FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Detection of Misuse and Malicious Behaviours through a Dialogue Analysis System

Beatriz Gonçalves Neto Carneiro de Brito

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Rui Camacho

Second Supervisor: Jorge Oliveira

October 29, 2022

Resumo

O reconhecimento de emoções humanas tornou-se um tópico de investigação popular, com sistemas de análise de sentimentos a surgir cada vez mais. Sistemas como estes podem ter diversas aplicações em diferentes campos, tais como segurança, saúde, educação, marketing e outros.

As emoções humanas são muito complexas e podem ser extremamente difíceis de decifrar, dado que cada pessoa se expressa de forma única. Embora muitos estudos detetem emoções a partir de expressões faciais ou da fala, fazê-lo através da análise de discurso é ainda um desafio.

Uma compreensão bem-sucedida dessas emoções pode levar ao desenvolvimento de ferramentas com propósitos essenciais, como a criação de sistemas de segurança inovadores e eficientes, capazes de detetar e prever comportamentos maliciosos. Esta deteção pode ser conseguida através da análise do discurso de uma pessoa, ao interpretar expressões ou palavras-chave que possam indicar más intenções.

Com isto em mente, foi desenvolvido um sistema de aprendizagem profunda capaz de analisar diálogos e descodificar os sentimentos e emoções transmitidos pelos participantes durante uma conversa. Foram aplicadas duas redes neurais artificiais, uma rede *multilayer perceptron* e uma *long short-term memory*, a uma base de dados já existente, constituída por diálogos com múltiplos intervenientes. Ambas foram criadas para atribuirem a cada frase um sentimento ou uma emoção, sem ter em consideração o contexto das conversas, tendo sido alcançado o valor de *F1-score* máximo de aproximadamente 30%.

Abstract

Recognising human emotions has become a trendy research topic, with sentiment analysis systems emerging more and more. Systems like this have many applications in different areas such as security, healthcare, education, marketing, and others.

Human emotions are very complex and can be extremely hard to decipher since each person expresses himself in a unique form. While many studies detect emotions from facial expressions or speech, doing so from only spoken words is still very challenging.

Successfully comprehending these emotions can lead to developing tools that serve essential purposes, such as creating innovative and efficient security systems capable of detecting and predicting malicious behaviours. This detection can be done by analysing a person's dialogue and interpreting expressions or keywords that can reveal bad-natured intentions.

With this in mind, a deep learning system was designed to analyse dialogues and retrieve the sentiments and emotions transmitted by the speakers during a conversation. Two artificial neural networks, a multilayer perceptron network and a long short-term memory network, were applied to an existing dataset consisting of multi-party dialogues. Both networks were designed to attribute a sentiment or emotion to each sentence without considering the conversational context, having achieved a maximum F1-score of approximately 30%.

Acknowledgements

I would like to thank my supervisors, Professor Rui Camacho and Professor Jorge Oliveira, for helping me and accompanying me from the beginning.

I am thankful for my parents and sister Vicky, and their motivation and belief in me during this dissertation's elaboration period, and also for my uncles and my cousin Tita, for always putting a smile on my face. I am very fortunate to have such a supportive family. I would also like to thank my dog Pazuzu for always keeping me company, no matter how long the night was.

Lastly, I would like to extend my sincere thanks to a person without whose help I would not have been able to write this, Francisco. I am grateful for his patience and aid at all times, and I thank him for always keeping my spirits high.

To all my family and friends, thank you.

Beatriz Brito

"The original question, 'Can machines think?' I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted"

Alan Turing

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AI	Artificial Intelligence
ANN	Artificial Neural Network
BRNN	Bidirectional Recurrent Neural Network
CMN	Conversational Memory Network
CNN	Convolutional Neural Network
COMET	COMmonsEnse Transformers
CSV	Comma-separated values
EAR	Emotion Association Rules
EGR	Emotion Generation Rules
GRU	Gated Recurrent Units
LSTM	Long Short-Term Memory
MELD	Multimodal EmotionLines Dataset
MLP	Multilayer Perceptron
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
POS	Parts of Speech
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RNTN	Recursive Neural Tensor Network
SAOOP	Sentiment Analysis Of Online Papers Sent
SCNN	Sequence-based Convolutional Neural Network
SMM	Separable Mixture Models
SS-BED	Sentiment and Semantic-Based Emotion Detector

Chapter 1

Introduction

This introductory chapter summarises the work done in this dissertation, its motivation, possible contributions, and lastly, the dissertation's structure.

1.1 Motivation

In the last few years, there has been a sudden growth in the research of sentiment analysis and emotion recognition systems. These systems can identify sentiments and emotions through facial expressions, spoken words, speech, posture, etc., and can be applied in different contexts.

Sentiment analysis in texts or dialogues analyses trends and opinions, allowing for monitoring social media and managing brand reputation. Thus augmenting marketing strategies, with companies keeping track of customers' feedback by analysing social media comments or surveys. The same goes for customer service, where negative emails or messages can be prioritised, avoiding frustrated clients. Another possible application is in market research since it can help to better understand the demands of the buyers [2].

Emotion recognition via facial expressions is another field of research, for example, for the automotive industry. Car safety can be improved by detecting what the driver awareness: if the driver is noticeably tired or drowsy, the car could warn him, preventing him from going out of a lane. This technique can also be used in health care for better triage, in online education by analysing the student's emotional feedback, or even in video game testing [3].

Another application that can emerge from this, using spoken words, is developing a dialogue analysis system capable of identifying malicious behaviours. A security system installed in workplaces could be used to monitor, predict and avoid misuse conduct.

During a conversation, an upset employee may express his discontent through specific keywords or expressions that the system would pick up on. After that, it would classify the worker as being, for example, angry. This simple recognition could help prevent situations of dissatisfaction from escalating to something worse, making the work environment safer.

Compared to speech or video, the advantage of analysing textual data is that it contains less additional information besides the written words, making it easier to process. Regardless, the automatic analysis of dialogue is still very challenging.

A natural conversation contains many variables that must be considered in order to create an efficient emotion recognition system. The topic of the conversation, the speaker's point of view, personality, intent, mental and emotional state, and the argumentation logic [4] are good examples of said variables.

On top of that, individuals can express themselves ironically during a dialogue. They can quickly shift their emotions or even fake them, making it extremely hard for an algorithm to accurately classify the sentiment being exhibited [5].

Also, comprehending the conversational context is a complex yet crucial step in emotion recognition, as it strongly influences the course of the discussion [5].

Despite all these challenges, the advantages of the applications of this type of system are undeniable. Therefore, the continuation of research in this area and the constant improvement of these systems are crucial to enhancing human-to-machine interaction systems.

1.2 Goals and Contributions

This work aimed to create a dialogue analysis system capable of sentiment and emotion recognition. The final work was restricted to the identification of a smaller number of sentiments/emotions, which was imposed by the available dataset. Nevertheless, the first steps towards creating a system capable of detecting misuse and malicious behaviours were given.

The system ended up predicting a total of three sentiments and seven emotions. A deep learning model was developed and applied to an existing dataset composed of dialogues transcribed from a TV show. This aspect came as an advantage given that this work's goal was to analyse spoken text; this way, the model could be trained in genuine conversations.

Taking a closer look into the project development, the first step of the process was preprocessing the textual data extracted from the dataset. This included removing non-alphabetical characters, numbers and spaces, normalizing letter case, and other tasks such as tokenization, stemming, lemmatisation, and others that will be mentioned in the next chapters. Afterwards, two vocabularies were created, by selecting key words for classification from the utterances in the dataset. The next step was defining the two models (a multilayer perceptron network and a long short-term memory network) and vectorizing and normalizing the data according to the model. The last steps were training the models and evaluating their performance, by analysing the values of the metrics obtained.

1.3 Dissertation Structure

This dissertation is divided into the following five chapters:

Chapter 1, Introduction.
Chapter 2, Literature Review – includes all the fundamental theory on text classification tasks and deep learning algorithms and describes the selected state-of-the-art models.
Chapter 3, Project Development – contains a description of the dataset and a thorough explanation of all the procedures that led to the elaboration of this project.
Chapter 4, Experiments and Results – covers the experiments carried out and a performance evaluation.
Chapter 5, Conclusions and Future Work – presents the main conclusions drawn from the proposed work and suggestions for improving it.

Chapter 2

Literature Review

This chapter contains the essential background topics required for comprehending the primary aspects of the developed work and the existing models that constitute the state-of-the-art.

2.1 Background

Existing conversational emotion recognition or sentiment analysis systems use attribute-based and deep-learning-based algorithms. It requires an input text, withdrawn from an available dataset, a feature vector containing information about said input and labels that the model will predict: using a set of texts as the data source the individual sentences are extracted and pre-processed; for the use of most algorithms, each sentence is transformed into a set of attribute values; each sentence is then assigned a label corresponding to an emotion.

To comprehend how these approaches are executed, it is essential to understand the principles of primary text classification.

2.1.1 Text-mining and Natural Language Processing

Text-mining is a common technique used for simple sentiment analysis that allows extracting information from textual data. It is usually used in collections of documents to identify relevant info and sort it, this way structuring the data. This allows, for example, for the creation of smaller and well-organised datasets.

Natural language processing (NLP) is perhaps one of the most important text-mining methods. It instils in computers the human ability to understand language [6], allowing them to comprehend complex concepts and recognise relationships and keywords in text.

Because there is usually a lot of irrelevant text items in the data, pre-processing it is a crucial step. Some fundamental tasks can be applied, such as:

- Tokenisation – separating the continuous text into sentences and then single words.
- Stemming – removing the endings of the words, leaving only the stem.
- Lemmatisation – like stemming, but it returns the base dictionary form of the word, known as the lemma.
- Part-of-speech tagging – attributing the part-of-speech (noun, verb, pronoun, etc.) to each word in a sentence.
- Named Entity Recognition – recognising named entities (person's name, organisation name, location, etc.).
- Chunking – transforming a group of tokens into chunks.
- Removing stop words – removing words that add no value to a sentence.
- Removing non-alphabetic characters, numbers and redundant spaces.
- Normalizing letter case.

2.1.2 Deep Learning

Artificial Neural Networks

Artificial neural networks are used in many fields of AI and deep learning because they grant computers the ability to recognise patterns by mimicking the behaviour of the human brain, making the process of classifying data much faster [7].

They are composed of node layers: an input layer, one or more hidden layers, and an output layer. The nodes are all connected, each with a certain weight and threshold. The node is activated when the output of a node exceeds its point, and only then does it send the data to the next layer, turning the result of one node into the input of the next. A common ANN architecture can be seen in Figure 2.1.

Neural networks require training to provide reliable results, which is achieved with backpropagation. This consists of fine-tuning the weights of each node by computing the gradient of the loss function obtained in each iteration, thus reducing the error rate and increasing generalization.

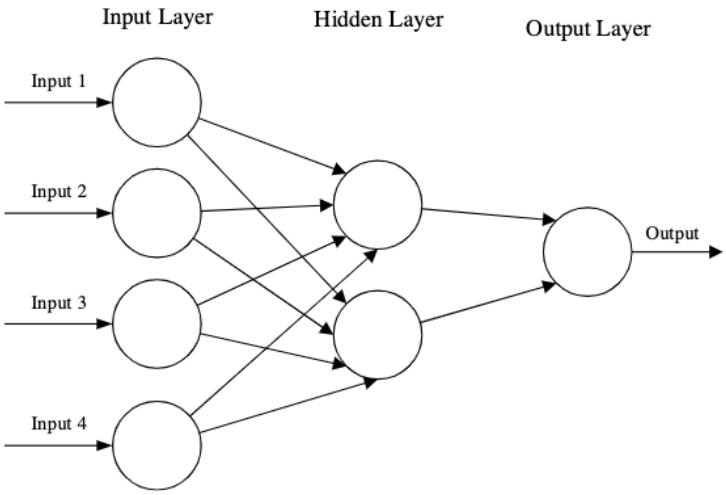

Figure 2.1: Artificial neural network architecture [1].

Convolutional Neural Networks

A convolutional neural network is a feedforward and multilayer network usually used for image processing but has been applied in text classification [8]. However, due to its basic architecture, it does not consider the data from previous nodes.

What distinguishes CNN from other neural networks is that it contains hidden layers called convolutional layers. These receive input, transform it through a convolutional operation, and send the output to the next layer. They can detect patterns due to filters, known as kernel functions. These filters can be seen as a matrix and can extract features from the data by sliding on it [8].

Besides the convolutional layer, CNNs can also have a pooling layer responsible for dimensionality reduction, the process of decreasing the number of parameters in the input. It can be either max or average pooling: max pooling selects the maximum value as the output and is usually preferred because it discards noisy activations; average pooling calculates the average value and establishes it as the output.

The process of classification is achieved in the fully connected layer. In this layer, as the name implies, each node is directly connected to a node from the previous layer [7].

Recurrent Neural Networks

A recurrent neural network is a network structure that contains a loop, which allows it to preserve information from layer to layer [8]. It is usually used in NLP tasks like text classification. They are beneficial when handling sequential data and can manage variable-length sequence inputs [9].

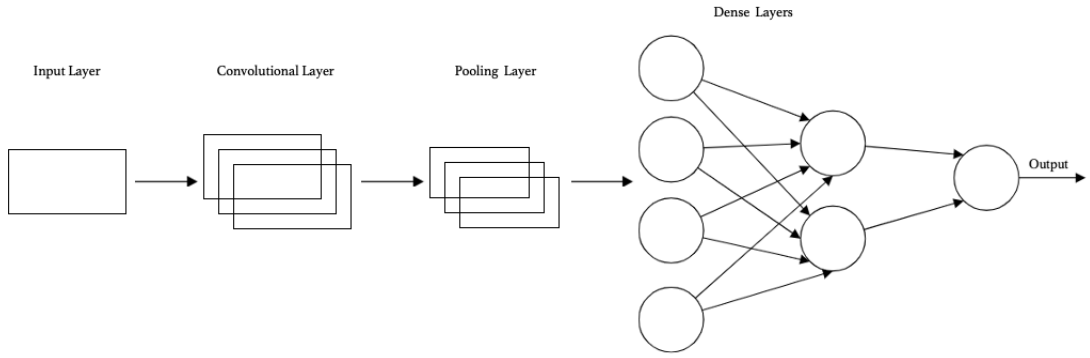

Figure 2.2: Convolutional neural network architecture.

RNN's architecture is like CNN but contains a looping mechanism that allows data to flow between states, as seen in Figure 2.3. The data from the previous inputs is stored in memory, called a recurrent hidden state, and it influences the output of the current node. This memory provides RNN with the ability to predict the following input. Unlike CNN, all nodes' weights are the same and are initialised with random values close to zero, only to be changed throughout training to perfect the network.

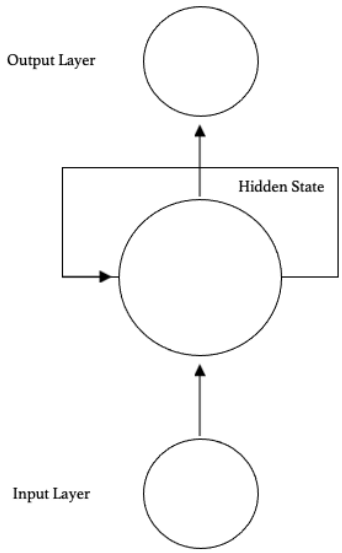

Figure 2.3: Recurrent Neural Network architecture.

A gradient is calculated in RNN that measures the changes between the output of a function and its corresponding input. When data goes through a vast number of layers, this gradient can either become too small and lose its value (vanishing gradients) or become way too large (exploding

gradients) [9]. Combined with memory limitations, these two factors represent the main problems when analysing long-term sequences with RNNs.

These networks can be described according to the number of inputs and outputs. They can have one information to many outputs, many inputs to only one output, or many inputs to many outputs. There are also different network architectures [10]:

- Bidirectional recurrent neural networks - future data is used to predict the current state. Improves the network's accuracy.
- Long Short-Term Memory - use three gates (input, output, and forget gate) to control the information, deciding whether to preserve or neglect it based on its weight. Stores memory over a more significant period, solving the long-term dependency problem.
- Gated recurrent units - use two gates (reset and update gate) to control the information flow. Solves the memory limitation problem.

2.2 State of the Art

The articles for analysis were selected using Google Scholar by researching keywords such as dialogue analysis systems, sentiment analysis, text classification, emotion detection, and emotion recognition. After carefully selecting and studying a primary batch of documents, more were added to the list by gathering papers from their bibliographic references.

The selection of an article obeyed the following criteria: the title and abstract indicated that it was related to the theme in question; considering the increased research on this topic in the past few years, it was essential to analyse updated works. Therefore, the article's date was a relevant factor; lastly, the paper was available online.

The selected articles that make up the state-of-the-art fall into two categories: feature-based and deep-learning models. Special attention was given to the last one. Feature-based models analyse keywords with exact emotional value [11]. Many models use this method because it is a very straightforward approach. Still, it faces many problems, such as ambiguous keywords, the presence of utterances that lack keywords, and statements without the semantic and syntactic information required for emotion recognition [12].

On the other hand, deep-learning-based methods use neural networks that provide highly reliable results in all domains: text, speech and image. Recurrent and convolutional neural networks have been used for emotion detection in different fields and have consistently proven their value.

Different models were studied to better understand how to create an emotion recognition system, unimodal (text) and multimodal (text, audio and video), but focusing on how they performed using only textual data. The conversational context is another variable that a few of the presented works consider.

Now follows a brief description of the selected documents.

Bag of Words

Bag-of-words is a well-known feature-based model for positive and negative sentiment classification tasks. El-Din et al. (2016) [13] suggests an Enhancement Bag-of-words model, Sentiment Analysis Of Online Papers Sent (SAOOP), that classifies data as very negative, negative, neutral, positive, and very positive. It was applied to a dataset of online reviews, which resulted in an accuracy of 82%.

This is a straightforward approach to sentiment analysis, so it is essential to acknowledge its existence. It can be used to set the baseline performance for emotion detection, like Colneric et al. (2020) [14] suggests.

Automatic Emotion Recognition Approach From Text

Wu et al. (2006) [12] uses two approaches for recognising emotions (happy, unhappy, or neutral): emotional keywords and semantic network. Both methods require a training phase - during which emotion association rules (EAR) are created, based on emotion generation rules (EGR) and an emotional corpus - and a test phase - where separable mixture models (SMM) are applied to the input text to classify the emotion. The EGR is manually annotated, and each sentence in the corpus is given a corresponding emotion label and a specific EGR.

The first corpus consists of students' dialogues, and the second one is a broadcast drama. Compared to state-of-the-art models from the time it was proposed, this method provided excellent results (75.14% accuracy for corpus A and 61.18% on corpus B). Still, nowadays, deep learning-based methods are usually preferred.

Convolutional Neural Network

Kim et al. (2016) [15] uses a baseline model that applies convolutional neural networks for text classification tasks, which can be applied to sentiment analysis. As mentioned, convolutional neural networks apply layers of convolving filters to local features. They are usually used for image classification but can be applied to text using a one-dimensional convolutional network. They have been proven very effective with NLP but represent a simple approach regarding emotion recognition in conversations.

Poria et al. (2018) [5] tested this text-CNN on the dataset MELD, where utterances had two different sets of labels: sentiment label (positive, negative and neutral); emotion label (anger, disgust, fear, joy, neutral, sadness and surprise). For sentiment analysis, the model achieved 64.25% F-score, and for emotion, 55.02%.

Recursive Neural Tensor Network

Socher et al. (2013) [16] proposes RNTN, a model for sentiment analysis that takes any length utterances as inputs, representing them through word vectors and a parse tree. It computes compositional vector representations used as features to classify each sentence.

The dataset consists of positive and negative movie reviews that fall into five categories: neutral, somewhat positive, somewhat negative, positive, and negative. Compared to standard RNN and baseline models, it proved to be more effective overall in shorter sentences, with an accuracy of 80.7%. Feature-based baseline models perform well with longer sentences.

Spoken Language Embedding Sub-Network

Zadeh et al. (2017) [17] created a Tensor Fusion Network for multimodal sentiment analysis that uses spoken language, gestures and voice. Focusing only on the sub-networks responsible for analysing unimodal features, in this case, spoken text, they propose the spoken language embedding sub-network.

Spoken text is filled with unreliable information. Therefore, the goal was to design a system capable of filtering the relevant parts of the speech. To achieve this, the model is fed information between words from an utterance to use as input in a deep neural network. Said information consists of data stored in memory (LSTM) since the beginning of the statement.

The results obtained were satisfactory when compared to other models. For binary classification (positive or negative), it acquired 74.8% accuracy and 38.5% for a five-class classification (from very negative to very positive) on the CMU-MOSEI dataset [18].

Knowledge Powered CNN

Wang et al. (2017) [19] suggests a joint model that uses two sub-networks, both CNNs, one for extracting the word concept and another for removing the features. Although the approach is not specific to emotion detection, the model was tested on tweets with sentiment labels (negative, neutral and positive) and provided good results (59.84% accuracy).

A similar approach comes from Poria et al. (2015) [20]. It uses the values from the CNN's hidden layer as features to be fed to a more advanced classifier. The CNN acts not as a classifier but as a trainable feature extractor. The dataset is labelled as negative, neutral and positive and consists of real online data from Morency et al. (2011) [21]. With this dataset, the model achieved 79.77% accuracy.

Sequence-Based CNN

Zahiri and Choi et al. (2018) [22] proposes the use of a sequence-based CNN. A simple CNN, as mentioned before, does not work with sequential data like RNNs do. Yet, these are usually slower and need more training data. Therefore, they suggest SCNN because it uses the emotion sequence from the previous utterances to detect the emotion of the current one. This is only possible because the corpus maintains the original line of the utterances.

The model generally works as follows: convolution and max pooling is applied to the current input; then, the resulting vector is concatenated with the previous utterances and suffers another

convolution; consequently, features extracted from the recent statement get fused with the ones from the earlier utterances.

It was applied to a dataset of transcripts from the tv-show Friends, with three-class and seven-class labels. In terms of performance evaluation, SCNN did not outperform a base CNN model from Kim et al. (2014) [15], but it did an RNN-CNN model inspired from Donahue et al. (2015) [23] because the corpus was too small to grant good results to the RNN. The results were 53.20% accuracy for three-class classification and 37.35% for seven-class.

Conversational Memory Network

CMN, proposed by Hazarika et al. (2018) [24], is a multimodal approach for emotion recognition, being the possible emotional states of anger, happiness, sadness and neutral. It extracts audio, visual, and textual features from video. The textual features extraction is done through a simple CNN with one convolutional layer and max pooling.

The emotion classification depends on the conversational history, which implies using memory cells like GRUs, that store information from previous states. These two gates (reset and update) control the combination criteria with the current utterance and the last hidden states.

It was tested on IEMOCAP dataset [25], but they only shared the results using the multimodal system.

LSTM based model

This method, proposed by Poria et al. (2019) [26], uses an LSTM network to obtain contextual information and allow for better sentiment analysis. It extracts audio, visual, and textual features from video without considering context and then feeds them to an LSTM to get contextual features. For the textual features extraction, a simple CNN was used.

Applied to IEMOCAP dataset, with the labels anger, happiness, sadness and neutral, it achieved 73.6% accuracy. On MELD, with labels anger, disgust, fear, joy, neutral, sadness and surprise, it only got 56.44%.

Dialogue RNN

Poria et al. (2019) [27] suggests DialogueRNN, a model capable of handling multi-party conversations, taking into account the conversational context. It states that there are three key elements to obtain an emotion accurately - the speaker and the context and emotion from the previous utterances.

DialogueRNN has three stages of gated recurrent units: *global*, updates context; *party*, updates speaker state through analysing the current utterance, speaker's previous state, and conversational context; and *emotion*, which models the information for classification. The utterances from the dataset are sent to the *global*, and *party* GRU, and the outcome is fed to *emotion* GRU.

The model is prepared to receive both textual data and audio, making it a highly effective model that outperformed all the baseline methods. On the IEMOCAP dataset, this model achieved 59.33% accuracy, and on MELD, 66.10% F-score for sentiment analysis and 57.03% for emotion.

Sentiment and Semantic-Based Emotion Detector Model

Chatterjee et al. (2019) [11] proposes SS-BED for detecting four emotions: happy, sad, angry and other. The model harnesses sentiment and semantic-based features for a more precise emotion prediction.

Each utterance goes through two LSTM layers using two-word embedding matrices (semantic and sentiment). The resulting representations are concatenated and sent to a hidden layer that outputs the probabilities corresponding to each emotion class.

Compared with CNN-based approaches and other models, SS-BED outperformed all due to its combination of sentiment and semantic features. Tested on a set of tweets, it achieved 71.34% F1-score.

2.3 Summary

The previous works can be summarised in Table 2.1. It exposes the name and publication year, the dataset used, the input format, the labels for sentiment/emotion classification (if applied), the approach, and the available results for textual data.

Table 2.1: State-of-the-art summary.

Name	Year	Dataset	Input	Labels	Approach	Results (for text only)
Enhanced bag-of-words model	2016	Online reviews	Textual	Very negative Negative Neutral Positive Very positive	Features and keywords extraction	82% accuracy
Automatic Emotion Recognition Approach From Text	2006	Students dialogues (corpus A) Broadcast drama (corpus B)	Textual	Happy Unhappy Neutral	Emotional keywords and semantic networks extraction	75.14% accuracy (on corpus A) 61.18% accuracy (on corpus B)
Convolutional Neural Networks for Sentence Classification	2014	Movie reviews Stanford Sentiment Treebank Subjectivity dataset TREC question dataset Customer reviews	Textual	-	Sentence-level classification tasks using CNN	-
Text-CNN on MELD	2019	MELD	Textual	Sentiment (positive, negative and neutral) Emotion (anger, disgust, fear, joy, neutral, sadness and surprise)		64.25% F-score (sentiment) 55.02% F-score (emotion)
Recursive Neural Tensor Network	2013	Positive and negative movie reviews	Textual	Negative Somewhat negative Neutral Somewhat positive Positive	RNN with a tensor-based composition function for all nodes	80.7% accuracy
Spoken Language Embedding Sub-Network	2017	CMU-MOSEI	Textual Visual Audio	Binary (positive and negative) 5-class (from very negative to very positive)	Tensor Fusion Network with a spoken language embedding sub-network using LSTM	74.8% accuracy (binary) 38.5% accuracy (5-class)
Knowledge Powered CNN	2017	Set of tweets	Textual	Negative Neutral Positive	Word concept and features extraction using two CNNs	59.84% accuracy
CNN for feature extraction from text	2015	Online data	Textual Visual Audio	Negative Neutral Positive	Feature extraction using CNN	79.77% accuracy
Sequence-Based CNN	2018	Transcripts from tv-show Friends	Textual	3-class (positive, negative and neutral) 7-class (sad, mad, scared, powerful, peaceful, joyful and neutral)	CNN that uses sequenced information to improve classification Considers the conversational context	53.20% accuracy (3-class) 37.35% accuracy (7-class)
Conversational Memory Network	2018	IEMOCAP	Textual Visual Audio	Anger Happiness Sadness Neutral	Features extraction using simple CNN Emotion classification using GRUs	-
LSTM based model	2019	IEMOCAP	Textual Visual Audio	Anger Happiness Sadness Neutral	Feature extraction using CNN and context extraction using LSTM Considers the conversational context	73.6% accuracy
		MELD		Anger Disgust Fear Joy Neutral Sadness Surprise		56.44% F-score
DialogueRNN	2019	IEMOCAP	Textual Audio	Happy Sad Neutral Angry Excited Frustrated	RNN with three GRUs Considers the conversational context	59.33% accuracy
		MELD		Sentiment (positive, negative and neutral) Emotion (anger, disgust, fear, joy, neutral, sadness and surprise)		66.10% F-score (sentiment) 57.03% F-score (emotion)
Sentiment and Semantic-Based Emotion Detector Model	2019	Set of tweets	Textual	Happy Sad Angry Other	Two LSTM layers using two-word embedding matrices (sentiment and semantics)	71.34% F1-score

Chapter 3

Project Development

This chapter covers a brief explanation of the dataset and a thorough description of the procedures carried out during the development of the work.

3.1 Dataset

As seen, there is a great variety of methods for developing a text recognition system. The level of complexity depends on the purpose for which the model is intended and the dataset in which it will be tested.

The most common datasets for emotion recognition in conversations have a small number of utterances and contain individual narratives. This makes the development of algorithms capable of extracting relevant information from dialogue extremely challenging.

3.1.1 EmotionLines Dataset

EmotionLines [28] is a dataset created for textual analysis that has more than 13,000 utterances and multi-party dialogues in English, withdrawn from the television series *Friends*.

The scripts from seasons one to nine were separated firstly per episode and afterwards per scene, each constituting a dialogue. A random sample of 1,000 dialogues was placed on Amazon Mechanical Turk to be analysed by five workers. They attributed an emotion category to each utterance, labelling them with Ekmans' six universal emotions: joy, sadness, fear, anger, surprise, and disgust; plus, neutral. The most voted emotion was chosen as the final label of the utterance.

3.1.2 Multimodal EmotionLines Dataset

An improvement of EmotionLines is MELD - Multimodal EmotionLines Dataset [5] since it includes the dialogues' corresponding visuals. Yet, for the development of the algorithm, only the textual conversations were analysed.

Another feature of this dataset is that it quantifies emotions. This can be done by analysing Russell's circumplex model [29], which portrays all emotions in the Valence-Arousal two-dimensional

space. However, only the seven previously mentioned emotions were characterised according to their valance, which can be negative, positive, or neutral.

A sentiment label is attached to every emotion in the following way: sadness, fear, anger, and disgust are considered negative, joy is positive, neutral is neutral, and surprise is a complex emotion that can be both positive and negative. MELD is divided into three sets: the train set, which contains 9989 utterances; the test set, including 2610; and the validation set with 1109. Tables 3.1 and 3.2 contain the number of sentences for each sentiment and emotion, respectively.

The advantage of using an already divided dataset is that you can accurately compare the performance of different models on the same dataset because the train, test and validation sets are the same.

Each CSV file contains relevant information regarding each utterance: ID, sentence, speaker, emotion label, sentiment label, dialogue ID, sentence ID, season, episode, start-time, end-time (hh:mm: ss, ms).

Table 3.1: Number of utterances for each sentiment label.

	Number of utterances		
Sentiment label	**Train**	**Test**	**Validation**
Positive	2334	521	233
Neutral	4710	1256	470
Negative	2945	833	406
Total	9989	2610	1109

Table 3.2: Number of utterances for each emotion label.

	Number of utterances		
Emotion label	**Train**	**Test**	**Validation**
Neutral	4710	1256	470
Angry	1109	345	153
Fear	268	50	40
Disgust	271	68	22
Surprise	1205	281	150
Sadness	683	208	111
Joy	1743	402	163
Total	9989	2610	1109

3.2 Methodology

The work proposal was to develop a deep learning model capable of examining the utterances from MELD and accurately predicting the correspondent sentiment and emotion.

Any text classification model requires three things: an input text; a feature vector; and labels that the model will predict. This model's inputs are the sentences from the dataset, and they first need to be pre-processed to eliminate noise. After that, it is necessary to vectorize the text input

and set the vocabulary, and the last step is retrieving the labels from the dataset and normalising them.

Having done this, it is time to define the model. Two models were developed, first a multilayer perceptron network, a feedforward artificial neural network, and then a long short-term memory, a type of recurrent neural network. A detailed description of all the processes is presented below.

All code was written in Python in the web application Jupyter Notebook [30] and later using the virtual servers of Paperspace [31]. The deep learning models were developed using the Python library Keras [32] that runs on top of TensorFlow, a platform for machine learning [33]. The remaining libraries used for the elaboration of this project are mentioned in the following sub-sections.

3.2.1 Pre-processing input text

As mentioned, the information contained in the dataset is divided into columns: identifier; utterance; speaker; emotion label, sentiment label; dialogue ID; sentence ID; season; episode; start-time; end-time. It is only necessary to look at the utterances for the pre-processing stage, so they were loaded from the second column of the CSV file into a list using the Python library pandas [34].

Text needs to be polished before being processed to eliminate irrelevant data. The first step was removing non-alphabetic characters, numbers and redundant spaces from each sentence, using the library Python RegEx [35], and normalising all the words to lower case.

Afterwards, each utterance was tokenised, meaning it was divided into words. For instance, given the sentence ["why do all your coffee mugs have numbers on the bottom"] the output, after the word tokenization would be ['why', 'do', 'all', 'your', 'coffee', 'mugs', 'have', 'numbers', 'on', 'the', 'bottom'].

The next step was removing stop words, which add no real value to a sentence. This was accomplished using NLTK, Natural Language Toolkit, a Python library for NLP [36]. NLTK has a list with 179 English stop words, consisting of a selection of the most common words in data [37], such as 'the', 'to', 'and', 'a', 'in', 'it', 'is', etc. Every sentence had these words removed. The output of the previous example would be ['why', 'coffee', 'mugs', 'numbers', 'bottom'].

The last step was the lemmatisation process, also using the NLTK library [38], which reduces noise in the text by swapping a word for its lemma. It was preferred over stemming because it converts the word to a meaningful base form and groups similar words, whereas stemming only removes the endings. Applied to the same example, the output would be ['why', 'coffee', 'mug', 'number', 'bottom'].

After applying these methods to each sentence, there were 9574 cleaned utterances left for training, 2512 utterances for testing, and 1054 for validation. With the text corpus ready to process, the next stage was setting up the vocabulary, the set of words used in the text. A word-based vocabulary allows for the representation of each sentence as a vector composed of its vocabulary words.

Table 3.3: Final number of utterances for each sentiment label.

	Number of utterances		
Sentiment label	**Train**	**Test**	**Validation**
Positive	2231	508	221
Neutral	4525	1202	449
Negative	2818	802	384
Total	9574	2512	1054

Table 3.4: Final number of utterances for each emotion label.

	Number of utterances		
Emotion label	**Train**	**Test**	**Validation**
Neutral	4525	1202	449
Angry	1055	335	139
Fear	253	46	40
Disgust	259	66	22
Surprise	1150	271	137
Sadness	665	199	110
Joy	1667	393	157
Total	9574	2512	1054

3.2.2 Delineating vocabulary

When withdrawing information from spoken text, in this case for sentiment analysis or emotion recognition, it is necessary to comprehend that many words in the text corpus may not contribute to that classification. And having large vocabularies with irrelevant words implies occupying resources unnecessarily [39]. With this in mind, the goal was to create a vocabulary containing only the right words for classification without generating many sparse vectors.

Two experiments were conducted using two distinct vocabulary sets: vocabulary A with 3659 words; and vocabulary B with only 490. Both were defined using train, test and validation data.

The first step for creating both lists was saving all the 57106 words from the pre-processed utterances in an array. After removing the duplicated words, a function was designed to analyse and compare each of the remaining 4800 words and their synonyms: if two words were synonyms, the second would be converted into the first one, decreasing the vocabulary size even further. The synonyms were obtained using wordnet [40]. The last step was attributing the part-of-speech tag to each word, meaning labelling them as nouns, verbs, adverbs, etc., using once again NLTK's library [41]. All the proper nouns, pronouns, prepositions, determiners and verbs were removed from the list since they are not considered relevant for sentiment analysis.

Vocabulary B is a portion of A that was obtained through selecting words that coincide with a list created by Liu et al. (2012) [42], containing approximately 7400 positive and negative words. For example, pure, pleasant and better are three words from the list that appear in the dataset

and suggest a positive feeling; awkward, rude and poison, on the other hand, are three negative indicators, also present in both the list and the dataset.

```
Total number of words:
    after pre-processing utterances - 57 106
    after removing duplicates - 4 800
    after synonym switch and removing irrelevant data - 3 659
    after selecting positive and negative words - 490
```

3.2.3 Normalising labels

Each sentence is associated with a sentiment and an emotion label in the form of strings: neutral, positive and negative; neutral, angry, fear, disgust, surprise, sadness and joy. The first step was retrieving the labels from the dataset (columns five and four, respectively) and converting them to integers since machine learning algorithms do not work with categorical data.

Sentiment analysis requires a 3-class model with labels [0, 1, 2] where 0 is neutral, positive is 1, and negative is 2. Emotion requires a 7-class model with labels [0, 1, 2, 3, 4, 5, 6] where 0 is neutral, 1 is angry, 2 is fear, 3 is disgust, 4 is surprise, 5 is sadness, and joy is 6.

The next step was applying one hot encoding to each label [43] to convert the integer values into a binary vector. This way, the sentiment label 0 is represented by [1. 0. 0.], label 1 by [0. 1. 0.] and label 2 [0. 0. 1.], the same for the emotion's labels.

All training, testing and validation labels were encoded, and the resulting shape was (total number of utterances, number of classes).

```
hotencoder = OneHotEncoder()
encoder= LabelEncoder()
labels = np.array(labels)
y_enc = encoder.fit_transform(labels)
y_train = hotencoder.fit_transform(y_enc.reshape(-1,1)).toarray()
#shape = (number of utterances, number of classes)
```

3.2.4 Generating class weights

In Tables 3.1 and 3.2, it is possible to observe that the number of utterances for each label is quite distinct, especially in the second table. The fact that some classes have a minimal number of samples leads to the problem of imbalanced classes. This can result in false predictions because the model will generalise in favour of the class with more samples.

There are various approaches to handle this problem [44]. One is undersampling, removing random data from the majority class to decrease the ratio between classes, but this can lead to the exclusion of relevant data for classification. On the contrary, with oversampling, the number of instances of the lowest classes increases by randomly copying its data, but it can cause overfitting.

Another technique is using the class weight argument in model.fit() from TensorFlow [45], which allows us to define the weights of each class. A function was used to generate the class weights for this multi-class problem that supported one hot encoded labels [46]. Each weight is calculated by dividing the number of samples by the number of classes, and multiplying it by the class frequency.

The function returns a dictionary with the labels and corresponding weights that serve as input to the parameter class weight used in the MLP network model fit() function.

The chosen approach for the LSTM was oversampling, which will be addressed later in the document.

```
#sentiment labels
class_weights = {0: 0.7052670349907919,
                 1: 1.4304497235918123,
                 2: 1.1324816654837946}

#emotion labels
class_weights = {0: 0.3029723991507431,
                 1: 1.2867448151487826,
                 2: 5.324626865671642,
                 3: 5.265682656826568,
                 4: 1.1842323651452282,
                 5: 2.089311859443631,
                 6: 0.8187033849684452}
```

3.2.5 Multilayer perceptron network model

Obtaining the feature vector

Every sentence is represented as a 1 by X vector, where X is the vocabulary size. If any word of the sentence is equal to, shares a lemma, or is a synonym of a term present in the vocabulary, then the vector appends the value 1 in that exact position. If the word does not appear in the vocabulary, then it appends 0. The final vector has shape (number of utterances, vocabulary size). An illustrative example can be seen below.

This process is done to training, testing and validation sets, and because the vectors are already binary, there is no need for normalising the data.

```
vocabulary = ['this', 'example', 'and', 'another', 'one']
utterances = ['this is an example', 'and this is another']
vector = [[1 1 0 0 0]
          [1 0 1 1 0]]
#shape = (number of utterances, vocabulary size) = (2,5)
```

Defining the model

The created model is a sequential model with five dense layers [47]. These are the most commonly used layers. Each neuron in this layer receives the input from the previous ones and outputs a vector with a particular dimension. This change in the dimensions of the vectors is the result of matrix-vector multiplication.

The first layer has an input shape equal to the size of the vocabulary and activation ReLu. It is followed by three more dense layers with activation ReLu, interspersed with batch normalisation layers [48]. They are used to normalise small sets of data between layers, speed up training, and facilitate learning.

The fifth and last dense layer has the number of cells equal to the number of classes and a Softmax function that normalises the model's output. This way, each output value is represented as a probability of belonging to a particular class [49].

The compile method uses categorical cross entropy, a loss function for multi-class classification [50], and an Adam optimiser with a learning rate of 0,001. The loss function evaluates how the algorithm models the dataset, and the optimiser function modifies attributes to increase accuracy. The model's performance is assessed according to its accuracy and depends on how often the predictions equal the labels [51]. The model summary can be seen in Figure 3.1.

The training required: training data (both features and targets); validation data (data on which the model will be evaluated); batch size equal to 32 (number of samples); number of epochs (iterations over data); and class weights.

Model predictions

The method predict generates the predictions for the input samples [52]. It outputs an array of shape (2512,), where each position represents the classification given to each sentence.

To gather relevant metrics, it was first necessary to obtain the confusion matrix, and the multilabel confusion matrix, using the scikit-learn library. After that, four functions were created to calculate each class's accuracy, precision, recall and F1 score. Further description of this process will be provided in the next chapter.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 1024)              3747840

 batch_normalization_13 (Bat  (None, 1024)             4096
 chNormalization)

 dense_16 (Dense)            (None, 512)               524800

 batch_normalization_14 (Bat  (None, 512)              2048
 chNormalization)

 dense_17 (Dense)            (None, 256)               131328

 batch_normalization_15 (Bat  (None, 256)              1024
 chNormalization)

 dense_18 (Dense)            (None, 128)               32896

 batch_normalization_16 (Bat  (None, 128)              512
 chNormalization)

 dense_19 (Dense)            (None, 64)                8256

 batch_normalization_17 (Bat  (None, 64)               256
 chNormalization)

 dense_20 (Dense)            (None, 32)                2080

 batch_normalization_18 (Bat  (None, 32)               128
 chNormalization)

 dense_21 (Dense)            (None, 3)                 99

=================================================================
Total params: 4,455,363
Trainable params: 4,451,331
Non-trainable params: 4,032
_________________________________________________________________
```

Figure 3.1: Multilayer perceptron network model summary (for vocabulary A).

3.2.6 Long short-term memory network model

Obtaining the feature vector

The textual input is vectorised so that each word is treated as a 1 by X vector within every utterance, where X is the vocabulary size. Every sentence is therefore represented by a vector of length equal to the number of words in the sentence.

A function was created to obtain these features. It receives as input the list of utterances and the list of words, and it compares all the words of a sentence with the ones that make up the vocabulary. If the word is equal to, shares a lemma, or is a synonym of a word in the vocabulary, then the vector appends the value 1. If not, it appends 0. The function returns an array of shape (number of utterances,) and each position of the array has shape (number of words in sentence, vocabulary size).

This function is applied to the training, testing and validation sets. After that, all inputs are ready to be fed to the model.

```
vocabulary = ['this', 'example', 'and', 'another', 'one']
utterances = ['this is an example', 'and this is another one']
vector = [[[1 0 0 0 0]
           [0 0 0 0 0]
           [0 0 0 0 0]
           [0 1 0 0 0]]

           [[0 0 1 0 0]
           [1 0 0 0 0]
           [0 0 0 0 0]
           [0 0 0 1 0]
           [0 0 0 0 1]]]

# vector[i] shape = (number of words, vocabulary size)
# vector[0] shape = (4,5)
# vector[1] shape = (5,5)
# vector shape = (number of utterances, ) = (2, )
```

Padding and masking VS generator

Since every sentence has a distinctive length, the shape of the array that represents each utterance fluctuates depending on the number of words, but the input data for an LSTM model must be a tensor of shape (batch size, vocabulary size). To overcome this, two methods were implemented.

First, padding and masking [53]. Padding is used to transform smaller samples by adding values to them until they all have the same size. In this case, post padding was used because the values were added at the end of each instance. This process creates noise in the input data, so it is

required to inform the model that these new values should be ignored. For that, a masking layer is necessary, and since RNNs support mask arguments, it is possible to pass it manually when calling the model's layers.

```
vocabulary = ['this', 'example', 'and', 'another', 'one']
utterances = ['this is an example', 'and this is another one']
vector = [[[1 0 0 0 0]
           [0 0 0 0 0]
           [0 0 0 0 0]
           [0 1 0 0 0]
           [-10 -10 -10 -10 -10]] #padded values

           [[0 0 1 0 0]
           [1 0 0 0 0]
           [0 0 0 0 0]
           [0 0 0 1 0]
           [0 0 0 0 1]]]

# vector[i] shape = (max number of words, vocabulary size) = (5,5)
# vector shape = (number of utterances, ) = (2, )
```

The other method consisted of developing a generator and passing it as the input of the fit method to train the model: the generator iterates across the inputs and outputs, selects a random sample, and yields the features and targets. When using the generator, the batch size is one.

Using the generator proved the best approach. Therefore, the results presented in Chapter 4 only concern the generator.

```
def gen(X,y):
    while True:
        ix = np.random.choice(len(X), len(y), False)
        for i in ix:
            XA = np.asarray(X)
            yield (np.expand_dims(XA[i], 0),
                   np.expand_dims(np.expand_dims(y[i],0),0))
```

Oversampling

With the previous network, it was possible to use class weights on method fit. However, with the LSTM and using the generator, it was necessary to do oversampling because instead of using fit like before, the fit generator was used. Although it supposedly permitted the class weight parameter, many problems arose when trying to use it. On top of that, because this method was deprecated, it was hard to find helpful information about it.

Having said that, the oversampling consisted of randomly copying data from the minority classes until there was the same number of utterances from each category. This method was chosen instead of undersampling due to the reduced number of instances of certain classes, which led to the model being under-fitted: the training accuracy was very low because the algorithm didn't have enough data to learn from.

Defining the model

LSTMs are RNNs that store memory and, therefore, can discard irrelevant information and save only valuable data. They are usually used with sequential data and are beneficial for text classification problems.

The model used with the generator is a sequential model with eight layers [47]. The first is an LSTM layer with 128 memory units, activation ReLu [49] and input shape (none, vocabulary size). The first parameter is none since the batch size cannot be specified at this stage due to the variable length of the samples.

Using padding and masking, the first layer is the masking layer with mask value -10 so that the network ignores the padded values, and input shape (length of the samples, vocabulary size). The second is the LSTM layer also with 128 memory units, and activation ReLu [49].

In both cases the LSTM layer is followed by six dense layers, also with the ReLu function, and the final dense layer has the number of cells equal to the number of classes and activation Softmax.

The compile method uses categorical cross entropy for the loss function, an Adam optimiser with a learning rate of 0,001, and accuracy metrics.

The fit generator method was used to train the model [52] and has the following arguments: input data from the generator; number of epochs; number of steps per epoch (equal to the number of batches in training); validation data from the generator; validation steps (equal to the number of batches in validation).

Both models' summary can be seen in Figures 3.2 and 3.3.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm (LSTM)                  (None, None, 128)         1939456

 dense (Dense)                (None, None, 1024)        132096

 dense_1 (Dense)              (None, None, 512)         524800

 dense_2 (Dense)              (None, None, 256)         131328

 dense_3 (Dense)              (None, None, 128)         32896

 dense_4 (Dense)              (None, None, 64)          8256

 dense_5 (Dense)              (None, None, 32)          2080

 dense_6 (Dense)              (None, None, 3)           99

=================================================================
Total params: 2,771,011
Trainable params: 2,771,011
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.2: LSTM network model summary, with generator (for vocabulary A).

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 masking_1 (Masking)          (None, 34, 3659)          0

 lstm_1 (LSTM)                (None, 34, 128)           1939456

 dense_5 (Dense)              (None, 34, 1024)          132096

 dense_6 (Dense)              (None, 34, 512)           524800

 dense_7 (Dense)              (None, 34, 256)           131328

 dense_8 (Dense)              (None, 34, 128)           32896

 dense_9 (Dense)              (None, 34, 64)            8256

 dense_10 (Dense)             (None, 34, 32)            2080

 flatten_1 (Flatten)          (None, 1088)              0

 dense_11 (Dense)             (None, 3)                 3267

=================================================================
Total params: 2,774,179
Trainable params: 2,774,179
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.3: LSTM network model summary, with masking layer (for vocabulary A).

Model predictions

The creation of a prediction function was a necessary step to obtain an array with the final predictions. With the method predict generator, each word gets associated with a probability of belonging to a class, unlike the previous model that predicted the final label for each sentence.

For sentiment classification, the function evaluates the probabilities and assigns each sentence a class according to the following criteria: if all words have a higher probability of being neutral, then the sentence is neutral; if at least one word has a higher probability of being negative/positive, then the sentence is negative/positive. For emotion classification the same logic was followed, but with the seven distinct classes. The most likely emotion was chosen as the final classification.

The confusion matrix and multilabel confusion matrix are obtained like before, and so are the metrics.

Chapter 4

Experiments and Results

This chapter exposes the metrics used to evaluate the model's performance, the experiments carried out, and their final results.

4.1 Metrics for performance evaluation

Both models' performance will be evaluated according to the same metrics, each of them explained below. In classification problems such as this, a comparison between the predicted and the actual values is performed. A simple way of doing this is through a confusion matrix.

4.1.1 Confusion Matrix

With a confusion matrix it is possible to visualise how many instances the model correctly classified for each class, and extract important metrics such as classification accuracy, precision, recall and F1-score.

In Table 4.1 each row represents the predictions, each column represents the actual classifications, and the four cells are defined as:

- *TruePositives* - number of positive class samples predicted correctly.
- *TrueNegatives* - number of negative class samples predicted correctly.
- *FalsePositives* - number of negative class samples predicted incorrectly.
- *FalseNegatives* - number of positive class samples predicted incorrectly.

Table 4.1: Confusion matrix for binary classification.

		Predictions	
		Yes	No
Real	Yes	True Positive	False Positive
	No	False Negative	True Negative

4.1.2 Precision

Precision [54] is defined as the number of positives predicted correctly (true positives) divided by the total number of positives predicted (sum of true positives and false positives). It represents how much the model can be trusted when it makes a positive prediction. It assumes values between 0 and 1.

$$Precision = \frac{TP}{TP+FP} \tag{4.1}$$

where:
TP is the number of *TruePositives*;
FP is the number of *FalsePositives*.

4.1.3 Recall

Recall [54] is defined as the number of positives predicted correctly (true positives) divided by the total number of positives (sum of true positives and false negatives). Values range between 0 and 1.

$$Recall = \frac{TP}{TP+FN} \tag{4.2}$$

where:
TP is the number of *TruePositives*;
FN is the number of *FalseNegatives*.

4.1.4 Accuracy

Classification Accuracy [54] is defined as the number of correct predictions (sum of true positives and true negatives) divided by the total number of predictions (sum of true positives, true negatives, false positives, and false negatives). It is comprehended between 0 and 1.

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \tag{4.3}$$

where:
TP is the number of *TruePositives*;
TN is the number of *TrueNegatives*;
FP is the number of *FalsePositives*;
FN is the number of *FalseNegatives*.

4.1.5 Balanced Accuracy

Balanced Accuracy [54] is defined as the arithmetic mean of each class' recall, and is calculated for imbalanced datasets because with this formula every class has the same weight.

$$BalancedAccuracy = \frac{\frac{TP}{Total_{row1}} + \frac{TN}{Total_{row2}}}{n_classes} \tag{4.4}$$

where:
TP is the number of *TruePositives*;
TN is the number of *TrueNegatives*;
n_classes is the number of classes.

While accuracy benefits the majority class because it treats all classes the same way, more importance is given to the minority classes with balanced accuracy.

4.1.6 F1-Score

F1-Score [54] is defined as the harmonic mean of precision and recall. It ranges between 0 and 1, and if the value is close to 1 then there's a good balance between precision and recall.

$$F1\text{-}Score = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right) \tag{4.5}$$

4.1.7 Weighted metrics

Weighted metrics take into consideration the weight of each class. These values can be calculated by multiplying each metric (precision, recall or F1-score) by the number of samples in each class, adding the results, and dividing it by the total number of samples.

4.2 Existing models

MELD, the dataset used in the development of this work, has already been used by Poria et al. (2018) [5], Poria et al. (2019) [26], and Poria et al. (2019) [27] for the same purpose, as seen in Chapter 2. The average F-score results are shown on Table 4.2.

Note that both the LSTM-based model and DialogueRNN can analyse the conversational context, which improves classification. Given that the developed system is context-independent, the model's performance can only be compared to TextCNN.

Table 4.2: Existing models' F-Score results on MELD.

	F-score	
Model	Sentiment	Emotion
TextCNN	64.25%	52.02%
LSTM-based model	-	56.44%
DialogueRNN	66.10%	57.03%

4.3 Results

Two different models were developed, a MLP and a LSTM model. The first was experimented on two sets of attributes: vocabulary A with 3659 words; and vocabulary B with 490; and the second only on vocabulary A. On top of that, there were two types of classification: 3-class, with labels neutral, positive, and negative; and 7-class, with labels neutral, anger, fear, disgust, surprise, sadness, and joy.

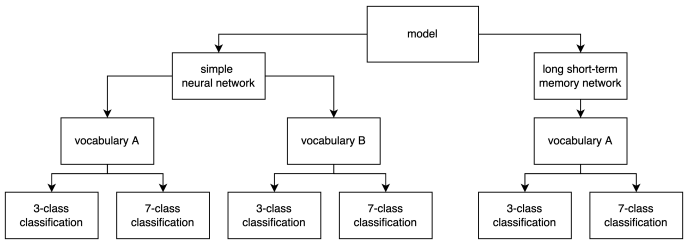

Figure 4.1: Representative diagram of the experiments performed.

4.3.1 First experiment

The first model to be analysed is the MLP. Four confusion matrix were obtained, shown in Figures 4.2 to 4.5 , as well as the values of the previously mentioned metrics, in Tables 4.3 and 4.5. The weighted metrics can be seen in Tables 4.4 and 4.6.

For vocabulary A, let us analyse the obtained results present in Table 4.3 for sentiment classification. In 100 predictions, our model predicted the correct classification approximately 50 times for neutral and 60 times for both positive and negative. In terms of precision, out of the 50 predictions for neutral, 43 were correct; for positive, 32 out of 60; and for negative only 23 out of 60. Out of all neutral values, our model predicted 43 correctly, which means it has a 48% recall when attributing a neutral classification. The same goes for positive and negative recall values of 19% and 31%, respectively. Observing the F1-Score, a value based on precision and recall, the neutral classification achieves better results than positive and negative.

The overall results are much worse using vocabulary B, although it seems to predict positives more accurately.

The results for emotion classification can be seen in Table 4.5. Once again, vocabulary A outperforms vocabulary B. For A, the classes with higher precision rates are neutral, disgust and joy. B also obtained a high precision in joy.

Table 4.3: Comparison of the performance of the 3-class model on both vocabularies.

	Vocabulary A				Vocabulary B			
Sentiment Label	**Accuracy**	**Precision**	**Recall**	**F1 score**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Neutral	50.60%	43.09%	48.19%	0.45	50.76%	15.64%	45.74%	0.23
Positive	59.24%	32.09%	19.36%	0.24	41.12%	63.98%	20.05%	0.31
Negative	59.28%	23.32%	31.43%	0.27	60.83%	18.58%	31.04%	0.23

Table 4.4: Weighted metrics of the 3-class model on both vocabularies.

	Vocabulary A	**Vocabulary B**
Weighted Precision	31.51%	39.71%
Weighted Recall	29.04%	28.75%
Weighted F1-score	29.20%	26.83%

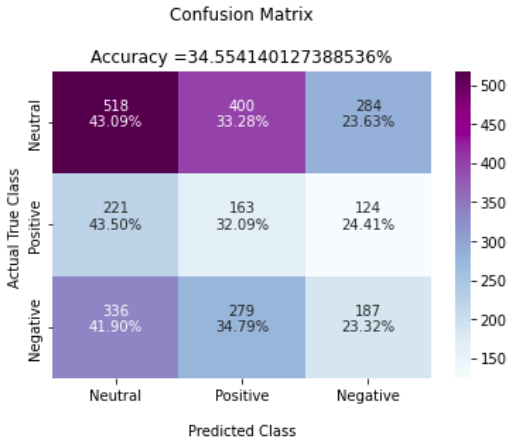

Figure 4.2: Three class confusion matrix for vocabulary A.

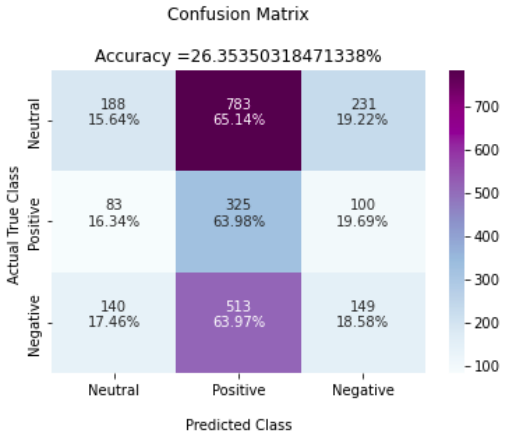

Figure 4.3: Three class confusion matrix for vocabulary B.

Table 4.5: Comparison of the performance of the 7-class model on both vocabularies.

	Vocabulary A				**Vocabulary B**			
Emotion Label	**Accuracy**	**Precision**	**Recall**	**F1 score**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Neutral	51.27%	30.12%	48.53%	0.372	51.59%	3.24%	42.39%	0.060
Anger	78.70%	9.55%	12.12%	0.107	84.51%	2.99%	13.51%	0.049
Fear	91.52%	6.52%	1.73%	0.027	88.50%	4.35%	0.81%	0.014
Disgust	73.17%	22.73%	2.35%	0.043	86.58%	12.12%	2.79%	0.045
Surprise	81.78%	8.49%	9.87%	0.091	86.03%	3.32%	9.18%	0.049
Sadness	85.71%	5.03%	5.56%	0.053	74.44%	14.07%	5.61%	0.080
Joy	76.63%	10.69%	15.11%	0.125	50.56%	46.56%	15.06%	0.228

Table 4.6: Weighted metrics of the 7-class model on both vocabularies.

	Vocabulary A	**Vocabulary B**
Weighted Precision	31.21%	18.99%
Weighted Recall	31.94%	27.43%
Weighted F1-score	22.75%	8.40%

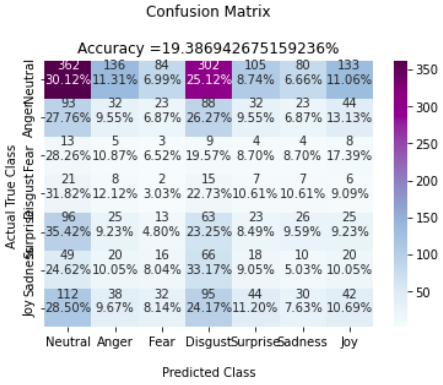

Figure 4.4: Seven class confusion matrix for vocabulary A.

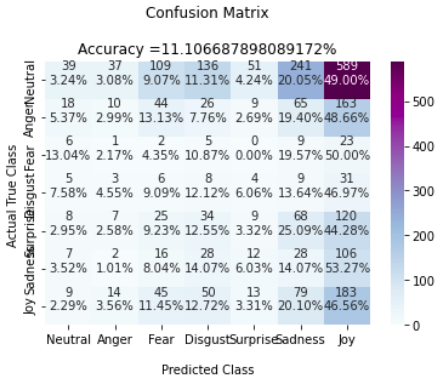

Figure 4.5: Seven class confusion matrix for vocabulary B.

Conclusions

A final comparison of the results can be seen in Figure 4.6. Overall, it is clear that vocabulary A has better results for sentiment and emotion classification, with an average accuracy of 34.55% and 19.39% respectively, than B, with an average accuracy of 26.35% and 11.11%.

This can be explained by the reduced size of vocabulary B. Although B is filled with words deemed relevant to sentiment analysis, the fact that very few patterns in the testing set exist, results in sparsity thus decreasing the training accuracy and leads to unreliable results.

Three-class classification also obtained better results than seven-class. Regarding the emotion classification, the number of samples corresponding to each class was highly unbalanced, resulting in the model's poor performance, specifically for the minority classes. This unbalance is very clear in emotion classification, where the majority class' accuracy (neutral) is far superior to the remaining classes.

In short, the developed MLP model achieved a maximum accuracy of 34.55% and 29.20% weighted F1-score for sentiment classification and 19.39% accuracy and 22.75% weighted F1-score for emotion.

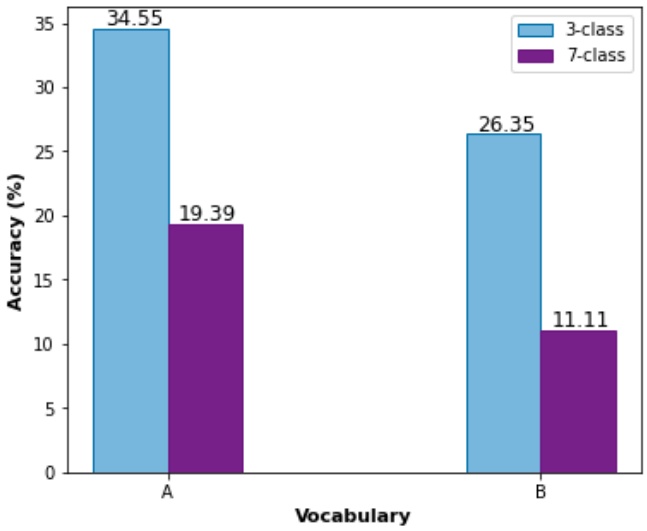

Figure 4.6: Average accuracy of the model on both vocabularies.

4.3.2 Second experiment

Considering how the textual input was vectorized for the LSTM, it did not make sense to experiment using vocabulary B due to the high number of sparse vectors. Therefore, all results obtained with this network only regard vocabulary A.

Just like before, four confusion matrix were obtained, yet only the results concerning sentiment classification will be presented, and can be seen in Figure 4.7 and in Table 4.7.

The reason for this is that the oversampling technique, used to compensate the imbalanced data, led the model to overfit. An overfitting behaviour can be characterised as when the model ends up memorizing data patterns, failing to generalize when facing unseen data. Due to this, instead of properly attributing labels to each utterance, the developed model ends up always predicting the same class - disgust.

Table 4.7: Performance of the 3-class model on vocabulary A.

	Vocabulary A			
Sentiment Label	**Accuracy**	**Precision**	**Recall**	**F1 score**
Neutral	52.07%	2.66%	48.49%	0.05
Positive	45.14%	60.63%	20.73%	0.31
Negative	53.82%	37.53%	31.35%	0.34

Table 4.8: Weighted metrics of the 3-class model on vocabulary A.

	Vocabulary A
Weighted Precision	41.53%
Weighted Recall	29.73%
Weighted F1-score	26.70%

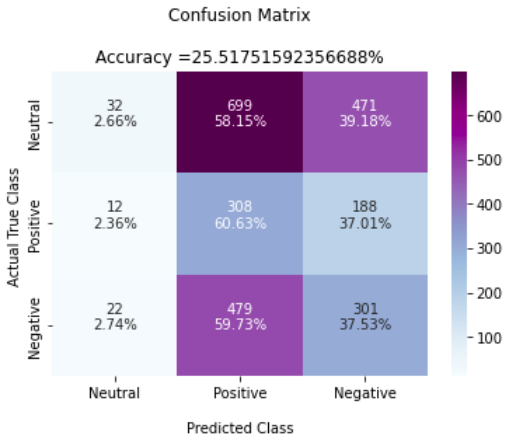

Figure 4.7: Three class confusion matrix for vocabulary A.

Conclusions

The LSTM achieved an overall accuracy of 25.52% and a 26.70% weighted F1-score for sentiment classification, values fairly close to the ones obtained with the MLP. Therefore one conclusion to be taken should be that the dataset was not appropriate for the developed work, since it resulted in equally low metrics despite testing two distinct deep learning methods.

Chapter 5

Conclusions and Future Work

The proposed work involved designing a system capable of detecting three sentiments – negative, neutral, and positive – and seven emotions – joy, sadness, fear, anger, surprise, disgust, and neutral – in a multiparty dialogue.

The initial phase of the project development relied on the comprehension of theoretical aspects required for the proper realisation of the proposed work and on understanding the difficulties and limitations that could arise in the process.

After that, a thorough investigation was conducted to comprehend how existing models work. This was accomplished by studying the literature regarding similar systems and exploring state-of-the-art solutions.

It was possible to start developing the system after doing the necessary research. The process was divided into stages: pre-processing textual data from the dataset; creating two distinct vocabularies; defining two models, a multilayer perceptron network and a long short-term memory network; vectorizing and normalizing data according to the model's needs; training the models; evaluating their performance.

5.1 Satisfaction of the objectives

The first conclusion that must be drawn is that the choice of the dataset was not the most appropriate considering the type of strategy chosen for the project.

First of all, it is a highly imbalanced dataset, especially regarding emotions, which caused a lot of setbacks during the work. Secondly, the words that make up the dataset are not very relevant in the sense that they do not represent any specific sentiment or emotion.

Regarding this, a previous analysis of the dataset showed that the words that occurred more frequently were the ones that were eliminated during the pre-processing phase because they were considered stopwords or words without any sentimental weight. The remaining words, through which vocabulary A was created (and from that vocabulary B), had such a low frequency, appearing only in one or another utterance, that it was impossible to find a relationship between these exact words and the sentiment/emotion associated with it. For this reason, the chosen method

of analyzing keywords that indicated a particular sentiment/emotion was not the most reasonable alternative.

Following this, it can also be concluded that picking the appropriate vocabulary is a crucial step in text classification. The chosen procedure was selecting words from the complete set that could imply some sentiment or emotion. Using this vocabulary A, the results obtained for sentiment classification were reasonable using both models.

The creation of vocabulary B took it a step further and compared the initial selection with a list of positive and negative words, keeping only those mutual to both lists. However, given the reduced occurrence of these words during the dialogues, this vocabulary did not generate great results.

Regarding emotion classification using the first model, the results were unsatisfactory. The dataset is highly imbalanced when it comes to emotion. Algorithms are designed to maximize accuracy, so whenever there is a disproportional ratio of samples per class, the model mainly predicts the majority class, disregarding the minorities. Even though weights were attributed to each class, the effect of this imbalance was still perceptible in the results.

The results obtained cannot be fairly compared with the other existing models tested on MELD, given that the chosen approach is quite distinct. Regardless, in terms of accuracy, the produced work did not outperform them, which can be justified by the feature extraction method. Although it seemed highly pertinent for sentiment and emotion classification, it may not have been the best fit for this particular dataset, as mentioned before.

5.2 Future work

This work was only the first step toward creating a dialogue analysis system capable of detecting misuse and malicious behaviours.

Future work includes choosing a more appropriate and less imbalanced dataset, in order to achieve better results.

Only after that it would be possible to start improving this project by developing a contextual model, which represents an increase in the system's complexity. In a dialogue, there is a high probability of interdependence between the utterances [28]. Therefore, the system must consider the current sentence, and the ones said previously to constantly update the context and the conversation flow.

On top of that, the model could be updated to learn interactions between the interlocutors like Ghosal et al. (2020) [55] proposes, using different elements of common sense such as personalities, mental states, intents, and emotions to improve the extraction of information from the dialogues.

Lastly, to provide robust emotion classification in the context of a multi-party conversation, the model could be combined with image and speech analysis systems, creating a multimodal approach that could serve as a powerful security system.

References

[1] Keiron O'shea and Ryan Nash. An introduction to convolutional neural networks. Technical report.

[2] Sentiment analysis applications in business | repustate. https://www.repustate.com/sentiment-analysis-applications/. Accessed on 10.03.2022.

[3] How can you benefit from using emotion recognition software? https://sightcorp.com/knowledge-base/emotion-recognition/. Accessed on 10.03.2022.

[4] Deepanway Ghosal, Navonil Majumder, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. Cosmic: Commonsense knowledge for emotion identification in conversations. *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*, pages 2470–2481, 10 2020. URL: https://arxiv.org/abs/2010.02795v1, doi:10.18653/V1/2020.FINDINGS-EMNLP.224.

[5] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. Meld: A multimodal multi-party dataset for emotion recognition in conversations. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 527–536, 10 2018. URL: https://arxiv.org/abs/1810.02508v6, doi:10.18653/v1/p19-1050.

[6] Elizabeth D Liddy. Natural language processing natural language processing natural language processing 1. 2001. URL: https://surface.syr.edu/istpub.

[7] What are neural networks? | ibm. https://www.ibm.com/cloud/learn/neural-networks. Accessed on 8.02.2022.

[8] Yuandong Luan and Shaofu Lin. Research on text classification based on cnn and lstm. *Proceedings of 2019 IEEE International Conference on Artificial Intelligence and Computer Applications, ICAICA 2019*, pages 352–355, 3 2019. doi:10.1109/ICAICA.2019.8873454.

[9] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. pages 3285–3292, 2019. doi:10.1109/BigData47090.2019.9005997.

[10] What are recurrent neural networks? | ibm. https://www.ibm.com/cloud/learn/recurrent-neural-networks. Accessed on 8.02.2022.

[11] Ankush Chatterjee, Umang Gupta, Manoj Kumar Chinnakotla, Radhakrishnan Srikanth, Michel Galley, and Puneet Agrawal. Understanding emotions in text using deep learning and big data. *Computers in Human Behavior*, 93:309–317, 4 2019. doi:10.1016/J.CHB.2018.12.029.

[12] Chung-Hsien Wu, Ze-Jing Chuang, and Yu-Chung Lin. Emotion recognition from text using semantic labels and separable mixture models. *ACM Transactions on Asian Language Information Processing*, 5:165–182, 2006.

[13] Doaa Mohey El-Din. Enhancement bag-of-words model for solving the challenges of sentiment analysis. *International Journal of Advanced Computer Science and Applications*, 7(1), 2016.

[14] Niko Colneric and Janez Demsar. Emotion recognition on twitter: Comparative study and training a unison model. *IEEE Transactions on Affective Computing*, 11:433–446, 7 2020. doi:10.1109/TAFFC.2018.2807817.

[15] Yoon Kim. Convolutional neural networks for sentence classification. Technical report, 2014. URL: http://nlp.stanford.edu/sentiment/.

[16] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. pages 1631–1642, 2013. URL: http://nlp.stanford.edu/.

[17] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis. 2017.

[18] Amir Zadeh, Paul Pu Liang, Jonathan Vanbriesen, Soujanya Poria, Edmund Tong, Erik Cambria, Minghai Chen, and Louis-Philippe Morency. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. pages 2236–2246. URL: https://github.com/A2Zadeh/CMU-.

[19] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. 2017. URL: https://concept.msra.cn/.

[20] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. pages 17–21, 2015. URL: www.gelbukh.com.

[21] Louis Philippe Morency, Rada Mihalcea, and Payal Doshi. Towards multimodal sentiment analysis: Harvesting opinions from the web. *ICMI'11 - Proceedings of the 2011 ACM International Conference on Multimodal Interaction*, pages 169–176, 2011. doi:10.1145/2070481.2070509.

[22] Sayyed M Zahiri and Jinho D Choi. *Emotion detection on tv show transcripts with sequence-based convolutional neural networks*. 2018.

[23] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. June 2015.

[24] Devamanyu Hazarika, Soujanya Poria, Amir Zadeh, Erik Cambria, Louis-Philippe Morency, and Roger Zimmermann. Conversational memory network for emotion recognition in dyadic dialogue videos. Technical report, 2018.

[25] Carlos Busso, Murtaza Bulut, Chi Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth S. Narayanan. Iemocap: interactive emotional dyadic motion capture database. *Language Resources and Evaluation 2008 42:4*, 42:335–359, 11 2008. URL: https://link.springer.com/article/10.1007/s10579-008-9076-6, doi:10.1007/S10579-008-9076-6.

[26] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Navonil Mazumder, Amir Zadeh, and Louis-Philippe Morency. Context-dependent sentiment analysis in user-generated videos. pages 873–883. URL: https://doi.org/10.18653/v1/P17-1081, doi:10.18653/v1/P17-1081.

[27] Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. Dialoguernn: An attentive rnn for emotion detection in conversations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6818–6825, Jul. 2019. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4657.

[28] Sheng Yeh Chen, Chao Chun Hsu, Chuan Chun Kuo, Ting Hao Kenneth Huang, and Lun Wei Ku. Emotionlines: An emotion corpus of multi-party conversations. *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pages 1597–1601, 2019. URL: https://github.com/declare-lab/MELD.

[29] Jorge Oliveira and Isabel Praça. On the usage of pre-trained speech recognition deep layers to detect emotions. 2021. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9319855.

[30] Jupyter project documentation — jupyter documentation 4.1.1 alpha documentation. https://docs.jupyter.org/en/latest/. Accessed on 16.07.2022.

[31] Paperspace docs | paperspace. https://docs.paperspace.com/. Accessed on 15.07.2022.

[32] About keras. https://keras.io/about/. Accessed on 25.06.2022.

[33] Tensorflow. https://www.tensorflow.org/. Accessed on 25.06.2022.

[34] Intro to data structures — pandas 1.4.3 documentation. https://pandas.pydata.org/docs/user_guide/dsintro.html. Accessed on 16.07.2022.

[35] re — regular expression operations — python 3.10.6 documentation. https://docs.python.org/3/library/re.html. Accessed on 16.07.2022.

[36] Nltk :: Natural language toolkit. https://www.nltk.org/. Accessed on 16.07.2022.

[37] Nltk stop words - python tutorial. https://pythonspot.com/nltk-stop-words/. Accessed on 16.07.2022.

[38] Python programming tutorials. https://pythonprogramming.net/lemmatizing-nltk-tutorial/. Accessed on 16.07.2022.

[39] Wenhu Chen, Yu Su, Yilin Shen, Zhiyu Chen, Xifeng Yan, and William Wang. How large a vocabulary does text classification need? a variational approach to vocabulary selection. pages 3487–3497.

[40] Nltk :: Sample usage for wordnet. https://www.nltk.org/howto/wordnet.html. Accessed on 25.06.2022.

[41] Python programming tutorials. https://pythonprogramming.net/part-of-speech-tagging-nltk-tutorial/. Accessed on 16.07.2022.

[42] Bing Liu. Sentiment analysis and opinion mining. 2012.

[43] sklearn.preprocessing.onehotencoder — scikit-learn 1.1.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html. Accessed on 15.07.2022.

[44] 1. introduction — version 0.9.1. https://imbalanced-learn.org/stable/introduction.html. Accessed on 20.01.2022.

[45] tf.keras.model | tensorflow core v2.9.1. https://www.tensorflow.org/api_docs/python/tf/keras/Model. Accessed on 25.06.2022.

[46] Method to generate class weights given a multi-class or multi-label set of classes using python, supporting one-hot-encoded labels. · github. https://gist.github.com/angeligareta/83d9024c5e72ac9ebc34c9f0b073c64c#file-generate_class_weights-py. Accessed on 16.07.2022.

[47] The sequential model. https://keras.io/guides/sequential_model/. Accessed on 25.06.2022.

[48] Batchnormalization layer. https://keras.io/api/layers/normalization_layers/batch_normalization/. Accessed on 25.06.2022.

[49] Layer activation functions. https://keras.io/api/layers/activations/. Accessed on 25.06.2022.

[50] Probabilistic losses. https://keras.io/api/losses/probabilistic_losses/#categoricalcrossentropy-class. Accessed on 25.06.2022.

[51] Accuracy metrics. https://keras.io/api/metrics/accuracy_metrics/#accuracy-class. Accessed on 25.06.2022.

[52] tf.keras.optimizers.adam | tensorflow core v2.9.1. https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam. Accessed on 25.06.2022.

[53] Masking and padding with keras | tensorflow core. https://www.tensorflow.org/guide/keras/masking_and_padding. Accessed on 25.06.2022.

[54] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. 8 2020. URL: https://arxiv.org/abs/2008.05756v1, doi:10.48550/arxiv.2008.05756.

[55] Deepanway Ghosal, Navonil Majumder, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. Cosmic: Commonsense knowledge for emotion identification in conversations. *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*, pages 2470–2481, 10 2020. URL: https://arxiv.org/abs/2010.02795v1, doi:10.18653/V1/2020.FINDINGS-EMNLP.224.