five statistical methods compared to two neural network-based methods.

Paper	Data source	Classes	Dataset size	Language
[3]	Subreddits: r/SuicideWatch, r/Depression, r/Anxiety, and r/ShowerThoughts	positive / negative	785	English
[30]	Weibo	5 binary classifiers of suicide risk factors: Suicide Probability Scale, depression, anxiety, stress, and Weibo suicide communication	-	Chinese
[12]	Twitter	Positive / negative	5,174	English, Filipino, Taglish
[102]	Twitter	Positive / negative	5,213	English
[32]	Twitter	suicide, campaign, flippant, support, memorial, reports, other	2,000	-
[89]	Twitter	strongly concerning, possibly concerning, safe to ignore	1,820	English
[35]	public self-stated data and donated through OurDataHelps.org	Positive / negative	197,615 (TP) + 197,615 (TN)	English

Table 3.6: Datasets for suicide detection in OHC

By observing Table 3.6, Twitter is the most used data source to build datasets, and English is the most common language chosen.

The amount of labeled data available in suicide detection often bottlenecks the performance, and, consequently, simpler models tend to behave better. Nevertheless, several approaches try to mitigate that by adding new features to detect suicide incidents. Moreover, some research overlooked the importance of false negatives. In these predictions, minimizing that value is essential since we do not want to leave people at risk undetected. If carried out in real-time, this detection could save thousands of lives every year and intervene and help patients during periods of crisis.

3.1.4 Surveillance

Adverse drug reaction detection is crucial to prevent and act promptly to prevent disastrous consequences. The process of understanding and preventing these events is named pharmacovigilance. Moreover, the consumption of other substances can also be monitored through social media. In terms of chronological order, before machine learning models were extensively explored, lexicon-based approaches were the most used [16, 119]. Unfortunately, they rely on lexicons, which hinders the possibility of analyzing any arbitrary number of drugs.

Several works focus on mining tweets for ADR. The collection of meaningful tweets can be accomplished using drug keywords, taking into account possible misspellings [51]. Similarly, a work validated their model by choosing five different drugs that appeared on tweets to perform side-effects detection [61]. They took into consideration 17 different possible side-effecting by leveraging the MetaMap software. Even though they experimented with three statistical classification methods, NB, SVM, and Maximum Entropy, that had similar performance, the latter was the best. Some works consider the problem of data imbalance and the combination of data from multiple sources to boost performance [98]. They also explored different feature extraction methods using a leave-one-out strategy and pointed out the usefulness of n-grams and UMLS semantic types. Out of three different statistical binary classifiers, SVM outperformed the others, including NB, contrary to some other works [51].

Besides binary classification, some conduct multiclass classification [88] also considering other aspects of using drugs. They created four classes: based on adverse drug reactions, beneficial effects, indication (why one is taking the drug), and other (mention or sign of symptoms). They joined word embeddings using word2Vec with negative features (checks whether the token is negated), context features (tokens around and current token), and POS tagging. These additional features only gave a small contribution when the training set was small. Using four different classes with SVM, they obtained a 0.72 accuracy on a Twitter dataset. Similarly, a work added extra features such as the presence or absence of URLs and the length of tweets, but they were not relevant [63]. On the other hand, the presence or absence of essential keywords boosted performance. Even workshops were conducted globally related to ADR where an already annotated dataset was given to the participants [99]. One of its objectives, besides dealing with social media data, was to classify the messages, as other works have already done, such as "ADR", "beneficial effect", "indication", or "other".

Some studies are based on e-cigarettes since people are shifting from using cigarettes to those, which may have led to people abusing them. A work used a simple methodology for classification: after sampling more than 81 thousand messages, they applied unigram and bigram word frequency to identify the most prevalent topics. They expanded that keyword list by using FastText, and the classification was entirely based on the presence of those keywords. Additionally, combining e-cigarette-related tags with other tags such as the "School" tag would allow seeing what students say about e-cigarettes [8]. Others developed a binary classifier using NB, SVM, Bayesian Logistic Regression, and Random Forests. This last one performed distinguishably well with a high area

under the receiver operating curve (ROC) [10]. This may be attributed to being an ensembling method.

Some works perform surveillance to predict outbreaks. An example of one is flu surveillance based on behavior analysis [34], which classifies health messages from social media into five different categories. They perform two other supervised learning techniques, namely SVM and NB. It is concluded that there is a high correlation between specific signals in tweets and diagnosing influenza. The categories used were avoidance behavior, increased sanitation, seeking pharmaceutical intervention, wearing a mask, and self-reported diagnosis.

Paper	Year	Feature extraction	Model	Evaluation
[99]	2016	Word embeddings, part-of-speech tags, n-grams, lexicons, spell-checking, and negations	Conditional random fields	61.1% (F-score)
[51]	2014	BOW	NB	78.1% (F-score)
[98]	2015	UMLS semantic features, polarity, ADR lexicon matches, sentiword scores, topic based, n-grams, language features	SVM	81.2% (F-score)
[88]	2015	context features, ADR lexicon, POS, Embedded cluster features	ADRMine	82.1% (F-score)
[69]	2016	n-grams, POS, token weights, isAllCaps, isElongated, isNegated, non-contiguous n-gram pair weights, TW cluster, drug name and minimum sentiment relative position	ADRMine	80.14% (F-score)
[61]	2013	tokenization, lexical lookup, shallow parsing, semantic type mapping and word sense disambiguation	Maximum Entropy	84.8% (F-score)
[10]	2016	n-grams	Bayesian Logistic Regression	0.9 (AUC)
[63]	2016	unigrams, POS, polarity scores, topic distribution, presense of certain terms	Logistic regression	91% (F-score)
[34]	2011	unigrams, regular expressions	SVM	74% (F-score)

Table 3.7: Surveillance in OHC

By observing Table 3.7, we can conclude that statistical models are the most prevalent. Moreover, there seems not to be a preferred model for this type of task. ADRMine is a model based on conditional random fields.

By analyzing the Tables 3.8 and 3.7, we can verify that the best multiclass classifiers perform fairly well partly because of the amount of data available and because they use ADRMine. Furthermore, we verify the significant incidence of the English language to the detriment of others.

The main motivations for this type of task are to perform surveillance and possibly amend some regulations to attenuate possible harmful consequences suggested by the individual's behavior, namely the abuse of substances.

Paper	Data source	Classes	Dataset size	Language
[99]	Twitter	has / has not ADR	2,607	English
[51]	Twitter	has / has not ADR	1,008	English
[98]	Twitter + DailyStrength + ADE corpus	has / has not ADR	10,573	English
[88]	Twitter + DailyStrength	B-ADR,I-ADR,B-Indication, I-Indication,and Out.	8,063	English
[69]	Twitter + DailyStrength	B-ADR,I-ADR,B-Indication, I-Indication,and Out.	8,061	English
[61]	Twitter	Detects side-effects	885	English
[10]	Twitter	positive / negative	728	English
[63]	Twitter	positive / negative	957	English
[34]	Twitter	asavoidance behavior, increased sanitation, seeking pharmaceutical intervention, wearing a mask, self-report	5,283	English

Table 3.8: Datasets for surveillance in OHC

3.1.5 Sentiment Analysis

In OHCs, sentiment analysis is often used as a proxy to voice users' opinions concerning specific diseases, treatments, and health-related matters.

A study tried to verify if there is an association between HIV drug treatment and sentiments [2]. In general, negative sentiment was captured. Moreover, they concluded there seems to be a relation between the sentiment and the type of drug used. Identically, a work used sentiments as a proxy for adverse drug reaction characterization [69]. They explored which features would contribute the most to the performance: contextual features, lexicon-based features (checks if the token exists in the ADR lexicon), parts of the speech, negation, and, lastly, embedding features. Each feature mentioned was evaluated separately to verify its contribution to the result. Sentiment analysis has also been conducted to analyze people's insight, perspective, or opinion of a topic, namely the view of people regarding alternative medicine and cancer [80] or concerning the HPV vaccine [79].

The impact of emotions on the prediction of suicide has been explored in Twitter posts [100]. They used a semi-supervised machine learning method named YATSI, enabling them to achieve state-of-the-art performance. It is a two-stage algorithm based on Random Forest. Although they used BOW as a feature representation with additional sentiment features, with just 4487 labeled posts, they obtained superior results to previous work. This reveals the potential of semi-supervised learning algorithms, especially with limited annotated data. The inclusion of sentimental features improved the performance of the classification process.

Predicting emotions of psychiatric questions was also a focus of a research [42]. What made this problem particularly interesting was the emphasis that the sequence of text was long, rendering the situation more challenging. Using 2,086 questions from WebMD and 5,528 from Healthtap,

a pre-trained GloVe embedding and a deep learning model based on LSTM and CNN achieved a state-of-the-art accuracy of 97.8%.

More complicated tasks such as predicting emotional tone over time also require a deep learning model that can capture the time dimension, namely LSTM [106]. The amount of data used was significant for this task, with data from 4 popular mental illness subreddits over six years.

A study leverages existing Chinese and English Dataset and uses a deep learning model, namely, an LSTM, to detect different sentiment polarities [74], achieving performance around the 90% mark of F-score. Its success is greatly attributed to the number of records the datasets contain.

Lastly, a study analyzes users' sentiments before and after a health threat [90]. They developed both a lexicon-based classifier and an ML-classifier in which the last one surpassed the performance of the other when using a bag of words and sentiment features. They concluded that there is a noticeable difference in the distribution of emotion prior to and after an outbreak event.

Paper	Year	Feature extraction	Model	Evaluation
[79]	2016	-	Moore-Penrose pseudoinverse (binary) + decision tree (multiclass)	vaccine sentiment (AUC=0.918), side effects (AUC=0.739), prevention (AUC=0.774)
[100]	2021	BOW, sentimental features	YATSI (semi-supervised algorithm)	86.97% (F-score)
[42]	2021	GloVe word embeddings	a model multi-head attention with bidirectional long short-term memory and convolutional neural network (MHA-BCNN)	WebMD: 78.3% (F-score) Health-tap: 89.5% (F-score)
[74]	2021	word embeddings	Double Channel-LSTM with hybrid attention	Sentiment140: 87.03% (F-score) cMedQA: 91.91% (F-score)
[106]	2021	word embeddings (DistilBERT), emotional tone (VADER)	RNN	outperformed baselines by over 12.9%

Table 3.9: Sentiment analysis in OHC

According to Table 3.9, word embeddings have been used together with deep learning methods. This is usually attributed to the fact that large datasets and tools concerning sentiment analysis are available.

According to both Table 3.9 and Table 3.10, we can verify that large datasets are correlated with better performance. However, even if they have less annotated data, semi-supervised learning methods can outperform other classifiers with relatively the same amount of data that use statistical methods.

3.1.6 Topic Classification

Regarding topic classification, a study characterizes discussions about antibiotics, proposing nine different classes [65]. They used a semi-supervised approach and a shallow neural network with

Paper	Data source	Classes	Dataset size	Language
[79]	Twitter	positive, negative, neutral (sentiment classes) and no mention, side-effects)	1,470	English
[100]	Twitter	anger, fear, sadness, joy, positivity and negativity scores	1,000 (labeled) + 4,487 (unlabeled)	English
[42]	Healthtap, WebMD	anxiety, insomnia, stress, depression (sentiment classes) and OCD, addiction	7,614	English
[74]	cMedQA and Sentiment140 datasets	cMedQA: 6 categories Sentiment140: 3 categories	CMedQA: 794,099 Sentiment140: 1.6M	CMedQA: Chinese Sentiment140: English
[106]	subreddits: r/Anxiety, r/Bipolar, r/Depression, r/SuicideWatch)	emotional tones	126,234 threads	English

Table 3.10: Datasets for emotional analysis in OHC

only three hidden layers with 700, 700, and 300 nodes. Using this and only 416 annotated tweets, they were able to get an accuracy of 70.4%.

Zhang et al. [121] used posts from breastcancer.org in which 1,008 posts and 9,016 sentences were annotated to create a classifier to identify topics of breast cancer over time. As topics can change from sentence to sentence, they use sentences as data samples instead of a full post. They proposed a multi-label topics classification that used CNN, outperforming the SVM model. Similarly, a study built a neural network classifier to distinguish different topics related to the Lyme disease [9].

Finally, a work argued about the lack of studies regarding the post-cancer analysis [13]. They analyzed the most important topics discussed by the patients to understand their experiences. They applied a methodology similar to our aim: they use self-training and deep learning models. A look-up table assisted the self-training in enhancing performance. The algorithm works by building a base classifier to augment the labeled data. To create some consistency, only after labeling the same instance with the same label several times is it added to the training set. More annotated posts generate an opportunity to use more complex machine learning models. They used deep learning models such as CNN, LSTM, and BiLSTM, getting an F1-score of 79.5%. They used 800 data samples for training, 100,000 were automatically marked, and 200 were used for testing.

The motivation for these is the capability to analyze behavioral patterns associated with specific topics. That capability can lead to understanding the patient's needs better. Moreover, this can be meaningful as doctors can be more conscious of the implications of people's beliefs.

The Table 3.11 shows the prevalence of neural network based approaches to the detriment of statistical methods.

We can conclude by observing Table 3.12 that all datasets have several classes since topics are diverse in OHCs. Moreover, the amount of annotated data for this task is limited, so several works

Paper	Year	Feature extraction	Model	Evaluation
[13]	2021	BERT word embedding	Ensemble of classical ML + BiLSTM	79.5% (F-score)
[65]	2015	BOW and some metadata (length tweet, etc)	Neural network with 3 hidden layers	70,4% (accuracy)
[121]	2017	word embeddings	CNN	60.3-78.1% (F-score) depending on the topic
[9]	2019	meta-level feature, polarity lexicon feature, content-free features	multi-class neural network	75.2% (accuracy)

Table 3.11: Topic classification in OHCs

Paper	Data source	Classes	Dataset size	Language
[13]	Breastcancer.org	context, support, treatment stories, adverse drug reaction effects, diagnosis stories, finance, clinical information	800 (labeled) + 74324 (unlabeled)	English
[65]	Twitter	Advice/Information, Other, General Use, Resistance, Wanting/Needing, misuse, side-effects, Animals, advertisement	412 (labeled) + 150,000 (unlabeled)	English
[121]	Breastcancer.org	ALTR: alternative medicine; DAIL: daily matters; DIAG: diagnosis; FIND: findings; HSYS: health systems; MISC: miscellaneous; NUTR: nutritions; PERS: personal life; RSRC: resources; TEST: laboratory tests; TREA: treatments	1,008 (9,016 sentences)	English
[9]	Twitter	asking about treatment, lyme infection confusion, lyme symptoms, confusion, depressed & frustrated, motivation and encouragement, seeking general information, other	1,474	English

Table 3.12: Datasets of topic classification in OHCs

used semi-supervised learning models.

3.2 Literature Review Summary

In this section, a quantitative analysis of some aspects of the literature will be made, and some tools used in the literature will be presented with a short description.

According to Figure 3.1, statistical methods are more common for intent classification, depression, suicide detection, and surveillance, with the latter being the one that has the most, with

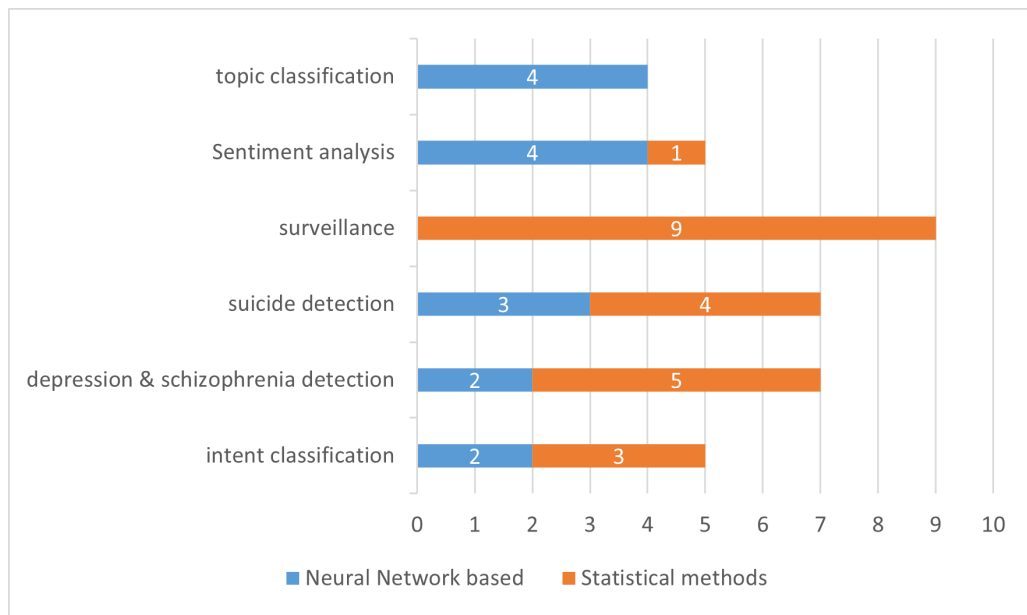


Figure 3.1: Health fields and corresponding approaches

nine different ones. Regarding neural network-based approaches, sentiment analysis and topic classifications tasks outnumber the statistical approaches. Both are the only tasks that apply semi-supervised learning methods: topic classification has two, and sentiment analysis has one work. Besides, sentiment analysis has a wide range of large annotated datasets, allowing the use of deep learning models. This shows that for new datasets, it may be possible to apply a deep learning model by leveraging semi-supervised learning methods.

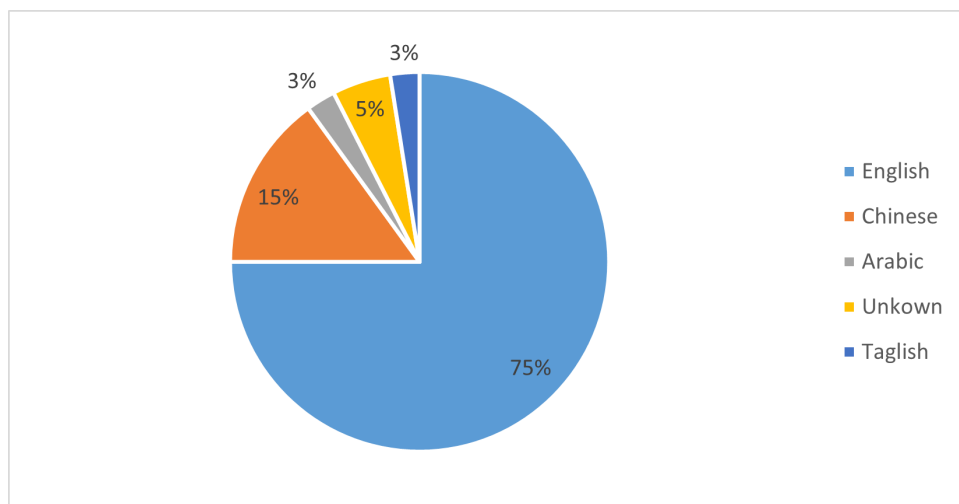


Figure 3.2: Target Languages

Figure 3.2 depicts the percentage of works that focus on a specific language. English is, undoubtedly, the preferred choice, with three-fourths of the total works targeting it. Chinese is the second language more used, with some works specifically focusing on Chinese social media.

Moreover, some papers also focus on other languages, such as Arabic and Taglish.

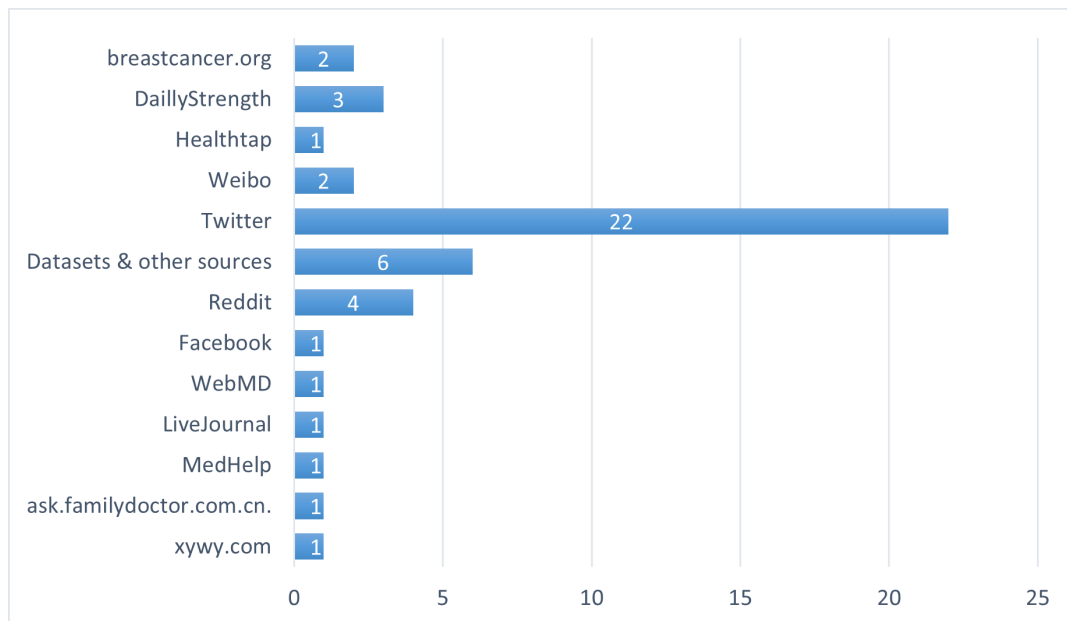


Figure 3.3: Data sources

Table 3.3 depicts the number of works that use a certain OHC as its source of information. Twitter is the most used data source. Weibo and xywy.com correspond to works that target the Chinese language. Besides existing datasets, physicians’ questions and data acquired in ourData-helps.org (currently not available) are included in other data sources.

3.3 Relevant Datasets and Schemes

To perform sentiment and polarity analysis, besides the tools already stated in the Background, some datasets are worth mentioning. The dataset Sentiment140 is an English dataset with around 1.6 million annotated tweets. The target labels are “negative”, “positive” and “neutral”. The annotation was not accomplished manually; they used a classification method that used Distant Supervision [52]. They argue that compared to other models, even though they may have lower precision, they have a higher recall. Their training data has emoticons as noisy labels. They strip the emoticons for the classifier to be able to learn from other features. We argue that even though they have a lot of data samples, the existence of noisy labels and other datasets makes this approach probably not the preferred one. Google developed a fine-grained emotion dataset called GoEmotions [40]. The dataset contains 58 thousand English Reddit comments. They demonstrated the quality of the annotation using Principal Preserved Component Analysis. They also showed that the dataset generalizes well to other domains. The dataset is constituted of 27 emotional categories and a neutral one.

Regarding topic classification, a previous work created a scheme based on a breast cancer forum [13]. They proposed a scheme that, even though it was based on what they did find in that

forum, could very well be applied to others. They proposed seven different classes presented in Figure 3.13.

Topic	Definition	Number of instances
Support	Support received: either emotional or information	211
Contextual	Entities encountered everyday, specially by ill people or survivors	184
Treatment stories	Discussion of treatments	154
Adverse drug effects	Negative reaction caused by a drug	109
Diagnosis stories	Discussion of diagnostic stories and histories	122
Finance	Financial matters and insurance	115
Clinical information	Information about clinics or clinicians	105

Table 3.13: Scheme proposed by Balakrishnan, Athira et. al (2021) [13]

Another interesting scheme is proposed by a work that performs longitudinal analysis of topics discussed in an online breast cancer community [121]. They formulated their work as a multiclass classification problem. The proposed scheme can be summarized in Table 3.14.

Topic	Definition	Number of instances
Alternative	Alternative and integrative medicine	302 + 113
Daily	Cancer-related experience	600
Diagnosis	Diagnostics and test results	1,127
Finding	Finding sign, symptom, side effects	1,195
Health Systems	System patient interact with: doctors, nurses, hospitals, insurance companies	864 + 104
Miscellaneous	What does not fit under the other labels	1,956
Nutrition	Nutrition	608 + 113
Personal	Personal Information	1,011
Resources	Pointer to an external information resource	568
Test	Testing procedures (not results)	295
Treatment	Treatments, including medication, procedures, therapeutic devices	2,078 + 104

Table 3.14: Scheme proposed by Balakrishnan, Zhang et. al (2019) [121]

As we can see from Table 3.14, in terms of multilabel instances, there are only two combinations, namely Nutrition with Alternative and Treatment with Health systems.

3.4 NLP Tools

In Table 3.15, several tools used throughout the literature are presented with a small summary of their purpose.

Tool	Description
Word length reducer [20]	Deals with the fact that words are longer by repeatedly using a letter which may act as a proxy for certain emotions. For instance, "coool" may emphasize excitement.
MetaMap	MetaMap is a highly customizable tool that maps biological text to Metathesaurus concepts and their associated semantic categories to make it easier for users to incorporate this information into their applications. The Unified Medical Language System (UMLS) Metathesaurus is the greatest thesaurus in the biomedical area, representing biomedical knowledge with over one million concepts categorized by semantic type and relationships between them.
Stanford Parser	The parser is software that determines the grammatical structure of sentences, such as which words belong together and the verb's subject or object. Probabilistic parsers strive to generate the most likely analysis of sentences by using linguistic information gathered from hand-parsed sentences. These statistical parsers are not perfect, but they are usually rather good.
SentiWord	It is a lexical resource for opinion mining and emotion classification as the result of annotating the wordnet's (lexical database) synsets (grouping of synonyms words) according to the degree of positivity, negativity, and neutrality.
LIWC (Linguist Inquiry and Word Count)	Extracts words from texts that fall in LIWC categories, giving social and psychological insights regarding ideas, feelings, personalities, and intentions.
NRC Affect Intensity Lexicon	It is a list of English words with score intensities for eight different emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, and trust).
SentiStrength	It is a Java tool to detect the strength of positive and negative sentiment in short texts, including informal language. It is implemented in NLTK, and the score values range from -5 to 5.
Stanford NER	Stanford NER, also known as CRFClassifier, is a Java implementation of a Named Entity Recognizer. Named Entity Recognition (NER) labels words in a text that are the names of something, such as a person, a company's name, a place, and even more.
Porter Stemming	It is a very well-known algorithm used for removing the morphological and inflexional endings from words in English. NLTK has its implementation.
Vader	Sentiment analysis for predicting polarity (positive/negative) and intensity of emotion. Nltk has its implementation.

Table 3.15: Tools used in the literature

3.5 Existing Online Health Communities

Several forums and social networks have health-related content, as shown in Table 3.16. In order to publish something, all of them required a user account. The availability of APIs, as a rule, is correlated with the platform's domain. Content-generic platforms, such as Reddit and Twitter, possess it. We can observe that certified professionals are only present in forums set up expressly to deal with health issues and concerns. The features developed bear in mind the target audience of those platforms. Aside from Twitter, discussions occur in threads. On Twitter, public conversations are often performed by mentioning specific users using "@" and topics are usually labeled by using hashtags.

3.6 Ethical Concerns

The growth of data available in social media and the possibility of using that data for NLP tasks has raised ethical concerns. The discussion of ethics about NLP was merely started in 2016 by Hovy and Spruit [59]. The authors emphasize the non-existence of guidelines for such tasks even though there are Institutional Review Boards (IRB) whose objective is to control and enforce ethical guidelines for human-involving experiences. They raise concerns regarding language being a gateway for users' inner characteristics and situations. In addition, the presence of dual-use problems should not be ignored. Even though they have a clear benefit, these are problems that can harm and have a negative impact if they are misused. They can exist for any research if the results are used differently than originally intended. For example, we have pre-trained machine learning models to create adversarial examples.

Furthermore, there exist some worries regarding algorithms' performance since machine learning models are built based on real-world data, which can be biased and not representative of the population [82, 28]. If the overgeneralization is based on false positives, it can have serious consequences depending on the task [112].

The twisted nature of these unsolved problems resulted in the healding of the first workshop on NLP ethics during the European Chapter of the Association for Computational Linguistics conference in 2017 [15]. Regarding using social media data for health matters, guidelines for ethical research were proposed. They consist of obtaining user permission whenever possible, carefully contemplating the outcomes of interactions with users or changes to their experience, protecting their data while researching and sharing it with other researchers, and concealing user data during analysis and presentation.

In 2018, the second workshop in NLP ethics was held [6]. It highlighted the addition of information to summaries about possible failure scenarios and future work, and there was a discussion regarding conversational system responses to sexual harassment.

More recently, there have been raised ethical concerns regarding crowdsourcing in NLP research, namely, the harm and risks for crowd workers [105]. This includes psychological harm via text, the exposure of their information, unmindfully including vulnerable populations to do

Name	Domain	API	Moderators	Professionals	Characteristics
Reddit	Generic, with health-related subreddits	Yes	Yes	No	Downvote/Upvote; Several type of media content; Coin gifting for good or funny content.
Twitter	Generic, hashtags used to infer content	Yes	Yes	No	Retweet and like posts; Several type of media content; Limited amount of chars per message.
Breastcancer.org	Breast cancer related divided into different topics, such as fitness, surgery.	No	Yes	The medical information published is reviewed by over 70 medical professionals.	Over 226k members and 83 forums; Latest research news; Personalized recommendations;
DailyStrength	Depends on the support groups. Examples are Anxiety, Bipolar disorder, Fibromyalgia.	No	Yes	Available for contact	Over 500 support groups; Write journals; private and public entries;
Sina weibo	Organized in super topics (chinese)	Yes	Yes	No	Several type of media; Stories and vlog videos; Limited amount of characters/post.
HealthTap	Health-related, divided into different communities.	Yes	No	Yes	Takes insurance into account; Symptom check; Schedule doctor appointment.
WebMD	Health-related, divided into different communities.	No	Yes	Yes	News and information about human-health are available.
xywy.com	Health-related, divided into different communities (chinese)	No	No	Yes	Payed one-to-one doctor appointment (that can be online).
LiveJournal	Generic, divided into different communities	Yes	Yes	No	Besides communities, it allows the creation of blogs.
Inspire	Health-related divided into different communities.	No	Yes	Partnership with associations	Connecting with other users; Partnership supports the communities needs.
PatientsLikeMe	Health-related divided into different communities	No	Yes	Partnership with associations	Review of drugs and treatments; Personal dashboard; Information about illnesses and conditions.
Meta	Generic, divided into different groups.	Yes	Yes	No	Several types of media; Groups can have different levels of privacy; Groups can have rules.
Patient	Health-related divided into different communities.	No	Yes	Clinical information reviewed by doctors.	Medicine and conditions leaflets.
Medhelp	Health-related divided into different communities.	No	Yes	Partnerships with doctors and health research institutions.	10.8MM health discussions; Extensive network with millions of users; There are forums only answered by professionals

Table 3.16: Popular websites that contain OHC

the work, the breaching of anonymity and privacy of the workers, and, finally, the triggering of addictive behavior (gamified crowdsourced tasks).

In conclusion, there has been a gradual shift from unconcerned usage of data from social media for research into a more intricate perception that recognizes the existence and impact of ethical and regulatory problems that emerge when utilizing NLP to conduct research. Efforts have been made to identify, explore and create frameworks and practical guidelines to conduct health-related NLP research.

Chapter 4

Problem Statement and Methodology

The current chapter will present the classification scheme we are using, show which one we are based on, state the problems as research questions, and, finally, present the overall methodology associated with each.

4.1 Previous Content-based Hierarchical Scheme

There is a work whose objective was to create a hierarchical classification scheme based on the analysis of content from an OHC called MedHelp [76]. For this, they annotated around 200 messages. The first level of the classification scheme is constituted by the classes offering support, seeking support categories, group interactions, and emotions. The dataset is multilabel, meaning that one message can have multiple categories. Their final proposal is presented in Table 4.1.

From the deeper level of it, we can infer what the higher-level classes should include. In the next section, we present the schema we used for our work which is very similar to this one with little nuances. Additionally, a description of each possible class will be exhibited. Furthermore, it is relevant to mention that a dataset with around 200 samples was made available to us, and, hence this is our starting point.

4.2 Adapted Hierarchical Scheme

First, we partitioned the scheme into two different ones. We separated the branch Emotions into a classification problem of their own. Due to the nature of emotions not being only negative or positive, we added a new branch, namely neutral. Moreover, since the branch happiness and negative do not encompass all positive and negative emotions, the dataset used as a starting point presented in the previous section was reclassified for this branch. This process will be better described in a future section.

In terms of the other high-level branches, they are left untouched. Therefore, our first classification task is presented as a three-class multilabel problem.

First-level	Second-level	Third-level
Offering Support	Information Support	Advice
		Recommendation
		Teaching
	Emotional Support	Affection
		Sympathy
		Encouragement
		Prayer
		Relief of Blame
	Esteem Support	Compliment
		Validation
	Network Support	Acess
		Presence
	Tangible Assistance	Perform direct task
		Express willingness
Seeking Support	Specific question	Specific question
	Reassurance	Reassurance
Group Interactions	Gratitude	Gratitude
	Congratulations	Congratulations
	Sharing Personal experiences	Sharing Personal experiences
Emotions	Negative	Anger
		Fear
		Sadness
	Hapiness	Hapiness

Table 4.1: Scheme proposed by Lopes & Guimarães (2019) [76]

In addition, we also want to build a second-level classifier. For this, we made a slight change in a branch of the classifier: instead of having the following classes proposed in the original schema, namely "Congratulations", "Gratitude", "Sharing Personal Experiences", "express willingness", we added "Agreement" and "Disagreement" and mapped all of them into "Interpersonal" (except sharing personal experiences), which usually relates to something that was said previously. Lastly, the branch "tangible assistance" was removed from the main scheme as it was practically unused, as stated by the original authors. Next, they are presented with a description of each class so that what each represents becomes clear.

4.2.1 High-level Classifier

In Figure 4.1, the first-level multi-label classification scheme is depicted.

To better understand it, we can derive the classes definitions to make everything clear.

- **Seeking Support:** People looking for a sort of support by asking a question, seeking comfort or making a request.

Example: Where can I find free insulin pump supplies and insulin?

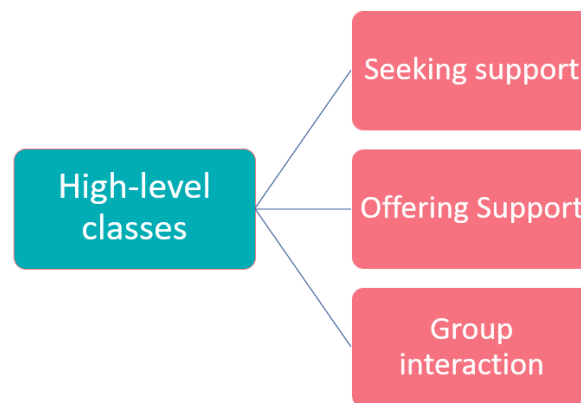


Figure 4.1: High-level classification classes

- **Offering Support:** It can be related to the share of knowledge, some advice or appraisals. It can also include some sort of emotional support, such as showing empathy or complimenting someone or validate. Finally, it can include the improvement of social contacts by meeting people in similar situations.

Example: I fully understand what you all are referring to when you talk about it being stressful mentally and emotionally. It's horrible being sick so regularly and not have an answer and a fix for it.

- **Network Interactions:** These include some sort of interactions and it is very much related to previous messages, such as agreement, disagreement, congratulations, and gratitude. All of them act as a reaction to some action, i.e, to something said previously. Finally, this class also includes the sharing of experiences related to a specific event of someone's life.

Example: I have the same kind of headache and I'm about one week from the start of my miscarriage. The pounding is terrible!!! I'm looking for answers as well.

4.2.2 Second-level Classifier

The categories of the messages from the second-level classifier can be seen in Figure 4.2.

To clarify this classification level, we present all the definitions of the classes above.

- **Informational Support:** Share of knowledge and reduction of uncertainties. It can include advice, situation appraisal and teaching.

Example: I highly recommend seeing a nutritionist that specializes in addiction too. They can recommend supplements that will replace the things that the drugs took from your system and will help the baby replenish what it hasn't been getting

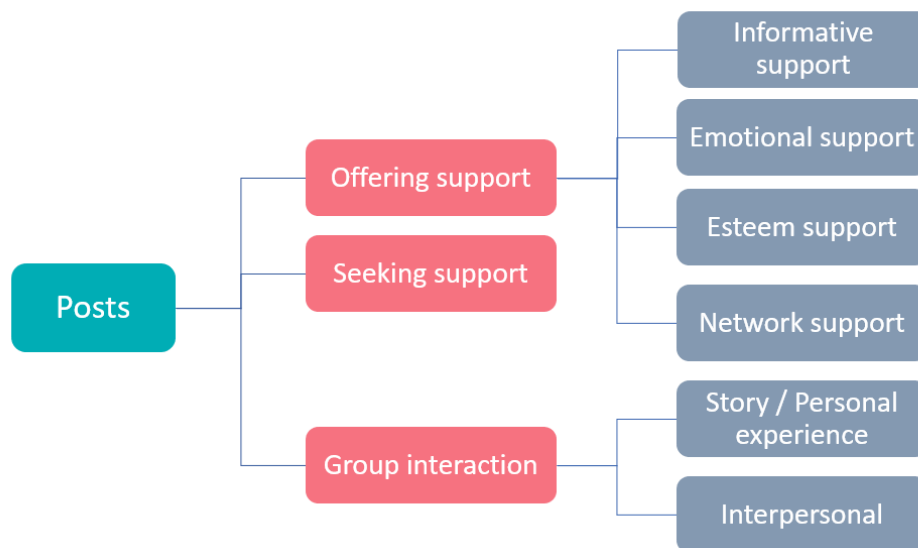


Figure 4.2: Hierarchical scheme with second-level categories

- Emotional Support:** Express empathy towards people. It can include sympathy, confidentiality, encouragement, prayer, relief of blame, virtual affection and relationship.
Example: I talked to my daughter about you and your wife. She feels awful for all of you because she knows what it's like for her and for her loved ones. But, she is hoping that your wife will agree to testing and get the help she needs.
- Esteem Support:** Allows to improve one's confidence. It can include compliments or validation.
Example: You are strong for standing up to it and should not feel embarrassed. Many women are in need of some help during and after pregnancy, it takes a brave and courageous woman to seek help. You can do it I promise.
- Network Support:** Allows a user to improve their social contacts and get to know other people in similar situations.
Example: Please keep me posted!!! And good luck,, Us 40 year olds need to stick together!!!!
- Seeking Support:** People looking for a sort of support by asking a question, seeking comfort or making a request.
Example: Where can I find free insulin pump supplies and insulin?
- Agreement:** Simply agreeing with something said previously.
Example: I certainly agree with that last perspective.
- Disagreement:** Disagreeing with what was previously said.
Example: I think you are in the wrong there.

- **Gratitude:** Express gratitude because of an answer, recognising the give help.
Example: Thank you everyone! Your answers have helped a lot!
- **Congratulations:** Expresses joy towards something accomplished.
Example: Congratulations! Hope all goes well :).
- **Sharing Experiences:** Share of experiences, thoughts, and feelings of a user related to a certain event in its life.
Example: I have the same kind of headache and I'm about one week from the start of my miscarriage. The pounding is terrible!!! I'm looking for answers as well.

4.2.3 Polarity Analysis

The difference between this scheme with the original scheme is that it corresponds to a task of its own. Rather than having positive and negative classes in a multi-label setting, it has positive, neutral, and negative classes in a multiclass one. In Figure 4.3, the emotional polarity classes are depicted.

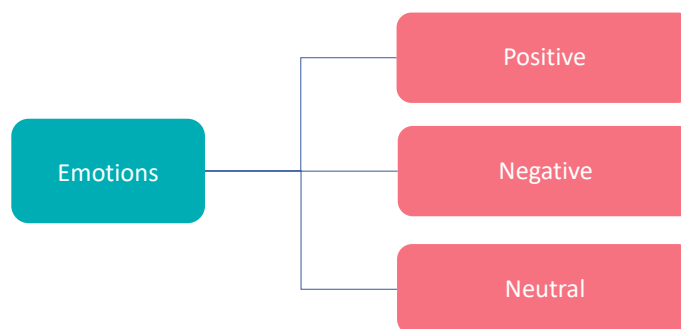


Figure 4.3: Emotion classification schema

The definition of each category is quite self-explanatory:

- **Positive:** Corresponds to positive emotions, such as happiness and excitement.
Example: I'm due June 24th I find out what I'm having Feb 1st! This will be my 3rd. Exciting yall!
- **Neutral:** Corresponds to the absence of positive or negative emotions.
Example: Where can I find free insulinpump supplies and insulin?
- **Negative:** Corresponds to negative emotions, such as sadness, sorrow or anger.
Example: I can't sleep, barely functioning, filled with dread and panic. Certainly what I am experiencing can be attributed to stages of grief but I question it has gone beyond that. My

behavior is manic as I either try to process the whole thing or desperately try to control it around others which makes it even worse.

Positive, negative, and neutral correspond to positive, negative sentiments, and neutral messages, respectively.

4.3 Research Questions

We can pose three different research questions, one for each classification task:

- Can we build a first-level classifier with acceptable results?
- Can we build a second-level classifier with acceptable results?
- Can current polarity analysis tools be used to classify the polarity of messages in OHCs? And how do they compare with classifiers built by us?

It is pertinent to clarify what acceptable is. For that, we should consider the results we found in the literature presented in Section 3. For that, we should take into account not only the size of the dataset but also the number of classes of each task and the type of problem (multiclass or multi-label).

4.4 Methodology

This section presents a high-level approach on how we dealt with each task. The overall pipeline is the same for the first and second-level categorization of messages. On the other hand, we also explore existing tools for polarity analysis.

4.4.1 First and Second Level Categorization of Messages

To determine the ideal preprocessing combination, many preprocessing procedures were tested. Performance comparisons were made between frequency-based and grouped embedding techniques (using mean and TF-IDF mean). We investigate the performance of traditional machine learning models and neural networks as classifiers using our dataset. Additionally, we experimented with feature engineering and Multi-label Synthetic Minority Oversampling Technique (MLSMOTE) for data augmentation. The overall pipeline is shown in Figure 4.4.

4.4.2 Polarity Analysis

In the first approach, we used all the data we have as testing set to verify how well different popular available tools behave. The entire process is shown in Figure 4.5. We tested with both having or not the neutral class.

Next, we created our models to compare its performance with existing approaches. The pipeline used for it similar to what was presented in Figure 4.4.

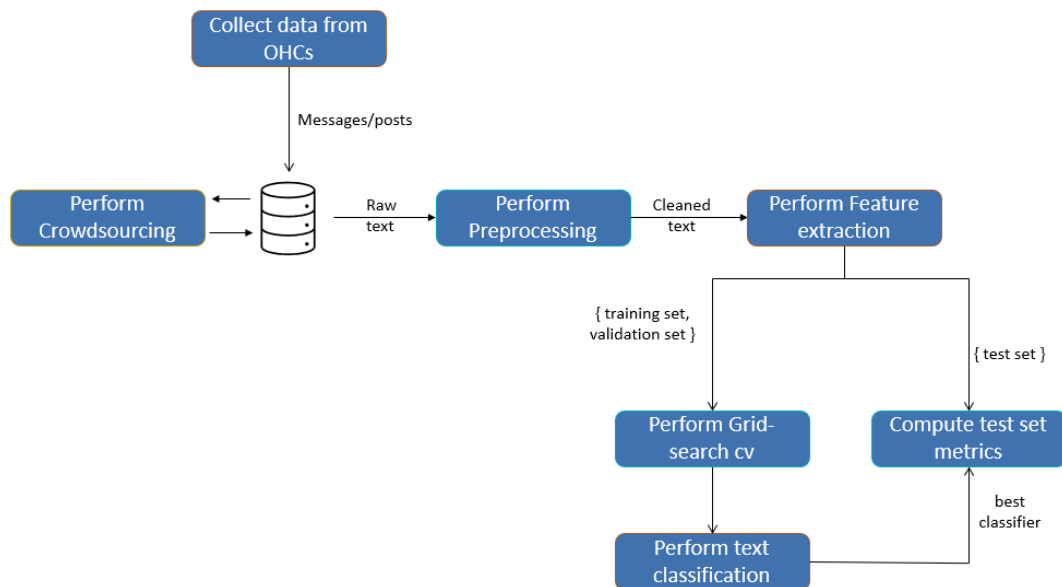


Figure 4.4: Pipeline for classical machine learning

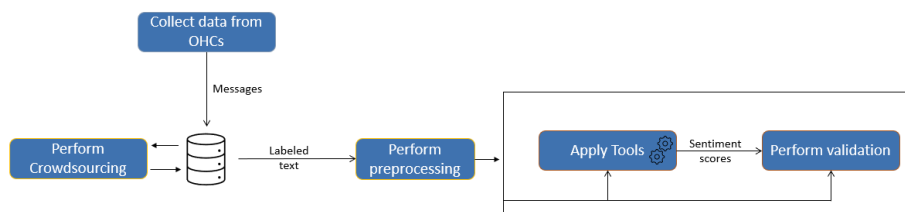


Figure 4.5: Pipeline to test different polarity analysis tools

We built a binary classifier to distinguish between positive and negative instances by leveraging not only universal sentence encoders but also an existent dataset of such samples. That process is depicted in Figure 4.6.

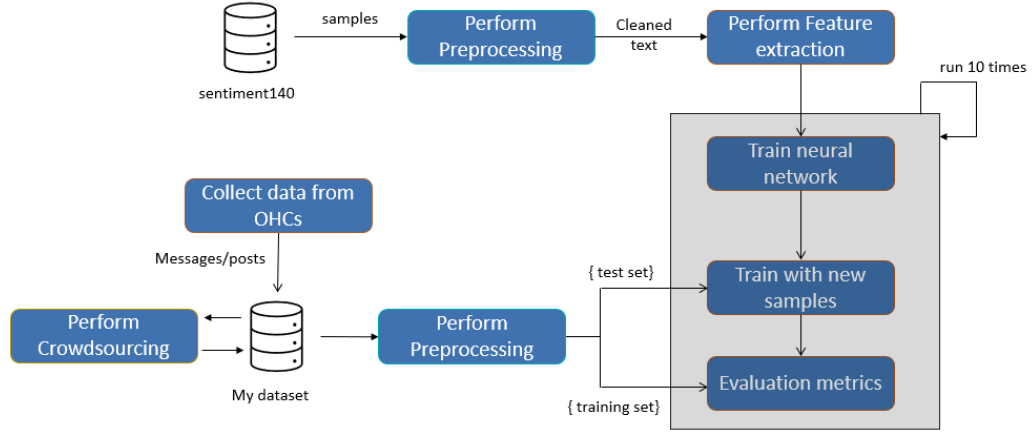


Figure 4.6: Neural network with transfer learning pipeline

4.4.3 Evaluation Metrics

To evaluate the polarity analysis Task, since we are dealing with a multiclass classification problem, Accuracy, Precision, Recall, and the F1-score was used. We also used Precision, Recall, and F1-score to deal with the Multilabel problems. However, the weighted average, i.e., the average per class of each metric, considering each class weight is different and dependent on the number of samples. Moreover, we also utilized the weighted Jaccard Score and Hamming Loss that are more guided towards multilabel classification. Jaccard Score allowed us to gauge the similarity between sample sets by performing the division between the intersection of samples dividing by the union of them. Hamming Loss is the fraction of wrong labels to the total number of labels. All of the metrics mentioned above were implemented in the Scikit-learn Python library.

Chapter 5

Dataset Building and Analysis

In this chapter, we will explain how we built the dataset and performed data annotation using crowdsourcing. Afterward, an analysis of the characteristics of the dataset shall be presented. This new information derived from each text sample can be subsequently used for Feature Engineering.

5.1 Data Collection

The data acquired for this work came from two different time instances. An initial batch of instances came from a previous work [76] whose focus was to create a generic classification scheme for OHCs taking their content into account. However, since the number of cases collected was limited, we expanded the dataset by acquiring new data samples to be labeled. We decided to extract textual samples from the same website the work mentioned above did, namely MedHelp. There were two significant reasons for that. The first is the data collection policy, and the second one is under the assumption that different websites can have data distributions that are too different. Moreover, in future work, we can expand the dataset to other ones when we have a good performance for this forum. It would probably be useful to study different types of forums as some may be quite similar while others not. We favor this study before choosing the next OHC since data labeling can be costly, and acquiring good data, instead of a redundant one (and hence probably biased towards a particular population), would be extensively more useful to have a more holistic classifier.

The selection of the samples to label is described next. The most popular communities assigned by the website were selected, and up to 100 most recent threads were extracted from each. To do that, web scrapping was conducted using Python and Selenium framework. The data collected from it consisted of the text itself, the user identifier, the date, and even a thread identifier was appended to be able to reconstruct them entirely using it together with the message publication date. The selection of messages to annotate was made by sampling 285 out of the total. The annotations were done from March until the end of May, and all the data was collected on the fifteenth of March. Moreover, two other scripts were created. One uses the Reddit API to extract data from OHCs, and another performs web scrapping out of the DailyStrength's most popular communities.

However, collecting and annotating data from them is also left as Future Work. With the 285 new messages, currently, the dataset has 487 annotated instances.

5.2 Data Labelling and Crowdsourcing Platform

Crowdsourcing was used to leverage many volunteers to aid us in labeling our collected data. For that, a full-stack application was built and deployed, following a MERN (MongoDB, Express, React, Node) stack, presented in Figure 5.1. To deploy the backend, the Heroku platform was used, while the frontend deployment platform used was Netlify.

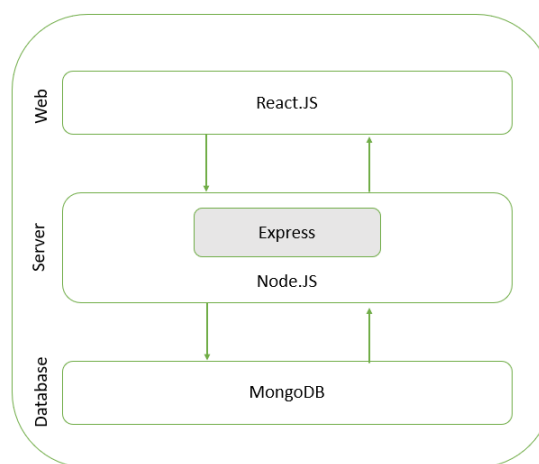


Figure 5.1: Website stack

The website¹ that we built is constituted by three pages. The Instruction page is the home page where users are firstly redirected and has all the information necessary to perform the task. Moreover, we give context about the work and an email to people to contact us in case they have some doubts or problems. The Participation Page is where users perform labeling of the text messages. We apply a threshold of 1,000 words in total so that users do not spend more than 10 minutes labeling (considering that the average English reader reads on average 238 words per minute for non-fiction text). After each message is labeled, users can submit it to be saved in the Database. In it, we also maintained a counter per instance corresponding to the number of annotations a message has. A text instance is only retrieved for labeling if they have less than three annotations. Finally, there is a Thanks Page where users can be redirected again to the Participation Page in case they want to collaborate again. The sitemap of the Website built is shown in Figure 5.2.

The Instruction page is shown in Figure 5.3.

In Figure 5.4, the Participation (Home) page is exhibit and in Figure 5.5, the Thanks Page.

¹<https://health-forum-label.netlify.app/>

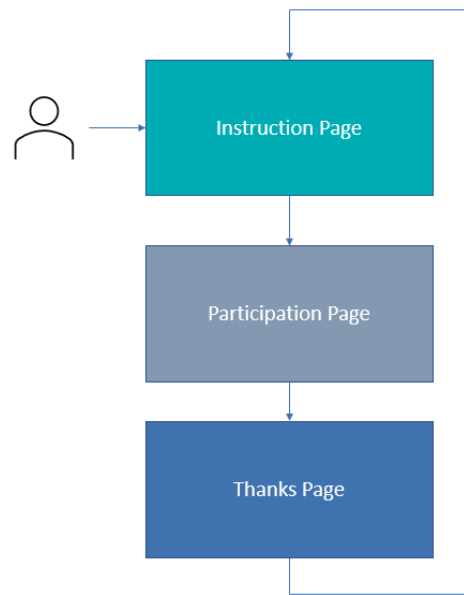


Figure 5.2: Site Map

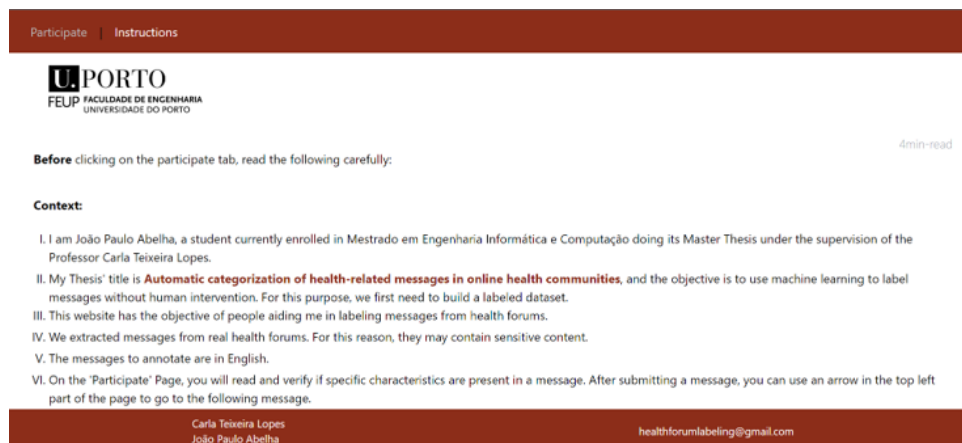


Figure 5.3: Instruction Page

To guarantee output quality, the paper [37] was utilized as a guideline to assess and ensure quality tangentially. Next, we will describe techniques that they describe and that we used to have a good quality output. Data quality is a great deal since we are building data-driven models. The output quality of them is dependent on the quality of their input [66]. The author showed that averaging the guesses of multiple people can contribute to an accurate overall guess [109]. In our work, we followed a voting strategy, using three different voters for labeling each instance. Each voter assesses the presence of a label in a text, and the majority decision for each label for each instance was taken. That representation can be seen in Table 5.6. This way is made to guarantee a better overall quality labeling output.

In addition, we can also filter workers, or, in other words, the people who perform crowdsourcing. This can be relevant to ensure that the labeler has the necessary skills, preparation and

Participate | Instructions

1/5

"Using" two different mathematical scales

What are the emotional polarity present in the text? **?**

Negative ☐ Neutral ☐ Positive ☐

Does the text depict network support? **?** -- select an option --

Does the text depict emotional support? **?** -- select an option --

Does the text depict informational support? **?** -- select an option --

Does the text depict esteem support? **?** -- select an option --

Does the text depict agreement? **?** -- select an option --

Does the text depict disagreement? **?** -- select an option --

Does the text depict gratitude? **?** -- select an option --

Carla Teixeira Lopes
João Paulo Abelha

healthforumlabeling@gmail.com

Figure 5.4: Participation Page

Participate | Instructions

Thanks for
your
contribution!

Contribute again

Carla Teixeira Lopes
João Paulo Abelha

healthforumlabeling@gmail.com

Figure 5.5: Thanks Page

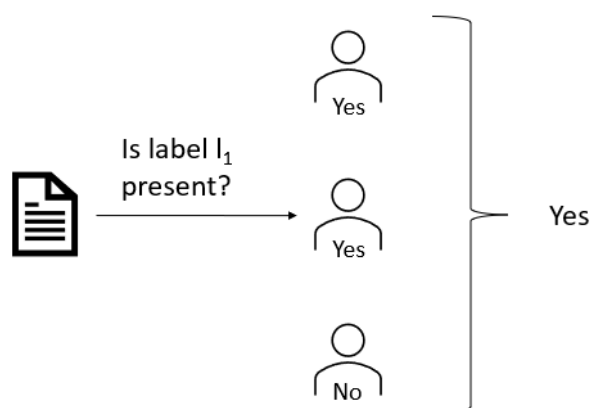


Figure 5.6: Labeling strategy using a majority vote to decide the presence of a label

attitude. There have been many strategies used to filter them in other works. For instance, the authors [7] considered their profiles and skills while others were based on demographics as they

were used as a proxy for skills [64]. Our strategy is more closely related to the last one. The website was shared among the students from the University of Porto and some medical students from the University of Minho. This guarantees that labelers are currently enrolled in a higher education institution; hence, a certain level of skills and capabilities are inferred. Another strategy is to assign workers instead of waiting for random workers to be selected. They claim that doing this can often be more effective. This can also be advantageous if we have information about their skills and capabilities when performing that task. There have also been more intelligent mechanisms of work assignments; for instance, the work [7] assigns stronger (better) workers for more demanding tasks. This strategy was performed on people who were more eager to be more related to the work and could easily contact us and clarify any existing doubts. Since they got more involved with the work, they were also more prone to participate several times. Moreover, intrinsic motivation, namely sharing a purpose, can attract some individuals if they can identify themselves with the work. By explaining the purpose of our thesis and by sharing it with people from the medical field, they revealed additional willingness to participate in the task. Another way to improve quality is to teach workers. This often means giving them the correct and appropriate instructions for them to be able to accomplish their tasks. For that, we give not only meaningful examples but also describe and define each label. Besides having a page for it, users can always consult this in the middle of the task in case they need some clarification, as can see in Figure 5.7.

The screenshot shows a web interface for data annotation. It contains three tasks, each with a question mark icon in a red box:

- Task 1: "What are the emotional polarity present in the text?" followed by radio buttons for "Negative", "Neutral", and "Positive". A "help" button is located between "Neutral" and "Positive".
- Task 2: "Does the text depict network support?" followed by a dropdown menu with the text "-- select an option --".
- Task 3: "Does the text depict emotional support?" followed by a dropdown menu with the text "-- select an option --".

Figure 5.7: Help button in each label during annotation

By clicking on the different questions marks, a pop-up appears explaining the label definitions and giving some examples as we can see in the Figure 5.8.

Another way is to improve task design which can be accomplished by dividing it into several sub-tasks. We divide the labels into understandable checkboxes (often more than the labels but more comprehensible), which are then mapped into the different labels. This is meant to aid people in understanding what they need to identify in the task, even if it is the first time they are performing it. Another aspect done to guarantee high-quality data is the validation of the inputs. This means we should verify if the UI's form fields are in synch with what we expect in the requirements. For that, we provide appropriate feedback. For instance, when the form is empty, the request is not sent, and the users receive a feedback message stating the problem.

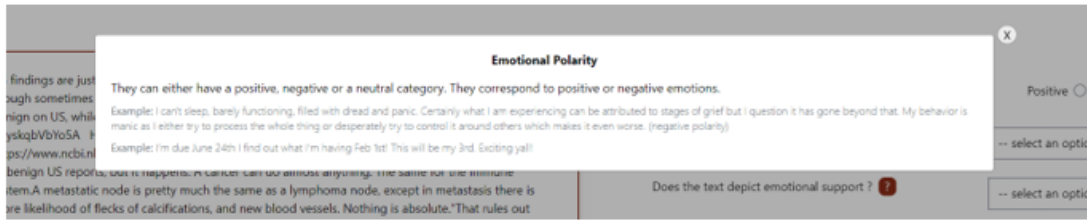


Figure 5.8: Help button Pop up

We can also improve website usability. This consists of following some UI guidelines, which were also mentioned above. Another example would be placing important content with the input fields so that tasks are not as time-consuming and not as memory demanding [4]. For this, we placed the content and labeling section side by side so that users can quickly decide without thinking or using their memory too much.

Overall, the process, from the collection, labeling, and collection of the annotated cases, is summarized in Figure 5.9.

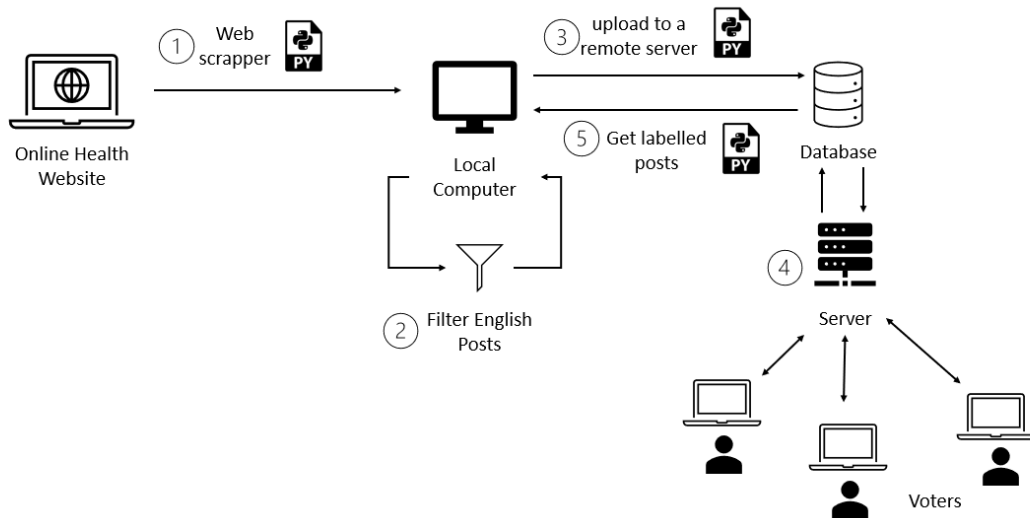


Figure 5.9: Entire process of message annotation

Since this is a novel dataset, there are no other publicly available datasets we can leverage for this work. As a result, we created a dataset with almost 500 labeled messages. Ideally, a larger dataset would be preferred; however, the limited resources for data labeling only allowed us to get this far. Hence, we place, as future work, the continuation of the expansion of the dataset. Next, posts with incomplete content (without labels or with empty texts) were filtered out, leaving 485 messages available for the task. Different python scripts were used to establish a connection between the local dataset and the one hosted on an online server. A relatively conservative number

of cases was chosen, namely 285, to ensure we had those instances labeled. Since we chose three different labels per instance, we required $285 * 3 = 855$ annotations.

5.2.1 Relabel Existing Messages for Polarity Analysis

To accomplish one of the work objectives, namely distinguishing neutral, positive, and negative messages, the messages needed to be relabelled into these categories. In the previous work, they tried to find six specific sentiments more connected to negative and positive sentiments. However, we argue that in the first level, that is, before going to that level of granularity, a high-level distinction is relevant, especially when few data cases annotated are available. Since they did not account for neutral messages, the mapping between what they had and what we were aiming for was not direct, and, hence, extra work was performed. Two different people, including myself, performed the classification independently. Afterward, another person intervened and served as a tiebreaker when a not conclusive decision (label) was achieved. In the end, the three of us reviewed those labels, and an agreement was reached.

5.2.2 Level of Agreement of the New Instances

For the different granularity levels, we can plot the frequency of votes they received to reach an agreement for each label. The labels not related to polarities only have two options: either they have two or three votes. We have another level of agreement for the polarity analysis labels, namely when three different people voted differently. This happens in this setting since sentiment polarity has three options while having three people voting. In Figure 5.10, the level of agreement related to the hierarchical scheme is shown while in Figure 5.11, we show that for sentiment polarity.

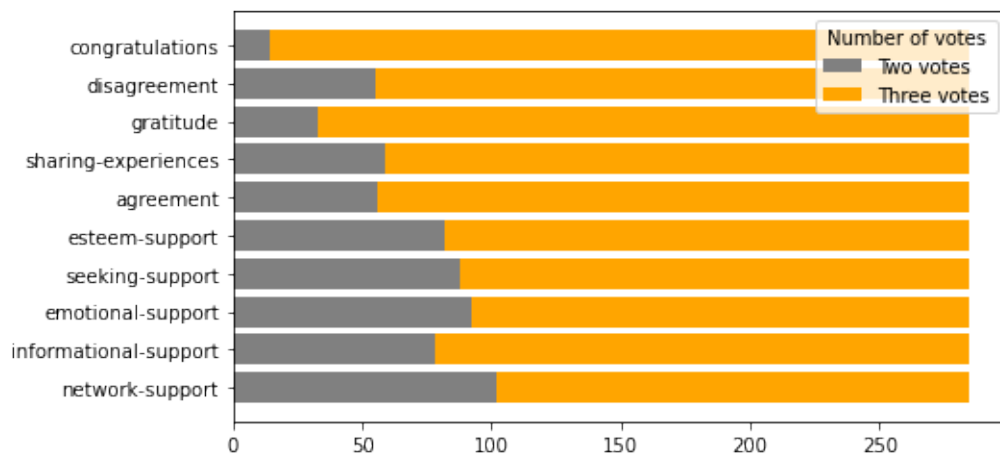


Figure 5.10: Multilabel classifier votes' distribution

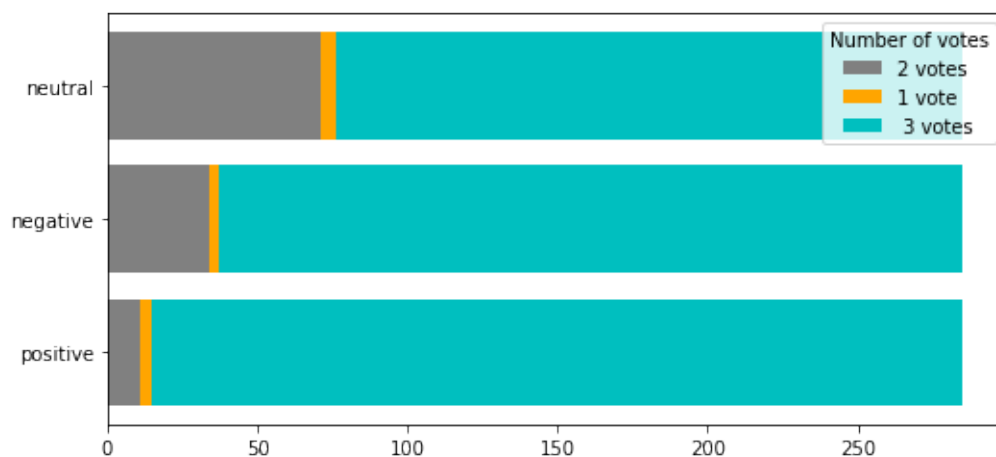


Figure 5.11: Sentiment polarity votes' distribution

5.3 Dataset Characterization

First, it is quite relevant to verify how imbalanced a dataset is and whose classes tend to go together. Moreover, we can also verify how common instances with several labels are in the dataset.

5.3.1 High-level Categories

To check the class distributions and how much each label appears with others, we can plot an upset diagram as shown in Figure 5.12.

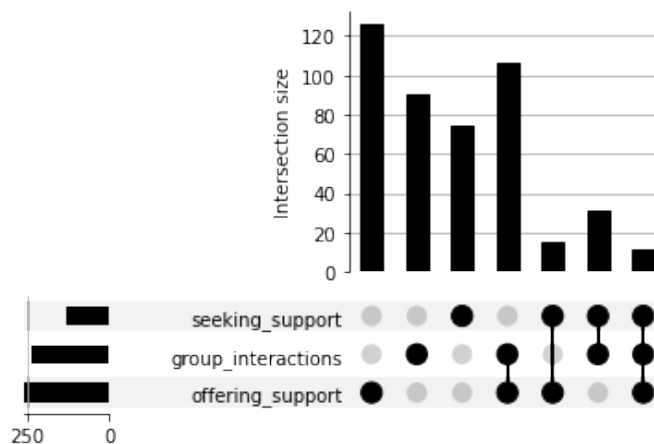


Figure 5.12: Upset diagram of first-level classes

Regarding offering support and group interactions, most of their instances are single-label. On the other hand, seeking support appears more together with group interactions than with only itself. All the other label combinations are rare, accounting for around 12.6% of the cases. The percentage of messages with one single label is around 64%.

We can verify if there is a correlation between the length of the message and the number of labels associated with this. This can make sense since the more we write, the higher likelihood we are of transmitting several ideas. We calculate the Pearson Correlation along with the two-tailed p -value, and we get a correlation coefficient of 27.7% with a significance level of 0.05. We obtain a p -value lower than 0.05. Therefore, we can conclude that there is a positive correlation between those two variables.

5.3.2 Second-level Categories

Since our dataset has 61 different label combinations for this current task, we will show in Figure 5.13 only the combinations with more than one instance. Since doing this, we are excluding around 6.4% of the total instances.

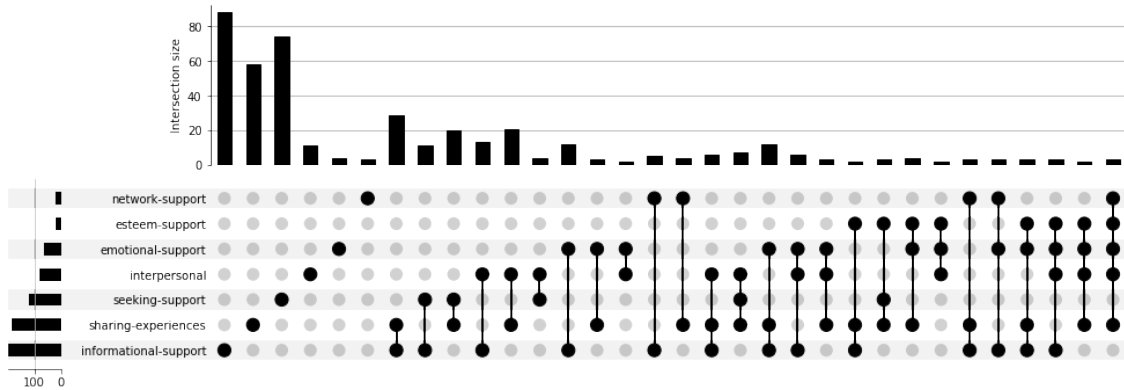


Figure 5.13: Upset diagram of the second-level classes

We can easily verify that some classes have a small amount of data. Most instances with a single label belong to informational support, sharing experiences, and seeking support. In Figure 5.14, we plot the frequency of each category, i.e., the class distribution, and we verified that even though interpersonal and emotional support does not appear exclusively in many messages, they do appear a lot together with other labels. Network support and esteem support are the least frequent, appearing in 37 (around 8.2%) and 34 posts (around 7.5%), respectively.

5.3.3 Polarity Analysis

Since we are dealing with a multiclass problem, it is easier to understand the class distribution, as shown in Figure 5.15.

From it, we can observe that neutral classes are the most common ones (around 59.5%), while negative messages (around 29.5%) are more prevalent than positives (around 11%).

5.3.4 Text Statistics

First, we can start by exploring the data's length (number of characters) distribution.

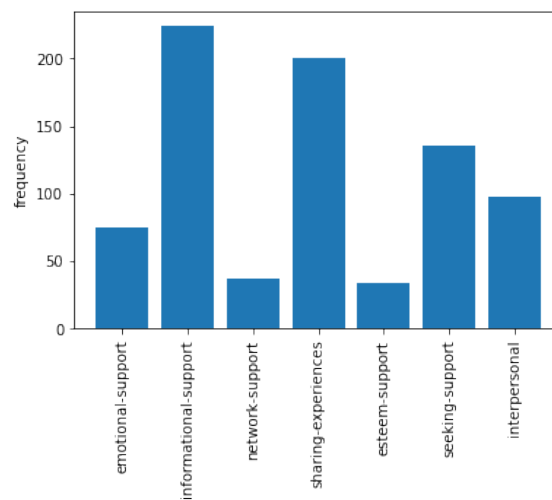


Figure 5.14: Classes distribution of the second-level of the hierarchical scheme

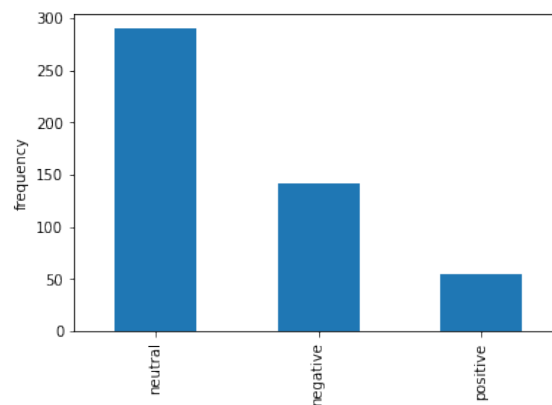


Figure 5.15: Polarity distribution

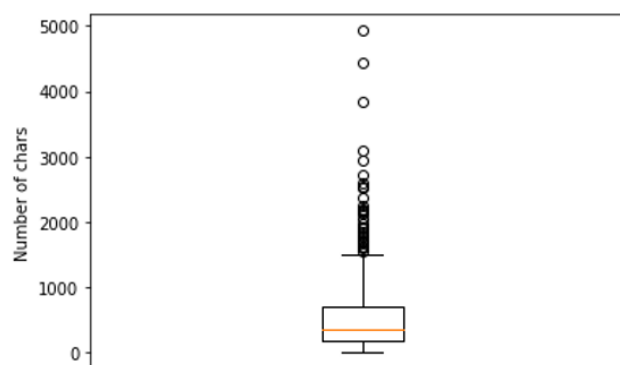


Figure 5.16: Length distribution of the textual data

By observing Figure 5.16, we verify that most values are within the 1500-character length, with slightly more than 92% of the instances. We can also observe a few rare samples whose length

is significantly greater than the others. Using the interquartile rule, we were able to conclude that there are 31 outliers regarding the length. This value corresponds to around 6.8% of the number of samples. We can also compute the length of a message but at a word-level length.

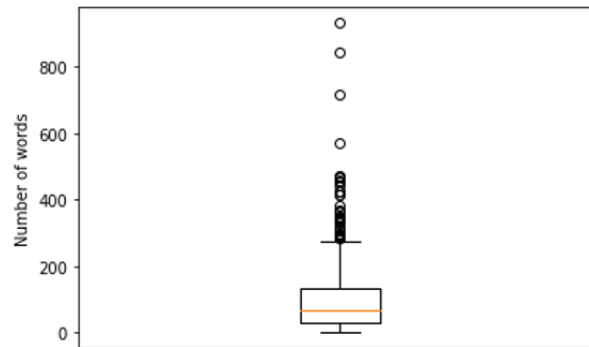


Figure 5.17: Number of words distribution of the textual data

As we can see in Figure 5.17, there are a few outliers that correspond to quite large messages and most likely correspond to those whose number of characters is significant, as seen in the previous Figure 5.16. We can see that most messages have below 200 words (almost 87% of them), and around 65% of them have less than 100 words. Using the interquartile rule again, we found that 30 cases are considered outliers due to their large number of words. After comparing the outlier instances from the length and number of word distributions, we discovered that 27 samples are shared between them. Moreover, both distributions are visually similar, so the previous observation was more and less expected.

Additionally, we can also explore the text distribution in terms of sentences. We used the library NLTK to perform sentence tokenization for that purpose. Figure 5.18 shows the number of sentence distributions.

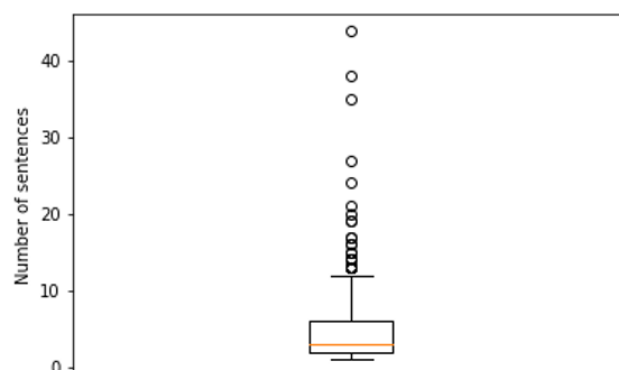


Figure 5.18: Number of sentences distribution

By analyzing it, we can see that most posts have less than ten sentences (around 90%). We can also observe that there are a couple of instances with a high number of sentences. We examine the existence of outliers by performing the interquartile range rule, and 37 posts fall within that category (around 8.6%).

Chapter 6

Data Preprocessing and Text Vectorization

First, we will present all the possible preprocessing steps we can perform on the raw text. Then, we will relate each preprocessing step to the different feature vectorization techniques. Finally, we showcase how we chose the preprocessing steps we took for each different classification pipeline we have.

6.1 Data Preprocessing

Textual data is considered unstructured data due to its non-standard representation. To tackle some of the issues related to its nature, a sequence of steps to normalize the texts are presented.

6.1.1 Remove HTML Tags

After the posts are collected from the web and before they are made public for data annotation, we perform HTML tag stripping in case they come with unwanted HTML tags. Since the text content is web scrapped, there are some chances that it can still contain some HTML tags. To accomplish this task, we use BeautifulSoup's HTML parser as it allows us to quickly get rid of them.

6.1.2 Dealing with Certain Patterns

We use regular expressions to remove and replace specific textual patterns as they do not contribute much to the classification by themselves as they would correspond simply to noise. We use rule-based matching combined with NER to extract the drug names and corresponding dosages. This is replaced by drug_usage token, but, in reality, it could also be used to verify if the dosages being administrated are correct if we wanted to perform knowledge extraction. In addition, we use a library called scispacy to replace chemicals and diseases with two different tags. This way, instead of needing a lot of examples for the classifier to understand that a specific word is a disease,

that problem is overcome by making this replacement. For this purpose, we use a model called `en_ner_bc5cdr_md`, which was trained on the BC5CDR corpus [72].

Moreover, we also used NER and replaced them with the entity they represent. The idea is quite similar to the one before, where instead of having multiple hours of the day in the middle of the message (for example, "10:00"), now we have a tag. For this, we use the NER implemented in `spaCy` and their corresponding tags as the replacement tokens.

Finally, we remove extra spaces, extra apostrophes, URLs, phone numbers, and names with digits, hashtags, and possible mentions. Moreover, we replace emoticons with a one-word equivalent in meaning.

6.1.3 Slang Expansion

Due to the nature of the generated text, abbreviations can be present, which can affect our classifier performance. Moreover, pre-trained models, which generally are trained with well-written text, would most likely discard these as they would be recognized as OOV. The file used to get the different acronyms was found on GitHub¹. Even though the file is limited in terms of the slang, its number of false-positive substitutions is lower compared with a more extensive list, such as that one found in `netlingo`². For example, "20" can mean "Location"; however, if someone's simply telling its age, changing the location would create a wrong substitution. Due to this reason, we use the more conservative list. Nevertheless, we experimented with the other list and added an abbreviation to the conservative list since we are sure they represent them, namely "IDK". The final dictionary is constituted of 68 common abbreviations.

6.1.4 Dealing with Contractions

English text, especially informal text, is full of contractions. The expansion of contractions can be seen as an extra normalization step. First, computers do not know that they represent a sequence of words; for example, "she is" and "she's" are considered distinct. Secondly, the document-term matrix increases when we do not deal with contractions. In the first phase, a dictionary that maps the contractions to their corresponding forms was used to implement the library contractions. However, this approach does not take the ambiguity of the contractions into account. For example, "ain't" can mean "am not", "are not", "is not", "has not", and "have not". Not considering this can lead to an error increase and reduction of the quality of our data. Only with the context would we be able to correctly predict what should be written. The library `pycontractions` take this into account. When there is no ambiguity, the contractions are simply expanded as they would in the other package mentioned above. When there is ambiguity, it generates all the possible options. They are all passed through a spell checker, and the Word Mover's Distance between the original sentence and each new one is calculated. The grammatically incorrect options are simply dropped, and if there is more than one option left, the one with the highest Word Mover's Distance

¹https://github.com/rishabhverma17/sms_slang_translator/blob/master/slang.txt

²<https://www.netlingo.com/acronyms.php>

Score is selected. This metric is the cumulative weighted cost required to move all words from the original text to each different hypothesis. The closest hypothesis is the minimum Euclidian distance between the contraction and expansion word pair in the embedding space. Comparing both packages, we could conclude that while pycontractions are slower, it is also more accurate, and if time is not crucial, it should be preferred.

6.1.5 Spell Correction

In real words, text generated by individuals without too much care for typos can produce very noisy text. That said, there is a need for an automatic spell corrector that changes what is wrong and does it without changing what is already correct. Before correcting, I analyzed which words were marked as not being English and concluded that some of them were health-related, such as names of medicines.

There is a paper [86] which compares 14 different spelling correction tools. Balancing the performance and considering we wanted an always-free-to-use spell corrector, we decided to go with JamSpell. It only stays behind paid versions or versions with limited API calls, such as Google Spell Checker API. Additionally, it is quite fast as it can process up to 5,000 words per second. It was trained on 300k Wikipedia sentences and 300k news sentences. Out of the total training data, 95% of it was used for training and 5% for testing (30k sentences). On their Github Page³, they also compare their tool against Hunspell and Norvig, which the above paper does not, and it shows its higher performance.

6.1.6 Tokenization

For this step, we are using the Python library named NLTK. To perform word tokenization, we use wordpunct tokenizer as it takes into account punctuation and emoticons. Consequently, this is preferred to a simple white space tokenization strategy. In terms of sentence tokenization, we are using the sentence tokenization that NLTK also provides.

6.1.7 Removal of Stop Words

Stop words correspond to the most common words in a language and often do not add much additional information to the text. Examples of stop words are “a”, “the”, and “so”. As they are abundant, when we remove them, we give more importance to other words and focus on more relevant information. When doing polarity analysis, we have to be especially careful if we remove any with a negative connotation, such as “not”. So, removing them may not always be the best option (or removing them mindlessly). Some libraries maintain a different list of stopwords. The most popular ones are NLTK and spaCy lists. However, by conducting some empirical testing, we concluded that simply removing them worsened the performance. Hence, we decide to consider different stop-word lists (each containing an integer identifier to ease explanation): the NTLK

³<https://github.com/bakwc/JamSpell>

default one (1), NLTK without the negation words (2), the NLTK words whose TF-IDF values are zero (3), and finally the NLTK without the negation words and without the words whose TF-IDF values are greater than zero (4).

6.1.8 Lowercasing

This step can be more worrisome when it comes to sentiment analysis. However, if it does not hurt performance, it is preferred since our dataset is small and, therefore, performing lowercasing helped reduce the size of vocabulary and possibly enhance the classifier performance.

6.1.9 Stemming/Lemmatizing

This task corresponds to assigning a syntactic category (e.g., noun, verb) to each word or token of the vocabulary. Part-of-speech is relevant for lemmatization to work because there are words whose meaning changes depending on the part of the speech. For example, we may not get the correct lemma for a verb if we do not specify explicitly that a word is a verb. However, we do not need to train a PoS tagger because NLTK includes one trained on a Treebank corpus. We may prefer this to a stemming step since the latter can produce a root form that may not exist in the English dictionary. A famous stemmer that we used to compare its performance with lemmatization is the Porter stemmer. Both stemmers and lemmatizers are a part of a normalization process and can usually condensate the vocabulary into a smaller one. However, there might be a case where, for example, different verb tenses can impact the classification task. An example of why Pos-tagging is important before lemmatization is exemplified in Figure 6.1.

```
print(lm.lemmatize("Books", pos="n"))  
Output: 'Books'  
  
print(lm.lemmatize("books", pos="v"))  
Output: 'book'
```

Figure 6.1: PoS-tagging when doing lemmatization

6.1.10 Punctuation

To deal with punctuation, we decided to try both maintaining them or removing them to verify if they impact the final model since some punctuation signs may be more associated with some classes than others. They are also vital if we want to perform sentence tokenization.

6.2 Text Vectorization

We tried several without changing the classifier to understand the impact of the different vectorization techniques. That way, the independent step in the whole pipeline is the text vectorization

step. By doing so, we can verify which one seems better depending on the classification setting inherent in the task. We tried three different methods: BOW, TF-IDF, and pre-trained embeddings (words and sentences).

6.2.1 Frequency-based Feature Extraction

When it simply comes to BOW and TF-IDF vectorizations, there are some variations we can try out, such as using words or chars as tokens. We can also experiment with different n-grams ranges at a larger vocabulary expense. These text transformations were done using a Python library named Scikit-learn. Notice that using at least 2-grams char tokens will make the algorithm learn word substructures rather than words. The parameters we used for BOW and TF-IDF are described in Table 6.2.

Count vectorizer	analyzer: {'char', 'char_wb'}
	n-gram range : { (1,1), (1,2), (1,3)}
TF-IDF	use idf: {True, False}
	norm: {'l1', 'l2'}

Table 6.1: Parameter set for Bow and TF-IDF vectorization

The only difference between BOW and TF-IDF vectorization lies solely in the "use idf" parameter. If it is set to false, it corresponds to BOW; otherwise, it is TF-IDF.

6.2.2 Word Embeddings

We can convert the word tokens into a vector of word embeddings using pre-trained word models, such as GloVe. Word embeddings are vectors whose representation meaning is based on word distribution, and therefore, words that appear in similar contexts have similar vectors. We can represent a group of words, a phrase, sentence, or whole text, using them by performing some operations on the word vectors. We experiment using the unweighted average of them, and we try to associate each word embedding with the weight the TF-IDF vector would give each word. Using this grouping technique in a single vector has been shown to work well in the past works [11]. The intuition is that the resulting vector will include information over all the components so that each component will pull the vector in a particular direction. This works well because of the characteristics of high-dimensional spaces since, in them, many embeddings are practically orthogonal among themselves, so it is difficult for two collections to have a similar average value. On the other hand, similar documents will have similar averages.

Besides GloVe, we experimented Word2Vec and FastText. Additionally, for GloVe, we tested with embeddings of different sizes to see which one performs the best. The reason for aggregating the values is because the feature space would get too big compared to the number of samples, which can lead to the curse of dimensionality. This way, we can maintain the feature space at the same size as each feature vector. Moreover, we would need to decide on a maximum length for

the sequence of words we can use. If we use too many words, their dimensional space will be enormous; if we use too few, we can lose information. An acceptable measure can be the median or average, but since our dataset is small, even those are deemed too large. Compared to BOW or TF-IDF, the entire feature space would be n times larger, with n the size of the word embedding, which is commonly at least 50.

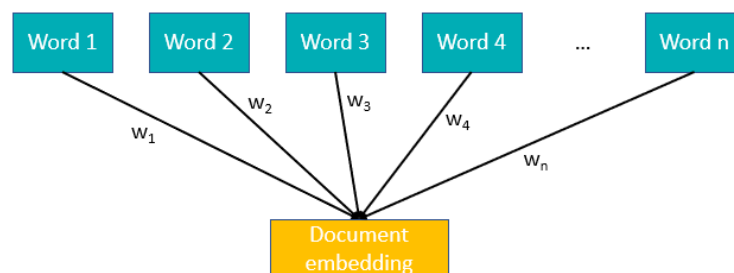


Figure 6.2: Grouping word embeddings to create document embeddings

The reason for choosing these three word-embedding models is because they represent three different ways of looking at word embeddings. Word2Vec takes textual data as training data fed into a small neural network. Word2Vec learns the embeddings by relating the words to their context. Nevertheless, the frequency of context words is ignored. Therefore, the embeddings capture whether words appear in similar contexts. However, for GloVe, the information about those occurrences is relevant. Hence, word embeddings are built so that a combination of word vectors is related directly to the probability of the words' co-occurrence in the training data. FastText uses as its building stones chars instead of words. With this, it is possible to avoid OOV as long as the words in the testing set have the same characters as those in the training set. Moreover, less information is needed because more information can be collected from textual data. To access these pre-trained models, we used a Python library named Gensim.

Since we have a small number of instances, we only use pre-trained embeddings that can generalize reasonably well.

6.2.3 Sentence Embeddings

One can argue that instead of grouping them for the whole text, we can do it sentence-wise. However, for that, we used a Universal Sentence Encoder developed by Google. The encoder is available at the TensorFlow hub. It can encode a variable-size text into a fixed-length vector (512), taking into account that the longer the text, the more diluted it becomes. As input to deep learning models, such as LSTMs, we can either determine a fixed number and truncate and padding as necessary, or we can use ragged tensors available at Keras since they allow having different-size inputs. We used these last ones as there is no need to truncate or pad the data. Moreover, inspired by the averaging of word embeddings, we performed the same for sentence embeddings rather than using the entire text as input to verify if there is a performance boost.

The pre-trained model embeddings we experimented can be summarized in the Table 6.2.

Model Name	Context sensitive	Learnt Representation	Dimension	Trained on
Word2Vec	No	Words	{300}	Google news
Glove-twitter	No	Words	{50,200}	Twitter
FastText	No	Words	{300}	Wikipedia
Universal sentence encoder	-	Sentence	{512}	Wikipedia, web news, Q&A pages, discussion forums

Table 6.2: Pre-trained embeddings

6.2.4 Feature Space Size

Regarding the feature space, if we consider each word as a token (feature), then we can say the following about the size of the feature space:

- BOW or TFIDF-vectorization: number of unique tokens
- Word embedding grouping: word embedding size
- Word embedding: Number of words of the sequence * word embedding size
- Universal sentence encoder (whole text): sentence encoder size
- Universal sentence encoder (per phrase): sentence encoder size * number of sentences of the sequence

6.2.5 Relation Between Preprocessing, Text Vectorization and Experiments

The processing steps should depend on the text vectorization technique we utilize. Comparing the word embedding with, for example, BOW, the only difference is that we do not replace words with patterns that would create OOV. Moreover, we should not use stemming since pre-trained models need to recognize existing words. Furthermore, in the case of OOV appearing, we ignore them. We also removed stop words from the samples, as they should not provide meaningful information to the final vector.

In terms of the sentence embeddings, the authors assert that it should not be necessary to perform any preprocessing to use them and, as we saw in Table 6.2, it was also trained on discussion forums.

For frequency-based approaches and word embedding, except for particular patterns, our steps are presented in Figure 6.3. The metrics shown at the end of the steps correspond to the situation we replace all the patterns.

A textual example of it can be seen in Figure 6.4.

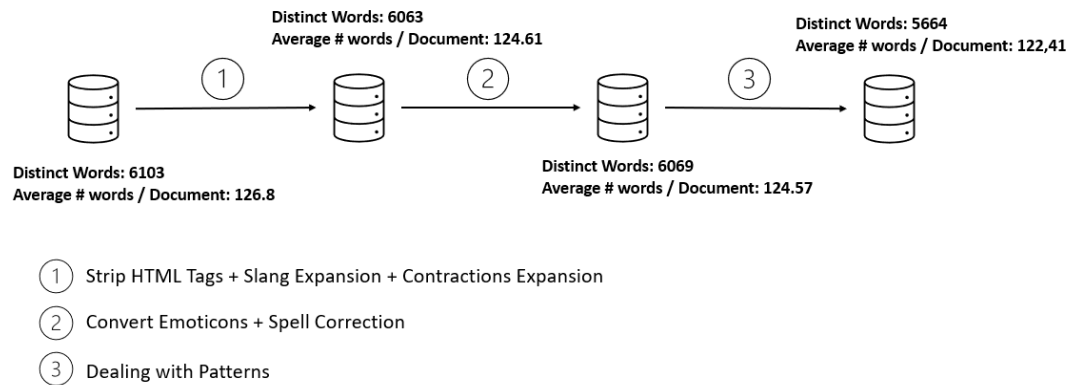


Figure 6.3: Pre-processing steps done for all the tasks

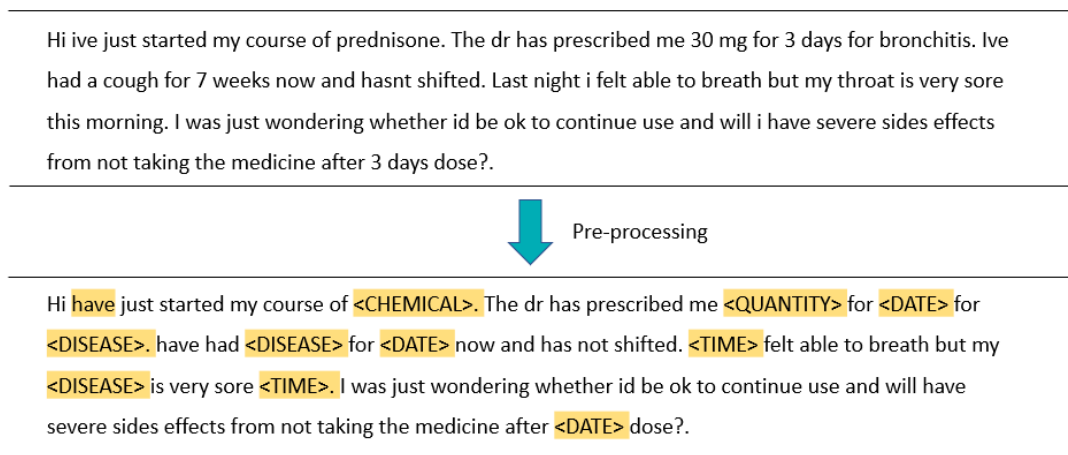


Figure 6.4: Pre-processing example

Starting with the preprocessed text after the steps described above, we utilize Classical machine learning classifiers and the TF-IDF vectorization technique to select the best preprocessing steps. Next, we choose the most appropriate ones via the mean and standard deviation of the F1 and Jaccard scores. The classical machine learning algorithms used were Decision Trees, Random Forest, SVM, KNN, Logistic Regression, and Gaussian NB in their default configuration, except for Logistic Regression, Random Forest, Decision Tree, and SVM, which have a parameter related to the class weighted, namely "balanced" which was passed. Moreover, since only Random Forest, Decision Trees, and KNN can perform multilabel and multiclass classification inherently, the others were run in a one vs. rest configuration, meaning that n binary classifiers were created to distinguish between the presence or absence of n classes.

In Table 6.3, we present some interesting statistics regarding their impact on some textual statistics. As we can see, lemmatization and stemming reduce the number of distinct words substantially while removing punctuation and some stop word lists reduce the average number of

words per document noticeably.

	Distinct words	Average # words / Document	Overall # words / sentence
Pre-processed text	5,664	122.410	19.997
Lowercased pre-processed text	5,068	122.410	19.997
Pre-processed text without punctuation	5,752	99.761	99.761
Pre-processed text with Porter Stemmer	4,574	122.270	20.110
Pre-processed text with WordNet Lemmatizer	4,794	124.690	19.940
stop word list (1)	5,533	76.140	8.440
stop word list (2)	5,539	77.780	8.550
stop word list (3)	5,591	116.400	11.550
stop word list (4)	5,594	116.850	11.670

Table 6.3: Impact of preprocessing steps on some textual statistics

The combination of these decisions was done to find patterns and the overall best preprocessing sequence. We will show the best combinations and some poor ones to explain our decision regarding the preprocessing steps.

Chapter 7

First-level Categorization

This section will explore different ways of performing multi-label classification of the first-level categories. We will show all the experiments we conducted, and we will discuss the results.

7.1 Procedure Steps

First, we performed data splitting into training and test sets, and we preprocessed the text. Then, we determined if the training set is representative of the test set by performing adversarial validation. Afterward, we conducted a series of experiments trying out different text vectorization techniques in conjunction with various machine learning models. To the best models, we also applied MLSMOTE to increase the dataset size, and feature engineering was used in an attempt to enhance the classifier’s capability.

7.1.1 Dataset Splitting

An appropriate way to divide a multilabel dataset into training and test sets has already been proposed in a previous work [103]. Single-label methods used to stratify data fail to create balanced dataset partitions, which can lead to classifiers not being able to generalize information properly. Therefore, we use an iterative stratification method whose objective is to provide a good balance distribution of label relationships up to a given order. By doing so, the label combinations proportions between the test and training set are as expected. The algorithm starts by calculating the desired number of examples at each subset, considering the proportion of the split. It also computes the desired number of each label for each subset. The iterative algorithm analyzes, at each iteration, the label with the fewest examples. The reason for doing this is that if these labels are not treated with priority, they may end up distributed in a not desired way. However, for more frequent samples, they can be later redistributed. Afterward, each sample needs to be put in a subset. The criteria start by following a greedy algorithm, namely the set with the highest deviation to the desired number of samples for the label at each set. If there is a tie between sets, another greedy algorithm acts as a tiebreaker: the subset whose proportions, regardless of labels, deviates the most is selected. Any more ties are solved randomly. Then, that example is removed from

the initial dataset, and the algorithm continues until the dataset is empty. After each iteration, the number of desired examples for each label of this example is decremented in that specific subset. This algorithm is implemented in a Python library called iterative-stratification.

The dataset was partitioned into a training and test size for our experiments using an 80:20 ratio.

7.1.2 Adversarial Validation

This subsection explains the algorithm used to guarantee that the training data is representative of the data we want to use it on (represented by the test data). There is a technique called Adversarial Validation used to verify if the training and test sets are similar or not. The algorithm consists of marking the training data as one and the testing data as zero, shuffling them, and then verifying if a binary classifier can correctly identify each. If it cannot correctly classify them, that is, it obtained an area under the ROC curve below 0.5, then they are indistinguishable; hence, we do not have a problem. On the other hand, if we can classify them, we may have a problem with the entire dataset or some of the features, which can be from different distributions in the training and test set. If there is a problem, we can check the more improper features. If the ROC contribution is especially high for a feature, removing it from the dataset is probably better.

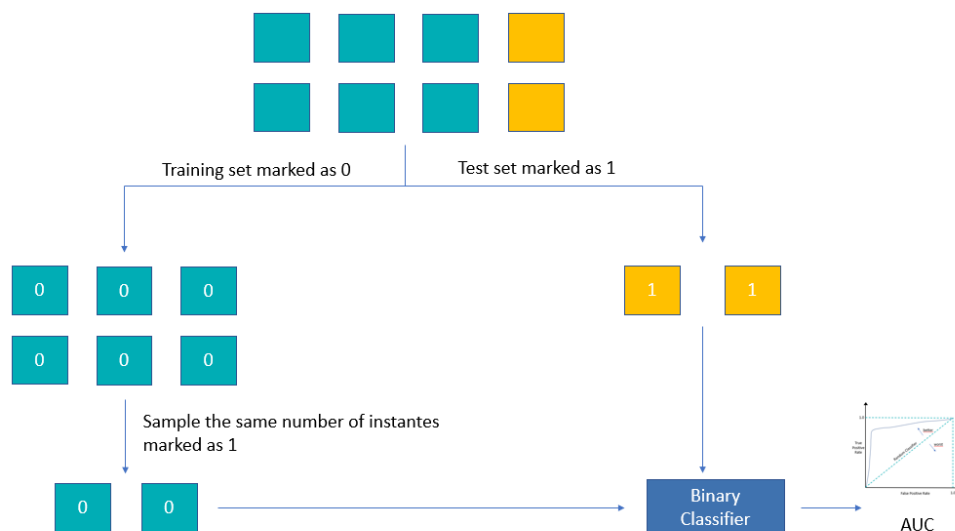


Figure 7.1: Adversarial Validation

Making sure we do not have this problem is relevant since our dataset is small, and possibly different sub-distributions can exist. In an attempt to verify this, we split the data ten different times, and we calculate the AUC. The binary algorithms used were SVM and a Random Forest classifier. We present the mean and standard deviation of those ten runs under each classifier in the experiments for each one. In addition, this was done for three different types of vectorization that we used, namely BOW, TF-IDF vectorization, and pre-trained embeddings.

7.1.3 Machine Learning Models

In this section, we present the different types of models we conducted our experiments with.

7.1.3.1 Classical Machine learning

To classify our data, we experimented with several classical machine learning algorithms: Naïve Bayes, Support Vector Machines, Random Forests, Decision Trees, and Logistic Regression. As seen in the literature, they have been used to classify texts from OHCs successfully, and their ability can be attributed to the fact that they can work quite well with textual data. We used them with frequency-based feature extraction techniques and pre-trained (words and sentence) embeddings.

The whole process can be summarized in the Image 7.2. We can see that the best classifier and text vectorization selection is based on the weighted F1-score.

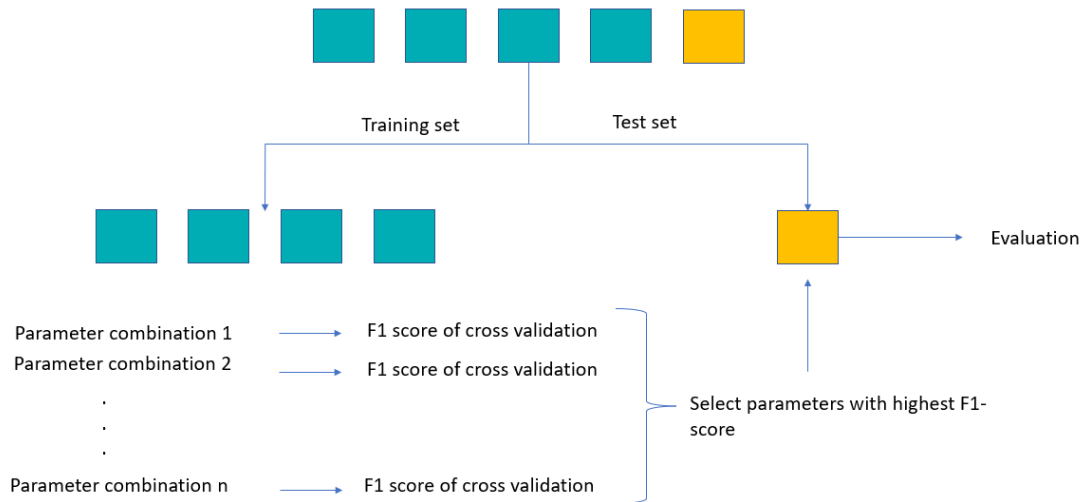


Figure 7.2: Polarity analysis pipeline

To perform this, we use a Python library called Scikit-learn and build different pipelines. For the training part, we performed grid search cross-validation with a cross-validation of five. We varied both the text vectorization arguments as well as the classifiers'. Afterward, we select the best classifier and corresponding text vectorization and apply it to the test set.

In Figure 7.3, we can see which algorithms directly support multilabel and multiclass classification. We use binary relevance, label powerset, or classifier chains for the algorithms that cannot deal directly with a multiclass or multilabel dataset. These last two take into account the labels' correlation. Hyperparameter tuning ensures we achieve the best prediction given the input. Table 7.1 presents the specific list of parameters for each classifier.

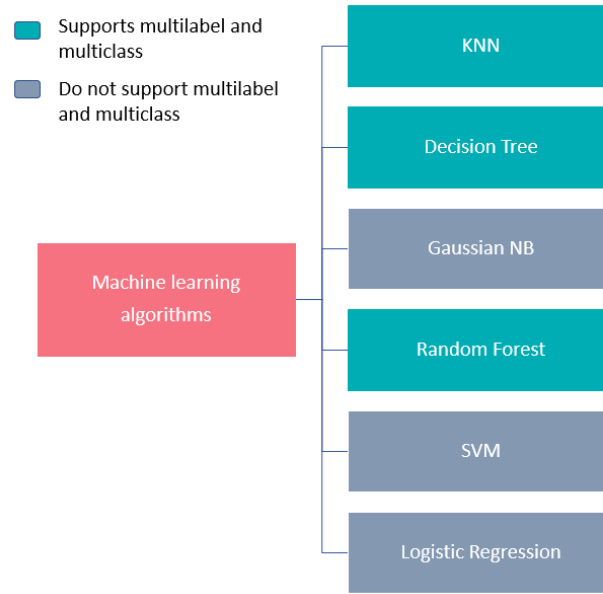


Figure 7.3: Tasks different ML algorithms support

{Classifier}	{Parameter set}
Decision Tree	'criterion' $\in \{ 'gini', 'entropy', 'log_loss' \}$
	'class weight' $\in \{ 'balanced' \}$
	'ccp alpha' $\in \{ 0, 0.005, 0.01, 0.02 \}$
Gaussian NB	'var smoothing' $\in \{ 1e-9, 1e-8, 1e-10 \}$
K Neighbors	'n neighbors' $\in \{ 3, 5, 10 \}$
	'weights' $\in \{ 'uniform', 'distance' \}$
	'p' $\{ 1, 2 \}$
SVM	'penalty' $\in \{ 'l2', 'l1' \}$
	'loss' $\in \{ 'hinge', 'squaredhinge' \}$
	'kernel' $\in \{ 'linear', 'rbf', 'sigmoid' \}$
	'C' $\in \{ 0.1, 1.0, 10, 100 \}$
	'class weight' $\in \{ balanced \}$
Logistic Regression	'penalty' $\in \{ 'l1', 'l2' \}$
	'C' $\in \{ 0.1, 1.0, 10, 100 \}$
	'class weight' $\in \{ 'balanced' \}$
	'solver' $\in \{ 'liblinear' \}$
Random Forest	'bootstrap' $\in \{ True, False \}$
	'max depth' $\in \{ 10, 40, 50, 60, 100, None \}$
	'n estimators' $\{ 200, 400, 600, 800, 1000 \}$

Table 7.1: Classical machine learning and their varying parameters

7.1.3.2 Neural Networks with Threshold moving

A fully connected neural network with one hidden layer was architected to perform multilabel classification. We are using as the loss function the binary cross-entropy, and the output layer

activation function is sigmoid.

The hidden size layer is the parameters related to the architecture of the neural network we are varying. Moreover, the input layer size can change depending on the size of the pre-trained embedding. The parameters set we are changing are presented in Table 7.2.

Classifier	Parameter set
Neural Networks	'hidden layer size' $\in \{16, 32, 64, 128, 256\}$

Table 7.2: Neural network settings

A major concern with this approach is the number of connections between the input layer and the first layer since the feature space can easily become massive. The number of connections is the size of the *feature space * number of neurons of the first layer*. When using this approach, using word embeddings without any grouping is not conceivable since the number of inputs would be *word embedding size * number of words * number of neurons of first layer*. But by grouping, we convert the number-of-words parameter to one, leaving the formula as *embedding size * number of neurons of first layer*. For this reason, we are not using words as input features (that includes frequency-based feature extraction techniques).

Next, we calculate the threshold that maximizes the F1-score, i.e., the value that separates the presence or absence of the class. So, to further improve the score, we can tune the threshold. By default, its value is 0.5. We can try different values by, for instance, threshold moving within a range. There are several reasons why using the default one may not be the most adequate, such as the class distribution being skewed, hence the relevancy of the need to change the threshold when interpreting predictions. The thresholds we tried started at 0.05 until 0.95 with a step of 0.05.

On the one hand, higher dimensionality offers a more precise differentiation of vectors than smaller ones. On the other hand, the number of parameters increases because of the input layer. We aim to verify whether there is a common tendency to choose smaller or larger pre-trained embeddings.

The full pipeline is shown in Figure 7.4.

7.1.3.3 LSTM

We also experiment with LSTMs (and BiLSTM) together with words and sentence embeddings (using Ragged tensors). Nevertheless, we do not include a trainable layer in the architecture to learn the word embeddings as it is associated with many trainable parameters, rendering it impossible with the amount of data we have. The pipeline is shown in Figure 7.5 using a BiLSTM architecture.

Each experiment was run ten times due to the random initialization conditions, resulting in very different results each time a given configuration is trained. The architectures were manually improved, considering the number of trainable parameters. The best model found will be shown. Hence, a diagnostic approach was conducted to explore different model configurations.

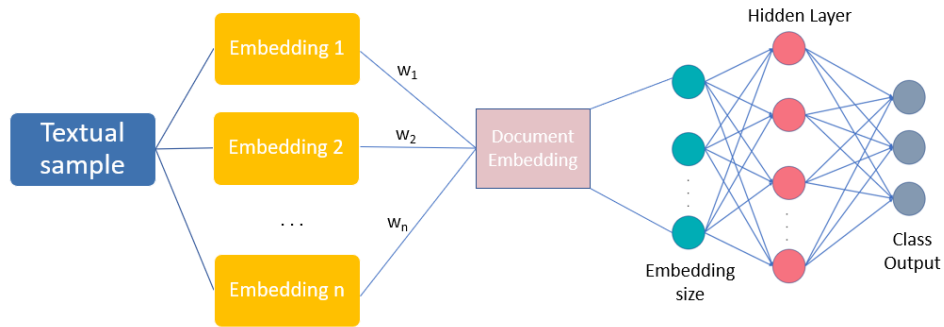


Figure 7.4: Neural network pipeline

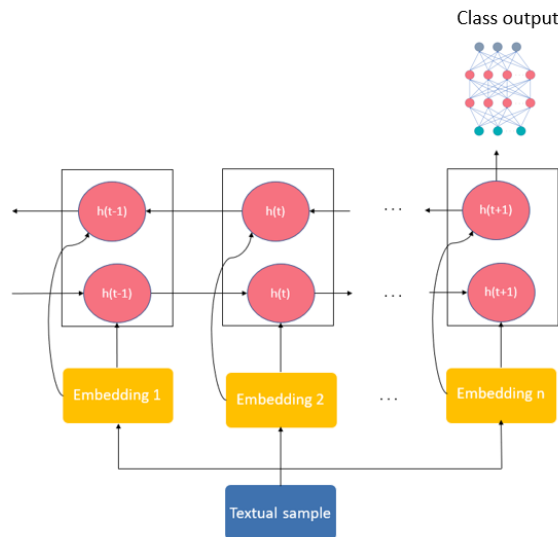


Figure 7.5: BiLSTM pipeline

Line plots of the model performance over the epochs were studied for insight into how a given configuration performs and how it may be configured to elicit better performance. The model was evaluated on both the train and the test datasets at the end of each epoch and the sequence of training and test scores were plotted.

7.1.4 MLSMOTE

To combat data imbalance, we can do several things, such as giving weights to classes, threshold moving, and data augmentation. We have already mentioned threshold moving and class weights. In this section, we will explain MLSMOTE [29]. MLSMOTE is a variation of SMOTE to work with a multilabel dataset. The data selection to augment is based on samples corresponding to the minority classes. The selection is based on two different metrics: IRPL (Imbalance ratio per label)

and MIR (Mean Imbalance ratio). The former is shown in Equation 7.1, and the latter in Equation 7.2.

$$IRPL(j) = \frac{\arg\max_{j'=L_1}^{L_{|L_1|}} (\sum_{i=1}^{|N|} h(j', Y_i))}{\sum_{i=1}^{|N|} h(j, Y_i)} \quad h(j, Y_i) = \begin{cases} 1, & j \in Y_i \\ 0, & j \notin Y_i \end{cases} \quad (7.1)$$

$$MIR = \frac{1}{|L|} \sum_{i=L_1}^{L_{|L|}} IRPL(i) \quad (7.2)$$

A label is selected as a minority if the IRPL of a certain label is greater than the mean. After selecting a minority instance, we select their K nearest neighbors on the feature space. The number of neighbors they recommend is five, like the original SMOTE algorithm. To create a new synthetic sample, an interpolation between a reference neighbor (selected randomly) and the data itself is calculated. The label set of the new instance is a cloning of the seed instance label set.

Since the algorithm is dependent on randomness, we ran each experiment ten times and created one new instance from each instance selected as a part of minority classes.

7.1.5 Feature Engineering

We selected the best model that uses a word or sentence embedding and another that uses a frequency-based feature extraction. In addition, since we are dealing with textual data, the new features we can create accrue from it. They can be used as extra features that can potentially enhance the classification model performance. The features we are using are presented in Table 7.3.

We tried adding each feature individually and all of them, and then we check the performance.

Besides, all the features were normalized since features with a more considerable range tend to play a more decisive role in models that are not rule-dependent as they are not affected by any monotonic transformation of variables (example: Decision Trees). We calculate the Min-Max Scaler presented in Equation 7.3 to overcome this.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (7.3)$$

7.2 Results and Discussions

This section presents the experiment results and their discussion about the first-level categorization task.

Length analysis (1)	There are several measures that can give us some insight. For instance, a certain category can be systematically longer or has more words per sentences. Since these types of metrics are easy to calculate and can be potentially useful, we are using the following as features: the number of chars, number of words, number of capital words, number of punctuation marks, number of sentences, number of unique words, average word length, average sentence length and the ratio between unique words and total number of words
Sentiment Score (2)	Even though this is directly related to the second task, we can still use scores from different open-source tools as additional features for the other tasks. We are using both VADER and TextBlob to obtain those respective values.
NER (3)	For this tool to become useful, we need to run a NER model for every sample in the dataset. Each recognized entity is going to correspond to a new feature (column in the dataset) and we can keep track of the number of times an entity appears in a document.
Topic Modeling (4)	We can use the Gensim package to perform topic modelling using the LDA model. We already showed in the data analysis Section how to choose the number of topics for a dataset. With that value in mind, we can run the algorithm and create a matrix where the probability of each document containing a topic is passed as a feature.
Pos Tagging (5)	The same strategy we used for PoS we also did it for this. For each document, we counted the number of times a part of speech appeared as used them as features. The model we used to identify them was the model implemented in the Python library NLTK.
Readability score (6)	For each instance the readability score, also shown in the data analysis section, was calculated, and passed as a feature. For this, the Python library Textstat was utilised.

Table 7.3: Features to be used as feature engineering

7.2.1 Preprocessing

In this section, we verify which preprocessing steps go better hand in hand with each other. We will present a table with the mean and standard deviation. The following Table 7.4 shows some interesting combinations, and the row whose values are bold depicts our selected combination for the first task (the best-weighted F1-score).

Overall, not performing any normalization or performing lemmatization together with lowercasing without removing punctuation seems consistently the best.

Stemming in conjunction with removing punctuation seems, in general, an above-average combination but not constantly the best.

In general, what influenced the classification the most was the stop word list used, being preferred not using them or using the ones whose TF-IDF values are solely zero. On the other hand, what seemed to influence performance the least was lowercasing.

With that being said, for the first task we choose the combination: `{lowercasing : True, normalization : None; stopwordslist : (3), remove_punctuation : False}`. Choosing lemmatization or None as normalization is preferred since stemming creates words that may not belong to the English dictionary. Therefore, when using word embeddings, that could lead us to a great amount

Lowercase	Remove punctuation	Normalization	Stop word list (Identifier)	Weighted F1-Score	Weighted Jaccard score
True	False	None	None	0.661 +- 0.042	0.520 +- 0.044
True	False	lemmatization	None	0.662 +- 0.045	0.520 +- 0.046
True	False	None	(1)	0.610 +- 0.041	0.460 +- 0.045
False	False	lemmatization	(2)	0.641 +- 0.027	0.490 +- 0.029
False	True	stemming	(3)	0.662 +- 0.040	0.510 +- 0.035
True	False	lemmatization	(4)	0.649 +- 0.046	0.510 +- 0.048
True	False	lemmatization	(3)	0.656 +- 0.041	0.510 +- 0.043
True	False	None	(4)	0.661 +- 0.035	0.520 +- 0.028
True	False	None	(3)	0.667 +- 0.029	0.520 +- 0.024
True	False	None	(2)	0.610 +- 0.310	0.460 +- 0.032

Table 7.4: Average performance and standard deviation of some noticeably interesting combinations for the first-level message categorization

of OOVs. Moreover, as we can see in Table 7.4 above, that combination has the least standard deviation showing its consistency along with the classifiers.

7.2.2 Adversarial Validation

Binary classifiers	Mean	Standard Deviation
Random Forest	0.485	0.063
SVM (linear)	0.479	0.073

Table 7.5: Adversarial Task on BOW

Binary classifiers	Mean	Standard Deviation
Random Forest	0.495	0.044
SVM (linear)	0.466	0.050

Table 7.6: Adversarial Task with TF-IDF vectorization

According to Table 7.5, Table 7.6, and Table 7.7 (using BOW, TF-IDF vectorization, and universal sentence encoder, respectively), the training and test set should follow a similar distribution since a binary classifier would not be able to distinguish between them.

Binary classifiers	Mean	Standard Deviation
Random Forest	0.490	0.061
SVM (linear)	0.502	0.049

Table 7.7: Adversarial Task with Universal Sentence Encoders

7.2.3 Comparison Between Pipelines

In this subsection, we display the best results we accomplished from our best pipelines, and we discuss and compare the results obtained among them all.

7.2.3.1 Frequency-based Feature Extraction with Classical Machine Learning Models

After running grid search cross-validation with a value of five for the training data, the best parameters were applied to the testing data. The best pipeline's performance and parameters are shown. The best classifiers found were SVM and Logistic Regression.

SVM does not inherently support multilabel classification, and after experimenting with binary relevance, classifiers chains, and label powerset, binary relevance came out on top (a bit superior compared to classifier chain). The best pipeline parameters are shown in Table 7.8.

SVM pipeline	Parameter value
C	1
kernel	rbf
TF-IDF norm	l2
TF-IDF use-idf	True
n-gram range	(1,2)
analyzer	char_wb

Table 7.8: Best configuration parameters for SVM (binary relevance) and frequency based vectorization technique

The classification report for the best configuration is presented in Table 7.9.

This classifier is one of the most balanced in terms of performance since for each class it has precision and recall values well fairly above the 50% mark.

Regarding the Logistic Regression classifier, the best algorithm configuration is described next, in Table 7.10, and the classifier chain had the best performance in comparison with binary relevance and label powerset.

The classification report for the best configuration is presented in Table 7.11.

Its performance is the most similar to SVM but slightly better. Next, we discuss the other classifiers we tried.

SVM (binary relevance)	Group interactions	Offering support	Seeking support
Precision	0.67	0.91	0.59
Recall	0.73	0.77	0.65
F1-score	0.70	0.83	0.62
Jaccard Score	0.59		
Hamming Loss	0.26		

Table 7.9: Classification report for SVM using binary relevance

Logistic Regression pipeline	Parameter value
C	100
solver	liblinear
TF-IDF norm	l2
TF-IDF use-idf	True
n-gram range	(1,2)
analyzer	char_wb

Table 7.10: Best configuration parameters for Logistic Regression (classifier chain) and frequency based vectorization technique

Logistic Regression (classifier chain)	Group interactions	Offering support	Seeking support
Precision	0.71	0.89	0.54
Recall	0.73	0.81	0.81
F1-score	0.72	0.85	0.65
Jaccard Score	0.62		
Hamming Loss	0.25		

Table 7.11: Classification report for Logistic Regression using classifier chains

The Decision Tree classifier performs better when dealing with the class "seeking support" than other classifiers. Also not supporting multilabel classification, Gaussian NB, similar to most classifiers, struggles to deal with instances with the class "seeking support", but it behaves pretty well for the other two classes, especially for offering support obtaining an F1-score of 80%. When considering the Random Forest classifier, even though the performance is quite good compared to SVM and LR, the discrimination of the seeking support class is worse since the recall is lower than theirs, even though it possesses higher precision (83% in offering support). For KNN, the algorithm's performance is better for the classes with more instances; however, it is relevant to consider that we might not have enough data samples to find the proper clusters in the feature space. For further clarification, tables containing the best configuration parameters and classification report for Decision Tree, Gaussian NB, Random Forest, and KNN can be found in the Appendix A.1.

7.2.3.2 Pre-trained Word Embeddings and Classical Machine Learning

In this section, we will show the best overall pipeline and results for each pre-trained embedding.

SVM (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
C=10 class_weight='balanced' kernel: 'rbf' (binary relevance)	Twitter-200	0.26	0.58	0.73
C=10 class_weight='balanced' kernel: 'rbf' (label powerset)	word2vec-google-news-300	0.26	0.58	0.72
C=100 class_weight='balanced' kernel='linear' (label powerset)	fasttext-wiki-news-subwords-300	0.23	0.63	0.77

Table 7.12: Classification report for SVM with word embeddings

Both LR and SVM classical machine learning models stand out when using word embeddings as input features.

According to Table 7.12, we can see that for SVMs, the larger feature spaces are preferred without any type of word weighting. This may be because finding linear separations in higher dimensions is easier. Among themselves, the FastText embedding is superior, with an additional 5% of weighted F1-score.

Logistic Regression (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
C=1 class_weight='balanced' solver='liblinear' (binary relevance)	Twitter-200	0.25	0.58	0.73
C=100 class_weight='balanced' solver='liblinear' (label powerset)	word2vec-google-news-300	0.26	0.58	0.72
C=100 class_weight='balanced' solver='liblinear' (binary relevance)	fasttext-wiki-news-subwords-300	0.23	0.63	0.77

Table 7.13: Classification report for Logistic Regression with word embeddings

For Logistic Regression, similar to SVM and KNN, according to Table 7.13, larger feature spaces are favored over smaller ones without any type of word weighting. Even though it is based on statistical approaches, compared to a geometrical one, as seen in SVM, both have practically the same performance for each embedding. Next, we discuss briefly the others classifiers.

While TF-IDF word weighting is not preferred in the majority of the word embeddings in Decision Tree, the word whose value is the highest uses it. Moreover, for a smaller word-embedding (GloVe-twitter) whose training data is more similar in comparison with a larger one (FastText), we obtained overall better results. In Gaussian NB, the TF-IDF is preferred when the word embedding is smaller. Moreover, its performance can rival a larger word embedding, namely FastText. Nevertheless, the performance of all of the embeddings is quite similar. Regarding Random Forest, its performance is superior to Decision Tree, especially when using FastText embedding. In the KNN classifier, larger feature spaces are preferred without word weighting. The performance of all of the word embeddings is quite similar, and all of them are a bit superior to Gaussian NB in terms of performance. Likewise, if further clarification is needed, Tables containing the best configuration parameters and classification report for Decision Tree, Gaussian NB, Random Forest, and KNN can be found in the Appendix A.1.

7.2.3.3 Universal Sentence Encoder with Classical Machine learning

First, we present the weighted F1-score, Hamming Loss, and Jaccard Score for each model whose parameter configuration was found via cross-validation with a five-fold. Finally, we present the classification report of the best classical machine learning model. For models that do not implement multilabel classification, we will show which one out of the binary relevance, classifier chain, and label powerset gives the best results.

By observing Table 7.14, we were able to obtain the best performance, namely the SVM (with linear kernel and C parameter as 10), together with universal sentence embedding (average over the sentence embeddings).

7.2.3.4 Pre-trained embeddings with neural networks

According to Table 7.15, we can conclude that TF-IDF does not significantly affect the weighted F1-score. We can also verify that when word embeddings are smaller, the hidden layer's best size tends to be larger, which leads us to think that the best number of parameters is in-between those values. Moreover, smaller layers (16, 32, and 64) were not favored. Both fastText and Universal Sentence Encoders are the best models to use with a single hidden layer neural network.

7.2.3.5 Deep learning

After experimenting with word and sentence embeddings, we concluded that the latter gave consistently better results. The former would often lead to a very high recall and very low precision result or vice-versa. Additionally, LSTM, compared to BiLSTM, was, in general, a bit worse.

Classifier	Approach	Weighted F1-score	Hamming Loss	Weighted Precision	Weighted Recall	Jaccard Index
Decision Tree	whole text	0.64	0.29	0.89	0.55	0.48
	sentence mean	0.70	0.24	0.89	0.60	0.54
Gaussian NB	whole text (label powerset)	0.78	0.21	0.80	0.77	0.65
	sentence mean (classifier chain)	0.78	0.23	0.76	0.79	0.64
KNN	whole text	0.72	0.26	0.77	0.70	0.57
	sentence mean	0.70	0.26	0.75	0.67	0.56
SVM	whole text (binary relevance)	0.76	0.23	0.79	0.74	0.62
	sentence mean (classifier chain)	0.82	0.18	0.83	0.82	0.70
Logistic Regression	whole text (binary relevance)	0.76	0.24	0.77	0.75	0.61
	sentence mean (binary relevance)	0.79	0.21	0.80	0.79	0.66
Random Forest	whole text	0.77	0.19	0.85	0.73	0.65
	sentence mean	0.8	0.17	0.89	0.76	0.68

Table 7.14: Best classical machine learning models using universal sentence encoders

Neural Networks (best parameter values)	Word Embedding	F1-score weighted
hidden layer size: 256 (TF-IDF)	GloVe-Twitter-50	0.70
hidden layer size: 128 (TF-IDF)	GloVe-Twitter-200	0.75
hidden layer size: 256 (TF-IDF)	Word2vec	0.72
hidden layer size: 256 (TF-IDF)	fastText	0.76
hidden layer size: 256	Universal (whole)	0.76
hidden layer size: 128	Universal (mean)	0.75
hidden layer size: 256	GloVe-Twitter-50	0.71
hidden layer size: 128	GloVe-Twitter-200	0.71
hidden layer size: 128	Word2vec	0.71
hidden layer size: 128	fastText	0.76

Table 7.15: Classification report for neural networks with pre-trained embeddings

Therefore, we only present the BiLSTM results. The configuration found was constituted by five BiLSTM cells, and the batch size was four.

In Figure 7.6, we show the loss of the algorithm when we run ten experiments. Although it

could not beat Universal Sentence Encoder with SVM, it had an overall good performance, as we can see in Table 7.16 the mean of the runs. Moreover, had we let the training continue for more epochs, it would overfit the training data.

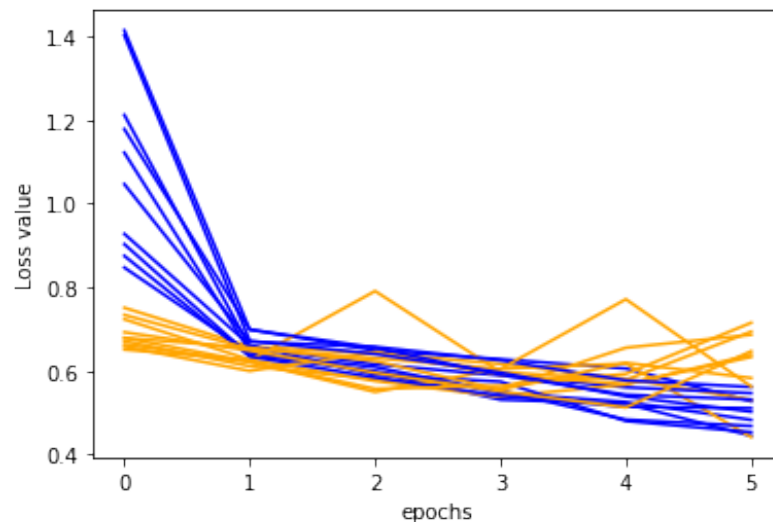


Figure 7.6: Loss of the training and test data

10 runs	Mean value
Weighted F1-score	0.75
Weighted Jaccard Score	0.61
Hamming Loss	0.24

Table 7.16: BiLSTM evaluation metrics

In Figure 7.6, we can also see that although it usually behaves fairly well, depending on the initialization weights, it can lead to erratic behavior.

7.2.4 Feature Engineering

The best pipelines which used frequency-based text vectorization were the ones using SVM and Logistic Regression classifier. Consequently, we tried both. The pipelines which used embedding techniques that were the best also include the same classifiers, SVM and Logistic Regression, with SVM being a bit superior, so we tried this last one.

According to Table 7.17, for SVM, PoS tagging is the unique extra feature that improves performance. Nevertheless, there are other interesting local maximums. For example, features (3) and (4) are the worst, but they get a slight boost when combined. A common trend is for them to diminish the precision of the original algorithm while increasing the recall.

SVM (best configuration)	Weighted F1-score	Hamming Loss	Weighted Precision	Weighted Recall	Jaccard Index
Original	0.74	0.26	0.75	0.73	0.59
Length analysis (1)	0.73	0.27	0.72	0.75	0.58
Sentiment Score (2)	0.70	0.32	0.68	0.78	0.55
NER (3)	0.58	0.35	0.65	0.55	0.43
Topic Modeling (4)	0.65	0.37	0.66	0.67	0.48
(3) + (4)	0.65	0.35	0.66	0.63	0.494
Pos Tagging (5)	0.76	0.25	0.75	0.79	0.63
Readability score (6)	0.69	0.31	0.71	0.74	0.53
All	0.65	0.37	0.68	0.67	0.49

Table 7.17: Feature engineering on SVM (binary relevance)

Logistic Regression (best configuration)	Weighted F1-score	Hamming Loss	cWeighted Precision	Weighted Recall	Jaccard Index
Original	0.76	0.25	0.75	0.78	0.62
Length analysis (1)	0.75	0.25	0.73	0.79	0.61
Sentiment Score (2)	0.75	0.28	0.71	0.80	0.60
NER (3)	0.73	0.26	0.74	0.72	0.58
Topic Modeling (4)	0.73	0.28	0.72	0.75	0.58
(3) + (4)	0.72	0.26	0.75	0.75	0.57
Pos Tagging (5)	0.75	0.26	0.74	0.74	0.61
Readability score (6)	0.75	0.26	0.74	0.77	0.61
All	0.70	0.31	0.70	0.72	0.54

Table 7.18: Feature engineering on Logistic Regression (Classifier Chain)

By observing Table 7.18, for LR, the impact they have on the original one is small. Nevertheless, we see the same trend as SVM, with the precision decreasing and the recall increasing. Unfortunately, feature engineering did not help this classifier, even though it revealed more stability in terms of performance than the SVM.

For Universal Sentence embedding, it seems that the score is always penalized with feature engineering, as seen in Table 7.19.

Universal Sentence Embedding + SVM (best configuration)	Weighted F1-score	Hamming Loss	Weighted precision	Weighted Recall	Jaccard Index
Original	0.82	0.18	0.83	0.82	0.70
Length analysis(1)	0.79	0.21	0.81	0.78	0.66
Sentiment Score (2)	0.79	0.22	0.77	0.81	0.66
NER(3)	0.71	0.26	0.77	0.67	0.56
Topic modeling (4)	0.81	0.19	0.83	0.79	0.68
(3) + (4)	0.73	0.24	0.79	0.68	0.58
PoS Tagging (5)	0.78	0.22	0.79	0.78	0.65
Readability Score (6)	0.81	0.19	0.81	0.81	0.68
(4) + (6)	0.79	0.20	0.80	0.79	0.66
All	0.75	0.23	0.75	0.77	0.62

Table 7.19: Feature engineering on SVM (Classifier Chain) with Universal Sentence Encoders

7.2.5 MLSMOTE

Since it is a method associated with some randomness, we ran the experiment ten times (to reduce variance). We only did it on the best models. For the Logistic Regression model (with the best parameters), the values are presented in Table 7.20.

Logistic Regression (classifier chain)	Group interactions	Offering support	Seeking support
Precision	0.69	0.88	0.60
Recall	0.69	0.87	0.81
F1-score	0.69	0.81	0.69
Jaccard Score	0.63		
Hamming Loss	0.24		

Table 7.20: Best Logistic Regression model performance with MLSMOTE

According to Table 7.20, if we compare without using MLSMOTE, we see that the increase was relatively small. A possible reason for this may be that we are using KNN in the algorithm, which can even lead to some errors being propagated. We can see that when using Universal Sentence Encoder with SVM in Table 7.21 since the performance dropped a bit.

Universal Sentence Encoder + SVM (best configuration)	Group interactions	Offering support	Seeking support
Precision	0.76	0.86	0.65
Recall	0.77	0.79	0.77
F1-score	0.76	0.83	0.70
Jaccard Score	0.64		
Hamming Loss	0.22		

Table 7.21: SVM with universal sentence encoders performance with MLSMOTE

Chapter 8

Second-level Categorization

This section is quite similar to the previous one, so we will focus on the results and discussions instead. The only difference between the methods of the two sections lies in the fact that, in this case, the deep learning model was not experimented with due to the lack of instances in some classes.

8.1 Results and Discussions

In this subsection, the results regarding the second-level message categorization are presented as well as their discussion.

8.1.1 Preprocessing

The table 8.1 illustrates some worth-mentioning combinations. The row with bold values represents the combination we selected for the third task (with the best F1-score and Jaccard Score as a tiebreaker). The values shown are the mean and standard deviation.

Overall, out of these parameters, the stop word list is by far the one that influences the outcome the most, with the lists (1) and (2) being noticeably worse than the others. The best options are not removing stop words or using the stop word list (3) or (4). Within these, performing lowercasing and removing punctuation is often better. Regarding normalization, not doing anything or performing lemmatization usually has superior results. With that being said, the option we chose was the one with the highest F1-score and Jaccard Score, namely $\{Lowercase : True, Remove\ punctuation : True, Normalization : None, stop\ word\ list : (4)\}$.

8.1.2 Adversarial Validation

After we carried out preprocessing, we performed adversarial validation using different input features.

Lowercase	Remove punctuation	Normalization	Stop word list (identifier)	Weighted F1-score	Weighted Jaccard Score
True	True	None	None	0.476 +- 0.097	0.353 +- 0.086
True	False	None	None	0.471 +- 0.096	0.348 +- 0.088
False	False	Stem	None	0.467 +- 0.103	0.348 +- 0.096
True	True	Lemma	None	0.469 +- 0.100	0.347 +- 0.089
True	True	None	(1)	0.426 +-0.105	0.301 +- 0.091
False	True	Lemma	(1)	0.456 +- 0.093	0.334 +- 0.085
False	True	None	(2)	0.454 +- 0.110	0.333 +- 0.098
False	True	Lemma	(2)	0.459 +- 0.094	0.336 +- 0.086
True	False	None	(3)	0.473 +- 0.106	0.352 +- 0.097
True	True	Lemma	(3)	0.472 +- 0.100	0.348 +- 0.086
True	True	None	(4)	0.476 +- 0.104	0.354 +- 0.096
False	True	Lemma	(4)	0.469 +- 0.096	0.349 +- 0.087

Table 8.1: Average performance and standard deviation of some noticeably interesting combinations for the second-level classifier

Binary classifiers	Mean	Standard Deviation
Random Forest	0.480	0.063
SVM (linear)	0.488	0.063

Table 8.2: Adversarial Task on BOW

Binary classifiers	Mean	Standard Deviation
Random Forest	0.485	0.064
SVM (linear)	0.481	0.045

Table 8.3: Adversarial Task with TF-IDF vectorization

Binary classifiers	Mean	Standard Deviation
Random Forest	0.511	0.058
SVM (linear)	0.480	0.055

Table 8.4: Adversarial Task with Universal Sentence Encoders

According to Table 8.2, Table 8.3, and Table 8.4 (using BOW, TF-IDF vectorization, and universal sentence encoder, respectively), the training and test set follow a similar distribution

since a binary classifier would not be able to distinguish between them.

8.1.3 Comparison Between Pipelines

The best outcomes from our best pipelines are presented in this subsection, along with a discussion and comparison of the results from each of them.

8.1.3.1 Frequency-based Feature Extraction with Classical Machine Learning Models

We ran grid search cross-validation with a value of five for the training data, and then the best parameters were applied to the test set. We show the best parameters for the corresponding classifier and the classification report. For this task, we show the individual values for each class. We also exhibit the support value corresponding to the total number of samples in the test set belonging to that class. We can infer the number of training samples the classifier acquired for training. We are doing this since the dataset is more imbalanced.

The best configuration for Gaussian NB is presented in Table 8.5 and the classification report in Table 8.6. Among binary relevance, classifier chain, and label powerset to deal with the multi-label setting of the problem, the classifier chain approach was the one with the best performance.

Gaussian NB pipeline	Parameter value
var smoothing	1e-3
TF-IDF norm	l2
TF-IDF use-idf	True
n-gram range	(1,1)
analyzer	char_wb

Table 8.5: Gaussian NB best configuration for the third task

classes (GN)	precision	recall	f1-score	support
emotional-support	0.22	0.80	0.35	15
esteem-support	0.07	0.57	0.12	7
informational-support	0.71	0.71	0.71	42
interpersonal	0.25	0.80	0.38	20
network-support	0.08	0.71	0.15	7
seeking-support	0.65	0.42	0.51	26
sharing-experiences	0.49	0.85	0.62	40

Table 8.6: Classification report for Gaussian NB with the best parameters

The Gaussian NB classifier, together with classifier chains, can predict at least one instance of each class correctly. In comparison with the Decision Tree classifier, it is practically better at distinguishing every class, even those which contain more support. Moreover, one possible reason

for being able to get right some minority classes is that it leverages those label correlations since it uses classifier chains.

The best configuration for SVM classifier is presented in the Table 8.9, and the classification report in the Table 8.8. Among binary relevance, classifier chain, and label powerset to deal with the multi-label setting of the problem, the binary relevance approach was the one with better performance.

SVM Pipeline	Parameter value
C	1
kernel	linear
TF-IDF norm	l2
TF-IDF use-idf	True
n-gram range	(1,3)
analyzer	word

Table 8.7: SVM best configuration for the third task

classes (SVM)	precision	recall	f1-score	support
emotional-support	0.50	0.07	0.12	15
esteem-support	0.00	0.00	0.00	7
informational-support	0.69	0.74	0.71	42
interpersonal	0.67	0.10	0.17	20
network-support	0.00	0.00	0.00	7
seeking-support	0.85	0.42	0.56	26
sharing-experiences	0.81	0.85	0.83	40

Table 8.8: Classification report for SVM with the best parameters

The performance of LR is very similar to that of the SVM. In terms of the classes more prevalent in the dataset, the F1-score rivals the ones of SVM. As for the rest, the two least representative classes are not classified correctly once in both classifications.

The best algorithm configuration and classification report is shown next in Table 7.10, and Table 8.10, respectively. Just like SVM, binary relevance obtained the best performance.

The performance of this classifier is very similar to that of the SVM. In terms of the classes more prevalent in the dataset, the F1-score rivals the ones of SVM, whereas, regarding the two least representative classes, it is unable to classify them correctly. The other classifiers are discussed next.

In regards to Decision Tree, we can see that this rule-based model, performs somewhat well on the samples whose class support is high in general, whereas regarding the rest, it is not able to recognize the data patterns and cannot distinguish the majority of the classes. For Random

Logistic Regression pipeline	Parameter value
C	100
solver	liblinear
TF-IDF norm	l2
TF-IDF use-idf	True
n-gram range	(1,3)
analyzer	word

Table 8.9: Logistic Regression best configuration for the third task

classes (Logistic regression)	precision	recall	f1-score	support
emotional-support	0.50	0.07	0.12	15
esteem-support	0.00	0.00	0.00	7
informational-support	0.70	0.76	0.73	42
interpersonal	0.67	0.10	0.17	20
network-support	0.00	0.00	0.00	7
seeking-support	0.91	0.38	0.54	26
sharing-experiences	0.83	0.85	0.84	40

Table 8.10: Classification report for Logistic Regression with the best parameters

Forest, as expected, the classes that have higher support get a better value, with an F1-score between 65% and 69% compared to 50% to 52% of the Decision Tree. In a sense, we can attest that having an ensemble of classifiers does help in the final performance. Nonetheless, even with the clear increase in performance, the classifier behaves poorly compared to the others. The number of neighbors for the KNN classifier was most likely skewed towards the number that maximizes the most classes whose support number is higher. That can especially affect the classes whose number of instances is very low, which were not predicted correctly. In case further clarification is needed, Tables containing the best configuration parameters and classification report for Decision Tree, Random Forest, and KNN can be found in the Appendix A.2. A summary of all the classifiers mentioned above used and their respective performance metrics can also be consulted on the Appendix A.2.

8.1.3.2 Pre-trained Eord Embeddings and Classical Machine Learning

For each type of pre-trained embedding, we are only showing the best overall pipeline and their results.

After analyzing Table 8.11, we see the same relation between the SVM and the KNN since both favors larger feature spaces without using the TF-IDF weights. The SVM configuration with the GloVe-Twitter-200 is one of the best among word embeddings with classical machine learning models for the current task.

SVM (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
C: 10 class_weight: 'balanced' kernel: 'linear' (binary relevance)	GloVe-Twitter-200	0.26	0.44	0.58
C: 20 class_weight: 'balanced' kernel: 'linear' (label powerset)	word2vec-google-news-300	0.27	0.38	0.53
C: 20 class_weight: 'balanced' kernel: 'rbf' (binary relevance)	fasttext-wiki-news-subwords-300	0.26	0.26	0.58

Table 8.11: Classification report for SVM with word embeddings

Logistic Regression (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
C: 10 penalty: 'l2' solver: 'liblinear' (classifier chain)	GloVe-Twitter-200	0.30	0.43	0.58
C: 100 penalty: 'l2' solver: 'liblinear' (binary relevance TF-IDF)	word2vec-google-news-300	0.26	0.38	0.52
C: 100 penalty: 'l2' solver: 'liblinear' (binary relevance)	fasttext-wiki-news-subwords-300	0.25	0.48	0.62

Table 8.12: Classification report for Logistic Regression with word embeddings

Similar to SVM and KNN, after observing Table 8.12, larger feature spaces are preferred to smaller ones, without any type of word weighting. The model, together with FastText, has the best performance among word embeddings with classical machine learning models tested.

In Decision Tree, the TF-IDF word weighting is not preferred in most of the embeddings. Moreover, comparing this same setting with the first-level classifier, the larger embeddings are always preferred here, contrary to the former. In Gaussian NB's case, the TF-IDF is not preferred in any of the cases. The word embedding selected and the parameter values of the classifier are the same ones as Task 1. The reason for the first classifier to have both higher hamming loss and Jaccard score is most likely because it sets more classes than the others to true, which can easily

lead to higher scores of hamming loss. After all, the majority of the classes are set to zero in a binary representation of labels, but, at the same time, the similarity of prediction and testing got higher. Therefore, indicating a better classifier here would depend on the importance given to a higher recall or higher precision. In comparison with Decision Tree, Random Forest, contrary to what we saw in the first-level classification, the ensemble method performs practically as well as the Decision Tree classifier. The KNN classifier, based on the distance of feature vectors, is better when we have a larger feature space without any type of word weighting. The performance of word2vec is slightly better than the others. Tables containing the best configuration parameters and classification report for Decision Tree, Gaussian NB, Random Forest, and KNN can be found in the Appendix A.2.

8.1.3.3 Universal Sentence Encoder with Classical Machine learning

Classifier	Approach	Weighted F1-score	Hamming Loss	Weighted Precision	Weighted Recall	Jaccard Index
Random Forest	whole text	0.36	0.19	0.42	0.32	0.28
	sentence mean	0.33	0.21	0.41	0.31	0.24
Gaussian NB	whole text (classifier chain)	0.60	0.23	0.57	0.64	0.45
	sentence mean (binary relevance)	0.60	0.29	0.54	0.73	0.46
KNN	whole text	0.47	0.20	0.50	0.46	0.35
	sentence mean	0.51	0.19	0.53	0.51	0.39
SVM	whole text (binary relevance)	0.52	0.17	0.58	0.51	0.40
	sentence mean (binary relevance)	0.60	0.19	0.58	0.62	0.46
Logistic Regression	whole text (classifier chain)	0.58	0.21	0.60	0.57	0.44
	sentence mean (binary relevance)	0.62	0.23	0.58	0.67	0.48
Decision Tree	whole text	0.44	0.28	0.43	0.46	0.30
	sentence mean	0.40	0.31	0.37	0.43	0.27

Table 8.13: Best classical machine learning models using universal sentence encoders

According to Table 8.13, the best algorithms for this approach are either using Gaussian NB or Logistic Regression. Both with binary relevance and using the mean of the sentence embeddings. The rule-based models were overall the ones that behaved the worst. In comparison with word embeddings, Gaussian NB is far better when using Universal Sentence Encoders, while for Logistic Regression, the best word embedding combination had a result that was similar to this one (just slightly worse).

8.1.3.4 Pre-trained Embeddings with Neural Networks

Neural Networks (parameter values)	Word Embedding	F1-score weighted
hidden layer size: '128'	GloVe-Twitter-50	0.38
hidden layer size: '256' (TF-IDF)	GloVe-Twitter-50	0.49
hidden layer size: '256' (TF-IDF)	GloVe-Twitter-200	0.50
hidden layer size: '256'	GloVe-Twitter-200	0.52
hidden layer size: '256' (TF-IDF)	word2vec-google-news-300	0.41
hidden layer size: '16'	word2vec-google-news-300	0.48
hidden layer size: '64'	fasttext-wiki-news-subwords-300	0.47
dropout rate: '0.1' hidden layer size: '256' (TF-IDF)	fasttext-wiki-news-subwords-300	0.40
hidden layer size: '256'	Universal	0.59
hidden layer size: '256'	Universal (mean)	0.51

Table 8.14: Classification report for neural networks with pre-trained embeddings

By analyzing Table 8.14, we can observe that the values declined quite a lot compared to the first-level classification. Moreover, we can see that the Universal Sentence Encoder, like in the previous task, is the embedding associated with higher performance. FastText which was as good as the Universal Sentence Encoder dropped its performance compared with the other embeddings. Interestingly, in this case, we see hidden layers whose size is far smaller than in the first-level categorization, which probably shows the need to have a simpler model when dealing with a more complex dataset with the same amount of labeled data.

8.1.4 Feature Engineering

We tried the best pipeline, which uses a frequency-based vectorization technique, and another that uses a pre-trained embedding. The selection was based on which model had the highest weighted F1-score. The former was the Logistic Regression (binary relevance) with a TF-IDF vectorization technique, while the latter was the Gaussian NB (binary relevance) with Universal Sentence Encoder (with mean operation).

From Table 8.15, we can see that performing feature engineering almost always leads to a slight performance decline. The only exception is when we add the PoS tagging, which increases both overall recall and precision.

Similarly to Table 8.15, Table 8.16 shows that performance almost always declines with again the exception of PoS tagging. Moreover, the readability score is the only metric that increased Precision.

SVM (best configuration)	Weighted F1-score	Hamming Loss	Weighted Precision	Weighted Recall	Jaccard Score
Original Text	0.53	0.16	0.66	0.50	0.41
Length analysis (1)	0.49	0.20	0.62	0.48	0.36
Sentiment Score (2)	0.47	0.19	0.63	0.45	0.34
NER (3)	0.53	0.18	0.59	0.53	0.41
Topic modeling (4)	0.53	0.18	0.62	0.50	0.40
PoS tagging (5)	0.54	0.16	0.68	0.52	0.42
Readability Score (6)	0.53	0.16	0.71	0.47	0.41
All	0.49	0.22	0.55	0.48	0.35

Table 8.15: Feature engineering for SVM with TF-IDF vectorization

Gaussian NB (best configuration)	Weighted F1-score	Hamming Loss	Weighted Precision	Weighted Recall	Jaccard Score
Original Text	0.61	0.22	0.58	0.64	0.47
Length analysis (1)	0.58	0.28	0.55	0.70	0.43
Sentiment Score (2)	0.57	0.22	0.56	0.59	0.43
NER (3)	0.59	0.23	0.55	0.64	0.44
Topic modeling (4)	0.61	0.23	0.57	0.66	0.46
PoS tagging (5)	0.62	0.24	0.58	0.70	0.48
Readability Score (6)	0.60	0.20	0.60	0.60	0.46
All	0.57	0.27	0.55	0.66	0.42

Table 8.16: Feature engineering for Gaussian NB with Universal Sentence Encoder

8.1.5 MLSMOTE

The metrics for each class are presented in Table 8.17. The experiment is the result of running it ten times to reduce variance.

Overall, the trend is similar to the first-level categorization when using MLSMOTE for a frequency-based text vectorization technique since it increased performance. In this experiment, there was on average a 4% improvement of the F1-score to 55%.

Regarding using a pre-trained embedding vector, we present the results in the Table when using universal sentence encoders with Gaussian NB classifier (best configuration) 8.18.

Universal Sentence Encoder + SVM (best configuration)	precision	recall	f1-score	support
emotional-support	0.80	0.27	0.40	15
esteem-support	0.00	0.00	0.00	7
informational-support	0.68	0.76	0.72	42
interpersonal	0.67	0.10	0.17	20
network-support	0.00	0.00	0.00	7
seeking-support	0.78	0.54	0.64	26
sharing-experiences	0.72	0.82	0.77	40

Table 8.17: SVM with TF-IDF using MLSMOTE

Universal Sentence Encoder + NB (best configuration)	precision	recall	f1-score	support
emotional-support	0.25	0.07	0.11	15
esteem-support	0.00	0.00	0.00	7
informational-support	0.77	0.55	0.64	42
interpersonal	0.40	0.10	0.16	20
network-support	0.00	0.00	0.00	7
seeking-support	1.00	0.27	0.42	26
sharing-experiences	0.72	0.45	0.55	40

Table 8.18: NB with universal sentence encoders with SMLSMOTE

As we can see from Table 8.18, the performance of the score, similar to the previous task when using universal sentence encoders, went down. The decline was steeper, most likely because the KNN error was higher for this case.

Chapter 9

Sentiment Polarity

Polarity analysis is a widely known NLP task that has been applied in various fields. Because of it, various out-of-the-box solutions have already been built. There has not been systematic work that reveals how well they behave in a health-domain task, more precisely in OHCs. We want to verify which tools are better at classifying the content of OHCs. Moreover, we defined baseline models using classical machine learning models with frequency-based techniques and even explored a transfer learning approach using a NN. In Figure 9.1, we show all the approaches explored.

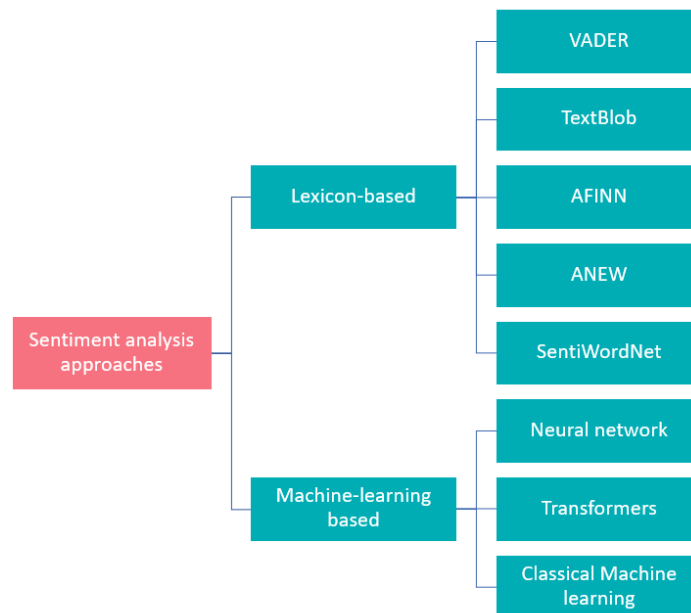


Figure 9.1: Polarity analysis approaches explored over the chapter

9.1 Lexicon-based Approaches

Since these approaches do not require any training data, all the available data we collected acts as the testing dataset. The approach followed was to extract the polarity value of each word in those dictionaries, and in the end, the median or mean could be carried out to get the overall polarity of the written post. Since those dictionaries are not ready to deal with negation in phrases, we follow the same approach as the authors used in [60]. We checked if there is a negation three words before the current word. If there is, the polarity of the current word is inverted. Another disadvantage is that the sentence's valence is limited by the number of words available in the dictionaries.

9.1.1 ANEW

The ANEW version we used was the 2013 expansion one, which contains 13,915 lemmas [115]. Following the rating for each non-stop word, we found the overall sentiment polarity by taking the median or average. It is relevant to notice that if the same lemma appears more than once, it contributes several times to the final decision. Hence, we not only consider the appearance or not appearance of lemmas (binary), but the number of their occurrences (similarly to Bag of Words). Since it is a dictionary of lemmas, after removing the stopwords, we performed lemmatization using the WordNetLemmatizer implemented in the NLTK library. We also used the part-of-speech tag implemented in the NLTK library to aid us in lemmatization. Since lemmas are all lowercased, we performed lowercasing over the whole text. Finally, we invert the polarity of a lemma when dealing with negatives, and for that, we follow the approach aforementioned. We rounded each instance's overall valence using a mean and median strategy using the step function where values up to four are considered negative, greater than six positive, and neutral if between four and six. We chose these values since the paper stated that when an instance is equivalent to five, it should correspond to completely neutral. We gave a broader range to neutral since getting precisely five in a whole instance can be too strict.

9.1.2 VADER

We used the SentimentIntensityAnalyzer to get the polarity scores for each post. The scores are normalized on a scale between -1 and 1. Positive values correspond to a positive valence, negative to negative ones, and values equal to zero or near correspond to a neutral valence. To create a wider boundary for neutral labels, the test set was run several times where the mapping from a score into a neutral label was done under the range [0,1] with a step of 0.05. Moreover, we also tried running the algorithm for the situation of not having to classify any neutral label to see how it behaves. In this case, we did not perform any preprocessing as they should deal with it.

9.1.3 TextBlob

We explored different threshold values to divide the range into classes similarly to VADER. Moreover, we ran the algorithm only considering the positives and negatives instances to see how it

behaves in that situation.

9.1.4 AFINN

We did not remove some medical conditions that should not have polarities, such as "disorder" and "tumor", as Yang et al. [118] did with his modification of this lexical tool - AFINN - since we did not propose the reanalysis of the lexicon dictionaries for the specific domain, as it would take a long time and, in the end, the results could have still been similar. Regarding AFINN, we use a Python library available called `afinn`. We passed as parameter emoticons as they can impact the overall score of a text and should be considered. Moreover, since all the words with scores are lowercased, we performed that as a preprocessing step. The method used is similar to ANEW. Currently, the wordlist approach outputs a number between -5 and 5. The open-source version currently has a vocabulary of over 3,300 lexicons. We experimented with different k values such that $[-k, k]$ correspond to neutral values.

9.1.5 SentiWordNet

After preprocessing the original text (lowercasing and lemmatization), each word's positive and negative scores are considered, and we get an output score. For values around zero, the label is neutral, and the rest of the process is similar to those already described. We also do a part-of-speech tag to verify if the word is a verb, noun, or adverb before getting the synset scores; otherwise, they are ignored. Additionally, if they are not in the synset, they are also ignored. We also experimented with different k values, such that $[-k, k]$ values are classified as neutral.

9.2 Machine learning-based Approaches

In this subsection, we will present the different machine learning models we built to classify messages in terms of their polarity.

9.2.1 Dataset Splitting

We use the Python library Scikit-learn to perform a stratified data split, i.e., the ratio of each label in the training and test set is the one passed as an argument. If we do not do the stratification, the test and training set separation is only done regardless of the number of samples present in a class. For this to work correctly, we cannot have single instances that contain a particular label. The split into training and test sets have an 80:20 ratio.

9.2.2 Baseline Models

After performing data splitting, similarly to the other classification tasks, we perform adversarial validation. Then, we define our baselines by building pipelines that use classical machine learning models with frequency-based text vectorization techniques. In terms of text vectorization, we

change the combinations presented in Table 6.2, and, in terms of fine-tuning parameters, we tried the ones in Table 7.1.

9.2.3 Neural Networks and Transfer Learning

We are leveraging the sentiment140 dataset to perform pre-training of a neural network. Since the dataset only has positive and negative samples, we discard the neutral ones during this task. The architecture of the NN used can be summarized in Figure 9.2.

Layer (type)	Output Shape	Param #
dropout (Dropout)	(None, 512)	0
dense (Dense)	(None, 64)	32832
dropout_1 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 64)	4160
dropout_2 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 64)	4160
dense_3 (Dense)	(None, 1)	65

Figure 9.2: Neural architecture used for transfer learning

The architecture presented in Figure 9.2 has 41,217 trainable parameters, and the Dropout layers act as regularizers. Other relevant configurations used are described in Table 9.1.

Settings	Pre-training	Fine-tuning
number of epochs	25	10
batch size	64	2

Table 9.1: Training configuration

We are also using the Google Sentence Encoder [27] as the input feature of the texts to keep the input size limited to 512 and because they showed that it tends to surpass word embedding performance in transfer learning tasks. Another reason for doing that is because the data length distribution of this dataset, as we can see in Figure 9.3, and ours is too different, so we want a feature extraction method that generalizes well for both (the length of the sample, in this case, are way smaller). Since the dataset has 1.6M instances, we are sampling 100k positive and 100k negative entries.

After that, we partition the dataset into correctly and wrongly classified examples. Half of the wrongly classified ones are part of the training and test set. For the right one, a train-test split of 0.2 is performed.

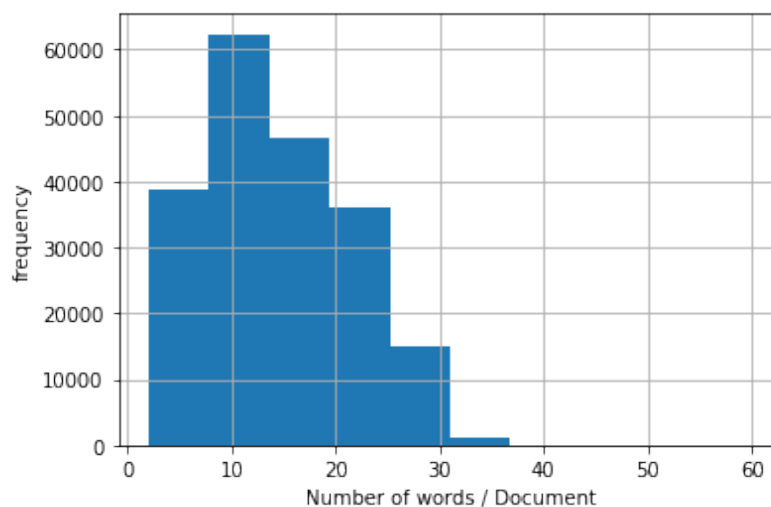


Figure 9.3: Sentiment140 sample word length distribution

By splitting this way, we want to make sure our algorithm learns more than we feed him, as it would not make sense to feed him all the wrong data (it would learn them) or not at all (it may have some problem learning as it can be a batch with different data distribution).

9.3 Transformers and Deep Learning

To use Transformers, we use a python package called `pysentimiento`. They provide state-of-the-art models for emotion and sentiment analysis. They experimented with *Bert-base*, *RoBERTA base*, *BERTweet* and some multilingual model such as *mBERT*. They showed that *BERTweet* was superior and suited for SocialNLP tasks. There is no need to perform any conversion as the classifier already returns a class (contrary to all seen lexicon approaches).

We also experimented with Flair. Flair provides pre-trained not only sentiment analysis models that are also explored but also text embedding representations. The pre-trained model it uses was trained on an IMDB dataset that is quite different from what we are working on, which can yield poor results. However, since the amount of data the model was trained on was quite extensive, it may make up for their domain difference. At the heart of Flair is a contextualized representation called string embeddings. They are obtained by training a large corpus broken into a sequence of characters to pre-train a bidirectional language model that learns embeddings at a character level. By doing this, the model can disambiguate between lower- and upper-case characters (we do not perform lowercasing or any other preprocessing step for these). We can use different word embeddings (such as ELMo) with the strings embeddings makes Flair quite powerful. The power to stack those embeddings into a fixed-size representation comes from the fact that the character-level embeddings can capture latent syntactic-semantic information (without having the notion of a word). In contrast, the pre-trained word embeddings provide the word context. Consequently, the model can identify a wide range of syntactic features in the text, allowing it to perform better

than classical word embedding models. Flair gives us different possibilities: either use an out-of-the-box model or build our own. We used an existing pre-trained model to verify how they behave with our dataset. There are two pre-trained models, both trained in movie and product reviews. One of them is transformer-based – the “en-sentiment” and the other is “RNN-based” – the “sentiment-fast” model, which is usually less performant. Due to the small amount of labeled data, we did not use Flair to create our model. Unfortunately, the sentiment models can only distinguish between positive and negative, so for validation purposes, we only use part of our datasets (we excluded neutral data instances).

9.4 Results and Discussions

In this section, we analyze and compare the different approaches to performing polarity analysis.

9.4.1 ANEW

First, we present the valence distribution of their lexicon dictionary, and similarly to the class distribution of our dataset, it also reveals that the neutral class is more present. However, a disadvantage of a lexicon approach can be seen by the fact that there can be several OOV words. While in the dataset, there are more than 48 thousand words, the vocabulary only finds around 19 thousand, which means that only around 28,7% of the words are taken into account, leading to misleading results.

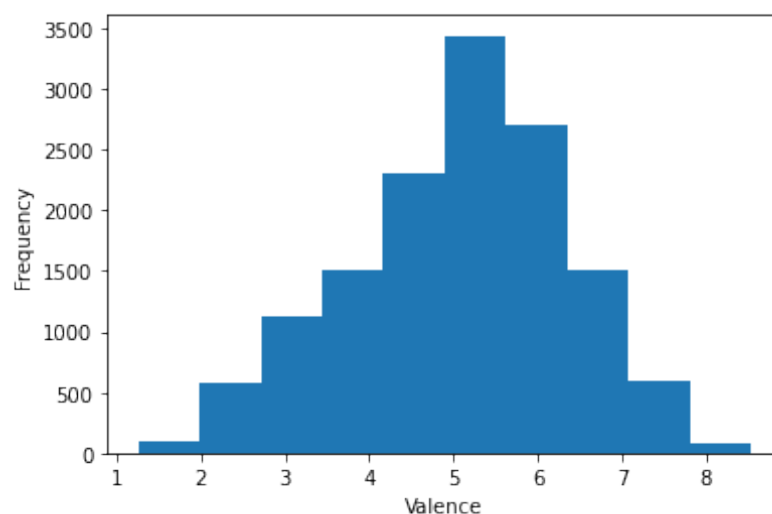


Figure 9.4: Valence distribution of the dataset

We can also analyze their lexicon dataset presented in Table 9.2, and we verify that the majority of lemmas correspond to neutral (around 55%), so it is more and less expected to find more of these lemmas as we saw in Figure 9.4.

The values regarding the mean and median valences found in the lemmas are presented in Table 9.3 and Table 9.4.

Range	Polarity	Number Lemmas
1-4	Negative	2,988
4-6	Neutral	7,865
6-9	Positive	3,347

Table 9.2: ANEW classification report using a mean strategy

ANEW with MEAN	negative	neutral	positive
precision	0.00	0.60	0.41
recall	0.00	0.86	0.47
F1-score	0.00	0.70	0.44
accuracy	0.56		

Table 9.3: ANEW classification report using a median strategy

ANEW with MEDIAN	negative	neutral	positive
precision	0.00	0.58	0.24
recall	0.00	0.74	0.47
F1-score	0.00	0.65	0.32
accuracy	0.50		

Table 9.4: ANEW lemmas frequency and their polarity

Both behave very poorly when identifying negative samples, not being able to find them at all. Both can identify the majority of neutral samples. However, this optimism may come from the fact that they are more biased toward neutral lemmas, and so does our dataset. Regarding positive samples, the mean approach is far better than the median when finding positive samples, both in precision and recall (and hence in F1-score). In terms of accuracy, the mean approach is a bit better than the median because it is mainly better at identifying positive instances.

9.4.2 AFINN

Similar to ANEW, we can start by analyzing the Valence distribution over their dataset.

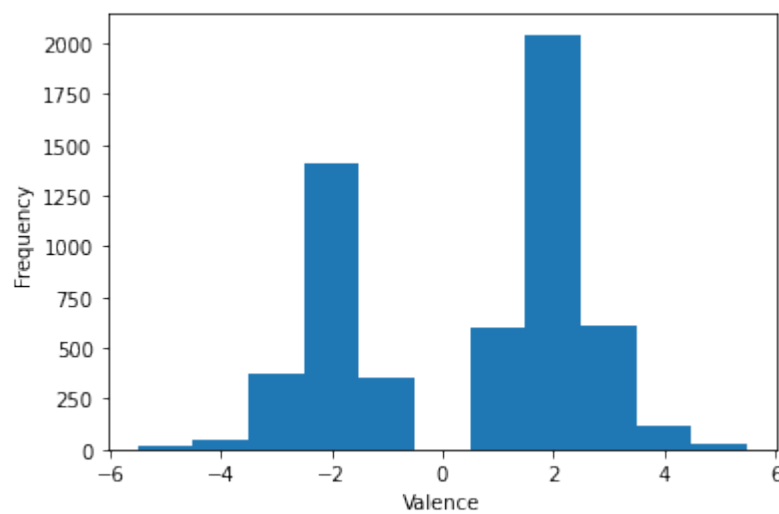


Figure 9.5: Valence distribution of the dataset

Instead of following a normal distribution like ANEW, the word list looks like it has two of them where valences of -2 and 2 are the most frequent. Curiously, the word list does not have any "pure" neutral values, i.e., whose value is zero. We explored different threshold values (between zero and four) for the classification, and we can see both the F1-score and accuracy in the following Image 9.6.

We can observe that a threshold of zero is undoubtedly the best option in terms of both analyzed metrics. We can see the classification report for the best threshold in Table 9.5.

AFINN	negative	neutral	positive
precision	0.48	0.73	0.20
recall	0.60	0.19	0.84
f1-score	0.53	0.30	0.32
accuracy	0.38		

Table 9.5: AFINN classification report for a threshold of zero

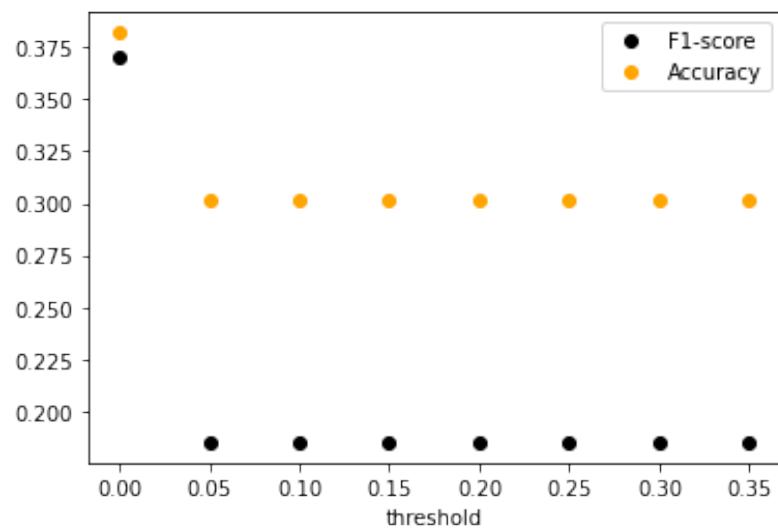


Figure 9.6: Accuracy and weighted F1-score for different threshold values

Compared to ANEW, the negative class is better identified. However, the overall accuracy of the classifier dropped mainly due to the low recall and high precision of the "neutral" class and the high recall and low precision of "positive" labels. This means that a great number of neutral labels are being classified as positive.

9.4.3 VADER

Since we did not have access to VADER's word list, we start by analyzing how the classifier's performance evolves over different threshold levels, as shown in Figure 9.7.

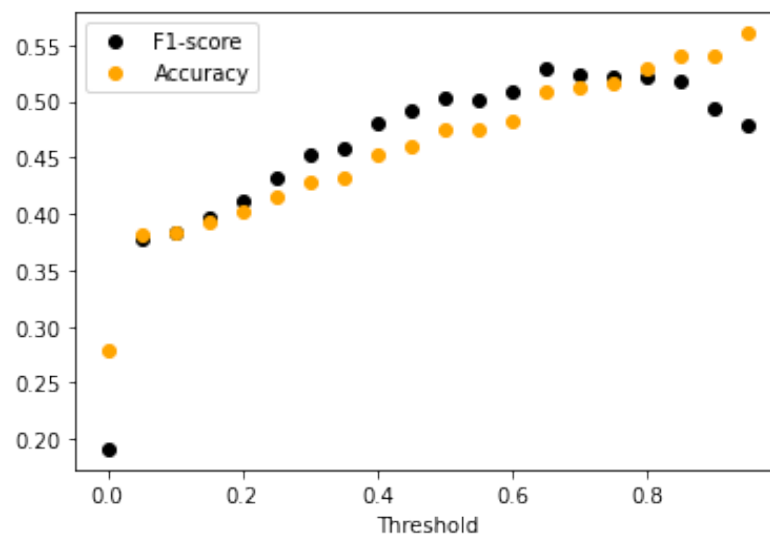


Figure 9.7: Accuracy and weighted F1-score for different threshold values of VADER

According to the chart in Figure 9.7, the best threshold value considering the F1-score weighted is 0.65. For that threshold value, we get the following classification report 9.6.

VADER	negative	neutral	positive
precision	0.63	0.34	0.44
recall	0.66	0.58	0.62
f1-score	0.21	0.60	0.31
accuracy	0.51		

Table 9.6: VADER classification report for a threshold of 0.65

Even though we get more than half of the samples correct with an F1-score of 0.53, we argue that the threshold value is biased towards the neutral values since its value is relatively high. To confirm that, we decided to perform the same analysis with our dataset, excluding the neutral samples. The chosen threshold value dropped to 0.3, and the new classification report is presented in Figure 9.7.

VADER	negative	positive
precision	0.91	0.54
recall	0.73	0.82
F1-score	0.81	0.65
accuracy	0.75	

Table 9.7: VADER classification report for a threshold of 0.3

This shows that the value of k is dependent on the number of samples we have on each class, and it acts to compensate for the lack of generalization the classifier has for the domain. Nevertheless, this classifier is quite capable compared to other lexicon approaches, such as ANEW or AFINN.

9.4.4 TextBlob

Similar to VADER, we start by analyzing how the performance of the classifier changes with different threshold values.

As seen in Figure 9.8, the accuracy increases with the threshold value. Nevertheless, we see that the F1-score peaks at the threshold value of 0.25. This means that the accuracy is only growing since our dataset is imbalanced. Considering the best classifier as the one with the highest F1-score, we obtain the classification report in Table 9.8.

We can see in Table 9.8 that the algorithm has difficulties finding negative samples in this setting compared to neutral and positive samples. This is rather interesting as there are fewer positive than negative samples, yet it is harder for the classifier to classify correctly the latter. However, this can be a side-effect of having too many neutral samples. If we perform the same process ignoring the neutral instances, we obtain the Table 9.9 for the best threshold experimented.

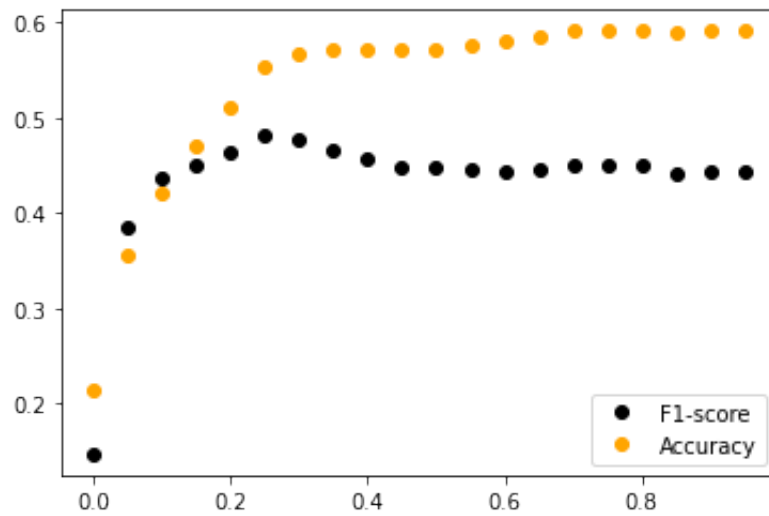


Figure 9.8: Accuracy and weighted F1-score for different threshold values of TextBlob

TextBlob	negative	neutral	positive
precision	0.40	0.61	0.33
recall	0.04	0.82	0.49
f1-score	0.08	0.70	0.40
accuracy	0.55		

Table 9.8: TextBlob classification report for a threshold of 0.25

TextBlob	negative	positive
precision	0.86	0.71
recall	0.90	0.62
F1-score	0.88	0.66
accuracy	0.82	

Table 9.9: Classification report for positive and negative samples using TextBlob

By observing Table 9.9, we could confirm our hypothesis. In this setting, the algorithm for our dataset seems to behave better than VADER.

9.4.5 SentiWordNet

By analyzing Figure 9.9, the threshold value that gives the best F1-score is 0.4.

The classification report for that threshold is presented in Table 9.10.

We can verify that the algorithm is more biased towards the neutral class. Both positive and negative classifications are not that great. Even if we ignore the neutral class (after finding the best k value, which was -0.2 in order to find the best F1-score), the results are not that phenomenal since we observe that while the negative class has high precision and lower recall, the positive class has the opposite situation as we can see in the Table 9.11.

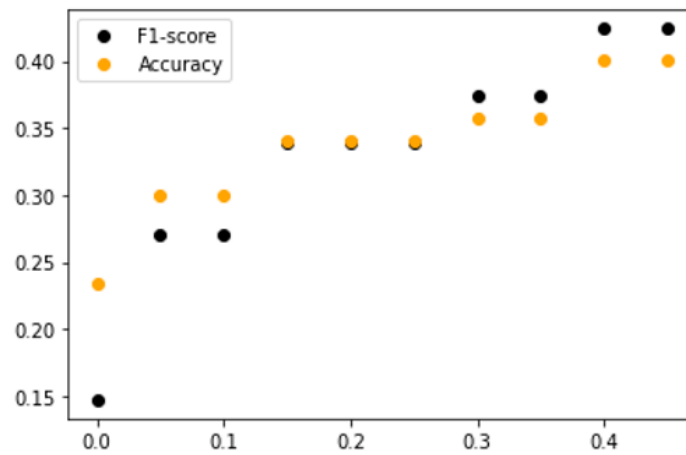


Figure 9.9: Accuracy and weighted F1-score for different threshold values of SentiWordNet

SentiWordNet	negative	neutral	positive
precision	0.37	0.63	0.18
recall	0.37	0.39	0.55
F1-score	0.37	0.48	0.27
accuracy	0.40		

Table 9.10: Classification report of SentiWordNet

SentiWordNet	negative	positive
precision	0.91	0.39
recall	0.48	0.87
F1-score	0.63	0.54
accuracy	0.59	

Table 9.11: Classification report of SentiWordNet for positive and negative classes

Overall, this lexicon approach does not seem to generalize well for our dataset.

9.4.6 Baseline Models

In this subsection, we show the performance of the models we trained.

9.4.6.1 Adversarial Validation

Binary classifiers	Mean	Standard Deviation
Random Forest	0.502	0.049
SVM (linear)	0.467	0.063

Table 9.12: Adversarial Task on BOW for polarity analysis

Binary classifiers	Mean	Standard Deviation
Random Forest	0.497	0.030
SVM (linear)	0.494	0.058

Table 9.13: Adversarial Task with TF-IDF vectorization for polarity analysis

According to Table 9.12 and Table 9.13 (using BOW and TF-IDF vectorization), the training and test set follow a similar distribution since a binary classifier would not be able to distinguish between them.

9.4.6.2 Frequency-based Feature Extraction with Classical Machine Learning

For this approach, the preprocessing steps we took are the ones we took for all the other tasks, including pattern Substitution. Since lowercasing and punctuation can be relevant, we did not remove them. As we are also defining baselines, we did not perform any stop word removal or text normalization based on lemmatization or stemming. Nevertheless, we will show the comparison between performing preprocessing or not.

We concluded that using words as tokens is preferred in terms of textual vectorization. Moreover, the n-grams whose range is more than a single word are commonly favored. In addition, TF-IDF vectorization, in general, is associated with better performance than BOW. We can also compare the performance of the different algorithms in the test set. The full results regarding which parameter set is the best for each classifier (when also changing the text vectorization parameters), with processing and not, can be seen in the Appendix A.3.

SVM	negative	neutral	positive
precision	0.66	0.79	0.50
recall	0.86	0.76	0.18
F1-score	0.75	0.77	0.27
accuracy	0.72		

Table 9.14: SVM classification report with preprocessing

Regarding the Multinomial NB, although it seemed that the algorithm got a bit worse with preprocessing, getting lower F1-scores and accuracy in comparison, this is not entirely true. Even though its classification performance declined when trying to find negative and neutral instances, text preprocessing can correctly predict some positive instances, which was impossible without it.

In Decision Tree, we concluded that the overall performance when discriminating between classes improved with preprocessing, achieving both higher F1-scores and accuracy.

Regarding KNN and Gaussian NB, both were affected negatively by performing preprocessing, but the classifiers could not classify positive instances correctly under any of the circumstances.

In SVM's case, not only did the discrimination of negatives and neutral instances improve with preprocessing, but also the positives, whose classifier could not classify.

In terms of Logistic Regression and Random Forest, both performances remained almost the same, with preprocessing aiding the discrimination of positives instances when using Logistic Regression.

Overall, preprocessing improved the performance of the algorithms, especially regarding positive instances. SVM plus TF-IDF vectorization with got overall the best performance, as seen in Table 9.14.

9.4.7 Neural Network and Transfer Learning

First, we train the neural network with the 200 thousand records using the universal sentence encoder to create a fixed size input of 512 for the neural network. Then, we can verify how well this classifier already classifies our positive and negative samples. The classification report is seen in Table 9.15.

Sentiment 140	negative	positive
precision	0.86	0.77
recall	0.93	0.62
F1-score	0.89	0.69
accuracy	0.84	

Table 9.15: Neural network classification report on negative and positive samples of our dataset

This shows the capacity the universal sentence embeddings have. For the same task but a rather different data distribution than ours, it was capable of achieving good results, surpassing any of the lexicons methods. Moreover, it also shows how important a great amount of data for a machine learning model is.

Next, we convert our training data and test data into sentence embeddings as described previously, and by feeding the training set to the NN, we can verify if the generalization capacity of our classifier improved. The evaluation metrics are presented in Table 9.16.

Settings	negative	positive
precision	0.91	0.86
recall	0.94	0.8
F1-score	0.93	0.83
accuracy	0.90	

Table 9.16: Neural network classification report after fine-tuning

Comparing Table 9.15 and Table 9.16, we can see that there was an improvement in both classes.

9.4.8 Transformers and Deep Learning

In this section, we will experiment with *BERTweet* to verify how well it generalizes for our dataset and other models included in the package Flair.

PYsentimiento	negative	neutral	positive
precision	0.54	0.75	0.57
recall	0.75	0.63	0.56
F1-score	0.62	0.70	0.57
accuracy	0.66		

Table 9.17: Classification report of *BerTweet*

The table 9.17 shows the generalization capabilities of the model in the health-domain. Its rather good performance has likely to do with the fact that it was training using tweets (social media text).

Now, we present how good the pre-trained models in Flair perform on our dataset.

By analysing Table 9.18 and Table 9.19, We can conclude that the transformer-based model from Flair is slightly better than the RNN-based one. Nevertheless, both models struggle the most with positive classes. Any of these models are superior to any of the lexicon-based models presented previously.

Word Embeddings en-sentiment	negative	positive
precision	0.88	0.70
recall	0.89	0.69
F1-score	0.88	0.70
accuracy	0.83	

Table 9.18: Classification report of transformer-based model *en – sentiment*

Word Embeddings sentiment-fast	negative	positive
precision	0.87	0.60
recall	0.82	0.67
F1-score	0.84	0.63
accuracy	0.78	

Table 9.19: Classification report of RNN-based model *en-fast*

Chapter 10

Conclusions and Future Work

After gathering data instances from MedHelp, a health-related network divided into distinct groups, we built a crowdsourcing platform to expand a dataset specialized for OHC content. We experimented with frequency-based feature extraction approaches, pre-trained words, and phrase embeddings in multiple pipelines.

Considering only small datasets (below 1,500 instances), seen in the Literature Review, except for sentiment analysis, whose number of classes is at least three, the F1-scores vary between 59.5% and 79.5%, considering that the majority of them are a multiclass problem. In a multi-label setting with just below 500 instances, for a three-class multi-label problem, we were able to obtain an F1-score of 82%. On the one hand, we have fewer instances and a multi-label setting. On the other hand, we are dealing with fewer classes. Nevertheless, we attained superior F1-scores. The class that, in general, all the pipelines struggled the most with was "seeking support", which corresponds to the one that has fewer instances. Although we achieved rivaling results compared to the existing literature, it is still quite relevant to be aware that the inherent nature of each categorization also influences the performance, since we are dealing with a different set of classes.

The second-level categorization was far more imbalanced. The classes that only had around 30 instances could not be correctly distinguished in general (specifically network support and esteem support). Apart from this, the ones around and with more than 100 instances achieved F1-scores above 60 and 70%, comparable to those found in the literature. Overall, our highest weighted F1-score (62%) was achieved when using Logistic Regression with n binary classifiers for the n classes together with Universal Sentence Encoders.

Regarding polarity analysis, lexicon-based approaches showed the poorest performance among all of the tested methods. More complex ones based on classical machine learning models or transformers were undoubtedly more successful. *BERTweet* was the out-of-the-box approach whose performance was the highest, achieving an accuracy of 66% when distinguishing between the three polarity classes. Nevertheless, our best pipeline, using SVM with TF-IDF vectorization, achieved the highest accuracy, surpassing the 72% mark. Even so, when distinguishing positive messages, *BERTweet* was still able to outperform our baselines. Overall, all the classifiers struggled the most when dealing with positive instances.

The reason why models may not work very well with positive samples in our context is that posts often describe issues and illnesses, which can lead to a more negative-biased classifier, when it does not have to be that way. This has to do with the health-related nature of the data. In terms of the social-media side of the data, there exists plenty of solutions and models which already consider that.

If dealing with only two polarities (positive and negative), our best approach consisted of training an initial neural network with 200 thousand cases from Sentiment140, feeding as input the Universal Sentence Encoding to those. Next, we partitioned our dataset and trained the neural networks with our data, achieving an accuracy of 90%. The length distribution of both datasets was very distinct. Nevertheless, the Universal Sentence Encoder was still able to overcome that obstacle, showcasing yet again its capabilities.

We showed the importance of preprocessing when using BOW or TF-IDF vectorization techniques for both first- and second-level categorizations. This was done by showing gains of over 5% in the F1-score metric depending on the combination of techniques used, with the stop-word list being the most impactful factor.

For the first-level categorization, regarding word embeddings, fastText was usually superior to others with values equated to frequency-based ones. We also tried Universal Sentence Encoders for each classifier, and they were consistently the best. Furthermore, calculating the mean of embedded sentences instead of encoding the whole text was generally the preferred approach. Besides this, they do not require preprocessing, which is also a huge advantage.

Regarding neural networks, the values obtained with all the embeddings were not very discrepant since they all ranged the weighted F1-score between 70 and 76%, being the highest score associated with both FastText and Universal Sentence Encoders (with mean). SVM and Logistic Regression obtained the best performance across different feature input types. Finally, the values obtained with the BiLSTM were similar to those of the best neural networks.

Regarding the second-level categorization, word embeddings in conjunction with the best classifiers (SVM and Logistic Regression) were able to accomplish better results than frequency-based text vectorization, surpassing the 60% mark for the weighted F1-score.

Regarding PoS tagging as feature engineering and MLSMOTE, we concluded that it is only worth using it with frequency-based feature extraction, in both aforementioned tasks (even though the performance does not surpass the best pipelines). Moreover, attributing weights to classes and performing threshold moving, aided us in combating the dataset imbalance.

It is essential to notice that it may be relevant to convey more importance to recall or precision depending on the category, especially for seeking support and offering support. To ensure that none of the people asking for support is neglected, we can give greater importance to recall, in the detriment of precision. This can be tuned by, for example, threshold moving. Doing the same, but the other way around, might make more sense if we are looking for answers: we may want to look for a message that we are sure is conveying some knowledge. As Future work, we can use this classification to distinguish where people offer support so as to perform knowledge extraction. This could contribute to collecting new information about specific health problems

and shared concerns in online communities. Moreover, as future work, we also consider relevant the expansion of the dataset, especially if we want to improve the second-level classifier. New techniques such as Virtual Adversarial Training and co-training multi-label algorithms can also be explored. Furthermore, we can also try to use autoencoders to diminish the size of the embeddings, while training them on OHC data. This way, we can not only reduce the input size, but we might also improve performance. Additionally, as our classifier gets better, we can use them to filter data before performing data labeling. We can attempt to select data that belongs to the minority classes, or we can only leave human annotation to those samples whose class confidence is low.

References

- [1] Ahmed Abbasi, Donald Adjeroh, Mark Dredze, Michael J. Paul, Fatemeh Mariam Zahedi, Huimin Zhao, Nitin Walia, Hemant Jain, Patrick Sanvanson, Reza Shaker, Marco D. Huesch, Richard Beal, Wanhong Zheng, Marie Abate, and Arun Ross. Social media analytics for smart health. *IEEE Intelligent Systems*, 29(2):60–80, 2014.
- [2] Cosme Adrover, Todd Bodnar, Zhuojie Huang, Amalio Telenti, and Marcel Salathé. Identifying adverse effects of hiv drug treatment and associated sentiments using twitter. *JMIR public health and surveillance*, 1(2):e7, 2015.
- [3] Ahmet Emre Aladağ, Serra Muderrisoglu, Naz Berfu Akbas, Oguzhan Zahmacioglu, and Haluk O Bingol. Detecting suicidal ideation on forums: proof-of-concept study. *Journal of medical Internet research*, 20(6):e9840, 2018.
- [4] Harini Alagarai Sampath, Rajeev Rajeshuni, and Bipin Indurkha. Cognitively inspired task design to improve user performance on crowdsourcing platforms. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3665–3674, 2014.
- [5] Maryam Mohammed Aldarwish and Hafiz Farooq Ahmad. Predicting depression levels using social media posts. In *2017 IEEE 13th international Symposium on Autonomous decentralized system (ISADS)*, pages 277–280. IEEE, 2017.
- [6] Mark Alfano, Dirk Hovy, Margaret Mitchell, and Michael Strube. Proceedings of the second acl workshop on ethics in natural language processing. In *Proceedings of the Second ACL Workshop on Ethics in Natural Language Processing*, 2018.
- [7] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2):76–81, 2013.
- [8] Jon-Patrick Allem, Likhith Dharmapuri, Jennifer B Unger, and Tess Boley Cruz. Characterizing juul-related posts on twitter. *Drug and alcohol dependence*, 190:1–5, 2018.
- [9] Rana Alnashwan, Humphrey Sorensen, Adrian O’Riordan, and Cathal Hoare. Accurate classification of socially generated medical discourse. *International Journal of Data Science and Analytics*, 8(4):353–365, 2019.
- [10] Yin Aphinyanaphongs, Armine Lulejian, Duncan Penfold Brown, Richard Bonneau, and Paul Krebs. Text classification for automatic detection of e-cigarette use and use for smoking cessation from twitter: a feasibility pilot. In *Biocomputing 2016: Proceedings of the Pacific Symposium*, pages 480–491. World Scientific, 2016.
- [11] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*, 2017.

- [12] Ghelmar Astoveza, Randolph Jay P Obias, Roi Jed L Palcon, Ramon L Rodriguez, Bernie S Fabito, and Manolito V Octaviano. Suicidal behavior detection on twitter using neural network. In *TENCON 2018-2018 IEEE Region 10 Conference*, pages 0657–0662. IEEE, 2018.
- [13] B Athira, Josette Jones, Sumam Mary Idicula, Anand Kulanthaivel, and Enming Zhang. Annotating and detecting topics in social media forum and modelling the annotation to derive directions-a case study. *Journal of Big Data*, 8(1):1–23, 2021.
- [14] Yi Ji Bae, Midan Shim, and Won Hee Lee. Schizophrenia detection using machine learning approach from social media content. *Sensors*, 21(17):5924, 2021.
- [15] Adrian Benton, Glen Coppersmith, and Mark Dredze. Ethical research protocols for social media health research. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 94–102, 2017.
- [16] Adrian Benton, Lyle Ungar, Shawndra Hill, Sean Hennessy, Jun Mao, Annie Chung, Charles E Leonard, and John H Holmes. Identifying potential adverse effects using the web: a new approach to medical hypothesis generation. *Journal of biomedical informatics*, 44(6):989–996, 2011.
- [17] Magdalena Berger, Todd H Wagner, and Laurence C Baker. Internet use and stigmatized illness. *Social science & medicine*, 61(8):1821–1827, 2005.
- [18] Margaret M Bradley and Peter J Lang. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Technical report C-1, the center for research in psychophysiology . . . , 1999.
- [19] Lucas Andreas Braun, Bijan Zomorodbakhsch, Christian Keinki, and Jutta Huebner. Information needs, communication and usage of social media by cancer patients and their relatives. *Journal of cancer research and clinical oncology*, 145(7):1865–1875, 2019.
- [20] Samuel Brody and Nicholas Diakopoulos. Coooooooooooooooooo!!!!!!!!!!!!!! using word lengthening to detect sentiment in microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 562–570, 2011.
- [21] Ruichu Cai, Binjun Zhu, Lei Ji, Tianyong Hao, Jun Yan, and Wenying Liu. An cnn-lstm attention approach to understanding user query intent from online health communities. In *2017 ieee international conference on data mining workshops (icdmw)*, pages 430–437. IEEE, 2017.
- [22] Andrew Caines, Sergio Pastrana, Alice Hutchings, and Paula J Buttery. Automatically identifying the function and intent of posts in underground forums. *Crime Science*, 7(1):1–14, 2018.
- [23] Jose Camacho-Collados and Mohammad Taher Pilehvar. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. *arXiv preprint arXiv:1707.01780*, 2017.
- [24] Herman Anthony Carneiro and Eleftherios Mylonakis. Google trends: a web-based tool for real-time surveillance of disease outbreaks. *Clinical infectious diseases*, 49(10):1557–1564, 2009.

- [25] Patricia A Cavazos-Rehg, Melissa J Krauss, Shaina Sowles, Sarah Connolly, Carlos Rosas, Meghana Bharadwaj, and Laura J Bierut. A content analysis of depression-related tweets. *Computers in human behavior*, 54:351–357, 2016.
- [26] Pew Research Center. Social Media Use in 2021. April 2021.
- [27] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.
- [28] Kai-Wei Chang, Vinod Prabhakaran, and Vicente Ordonez. Bias and fairness in natural language processing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): Tutorial Abstracts*, 2019.
- [29] Francisco Charte, Antonio J Rivera, María J del Jesus, and Francisco Herrera. Mlsmote: Approaching imbalanced multilabel learning through synthetic instance generation. *Knowledge-Based Systems*, 89:385–397, 2015.
- [30] Qijin Cheng, Tim MH Li, Chi-Leung Kwok, Tingshao Zhu, and Paul SF Yip. Assessing suicide risk and emotional distress in chinese social media: a text mining and machine learning study. *Journal of medical internet research*, 19(7):e243, 2017.
- [31] Jackie Chi Kit Cheung and Xiao Li. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 383–392, 2012.
- [32] Fatima Chiroma, Han Liu, and Mihaela Cocea. Text classification for suicide related tweets. In *2018 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 2, pages 587–592. IEEE, 2018.
- [33] Wen-ying Sylvia Chou, Abby Prestin, Claire Lyons, and Kuang-yi Wen. Web 2.0 for health promotion: reviewing the current evidence. *American journal of public health*, 103(1):e9–e18, 2013.
- [34] Nigel Collier, Nguyen Truong Son, and Ngoc Mai Nguyen. Omg u got flu? analysis of shared health messages for bio-surveillance. *Journal of biomedical semantics*, 2(5):1–10, 2011.
- [35] Glen Coppersmith, Ryan Leary, Patrick Crutchley, and Alex Fine. Natural language processing of social media as screening for suicide risk. *Biomedical informatics insights*, 10:1178222618792860, 2018.
- [36] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.
- [37] Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR)*, 51(1):1–40, 2018.
- [38] Kathryn P Davison, James W Pennebaker, and Sally S Dickerson. Who talks? the social psychology of illness support groups. *American Psychologist*, 55(2):205, 2000.

- [39] Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. Predicting depression via social media. In *Seventh international AAAI conference on weblogs and social media*, 2013.
- [40] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions. *arXiv preprint arXiv:2005.00547*, 2020.
- [41] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [42] Kodati Dheeraj and Tene Ramakrishnudu. Negative emotions detection on online mental-health related patients texts using the deep learning with mha-bcnn model. *Expert Systems with Applications*, page 115265, 2021.
- [43] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.
- [44] John W Ely, Jerome A Osheroff, Mark H Ebell, George R Bergus, Barcey T Levy, M Lee Chambliss, and Eric R Evans. Analysis of questions asked by family doctors regarding patient care. *Bmj*, 319(7206):358–361, 1999.
- [45] Elena V Epure, Dario Compagno, Camille Salinesi, Rebecca Deneckere, Marko Bajec, and Slavko Žitnik. Process models of interrelated speech intentions from online health-related conversations. *Artificial Intelligence in Medicine*, 91:23–38, 2018.
- [46] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the fifth international conference on language resources and evaluation (LREC’06)*, 2006.
- [47] Gunther Eysenbach. The impact of the internet on cancer outcomes. *CA: A cancer journal for clinicians*, 53(6):356–371, 2003.
- [48] George Forman et al. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3(Mar):1289–1305, 2003.
- [49] Susannah Fox. The social life of health information, 2014.
- [50] Shreya Ghosh and Tarique Anwar. Depression intensity estimation via social media: A deep learning approach. *IEEE Transactions on Computational Social Systems*, 2021.
- [51] Rachel Ginn, Pranoti Pimpalkhute, Azadeh Nikfarjam, Apurv Patki, Karen O’Connor, Abeed Sarker, Karen Smith, and Graciela Gonzalez. Mining twitter for adverse drug reaction mentions: a corpus and classification benchmark. In *Proceedings of the fourth workshop on building and evaluating resources for health and biomedical text processing*, pages 1–8. Citeseer, 2014.
- [52] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.
- [53] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

- [54] Vairaprakash Gurusamy and Subbu Kannan. Preprocessing techniques for text mining. In *Conference Paper RTRICS, India*, 2014.
- [55] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.
- [56] Anam Habib, Nosheen Jelani, Asad Masood Khattak, Saima Akbar, and Muhammad Zubair Asghar. Exploiting deep neural networks for intention mining. In *Proceedings of the 2020 9th international conference on software and computer applications*, pages 26–30, 2020.
- [57] Andrea Hartzler, Wanda Pratt, et al. Managing the personal side of health: how patient expertise differs from the expertise of clinicians. *Journal of medical Internet research*, 13(3):e1728, 2011.
- [58] Md Rahat Hossain, Amanullah Maung Than Oo, and ABMS Ali. The combined effect of applying feature selection and parameter optimization on machine learning techniques for solar power prediction. *American Journal of Energy Research*, 1(1):7–16, 2013.
- [59] Dirk Hovy and Shannon L Spruit. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, 2016.
- [60] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.
- [61] Keyuan Jiang and Yujing Zheng. Mining twitter data for potential drug effects. In *International conference on advanced data mining and applications*, pages 434–443. Springer, 2013.
- [62] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.
- [63] Ramakanth Kavuluru and AKM Sabbir. Toward automated e-cigarette surveillance: Spotting e-cigarette proponents on twitter. *Journal of biomedical informatics*, 61:19–26, 2016.
- [64] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. The face of quality in crowdsourcing relevance labels: Demographics, personality and labeling accuracy. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2583–2586, 2012.
- [65] Rachel Lynn Kendra, Suman Karki, Jesse Lee Eickholt, and Lisa Gandy. Characterizing the discussion of antibiotics in the twittersphere: what is the bigger picture? *Journal of medical Internet research*, 17(6):e4220, 2015.
- [66] Roman Khazankin, Benjamin Satzger, and Schahram Dustdar. Optimized execution of business processes on crowdsourcing platforms. In *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 443–451. IEEE, 2012.
- [67] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

- [68] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273–324, 1997. Relevance.
- [69] Ioannis Korkontzelos, Azadeh Nikfarjam, Matthew Shardlow, Abeer Sarker, Sophia Ananiadou, and Graciela H Gonzalez. Analysis of the effect of sentiment analysis on extracting adverse drug reactions from tweets and forum posts. *Journal of biomedical informatics*, 62:148–158, 2016.
- [70] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.
- [71] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.
- [72] Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wieggers, and Zhiyong Lu. BioCreative v CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016:baw068, 2016.
- [73] Xiao Li. Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1337–1345, 2010.
- [74] Shengbin Liang, Xinan Chen, Jixin Ma, Wencai Du, and Huawei Ma. An improved double channel long short-term memory model for medical text classification. *Journal of Healthcare Engineering*, 2021, 2021.
- [75] Yang Liu, Songhua Xu, Hong-Jun Yoon, and Georgia Tourassi. Extracting patient demographics and personal medical information from online health forums. In *AMIA Annual Symposium Proceedings*, volume 2014, page 1825. American Medical Informatics Association, 2014.
- [76] Carla Teixeira Lopes and Bárbara Guimarães Da Silva. A classification scheme for analyses of messages exchanged in online health forums. 2019.
- [77] Thai-Le Luong, Minh-Son Cao, Duc-Thang Le, and Xuan-Hieu Phan. Intent extraction from social media texts using sequential segmentation and deep learning models. In *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*, pages 215–220. IEEE, 2017.
- [78] Thai-Le Luong, Nhu-Thuat Tran, and Xuan-Hieu Phan. Improving intent extraction using ensemble neural network. In *2019 19th International Symposium on Communications and Information Technologies (ISCIT)*, pages 58–63. IEEE, 2019.
- [79] Philip M Massey, Amy Leader, Elad Yom-Tov, Alexandra Budenz, Kara Fisher, and Ann C Klassen. Applying multiple data collection tools to quantify human papillomavirus vaccine communication on twitter. *Journal of medical Internet research*, 18(12):e318, 2016.
- [80] Mauro Mazzocut, Ivana Truccolo, Marialuisa Antonini, Fabio Rinaldi, Paolo Omero, Emanuela Ferrarin, Paolo De Paoli, and Carlo Tasso. Web conversations about complementary and alternative medicines and cancer: content and sentiment analysis. *Journal of medical Internet research*, 18(6):e5521, 2016.

- [81] Mary L McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282, 2012.
- [82] Jude Mikal, Samantha Hurst, and Mike Conway. Ethical issues in using twitter for population-level depression monitoring: a qualitative study. *BMC medical ethics*, 17(1):1–11, 2016.
- [83] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [84] Dhiaa A Musleh, Taef A Alkhales, Reem A Almakki, Shahad E Alnajim, Shaden K Almarshad, Rana S Alhasaniah, Sumayh S Aljameel, and Abdullah A Almuqhim. Twitter arabic sentiment analysis to detect depression using machine learning. *CMC-COMPUTERS MATERIALS & CONTINUA*, 71(2):3463–3477, 2022.
- [85] Moin Nadeem. Identifying depression on twitter. *arXiv preprint arXiv:1607.07384*, 2016.
- [86] Markus Näther. An in-depth comparison of 14 spelling correction tools on a common benchmark. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1849–1857, 2020.
- [87] Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.
- [88] Azadeh Nikfarjam, Abeed Sarker, Karen O’connor, Rachel Ginn, and Graciela Gonzalez. Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 22(3):671–681, 2015.
- [89] Bridianne O’dea, Stephen Wan, Philip J Batterham, Alison L Cate, Cecile Paris, and Helen Christensen. Detecting suicidality on twitter. *Internet Interventions*, 2(2):183–188, 2015.
- [90] Bahadorreza Ofoghi, Meghan Mann, and Karin Verspoor. Towards early discovery of salient health threats: A social media emotion classification technique. In *biocomputing 2016: proceedings of the Pacific symposium*, pages 504–515. World Scientific, 2016.
- [91] World Health Organization et al. Depression and other common mental disorders: global health estimates. Technical report, World Health Organization, 2017.
- [92] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [93] Sebastián Pérez-Vera, Rodrigo Alfaro, and Héctor Allende-Cid. Intent classification of social media texts with machine learning for customer service improvement. In *International Conference on Social Computing and Social Media*, pages 258–274. Springer, 2017.
- [94] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

- [95] Tu Minh Phuong, Zhen Lin, and Russ B Altman. Choosing snps using feature selection. In *2005 IEEE Computational Systems Bioinformatics Conference (CSB'05)*, pages 301–309. IEEE, 2005.
- [96] Hemant Purohit, Guozhu Dong, Valerie Shalin, Krishnaprasad Thirunarayan, and Amit Sheth. Intent classification of short-text on social media. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pages 222–228. IEEE, 2015.
- [97] Yvan Saeys, Inaki Inza, and Pedro Larranaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.
- [98] Abeed Sarker and Graciela Gonzalez. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. *Journal of biomedical informatics*, 53:196–207, 2015.
- [99] Abeed Sarker, Azadeh Nikfarjam, and Graciela Gonzalez. Social media mining shared task workshop. In *Biocomputing 2016: Proceedings of the Pacific Symposium*, pages 581–592. World Scientific, 2016.
- [100] Samer Muthana Sarsam, Hosam Al-Samarraie, Ahmed Ibrahim Alzahrani, Waleed Alnumay, and Andrew Paul Smith. A lexicon-based approach to detecting suicide-related messages on twitter. *Biomedical Signal Processing and Control*, 65:102355, 2021.
- [101] Ranjan Satapathy, Claudia Guerreiro, Iti Chaturvedi, and Erik Cambria. Phonetic-based microtext normalization for twitter sentiment analysis. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 407–413. IEEE, 2017.
- [102] Ramit Sawhney, Prachi Manchanda, Raj Singh, and Swati Aggarwal. A computational approach to feature extraction for identification of suicidal ideation in tweets. In *Proceedings of ACL 2018, Student Research Workshop*, pages 91–98, 2018.
- [103] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 145–158. Springer, 2011.
- [104] Tiancheng Shen, Jia Jia, Guangyao Shen, Fuli Feng, Xiangnan He, Huanbo Luan, Jie Tang, Thanassis Tiropanis, Tat Seng Chua, and Wendy Hall. Cross-domain depression detection via harvesting social media. *International Joint Conferences on Artificial Intelligence*, 2018.
- [105] Boaz Shmueli, Jan Fell, Soumya Ray, and Lun-Wei Ku. Beyond fair pay: Ethical implications of nlp crowdsourcing. *arXiv preprint arXiv:2104.10097*, 2021.
- [106] Bárbara Silveira, Henrique S Silva, Fabricio Murai, and Ana Paula C da Silva. Predicting user emotional tone in mental disorder online communities. *Future Generation Computer Systems*, 125:641–651, 2021.
- [107] Jasmeet Singh and Vishal Gupta. A systematic review of text stemming techniques. *Artificial Intelligence Review*, 48(2):157–217, 2017.
- [108] Russel Smith. Five reasons to ditch manual data classification methods, 2018.
- [109] James Surowiecki. *The wisdom of crowds*. Anchor, 2005.

- [110] Nhu Thuat Tran, Huu Hong Nguyen, Thanh Tung Nguyen, Tien Son Dang, Thai Le Luong, and Xuan Hieu Phan. Domain-independent intent extraction from online texts. Technical report, VNU-UET, 2018.
- [111] Marcel Trotzek, Sven Koitka, and Christoph M Friedrich. Utilizing neural networks and linguistic metadata for early detection of depression indications in text sequences. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):588–601, 2018.
- [112] Marcel Trotzek, Sven Koitka, and Christoph M. Friedrich. Utilizing neural networks and linguistic metadata for early detection of depression indications in text sequences. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):588–601, 2020.
- [113] S Vijayarani, Ms J Ilamathi, Ms Nithya, et al. Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2015.
- [114] Jinpeng Wang, Gao Cong, Xin Wayne Zhao, and Xiaoming Li. Mining user intents in twitter: A semi-supervised approach to inferring intent categories for tweets. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [115] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207, 2013.
- [116] Chaochen Wu, Guan Luo, Chao Guo, Yin Ren, Anni Zheng, and Cheng Yang. An attention-based multi-task model for named entity recognition and intent analysis of chinese online medical questions. *Journal of Biomedical Informatics*, 108:103511, 2020.
- [117] Ke Wu, Song Yang, and Kenny Q Zhu. False rumors detection on sina weibo by propagation structures. In *2015 IEEE 31st international conference on data engineering*, pages 651–662. IEEE, 2015.
- [118] Fu-Chen Yang, Anthony JT Lee, and Sz-Chen Kuo. Mining health social media with sentiment analysis. *Journal of medical systems*, 40(11):1–8, 2016.
- [119] Andrew Yates and Nazli Goharian. Adrtrace: detecting expected and unexpected adverse drug reactions from user reviews on social media sites. In *European Conference on Information Retrieval*, pages 816–819. Springer, 2013.
- [120] Hong Yu, Carl Sable, and Hai Ran Zhu. Classifying medical questions based on an evidence taxonomy. In *Proceedings of the AAAI 2005 workshop on question answering in restricted domains*, 2005.
- [121] Shaodian Zhang, Edouard Grave, Elizabeth Sklar, and Noémie Elhadad. Longitudinal analysis of discussion topics in an online breast cancer community using convolutional neural networks. *Journal of biomedical informatics*, 69:1–9, 2017.
- [122] Thomas Zhang, Jason HD Cho, and Chengxiang Zhai. Understanding user intents in online health forums. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 220–229, 2014.
- [123] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626, 2015.

- [124] Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.
- [125] Yishi Zhang, Shujuan Li, Teng Wang, and Zigang Zhang. Divergence-based feature selection for separate classes. *Neurocomputing*, 101:32–42, 2013.

Appendix A

Appendix

A.1 First-level Categorization

Decision Tree pipeline	Parameter value
ccp_alpha	1e-8
criterion	gini
TF-IDF norm	11
TF-IDF use-idf	True
n-gram range	(1,3)
analyzer	char_wb

Table A.1: Best configuration parameters for Decision Tree and frequency based vectorization technique

Decision Tree	Group interactions	Offering support	Seeking Support
Precision	0.88	0.90	0.67
Recall	0.31	0.69	0.77
F1-score	0.46	0.78	0.71
Jaccard Score	0.494		
Hamming Loss	0.272		

Table A.2: Classification report for Decision Tree using binary relevance

Gaussian NB pipeline	Parameter value
var smoothing	1e-10
TF-IDF norm	11
TF-IDF use-idf	False
n-gram range	(1,3)
analyzer	word

Table A.3: Best configuration parameters for Gaussian NB (binary relevance) and frequency based vectorization technique

Gaussian NB (binary relevance)	Group interactions	Offering support	Seeking support
Precision	0.67	0.73	0.78
Recall	0.83	0.88	0.27
F1-score	0.74	0.80	0.40
Jaccard Score	0.551		
Hamming Loss	0.276		

Table A.4: Classification report for Gaussian NB using binary relevance

Random Forest pipeline	Parameter value
bootstrap:	false
max depth	10
n estimators	200
TF-IDF norm	12
TF-IDF use-idf	True
n-gram range	char_wb
analyzer	(1,1)

Table A.5: Best configuration parameters for Random Forest and frequency based vectorization technique

Random Forest	Group interactions	Offering support	Seeking support
Precision	0.75	0.83	0.73
Recall	0.75	0.83	0.42
F1-score	0.75	0.83	0.54
Jaccard Score	0.60		
Hamming Loss	0.23		

Table A.6: Classification report for Random Forest

KNN pipeline	Parameter value
n neighbors	10
p	2
weights	distance
TF-IDF norm	l2
TF-IDF use-idf	False
n-gram range	word
analyzer	(1,1)

Table A.7: Best configuration parameters for KNN and frequency based vectorization technique

KNN	Group interactions	Offering support	Seeking support
Precision	0.72	0.87	0.56
Recall	0.71	0.75	0.58
F1-score	0.72	0.80	0.57
Jaccard Score	0.50		
Hamming Loss	0.31		

Table A.8: Classification report for Random Forest

Decision Tree (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
ccp_alpha: 0.01 class_weight: 'balanced' criterion: 'entropy'	Twitter-50	0.35	0.50	0.65
ccp_alpha: 0.01 class_weight: 'balanced' criterion: 'gini' (TF-IDF)	word2vec-google-news-300	0,25	0.57	0.71
ccp_alpha: 0.02 class_weight: 'balanced' criterion: 'entropy'	fasttext-wiki-news-subwords-300	0.39	0.44	0.59

Table A.9: Classification report for Decision Tree with word embeddings

Gaussian NB (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score
var_smoothing=0.001 (binary relevance, TF-IDF)	GloVe-Twitter-50	0.29	0.54	0.70
var_smoothing=0.01 (binary relevance)	word2vec-google-news-300	0.30	0.53	0.67
var_smoothing=0.001 (binary relevance)	fasttext-wiki-news-subwords-300	0.28	0.56	0.71

Table A.10: Classification report for Gaussian NB with word embeddings

Random Forest (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
bootstrap: False class_weight: 'balanced' max_depth: 10 n_estimators: 400	Twitter-50	0.26	0.54	0.68
bootstrap: False class_weight: 'balanced' max_depth: 10 n_estimators: 600	word2vec-google-news-300	0.26	0.54	0.67
bootstrap: False class_weight: 'balanced' max_depth: 10 n_estimators: 600	fasttext-wiki-news-subwords-300	0.23	0.58	0.73

Table A.11: Classification report for Random Forest with word embeddings

KNN (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
n_neighbors: 10 p: 1 weights: 'distance'	Twitter-200	0.28	0.55	0.70
n_neighbors: 5 p: 2 weights: 'uniform'	word2vec-google-news-300	0.30	0.54	0.69
n_neighbors: 10 p: 1 weights: 'distance'	fasttext-wiki-news-subwords-300	0.25	0.58	0.73

Table A.12: Classification report for KNN with word embeddings

A.2 Second-level Categorization

Decision Tree Classifier	Parameter Value
ccp_alpha	0
criterion	entropy
TF-IDF norm	11
TF-IDF use-idf	False
n-gram range	(1,2)
analyzer	word

Table A.13: Decision Tree best configuration for the third task

classes (DT)	precision	recall	f1-score	support
emotional-support	0.23	0.20	0.21	15
esteem-support	0.00	0.00	0.00	7
informational-support	0.44	0.57	0.50	42
interpersonal	0.00	0.00	0.00	20
network-support	0.00	0.00	0.00	7
seeking-support	0.40	0.38	0.39	26
sharing-experiences	0.58	0.47	0.52	40

Table A.14: Classification report for DT with the best parameters

Random Forest pipeline	Parameter value
bootstrap:	false
max depth	20
n estimators	400
TF-IDF norm	12
TF-IDF use-idf	False
n-gram range	char_wb
analyzer	(1,1)

Table A.15: Random Forest best configuration for the third task

classes(RF)	precision	recall	f1-score	support
emotional-support	0.00	0.00	0.00	15
esteem-support	0.00	0.00	0.00	7
informational-support	0.75	0.64	0.69	42
interpersonal	0.00	0.00	0.00	20
network-support	0.00	0.00	0.00	7
seeking-support	0.00	0.00	0.00	26
sharing-experiences	0.65	0.65	0.65	40

Table A.16: Classification report for Random Forest with the best parameters

KNN pipeline	Parameter value
n neighbors	10
p	1
weights	distance
TF-IDF norm	l1
TF-IDF use-idf	True
n-gram range	word
analyzer	(1,2)

Table A.17: KNN best configuration for the third task

classes	precision	recall	f1-score	support
emotional-support	0.33	0.07	0.11	15
esteem-support	0.00	0.00	0.00	7
informational-support	0.66	0.69	0.67	42
interpersonal	0.33	0.05	0.09	20
network-support	0.00	0.00	0.00	7
seeking-support	0.62	0.31	0.41	26
sharing-experiences	0.93	0.68	0.78	40

Table A.18: Classification report for KNN with the best parameters

Decision Tree (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
ccp_alpha: 0 class_weight: 'balanced' criterion: 'entropy'	GloVe-Twitter-200	0.19	0.30	0.39
ccp_alpha: 0 class_weight: 'balanced' criterion: 'gini' (TF-IDF)	word2vec-google-news-300	0.20	0.25	0.33
ccp_alpha: 0 class_weight: 'balanced' criterion: 'gini'	fasttext-wiki-news-subwords-300	0.19	0.31	0.40

Table A.19: Classification report for Decision Tree

Random forest (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
bootstrap: True class_weight: 'balanced' max_depth: 20 n_estimators: 600	GloVe-Twitter-50	0.19	0.29	0.39
bootstrap: False class_weight: 'balanced' max_depth: 20 n_estimators: 600	word2vec-google-news-300	0.21	0.27	0.37
bootstrap: True class_weight: 'balanced' max_depth: 20 n_estimators: 600	fasttext-wiki-news-subwords-300	0.19	0.30	0.39

Table A.20: Classification report for Random Forest with word embeddings

GNB (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
var_smoothing=0.001 (classifier chain)	GloVe-Twitter-50	0.41	0.40	0.55
var_smoothing=0.001 (label powerset)	word2vec-google-news-300	0.38	0.31	0.45
var_smoothing=0.001 (label powerset)	fasttext-wiki-news-subwords-300	0.33	0.29	0.43

Table A.21: Classification report for Gaussian NB with word embeddings

KNN (parameter values)	Word Embedding	Hamming Loss	Jaccard Score	F1-score weighted
n_neighbors: 10 p: 2 weights: 'uniform'	GloVe-Twitter-200	0.21	0.25	0.36
n_neighbors: 10 p: 2 weights: 'uniform'	word2vec-google-news-300	0.22	0.29	0.42
n_neighbors: 5 p: 2 weights: 'uniform'	fasttext-wiki-news-subwords-300	0.25	0.27	0.38

Table A.22: Classification report for KNN with word embeddings

Best configurations	Hamming Loss	Jaccard Score	Weighted F1-Score
Decision Tree	0.236	0.329	0.47
Gaussian NB	0.164	0.376	0.53
KNN	0.183	0.354	0.47
SVM (binary relevance)	0.163	0.412	0.53
Logistic Regression (binary relevance)	0.176	0.427	0.51
Random Forest	0.197	0.264	0.35

Table A.23: Small summary of the metrics related to frequency-based vectorization methods and classical machine learning

A.3 Polarity Analysis

Classifier	Best parameters
Multinomial NB (without preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 2), 'alpha': 0.001, 'tfidf use idf': False}
Multinomial NB (with preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 2), 'alpha': 0.01, 'tfidf use idf': True}
Decision Tree (without preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 1), 'ccp alpha': 0.01, 'class weight': 'balanced', 'criterion': 'entropy', 'tfidf use idf': True}
Decision Tree (with preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 1), 'ccp alpha': 0.01, 'class weight': 'balanced', 'criterion': 'gini', 'tfidf use idf': True}
Gaussian NB (without preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 3), 'var smoothing': 1e-09, 'tfidf use idf': False}
Gaussian NB (with preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 3), 'var smoothing': 1e-09, 'tfidf use idf': True}
K Neighbors (without preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 1), 'n neighbors': 10, 'p': '2', 'weights': 'distance', 'tfidf use idf': True}
K Neighbors (with preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 2), 'n neighbors': 10, 'p': '2', 'weights': 'distance', 'tfidf use idf': True}
SVM (without preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 3), 'c': 1.0, 'class weight': 'balanced', 'kernel': 'sigmoid', 'tfidf use idf': True}
SVM (with preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 2), 'c': 1.0, 'class weight': 'balanced', 'kernel': 'sigmoid', 'tfidf use idf': True}
Logistic Regression (without preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 2), 'c': 100, 'class weight': 'balanced', 'penalty': 'l2', 'solver': 'liblinear', 'tfidf use idf': True}
Logistic Regression (with preprocessing)	{'bow analyzer': 'word', 'bow ngram range': (1, 1), 'c': 1.0, 'class weight': 'balanced', 'penalty': 'l2', 'solver': 'liblinear', 'tfidf use idf': False}

Table A.24: Pipeline best parameters

Multinomial NB	negative	neutral	positive
precision	0.61	0.79	0.00
recall	0.79	0.74	0.00
F1-score	0.69	0.74	0.00
accuracy	0.67		

Table A.25: Multinomial NB classification report without preprocessing

Multinomial NB	negative	neutral	positive
precision	0.54	0.71	0.33
recall	0.72	0.69	0.09
F1-score	0.62	0.70	0.14
accuracy	0.63		

Table A.26: Multinomial NB classification report with preprocessing

Decision Tree	negative	neutral	positive
precision	0.43	0.63	0.22
recall	0.52	0.59	0.18
F1-score	0.47	0.61	0.20
accuracy	0.52		

Table A.27: Decision Tree classification report without preprocessing

Decision Tree	negative	neutral	positive
precision	0.47	0.71	0.17
recall	0.52	0.59	0.27
F1-score	0.49	0.64	0.21
accuracy	0.53		

Table A.28: Decision Tree classification report with preprocessing

Gaussian NB	negative	neutral	positive
precision	0.51	0.72	0.00
recall	0.69	0.72	0.00
F1-score	0.59	0.72	0.00
accuracy	0.63		

Table A.29: Gaussian NB classification report without preprocessing

Gaussian NB	negative	neutral	positive
precision	0.48	0.65	0.00
recall	0.55	0.71	0.00
F1-score	0.52	0.68	0.00
accuracy	0.58		

Table A.30: Gaussian NB classification report with preprocessing

KNN	negative	neutral	positive
precision	0.59	0.73	0.00
recall	0.76	0.74	0.00
F1-score	0.67	0.74	0.00
accuracy	0.66		

Table A.31: KNN classification report without preprocessing

KNN	negative	neutral	positive
precision	0.49	0.71	0.00
recall	0.76	0.62	0.00
F1-score	0.59	0.66	0.00
accuracy	0.59		

Table A.32: KNN classification report with preprocessing

SVM	negative	neutral	positive
precision	0.62	0.76	0.00
recall	0.79	0.78	0.00
F1-score	0.70	0.77	0.00
accuracy	0.69		

Table A.33: SVM classification report without preprocessing

Logistic regression	negative	neutral	positive
precision	0.67	0.75	0.00
recall	0.76	0.81	0.00
F1-score	0.71	0.78	0.00
accuracy	0.70		

Table A.34: Logistic Regression classification report without preprocessing

Logistic regression	negative	neutral	positive
precision	0.64	0.74	0.33
recall	0.72	0.79	0.09
F1-score	0.68	0.77	0.14
accuracy	0.69		

Table A.35: Logistic Regression classification report with preprocessing

Random forest	negative	neutral	positive
precision	0.64	0.64	0.00
recall	0.31	0.93	0.00
F1-score	0.42	0.76	0.00
accuracy	0.64		

Table A.36: Random Forest classification report without preprocessing

Random forest	negative	neutral	positive
precision	0.75	0.64	0.00
recall	0.31	0.93	0.00
F1-score	0.44	0.76	0.00
accuracy	0.64		

Table A.37: Random Forest classification report with preprocessing