THE USE OF ARTIFICIAL INTELLIGENCE FOR THE INTERPRETATION OF EMOTION IN TEXT

Diploma Thesis

Paraskevi Maria Bakami

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Supervisor:

Prof. Aspassia Daskalopulu

Volos, 2021

THE USE OF ARTIFICIAL INTELLIGENCE FOR THE INTERPRETATION OF EMOTION IN TEXT

Diploma Thesis

Paraskevi Maria Bakami

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Supervisor:

Prof. Aspassia Daskalopulu

Volos, 2021

ΧΡΗΣΗ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ ΓΙΑ ΤΗΝ ΕΡΜΗΝΕΥΣΗ ΣΥΝΑΙΣΘΗΜΑΤΩΝ ΣΕ ΚΕΙΜΕΝΟ

Διπλωματική εργασία

Παρασκευή Μαρία Μπακάμη

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Επιβλέπουσα:

Ασπασία Δασκαλοπούλου

Βόλος, 2021

Approved by the Examination Committee:

Supervisor	**Aspassia Daskalopulu** Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Michael Vassilakopoulos** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Panagiota Tsompanopoulou** Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Date of approval: 09/07/21

Dedicated to my family

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to my supervisor, Prof. Aspassia Daskalopulu for giving me the opportunity to work on this subject and for her consistent, invaluable advice and support throughout the effort of writing this thesis. I would also like to thank Prof. Panagiota Tsompanopoulou and Prof. Michael Vassilakopoulos for putting their trust in me and accepting to be part of the examination committee. Last but certainly not least, I would like to express my deepest appreciation for the work done by the mental health counselor and external collaborator of the Psychology Lab of the University of Thessaly, Viki Steka, who provided me with significant support and encouragement during the difficult years of my studies.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The point where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The Declarant,

Bakami Paraskevi Maria, 09/07/21

ABSTRACT

Due to the increasing user engagement in several online communities, like social networks and microblogging platforms in recent years, emotion detection from text is developing into a field of high significance. Emotions offer unique insight into the thoughts, behaviour and motivations of humans. At the same time, automatic emotion detection is an indispensable step towards developing truly intelligent machines. Despite the plethora of available datasets and research for sentiment polarity detection from text, not enough attention has been given to the more detailed detection of multiple emotion classes. This thesis aims to investigate the prevalent emotion categorization approaches, collect and compare the most notable datasets and evaluate the performance of four supervised machine learning algorithms on this task. Based on the results of the comparative analysis of the datasets, three were selected, preprocessed and combined into two different ways, forming a final balanced and an unbalanced dataset on which we run our experiments. Moreover, TF-IDF, Bag-of-Words and n-grams were the vectorization approaches used to translate the text examples into numerical values before training the machine learning algorithms on them. Furthermore, the metric used to assess the performance of each algorithm was the F1 score. The best performing algorithm overall turns out to be the Linear Support Vector Machine, with Linear Regression being a close second that mostly fails to classify correctly the underrepresented emotions in the unbalanced dataset. Surprisingly, Multinomial Naive Bayes performs better than the Decision Trees and in spite of the simplicity of the former. Even though the results demonstrated some satisfying F1 scores, the Decision Trees underperformed and, most importantly, all approaches retained and extracted very little contextual and semantic information. On this basis, it is recommended that future research focuses on the use of ensemble learning methods that utilize multiple Decision Trees as well as word embeddings and neural network architectures.

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια η αυξανόμενη δημοφιλία των διάφορων διαδικτυακών κοινοτήτων, όπως τα μεσα κοινωνικής δικτύωσης και οι πλατφόρμες μικρο-blogging, έχει σαν αποτέλεσμα η αναγνώριση συναισθημάτων από κείμενο να εξελίσσεται σε έναν τομέα μεγάλης ερευνητικής σημασίας. Η γνώση των συναισθημάτων προσφέρει μοναδική διορατικότητα όσον αφορά στις σκέψεις και τα κίνητρα που καθορίζουν την ανθρώπινη συμπεριφορά. Συγχρόνως, η αυτόματη αναγνώριση τους αποτελεί απαραίτητη προϋπόθεση για την ανάπτυξη πραγματικά ευφυών μηχανών. Παρόλο που υπάρχει πληθώρα ερευνητικού υλικού και συνόλων δεδομένων για την ανίχνευση συναισθηματικής πολικότητας από κείμενο, δεν έχει δοθεί αρκετή προσοχή στην πιο λεπτομερή αναγνώριση πολλαπλών συναισθημάτων. Η εργασία αυτή σκοπεύει να διερευνήσει τα κυρίαρχα μοντέλα κατηγοριοποίησης των συναισθημάτων, να συλλέξει και να συγκρίνει τα πιο σημαντικά σύνολα δεδομένων και να αξιολογήσει την επίδοση τεσσάρων αλγορίθμων Επιβλεπόμενης Μηχανικής Μάθησης (Supervised Machine Learning). Με βάση τα αποτελέσματα της συγκριτικής ανάλυσης τρία σύνολα δεδομένων επιλέχθηκαν, προ-επεξεργαστηκαν και ενώθηκαν σε δύο τελικά, ένα ομοιογενές και ένα ανομοιογενές πάνω στα οποία τρέξαμε τα πειράματα. Ακόμη, για την αναπαράσταση των κειμένων σε αριθμητικές τιμές πριν την εκπαίδευση των αλγορίθμων, οι προσεγγίσεις που χρησιμοποιήθηκαν ήταν οι Bag-of-Words, TF-IDF και n-grams. Επιπρόσθετα, το μέτρο αξιολόγησης της επίδοσης των αλγορίθμων που χρησιμοποιήθηκε ήταν το F1 score. Την καλύτερη επίδοση σημειώνουν οι γραμμικές Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines) καθώς και η Λογιστική Παλινδρόμηση (Logistic Regression) η οποία αποτυχαίνει να κατηγοριοποιήσει σωστά κυρίως τις κλάσεις για τις οποίες το σώμα κειμένων (corpus) διαθέτει τα λιγότερα δείγματα. Παραδόξως, ο αλγόριθμος Αφελής Bayes, παρόλη την απλότητα του ξεπερνά σε επίδοση τα Δέντρα Απόφασης. Μολονότι τα αποτελέσματα περιείχαν κάποιες ικανοποιητικές τιμές F1, τα Δέντρα Απόφασης επέδειξαν πολύ χαμηλή αποδοτικότητα στο πρόβλημα μας και όλες οι αλγοριθμικές προσεγγίσεις που εφαρμόστηκαν έχουν τον περιορισμό του ότι εξάγουν και διατηρούν ελάχιστη πληροφορία σε σχέση με τα συμφραζόμενα και τη σημασιολογία των λέξεων στο κείμενο. Για το λόγο αυτό προτείνεται η μελλοντική διερεύνηση Συνδιαστικών Μεθόδων (Ensemble Methods) που αξιοποιούν πολλαπλά Δέντρα Αποφάσεων καθώς και διανυσμάτων λέξεων (word embeddings) και αρχιτεκτονικών Νευρωνικών Δικτύων (Neural Networks).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AI Artificial Intelligence

AC Affective Computing

BoW Bag-of-Words

CBoW Continuous Bag-of-Words

IG Information Gain

ISEAR International Survey on Emotion Antecedents and Reactions

ML Machine Learning

MLR Multinomial Logistic Regression

MNB Multinomial Naive Bayes

NLP Natural Language Processing

NLTK Natural Language Toolkit

OCC Ortony, Clore and Collins

OvA One-vs-All

OvO One-vs-One

POS Part-Of-Speech

SVM Support Vector Machine

TF-IDF Term Frequency - Inverse Document Frequency

CHAPTER 1

INTRODUCTION

Emotions are intertwined with intelligence and intelligent processing. They play a vital role in the regulation of attention. They also strongly influence memory and learning, decision-making as well as our prioritization capabilities. They interact with the background regulatory mechanisms that are important for functioning intelligently. More importantly, they are a crucial source of information and feedback that helps to direct our behaviour and social interactions. Therefore, it is no wonder that coordinated efforts are taking place to integrate representations and processing of emotion via the use of **Artificial Intelligence** techniques into a variety of tools.

1.1 Problem definition

The purpose of this thesis is to examine some of the most prominent components that take part in the systems that detect emotion from text. The focus is both on the theoretical elements, meaning the psychological theories and models that provide us with a basis for understanding and classifying the emotional states, as well as on the computational aspects, in the form of the algorithms and practices that prepare the textual data, represent it in a proper, numerical manner and implement its classification into emotion categories.

A simple and common form of emotion detection is sentiment analysis which is being used to distil consumer feedback, user preferences and even predictions of public opinion,

attracting the interest of multiple disciplines ranging from the social sciences to marketing and communication. The terms emotion, affect and sentiment are often used interchangeably, as umbrella terms, by researchers to describe similar meanings. In fact, sentiment analysis or opinion mining is an older and well-established area of Artificial Intelligence and Natural Language Processing in particular, focused on acquiring polarity from textual inputs, classifying users' texts as either positive, negative or neutral. Emotion recognition, on the other hand, which is the problem this thesis examines, is the automatic categorization of feeling expressions into more than just two classes, offering finer-grained control and useful insights into the complexity of the human emotional state. For instance by providing more accurate information than a casual binary sentiment classifier it can deal with dual polarity text such as "*I hated the plot of the movie… but, at least, the direction was superb.*", which can confuse the simple binary sentiment classifiers giving incorrect predictions.

1.2 Contribution to knowledge

The contribution of this work to the field of emotion detection can be summed up as follows. Various theories and models of emotion were examined, along with the ways in which they can and are being utilized by engineers to enhance automatic emotion detection from text. Moreover, all the prominent, free, emotionally labelled datasets were collected and compared. Eventually, this comparison led to the selection of three of them to run experiments upon. During the practical implementations, the data handling capabilities of the Python programming language and its libraries were also explored in order to clean thoroughly, structure and represent numerically the text data via three methods, namely, the Bag-of-Words, TF-IDF and n-grams. Four different algorithms were trained to make predictions regarding the underlying emotion of a piece of text. After the comparative analysis of their performance on the test data, the Linear Support Vector together with the TF-IDF vectorization proved to be the most robust approach, with Linear Regression paired with Bag-of-Words being a close second. The problem was approached by utilizing all the possible combinations of the distinct techniques analyzed in the chapters of this thesis and evaluated using the appropriate metrics, namely, the accuracy and the F1-score.

1.3 Background and related work

Originally, the term **Artificial Intelligence (AI)** [1] was invented in the 1950's to describe the ambition of creating, in software and hardware, entities that are endowed with intelligence resembling that of humans. Moreover, **Affective Computing (AC)** [2] is the subfield of AI that is trying to incorporate feeling into thinking machines. To be more precise, the three pillars of Artificial Emotional Intelligence, as shown in Fig. 1.1, are the detection of human affect, the instilment of emotions in artificial conversational entities and robots, as well as the enhancement of existing machine learning algorithms with the use of emotion-related concepts [3]. In other words, the perception, modeling and expression of emotion by machines are all issues that fall within the scope of the AC research.

The extraction of knowledge from such seemingly disorderly information as text is no easy task for machines due to the complexity, ambiguity and overall unpredictability of language. This has long been one of the objects of research of **Natural Language Processing (NLP)** which is the subfield of AI concerned with rendering machines capable

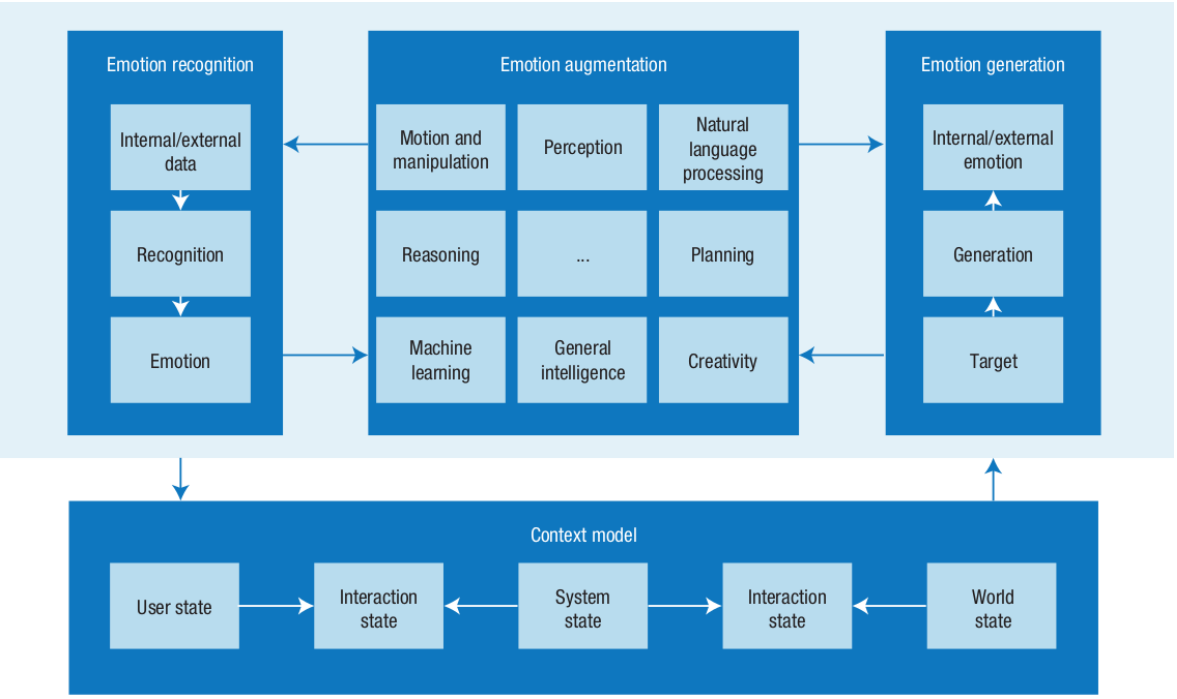

Fig. 1.1: The main domains occupied with the integration of emotion into AI [3].

of detecting, understanding, translating, summarizing language and producing language so as to communicate with people in a better way. Moreover, in recent years, the increasing user participation in social networks, microblogging platforms, blogs and other online

communities has resulted in automatic emotion detection, from textual resources in particular, to be growing into an area of great interest. Such user-generated content is an excellent source of emotional information and since it is available in abundance, it is being utilized by NLP in conjunction with another subfield of AI called **Machine Learning (ML)**.

The term ML is generally used to describe the set of algorithms that learn from data or, to put it differently, that improve their predictions based on previous observations. The supervised ML algorithms that are the focus of this thesis, require labelled data, where each label corresponds to a class. The assignment of the labels can take place either manually by people or by using various automatic techniques, such as the recognition of predominant keywords. The former is the slowest but most reliable practice while the latter is faster but error-prone and usually is succeeded by human double checking the machine assigned labels.

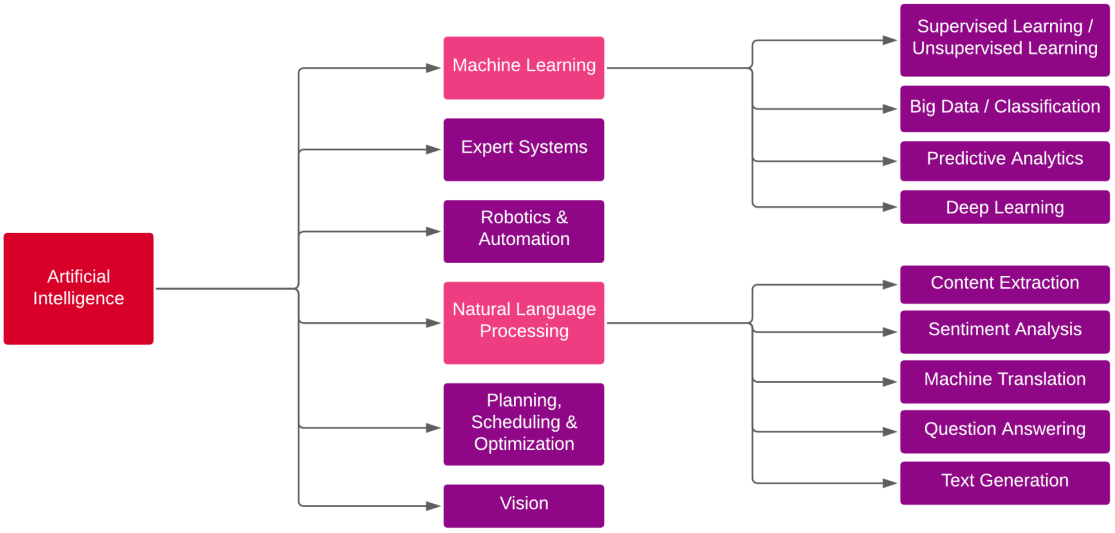

Fig. 1.2 Schematic representation of the field of AI and, indicatively, some of its subset areas of interest.

Related previous work on emotion detection and analysis from text using supervised ML varies. In [4] the Multinomial Naive Bayes (MNB) was used on only one dataset, the standard ISEAR. It was concluded that unigrams together with the Bag-of-Words vectorization gave the highest accuracy overall. Moreover, [5] used TF-IDF for feature engineering and trained multiple algorithms on the same dataset. Logistic Regression performed the best with an F1 score of 66.5%. In addition to these, other efforts focus on

utilizing microblog data. More specifically, [6] created a dataset from Chinese text posts that was labelled according to markers included in text, such as smilies and emoticons and trained a linear Support Vector Machine (SVM) to recognize mainly the *angry*, *happy* and *sad* emotions. They underlined the need for more data and the fact that bigrams and trigrams performed better than unigrams. Furthermore, the researchers of [7] collected 4,000 Twitter posts and manually assigned labels that classified them into nine categories (*anger, disgust, fear, guilt, interest, joy, sad, shame, surprise*). They showed that indeed the bigram model performed better than the Bag-of-Words and that, surprisingly, Multinomial Naive Bayes outperformed their simple Artificial Neural Network, a fact they attributed to the use of the TF-IDF vectorizer. In the end, they highlighted the need for established datasets that include multiple emotion classes, as well as to the overall superiority of the SVM for text related tasks.

1.4 Thesis roadmap

The remainder of this work is organized as follows: in Chapter 2 a study of the historically most prominent theories and models around emotion is presented, Chapter 3 introduces the best datasets with multiclass emotion labels, common preprocessing methods, as well as the different ways in which the data is usually vectorized, Chapter 4 explains in detail four supervised learning algorithms and, lastly, Chapter 5 clarifies the experiments that took place in the context of this thesis, demonstrating the preprocessing of the three datasets and the results of each algorithm that indicate Logistic Regression and Linear Support Vector performed the best. In the final paragraph of Chapter 5 suggestions towards future research and experiments are made. Specifically, the performance of ensemble methods and neural network architectures is recommended to be tested henceforth.

CHAPTER 2

THEORY OF EMOTION

2.1 Theoretical perspectives

The nature of emotions had first been pondered upon by humans long before neuroscience and psychiatry emerged as scientific fields. The pre-Socratic philosophers regarded emotions to be occurrences that served as distractions from logic and reason. Similarly, Plato argued that “passions” affect sensibility and decision-making mostly negatively and must, therefore, be put under the control of the rational part of human nature. In fact, in one of his most famous analogies he even described reason as a charioteer who has to handle two horses, representing the “passions” of a mortal’s nature, that pull the chariot in opposite directions. From the ancients to R. Descartes, the father of modern mathematics and philosophy, emotions tended to be dismissed as erroneous judgements and as a distinct, even beastial, part of human existence [8].

Historically, such dualistic views supporting the idea that reason and feelings are separate functions of the individual, dominated the Western thought for millennia. For instance, S. Freud’s influential psychological model, where the unconscious “id”, indicating the primordial instinctive human desires, is juxtaposed with the conscious “ego” and “super-ego” that underlie the more realistic and critical agents of the our mental structure respectively, also constitutes a substantially dualistic position [9]. It is not surprising that this apparent philosophical divide regarding the separate natures of affect and cognition did not leave the following major emotion theories unaffected.

2.1.1 Evolutionary perspective

Emotions did not develop all at once but, just like other characteristics, were evolved according to our ancestors' environmental needs, in nature's effort to increase species survival and reproduction chances [10]. Their purpose was to facilitate social interactions that were beneficial to the survival of the individual as well as its group. More specifically, evolutionary theorists argue that all of the members of a species will display the same behaviour when they feel a particular emotion and emphasis is given to the non-verbal communication of emotion like facial expressions and bodily stance and movements. The expression of emotions is considered to be a genetically determined physiological response to stimuli and each emotion is considered to be a combination of a few basic ones (namely, *anger*, *fear*, *surprise*, *disgust*, *happiness* and *sadness*). For instance, *terror* can be considered a more acute version of *fear*. Through its bodily manifestation (contraction of facial muscles, pupil dilation, etc.) it communicates to ourselves and to others the existence of danger to be avoided.

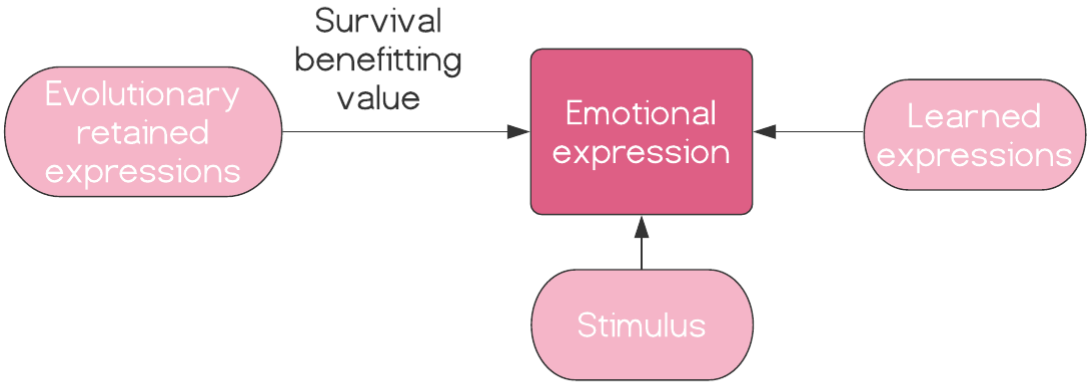

Fig. 2.1: Diagram of the Darwinian theory highlighting the survival factor in emotional expression.

2.1.2 James-Lange theory

The common-sense view suggested that a stimulus prompts an emotion which, in turn, prompts some bodily arousal. This theory has been contradicted by the influential theory developed independently by [11,12] that argued that the external factor, instead of triggering the emotion, actually activates the bodily arousal which, in turn, causes the emotion. In other words, the perception of the physiological arousal by the brain is what

ultimately causes the emotions. Moreover, modern neuroscience confirms some variations of this theory emphasizing the important part feelings play in decision-making [13].

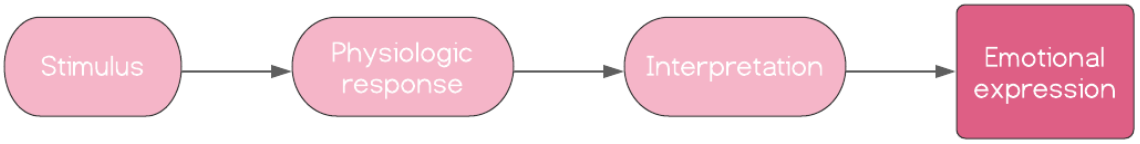

Fig. 2.2: Diagram of the James-Lange theory that highlights the gravity of bodily changes in the process of emotional expression.

2.1.3 Cannon-Bard theory

The physical and the emotional responses take place simultaneously and independently of each other. The argument behind this theory is that the thalamus sends the messages to the Autonomic Nervous System (ANS) and to the cerebral cortex at the same time. Since there have been various reports demonstrating that physiological responses can, indeed, cause emotional ones, it is highly unlikely, as this thalamic theory proposes, that such events are completely disconnected from each other [14,15].

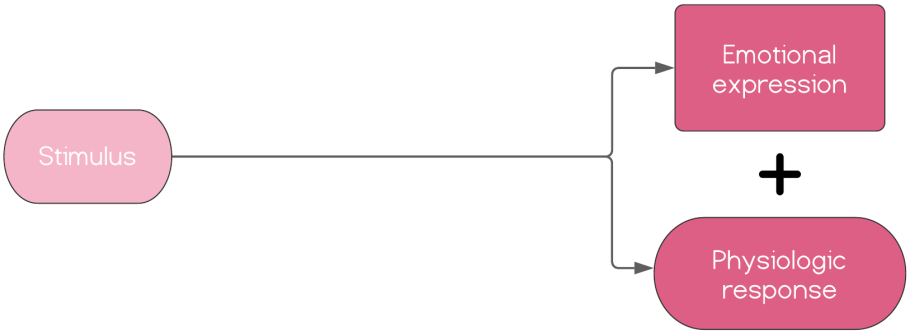

Fig. 2.3: The Cannon-Bard theory emphasizes that bodily changes and emotion take place simultaneously

2.1.4 Two-factor theory

A stimulus elicits arousal which, in turn, activates some cognition that interprets the source of the arousal. It is noteworthy that some emotions, like fear and excitement, involve very similar physiological arousal components. This theory [16] emphasizes the element of cognition which facilitates the deciphering of the environmental and bodily cues in order to

understand which emotion is fitting to the situation. Therefore, it argues that feelings are produced through a combination of both, bodily arousal and, primarily, rational assessment.

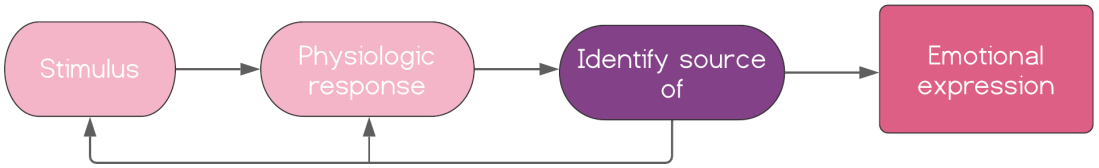

Fig. 2.4: The Schachter-Singer theory focuses on the cognitive label of the stimulus that caused the response.

2.1.5 Different routes to emotion

There are two prominent schools of thought regarding the connection between how we think (cognition) and how we feel (emotion). On one hand, [17] concluded that simpler, more primitive emotions, like anger and fear, are processed without any thinking taking place. In other words, according to this theory, emotions can occur before cognition takes place. An example of this is when we fear a snake even though we know it is an innocuous one. As shown in Fig 2.5(b), our fear stimulus, e.g. the image of a snake, is transmitted to the thalamus which is the relay centre of the brain. The thalamus then sends this information to the amygdala, which is the brain's first responder to stress, completely bypassing the cortical areas involved in thinking.

On the other hand, [18] others agree with an evolved version of the two-factor theory, supporting that an emotion does not exist until we label it. However, it is claimed that the cognitive labelling occurs unconsciously and we are not aware that we assign a specific labelling and feel an emotion in a specific context. For instance, we might feel guilty but be unable to pinpoint the exact reason. Put differently, complex emotions like guilt and love involve at least some, even effortless, appraisal of the situation. As shown in Fig 2.5 (a), in this case the information of the stimulus, after being received by the thalamus, is sent to the cortical neural pathways of the brain to be evaluated before finally being sent to the amygdala to activate the stress response.

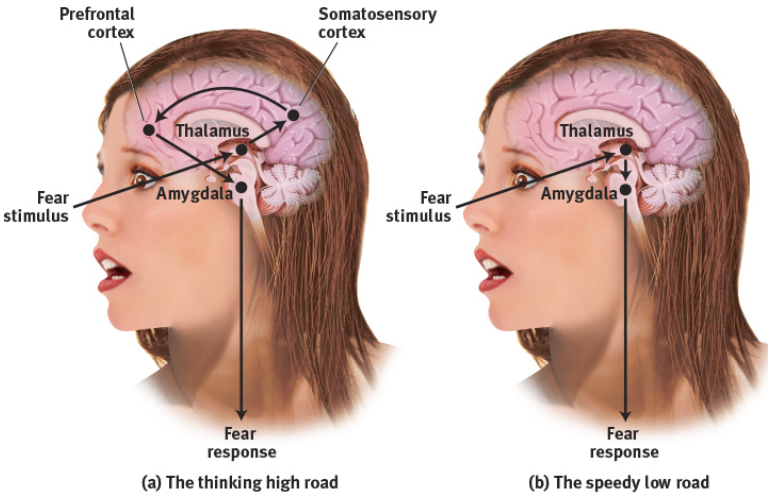

Fig 2.5: The two-track brain processes sensory input on two different pathways. (a) Some input travels to the cortex for analysis and is then sent to the amygdala. (b) Other input travels directly to the amygdala for an instant emotional reaction. [19]

2.2.6 The takeaway for emotion detection

How do these theories benefit the Natural Language Processing scientist?

One could argue that all these theories seem to oppose each other. Actually, what they do is concentrate on and prioritize different aspects of the complex, multidimensional phenomenon of the emotional experience. The NLP researchers have to deduce which theory offers the most appropriate basis depending on the nature of the problem they are researching and the data available.

For instance, Darwin's theory of emotion highlights the expression of emotion through non-verbal means like facial and bodily expressions that are ubiquitous across species [10]. In NLP this fact would be utilized via the consideration of texts such as "They are frowning at me" or emojis. A prerequisite for the implementation of such elements of evolutionary theories in NLP emotion detection is the provision of visual pointers in text.

Likewise, aspects of the James-Lange theory that points out that physiological changes go hand-in-hand with emotion expression can be employed by taking into account phrases that outline such bodily alterations. Some text examples would be "My heart started racing" that might reference anticipation and "I started trembling" that denotes panic.

Additionally, the neurophysiological and cognitive-appraisal approaches presented earlier pose a different challenge to adopt, especially emotion recognition in text. In the first case, this is due to the emphasis given on the lack of awareness regarding the stimulus while in

the case of the cognitive-appraisal theories, the implementation difficulties arise from the subjective nature of stimuli assessment which they emphasize. In conclusion, these theories offer varying frames of mind as well as potential implications regarding the methods used in emotion detection in text.

2.2 How do we classify emotion?

It is imperative that we examine the major emotion models because of the ways in which they affect the development of systems. The models related to the purpose of this thesis are separated into two broad categories depending on the way they represent and specify the emotional states: dimensional and discrete. Both of them have built upon the basis provided by the aforementioned emotion theories, as well as contemporary brain studies [20].

2.2.1 Dimensional models

These models presume that emotions are not independent of one another and approach them as entities that can be described in terms of dimensions with the number of dimensions varying from model to model. Dimensional models are especially useful for the discernment of similarities between emotions [21].

Russell's model of affect [22] is a two dimensional, circular plotting of emotions. One axis represents valence, meaning how positive or negative an emotion is, while the other constitutes the arousal or activation axis (see Fig. 2.6). Moreover, the wheel of emotion [23] is an augmentation of this model, which includes eight color-coded, basic emotions presented as four pairs of opposite emotions (see Fig. 2.7).

However, the two-dimensional models have been criticized as being too superficial, unable to capture the complexity of emotions. To amend this, variations with the addition of a third dimension, namely depth, have been proposed, in order to represent how strong or dominant a feeling is as seen in Fig. 2.7.

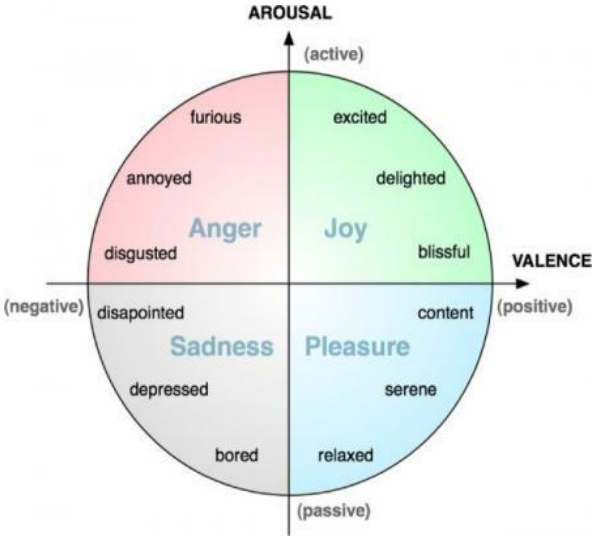

Fig. 2.6

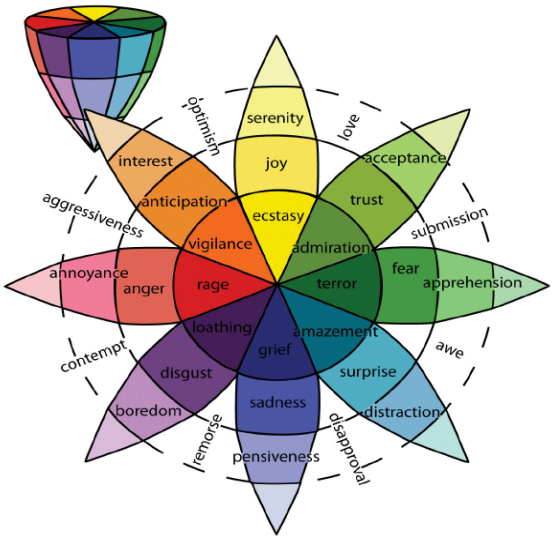

Fig. 2.7

Fig. 2.6: 2-D circumplex of affect [24]. Fig. 2.7: The cone-like wheel of emotions depicting the relations between emotions, depending on their relative position to each other. The depth of their position in the cone represents their intensity, photo taken from [23].

2.2.2 Discrete models

Models that come under this group classify emotions into distinct categories based on the idea that there are some primitive emotions with innate and universal qualities. More elaborately, discrete, basic models of emotion propose that human beings have a specific group of emotions that are basic in a biological and a psychological manner, each being reflected in the brain activations, as well as in patterns of behaviour [25]. Although there is general disagreement on which are actually these core emotions, the two prevailing models most commonly adopted by AI researchers are presented below.

Paul Ekman proposes one of the most prominent discrete psychological models [26]. Based on his cross-cultural research on facial expressions and their universality and innateness, he labeled six emotions as primary: *happiness, surprise, sadness, anger, disgust,* and *fear* and he later expanded this list to include *pride, shame, embarrassment,* and *excitement* [27]. Moreover, this research argued that some emotions are universal across cultures, even though some lack the exact words to describe them and that what varies is actually the degree of expression of each emotional expression. These basic emotions are viewed as atomic components, that is, they cannot be further reduced. It is also suggested that they emerge from separate neural systems and that the occurrence of

more complicated emotions e.g. *despair* is the result of the combination of these more primitive emotional states: fear and sadness. In contrast to the basic emotions though, they vary greatly across individuals as they are considered to be formed by sociocultural factors [28].

Some agree on the premise of basic emotions and even extend the previous model by adding more, such as *acceptance* and *anticipation*, like Plutchik's wheel of emotion [23], a hybrid of a dimensional and discrete model presented earlier in Fig. 2.7. However, others, like Ortony, Clore and Collins, the creators of the OCC model, disputed the idea of fundamental emotions, at least in the form suggested by the aforementioned studies [29]. Instead, their cognitive model, drawing from physiology, language, self-report and behavior includes an expansive classification of additionally up to sixteen emotion classes to the nine of Ekman as shown in the box-like leaves of the tree in Fig. 2.8 below.

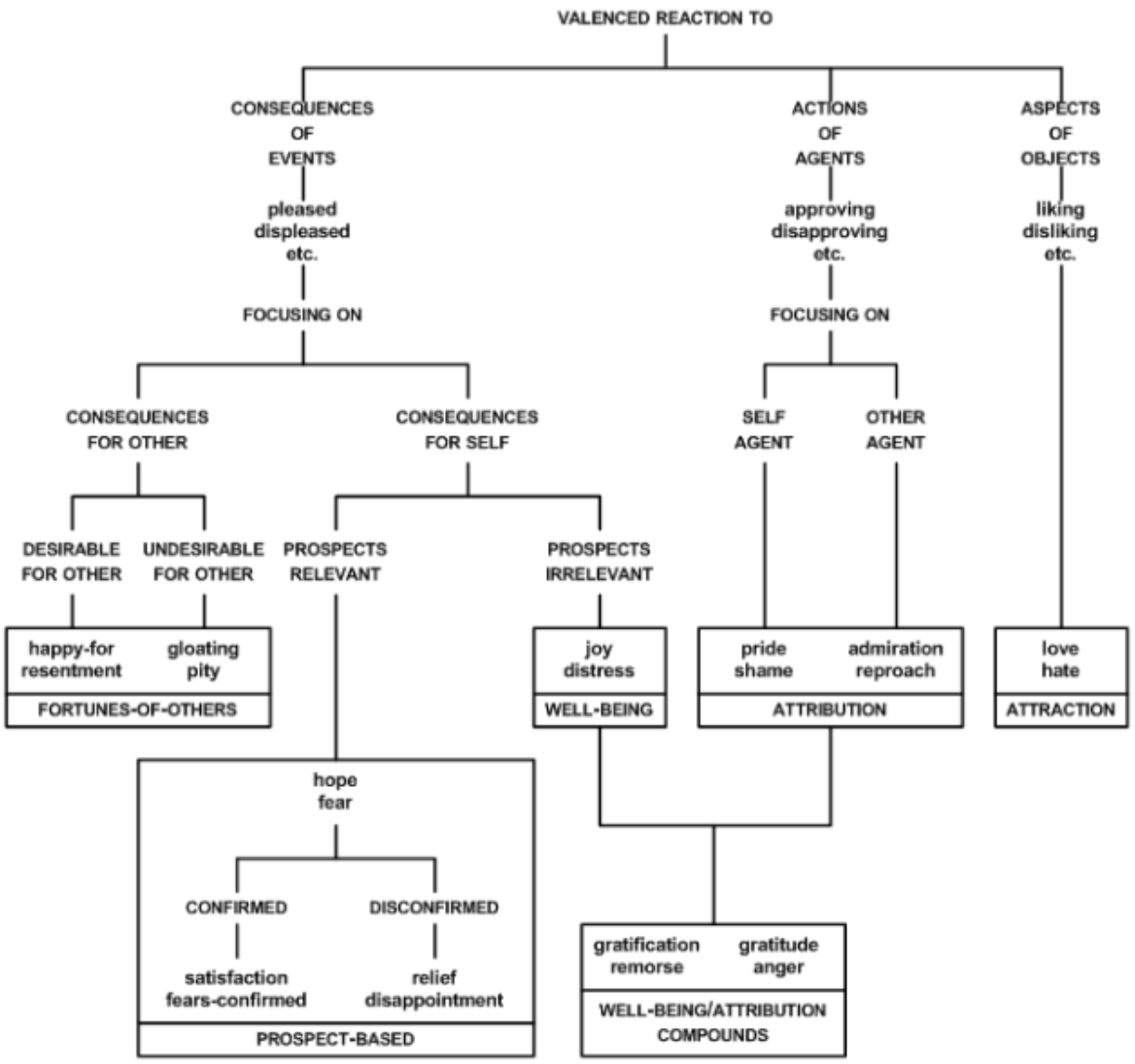

Fig. 2.8 The OCC model of emotion [29]

This model outlines a structure that contains three branches, specifically, emotions that are triggered depending on: the effects of events, such as *pity*, the actions of agents, such as *pride* and the aspects of objects, such as *love*. Moreover, some branches get combined, constituting a set of complex emotions, meaning emotions, such as *gratitude*, concerning outcomes of events brought about by the actions of agents. Because events, actions and objects are central ideas used in agent models, this renders the OCC model fitting to utilize in artificial agents too. In NLP for emotion detection the OCC model is employed either in implementations that are purely rule-based [30] or in conjunction with statistical methods and dictionaries in order to obtain an emotion-dimension dictionary [31].

In general, discrete categorical models, although sometimes criticized for their simplicity, are chosen more frequently by researchers in emotion detection tasks, especially the Ekman's list of basic emotions and the OCC model that offer a wider representation of the emotional states [32].

CHAPTER 3

COMPUTATIONAL RESOURCES FOR EMOTION DETECTION IN TEXT

3.1 Labelled data

After determining the psychological model that fits the specific emotion detection problem, data collection is of utmost importance in all such classification tasks. The existing open-source datasets most frequently used in textual emotion detection have varying sizes, linguistic methods as well as underlying emotion models. In contrast to the well established sphere of sentiment analysis, for which a plethora of corpora have been compiled, the labeled data intended for the more recent field of NLP emotion classification is in short supply. What Table 3.1 demonstrates is a synopsis of the current, textual data corpora annotated for multi-class emotion recognition that are also accessible to the general public.

Due to this paucity of standard corpora with thorough linguistic generality, as well as with widely agreed upon labels, scientists are turning to the extraction of self expressive posts from microblogging platforms, even utilizing emoticons and hashtags [33].

Table 3.1: The most prominent, freely accessible datasets for textual affect recognition.

Dataset	Source	Samples	Emotion model	Dimensions
ISEAR [34]	self-report from 3000 people around the world	7,665	Discrete	joy, sadness, fear, anger, guilt, disgust and shame
SemEval-2017 Task 4 [35]	Tweets, news headlines, newspapers	1,250	Discrete	Ekman's six emotions
EmoBank [36]	news headlines, blog posts, travel guides, essays, letters, fiction	10,000	Dimensional	valence, arousal, dominance
EmoInt [37]	Tweets	7,097	Discrete	emotion intensity values for: anger, fear, joy, sadness
SMILE [38]	Tweets about the British museum	3,085	Discrete	anger, disgust, happiness, surprise and sadness
CrowdFlower [39]	Tweets	39,740	Discrete	thirteen emotions
DailyDialog [40]	daily life conversations	13,118	Discrete	Ekman's six emotions plus neutral category
The Valence and Arousal dataset [41]	Facebook posts	2,895	Dimensional	valence, arousal
Grounded emotions [42]	Tweets	2,557	Discrete	happy, sad
MELD [43]	utterances from Friends TV show	13,000	Discrete	Ekman's six emotions plus neutral and non-neutral
Emotion lines [44]	utterances from Friends TV show & Facebook chats	29,245	Discrete	Ekman's six emotions plus neutral
Emotion Stimulus [45]	Built from data containing emotions & stimuli	1,594	Discrete	annotated with Ekman's six emotions plus shame

3.1.1 Data preprocessing

In a lot of cases, in order to increase the performance of the NLP algorithms the raw text data has to be preprocessed using various techniques. The main purpose of data preparation is identifying the meaningful parts in our corpus. The preprocessing consists of the cleaning and normalization of text data with the purpose of transforming it into a form that is predictable and decipherable depending on our corpus and task at hand.

Firstly, the simple step of **removing noise** from our data takes place. Noise can be punctuation marks, multiple space characters, URLs or numbers. This step is completely dependent on the data and the problem we are trying to solve. In the case of emotion detection and if the data comes from a Twitter feed in its original crude structure, punctuation removal should be implemented with caution. This happens because hashtags and emojis are often indicators of the emotional state of the user and therefore carry useful information. At the same time, there is a need for the removal of special characters and keywords like “RT” (which stands for retweet) and HTML tags, if our data is scraped from the web [46].

A standard practice for preparing our text data is **tokenization**. This is the process of segmenting text into smaller units called tokens. Words, numbers, punctuation marks can all be considered tokens. One of the most important and frequently utilized tokenization methods is splitting the text to tokens according to the white space between them. Other, more complicated techniques, provide additional versatility by allowing the specification of auxiliary rules that take priority over the tokenization operations.

Another usual and straightforward practice that takes place in order to avoid having duplicate occurrences and inconsistencies in our dataset is **lower casing** all the text inputs. Converting all text to lower or upper case ensures our algorithm will not be using different

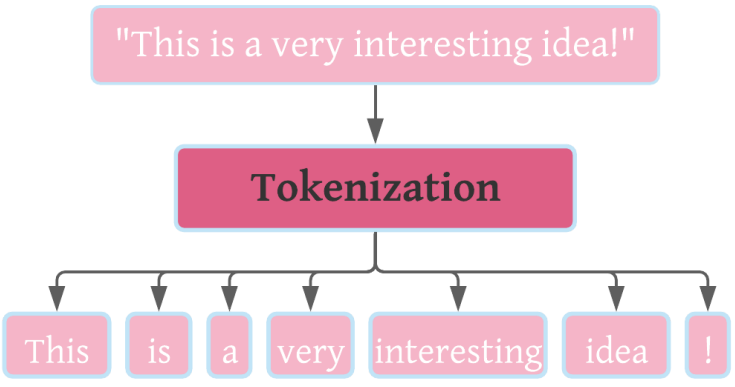

Fig. 3.1: Example of what the result is after we tokenize a sentence.

forms of one word because of mixed cases. This technique is also helpful for dealing with instances when we do not have a particularly large dataset in our disposal [46].

Likewise, another important method is **text normalization**. The two ways this usually takes place are: either with **stemming** or with **lemmatization**. More elaborately, stemming is the process of removing the affixes from the words in an effort to transform them into something resembling their root form. The outputs of the stemming procedure are not always meaningful, existing words but often a raw version of the original word. The most widely used method for stemming of english texts is the rule-based, sequential Porter's algorithm [47].

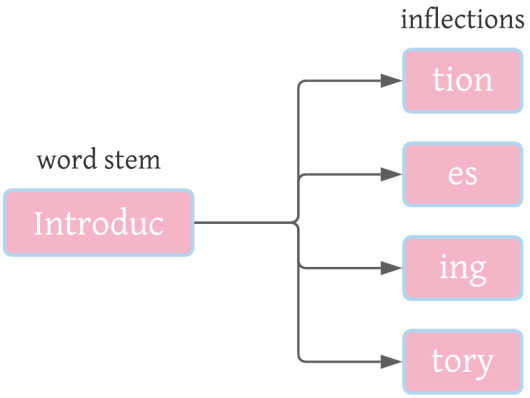

Fig. 3.2: Example of a stemmer that brings words such as "introducing", "introduction", "introduced" to the common representation "introduc".

Lemmatization, on the other hand, instead of simply cutting off the inflection of the token, performs morphological analysis on it and attempts to bring it to its proper, dictionary form. In order for a lemmatizer to work properly a **part-of-speech (POS)** tagging should take place first. This process provides additional information along with each token besides just the characters it consists of. An example of the importance of POS tagging would be a sentence such as "Mr. Barnes reluctantly agreed to visit the barns". The information of whether the word Barnes is a proper noun or not matters to the lemmatizer, because in the first case it will not lemmatize it to "barn".

Furthermore, some of the most frequently repeated words in natural language, such as “the, a, for, and, will, be, was, on, its, to…”, also called **stop words**, are usually removed from the texts that are to be the inputs to our algorithms. This happens because they are words that offer no additional meaning and just render the performance of our system slower. The appropriate stop words to be removed vary greatly from corpus to corpus. Depending on the context of the problem we are facing, we might wish to keep negation words [48]. Instead, we might wish to remove, for example in the case of a customer support assessment service, the phrase “customer support” from the comment “The inefficiency of your customer support is disappointing”. It would be preferable to focus on the important words “inefficiency” and “disappointing”.

In conclusion, thorough cleaning and manipulation of the initial data is a crucial and often overlooked part of the NLP pipeline. There is no master combination of preprocessing methods that fits all kinds of corpora and tasks so it is up to the emotion detection scientist to figure out the right one.

3.2 Feature extraction

Now that we have cleaned our data, how do we insert it to a suitable algorithm? A lot of the state-of-art algorithms and, particularly the machine learning models that have proved to be the most efficient in emotion detection and that will be analyzed later in this work, cannot process information in the form of text. They, instead, expect inputs to be in a numerical representation and more specifically, vectors. The process of transforming textual inputs into numbers in a way that maintains their original linguistic relations and attributes is called *feature encoding* or *extraction*.

3.2.1 Bag-of-words (BoW)

This is a very straightforward and adjustable method, the basic components of which are a vocabulary of known words and a measure of their occurrences. It is called a bag since this method provides no information regarding the position where a word appears, neither of the general arrangement of the input document. The idea is that the information about the frequencies of the different words carries itself useful meaning [49].

The first step is to create a vocabulary: an array that represents all of the unique words in our corpus. Initially, all of the entries will be equal to zero. The next step is to mark the presence of a word in a text by setting the value of its corresponding vocabulary position to

one. This is not related to the position of the word in the text but it has to do exclusively with the order of the words in the vocabulary array. The words that do not have corresponding entries into the vocabulary are ignored as shown in the example of Fig. 3.3.

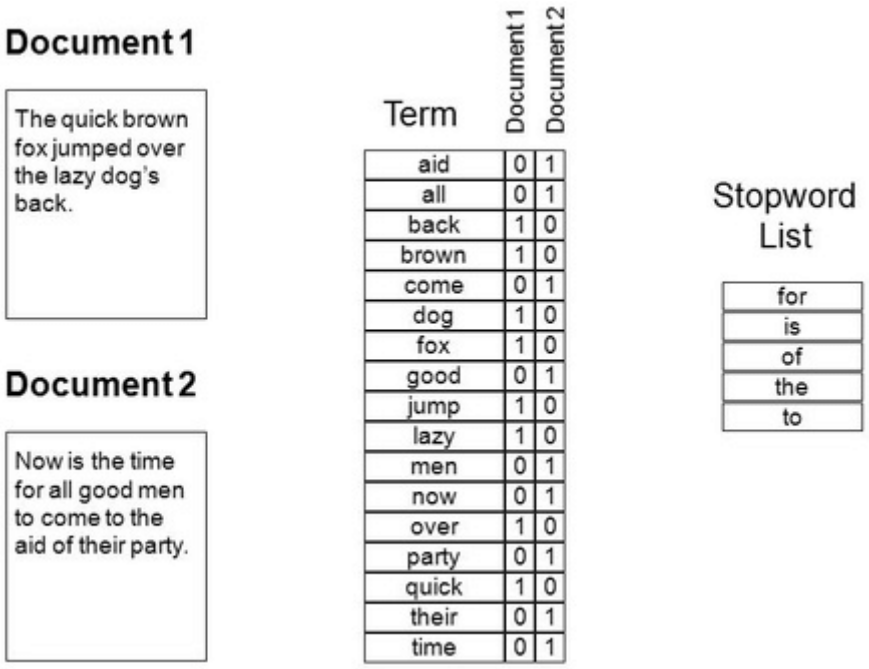

Fig. 3.3: Example of how Bag-of-Words represents the two different documents of a corpus. Image taken from [50].

This process results in a vector the size of the entire vocabulary which represents one sample of our data, ready to be fed into our model. For very large datasets and therefore vocabularies, this method usually produces equally long and sparse representations for each corpus entry. This is why an efficient cleaning of trivial words, in order to reduce the size of the vocabulary, is of paramount importance.

3.2.2 N-grams

This is a more complicated approach that somewhat amends the drawbacks of the BoW and suggests that each entry in the vocabulary should be a group of tokens instead of a single word. This group is called an n-gram where n is the number of tokens it consists of. What this method does is it captures more meaning by providing a context window for each word. The longer this context window is, though, the harder it is to pick up on words that frequently appear through the text. N-grams are especially useful when we wish to capture sarcasm and negative meaning.

In general, pairs of tokens or bi-grams are the most common choice. Despite being unable to capture the long term dependencies of language, bi-grams have been proven to be greatly more efficient than the traditional BoW approach [49].

- unigram (1-gram):

a	swimmer	likes	swimming	thus	he	swims

- bigram (2-gram):

a swimmer	swimmer likes	likes swimming	swimming thus	...

- trigram (3-gram):

a swimmer likes	swimmer likes swimming	likes swimming thus	...

Fig. 3.4: Tokenization examples utilizing n-grams. Taken from [51]

After we have created our vocabulary that consists of n-grams instead of single tokens, we need to decide on a method to count the occurrences of each vocabulary entry in a text. The simplest approach to this is binary registration of the presence of specific word combinations. Other often used methods include scoring the frequencies or counting the occurrences of each token in a specific document.

3.2.3 TF-IDF

TF-IDF [52] is a measure of the importance of a word that compares the number of times a word appears in a document with the number of documents the word appears in. It was presented by Karen Spärck Jones in 1972. The intuition behind this method is that less frequently used words carry more valuable information and thus, should be assigned bigger weights and vice versa.

It consists of two statistical components that are multiplied together:

$$tfidf(t, D) = tf(t, d) \cdot idf(t, D)$$

The first calculation is TF which stands for **Term Frequency** and which simply counts how frequently the term t, appears in the text, d:

$$tf(t, d) = f_{t|d}$$

While computing TF all tokens are considered of equal value.

IDF stands for **Inverse Document Frequency** and it represents how common or not a term is across the entire dataset. IDF is the total number of documents in the corpus, divided by the number of documents that contain the term,t.

$$idf(t,\ D)\ =\ \log(\frac{|D|}{|\{d \in D : t \in d\}|})$$

In the case that the token, t does not appear at all in our dataset and in order to avoid a zero denominator the adjustment of the denominator is a usual practice: $1\ +\ |\{d \in D :\ t \in d\}|$. As is apparent from the IDF formula, if a token appears in a large number of samples, the ratio inside the logarithm is close to 1. Therefore, the IDF value is almost 0. Similarly, the terms that appear in the least number of documents, the rarest terms in the corpus are assigned a high score by the IDF.

In conclusion, the TF-IDF vectorizer allocates high scores to elements that appear a lot of times in only a few documents and low weights to those that occur in many documents. In other words, it filters out the most common and least valuable words. While we still do not have a representation that carries enough information to deduce e.g. the similarities between words like “ecstatic” and “enthusiastic”, at least TF-IDF offers a vectorization of natural language that proves to be an input to machine learning algorithms that is far more efficient than the BoW approach [49].

3.3 Word embeddings

“You shall know a word by the company it keeps.” − J. R. Firth [53]

Unlike the BoW and the TF-IDF methods, which do not store semantic information, this very effective group of techniques captures approximations of meaning from words by transforming them into numeric vector inputs. Moreover, this transformation takes place in a way that allows words with similar meaning to have similar representations in a predefined, continuous vector space. The word embeddings approach is based on the distributional hypothesis in semantics which argues that words that are more similar to each other, tend to occur together more often.

More elaborately, word embeddings are a category of feature learning methods that provide a dense, distributed representation for each word. The embeddings are usually learned either with the utilization of Machine Learning algorithms or document statistics. The feature vector υ represents various characteristics of the word and each word is

associated with a point in the vector space. It is of great significance to also note that the number of features is a lot smaller than the length of the vocabulary [54], $\upsilon \in \mathbb{R}^d$, $d \ll |V|$.

The following embedding methods have been some of the most influential of the past few years and emotion detection from text has leveraged them, or their improvements, as input to various neural network architectures that will be presented in the next chapter.

3.3.1 Word2Vec

In this approach, each word is represented as a vector of 32 or more dimensions, instead of a single number. The embeddings are learned via a shallow neural network. The objective function of the Word2Vec neural net causes words with similar **context** to have similar embeddings. The length of the context or, to put it differently, the window of the surrounding words varies and can be set as a parameter of the Word2Vec models.

There are two different models regarding the context that the Word2Vec approach [55] takes into account in order to generate the vectors: the **Continuous Bag-Of-Word (CBOW)** and the **Skip-gram**. The CBOW learns the embeddings by predicting the current word, given the context while the Skip-gram learns by predicting the neighbouring words, given the current word.

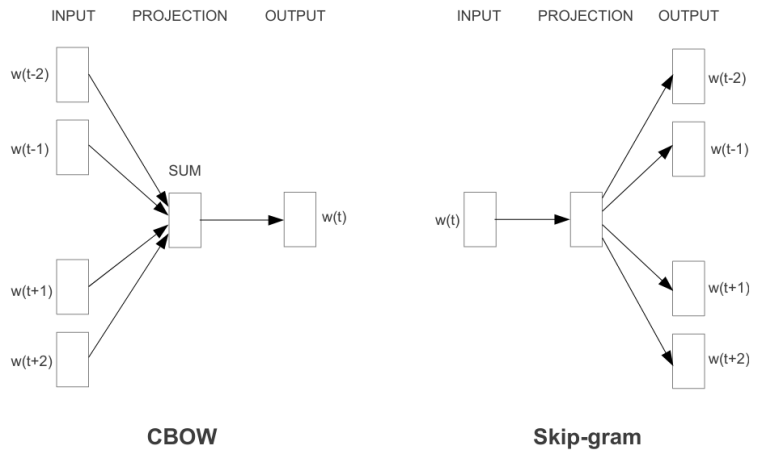

Fig. 3.5: The two Word2Vec training models [55].

Moreover, the researchers that presented this model also gave emphasis on the relations between the trained representations. More precisely, based on the arithmetic operations between learned vectors it was presented that various relationships between the words were characterized by an offset between their respective vectors. In other words, a linear

operation on the vectors translates to some meaning shift or semantic correlation between a start and destination, as shown in the very famous example:

$$\upsilon(king) - \upsilon(man) + \upsilon(woman) = \upsilon(queen)$$

The most substantial attributes for the Word2Vec methods are shown in Table 3.2 below. As we can see, where CBOW fails, Skip-gram shines and vice versa.

Table 3.2: The pros and cons of using each method to train word embeddings.

	(+) Advantages	(-) Disadvantages
CBOW	✓ Fast ✓ More precise for usual words ✓ Consumes less memory	x Requires a large amount of training data x Fails to represent rarer words with high precision
Skip-gram	✓ Efficient with small corpora for training ✓ Satisfying results even with rarer phrases	x Slow x Memory inefficient

3.3.2 GloVe

Matrix factorization embeddings, meaning methods that approximate corpus statistics on term-term or term-doc matrices, date back to the 1960's and are still very much relevant in NLP research today. However, the problems as noted by the authors of the GloVe paper [56], come from their inability to capture contextual data in the word's neighbourhood. Also, the fact they are simply taking the probability of word occurrences results in them failing to distinguish between any secondary meanings a word may have.

Another set of approaches, like the Word2Vec which was presented earlier, is based on the **shallow window,** where word representations are learned so that they are able to make predictions within a local contextual window.

In theory, they ameliorated the matrix factorization methods since they were able to learn language patterns and semantics as relationships between the learned vectors. But these schemes move the context window across the entire text collection, which means the recurrence of words and phrases are not utilized, hence these methods do not account for co-occurrence statistics.

GloVe aims to improve upon all of these by combining both, the global statistics as well as the shallow window approach. It achieves this by capturing the context of the word in the embedding through explicitly capturing these co-occurrence probabilities. This is empirically shown in the paper through Table 3.3 below:

Table 3.3 Example of the GloVe embeddings capturing the context of words [56].

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

The words on the left are compared to the exploratory words k. Interestingly, we can see the word ice is more strongly related towards $solid$ than it is for gas and the converse is true for $stream$, as seen by the ratio calculated at the bottom row.

All this argues the point that embeddings should be built not on just the word probabilities but the co-occurrence probabilities within the context.

At first, GloVe, based on the distance of the words and the vocabulary$V$, creates the co-occurrence matrix with dimensions $Vx\,V$. A window with set size, taken as the context, is moved across the corpus and for every word i, we wish to learn, we tally the presence of another word j. Instead of simply counting up the occurrences in the corpus as statistical measure or count within a set context, GloVe incorporates a notion of **contextual distance** between the words. This is achieved by weighting the X_{ij} according to the lexicographic distance between the words i and j , as shown in the example in Table 3.4.

Table 3.4 Example of the way contextual distances are calculated with GloVe.

i	j	x_{ij}
We	aren't	1
We	dealing	1/2
We	with	1/3
aren't	dealing	1

Fig. 3.6: This weighting mechanism ensures that words that are far apart do not count as much as words that are close to each other.

The co-occurrence matrix that is constructed through this process, contains values that represent how often a word occurs within a specific context (e.g. other word or phrases). The next step is to attempt to recreate this word-context matrix as the product between two other matrices that contain the values for the word-features and the features-context pairs.

For this factorization to take place, the two matrices are initialized and multiplied, producing an estimation of the co-occurrence matrix we wish for. Then this estimation is compared to the co-occurrence matrix and the values of the two matrices are updated accordingly. This process is repeated until the two matrices produce the desired result. Finally, the word-features matrix is the one that contains the trained embeddings.

CHAPTER 4

MACHINE LEARNING FOR EMOTION DETECTION

Machine learning is the use of methods that leverage the properties of the data to solve for the most optimal value of the parameters. More specifically, supervised learning algorithms usually take a training set, x as input and produce a hypothesis function, denoted by convention as$h$ which maps the input x to some output y.

4.1 Naive Bayes Classifier

Naive Bayes (NB) is a set of algorithms based on Bayes' Rule. Usually, the Naive Bayes classifier the NLP researchers refer to is the **Multinomial Naive Bayes (MNB)**. Some other prominent types of this category of algorithms are the Gaussian and the Bernoulli Naive Bayes which are used in cases when our features are continuous and binary, respectively. For our problem of emotion detection in text and due to words in NLP problems being discrete features, the Multinomial Naive Bayes (MNB) is the appropriate version of the Naive Bayes classifier to implement.

MNB is one of the most straightforward classifiers and it is based on two simplifying assumptions. The first is it considers the order of the words in the text irrelevant and does not take it into account. The second assumption and the reason why it was given the 'naive' adjective is the assumption the features are mutually independent, which in many cases is far from the truth [57].

The emotion detection problem is the classification of a document, d, to the most fitting class, c, out of the set of all classes, C. In other words, what interests us is the calculation of the maximum a posteriori or most likely class, c.

$$c_{NB} = \underset{c \in C}{argmax}\ \boldsymbol{\mathcal{P}}(c \mid d)$$

Due to Bayes' Rule, this becomes:

$$c = \underset{c \in C}{argmax}\ \frac{\boldsymbol{\mathcal{P}}(d \mid c)\,\boldsymbol{\mathcal{P}}(c)}{\boldsymbol{\mathcal{P}}(d)} \propto \underset{c \in C}{argmax}\ \boldsymbol{\mathcal{P}}(d \mid c)\,\boldsymbol{\mathcal{P}}(c)$$

The probability of the document, $\boldsymbol{\mathcal{P}}(d)$, is a constant since it is identical for the calculations of all the classes, and thus, it is common practice to eliminate it from the denominator. In the end, the most likely class, c , is the one that maximizes the product of the likelihood, $\boldsymbol{\mathcal{P}}(d \mid c)$ and the prior $\boldsymbol{\mathcal{P}}(c)$.

As we saw earlier, a piece of text is represented as a vector of features $x_a, x_b, \ldots, x_n$:

$$c = \underset{c \in C}{argmax}\ \boldsymbol{\mathcal{P}}(x_1, x_2, \ldots, x_n \mid c)\,\boldsymbol{\mathcal{P}}(c)$$

The probability of the class, $\boldsymbol{\mathcal{P}}(c)$, is easy to compute and can be done just by counting the relative frequencies of the classes in the corpus. Due to the conditional independence assumption of Naive Bayes, the probabilities of the features given the class become:

$$\boldsymbol{\mathcal{P}}(x_1, x_2, \ldots, x_n \mid c) = \boldsymbol{\mathcal{P}}(x_1 \mid c) \cdot \boldsymbol{\mathcal{P}}(x_2 \mid c) \cdot \ldots \cdot \boldsymbol{\mathcal{P}}(x_n \mid c)$$

In other words, the most likely class using the Naive Bayes assumption is:

$$c_{NB} = \underset{c \in C}{argmax}\ \boldsymbol{\mathcal{P}}(c_j) \prod_{x \in X} \boldsymbol{\mathcal{P}}(x \mid c)$$

In spite of the simplicity in the NB approach and due to its fast training times, it has repeatedly proven to be an effective starting point for emotion detection in text, even in its simplest form e.g. where it considers words to be not only independent but of equal weights too [58]. In [58], the Naive Bayes classifier was implemented on four differently preprocessed versions of the same dataset, which consisted of chinese song lyrics collected from the web and assigned emotions with the dimensional model. The results showed accuracy of up to 68% in the better cleaned dataset, which is not a bad outcome, bearing in mind the lack of complexity of the method.

Moreover, others [59] have presented even higher accuracy which is clearly attributed to the high quality of the training data. Many implementations of NB [60] highlight the importance of the extensive implementation of preprocessing techniques on the input data, something that had been previously overlooked, resulting in poor performance, especially for simpler models like NB.

Furthermore, some attempts to incorporate other feature representation methods, such as bi-grams [60], have taken place without noting particularly higher performance. One explanation for this is probably the fact NB methods require large training text collections and vocabulary, in order to achieve more accurate predictions [59], which results in sparse bi-gram vectors especially in cases when the samples are short, e.g. Tweets.

In conclusion, NB is an easy to implement probabilistic classifier that is fast but makes unrealistic assumptions about the data. This renders it less reliable than other algorithms for multiclass classification in text. Some solutions have been proposed to increase its accuracy. Most suggestions focus primarily on increasing the quality and balance of the emotion classification datasets.

4.2 Decision Trees

A decision tree is a tree-like arrangement of a selection of features to be compared, so that a prediction outcome occurs. As shown in the binary tree example of Fig. 4.1 below, this hierarchical structure has two types of nodes, the **internal decision nodes** and the **leaves** which are the predictions. The decision nodes examine the value of a given feature $X^{(i)}$ and whether it surpasses a certain threshold or split value, v .

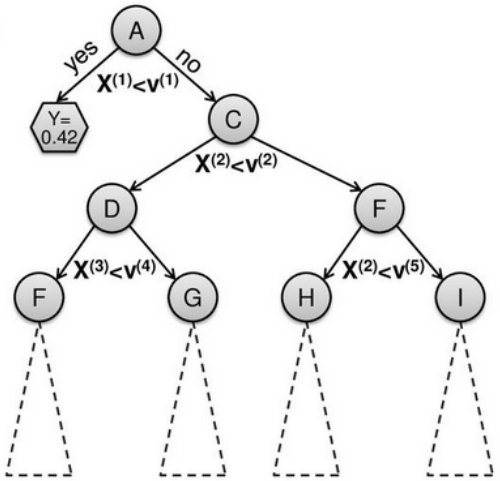

Fig. 4.1 Example of a binary decision tree taken from [61].

The part of making a prediction after having constructed the tree is straightforward. The input is compared to the various thresholds at each node, beginning from the root and then proceeding downwards, until a leaf and therefore a prediction, is reached [62]. So as to achieve this prediction, though, we have to determine the order in which the testing of the features will take place. In other words, we have to decide which question should be asked at each node. This is where the concept of **information gain** comes in.

4.2.1 Entropy

The best question to ask at every node is the one that reduces uncertainty the most. Entropy, H, is a measure of uncertainty or disorder. The higher it is, the more difficult it is to draw conclusions on the data. Information gain, IG, on the other hand, is a number that describes how much a question helps to unmix the labels at a node. It determines which features are most relevant so that they are tested at nodes closer to the root. We begin by calculating the uncertainty of a starting set of examples, using Shannon's information entropy equation:

$$H = -\sum_{i}^{C} p_i log\ p_i$$

where p_i is the probability of randomly drawing an example of class i. As we notice, if $p$is close to zero, such as in the case when the class i contains only a few training examples, the $logp$ becomes a large negative number. Due to p being close to zero, the entropy is also close to zero. In a similar manner, when most of the examples fall into one class, p is close to one and $logp$ is close to zero. Thus, we see the entropy is very low when the uncertainty, regarding which class a sample is most likely to belong in, is high.

For each question we ask, we partition the data accordingly and calculate the uncertainty of the child nodes that result. Then, we subtract the weighted average uncertainty, $H(Y|X)$, of the feature X we wish to split on, from the starting uncertainty, $H(Y)$ and the result is the information gain:

$$IG(Y, X) = H(Y) - H(Y|X)$$

In other words, the higher the $IG(Y, X)$ is, the more entropy is removed. As we proceed, we keep track of the feature that produces the most information. The attribute whose split reduces uncertainty the most, is set at the root. We continue growing the tree in a similar

manner, testing the next most important feature and choosing a value to split on at every node, except the leaves [63]. This process arrives at an end when either a specific depth has been reached or when no more splits can take place. Our tree is ready to make predictions then.

4.2.2 Gini Impurity

An alternative splitting criterion to use instead of entropy is Gini impurity. It is the measure of how often a sentence, randomly drawn from the dataset, would be assigned an incorrect emotion label according to the distribution of the labels in the dataset. Gini impurity is given by the following equation:

$$GI = 1 - \sum_{i=1}^{C} p_i^2$$

where p_i is the probability of class i and C the number of classes. Due to the square within the sum, it is computationally less costly than the more complex logarithm of entropy and we can expect it to be faster.

The minimum value GI can assume is 0. In this case the node is called pure. This means that all the samples of the node belong to a single class. Then our splitting criterion is optimal and the node will not proceed to split again.

Decision trees are appropriate for multiclass prediction problems such as ours. Moreover, they do not require extensive pre-processing of the data. Their performance is not affected by missing values and making predictions is fast and easy after the tree has been constructed. Furthermore, they are flexible since they are used for the classification of numeric, as well as categorical data without making any assumptions regarding the linearity of the data.

At the same time and despite being fast at making predictions, decision trees have been shown to be expensive [64], in terms of time and space to train. Due to their flexibility they are also prone to overfitting the data, resulting in lower accuracy performance when compared to other methods such as Naive Bayes and SVMs , as demonstrated in [64].

4.3 Logistic Regression

The name logistic regression is misleading as this statistical method is not used to model regression problems but classification ones. Logistic regression is inherently used on binary problems [65].

The hypothesis function used in linear regression is the weighted sum of the input features:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_n x_n$$

where n is the number of features and θ is the $n + 1$ dimensional vector of parameters. The outputs of a linear model such as this cannot be interpreted as probabilities. The purpose of linear regression is instead, to look for the hyperplane that minimizes the distance between the hyperplane and the features x_i, $i = 0,\ldots,n$.

This approach cannot generalize for multiclass problems. Also, these outputs can be greater than one and lower than zero, which results in obscurity regarding their interpretations. Logistic regression amends these issues by using the sigmoid or logistic function which is shown graphically in Fig. 4.2 and is given by the following equation:

$$g(z) = \frac{1}{1+e^{-z}}$$

The logistic regression hypothesis function is:

$$h(x) = \frac{1}{1+e^{-\theta^T x}}$$

Moreover, $h(x)$ is interpreted as the predicted probability that $y = 1$, given a particular feature x , parametrize by the weight θ: $h(x) = P(y = 1|x_i \theta)$.

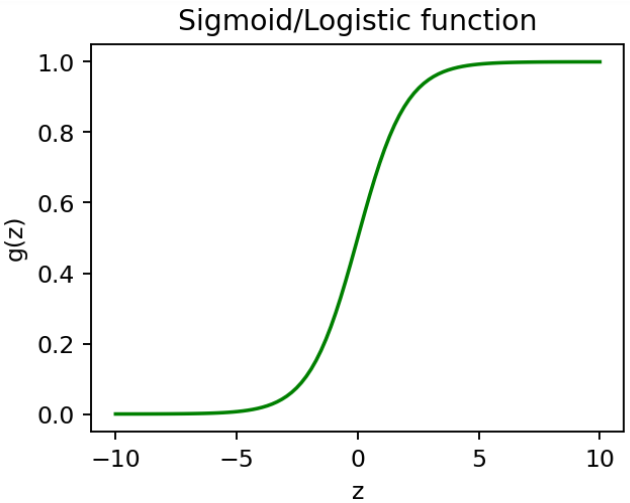

Fig. 4.2 The sigmoid function maps real values into $[0, 1]$. It is almost linear around 0. We observe that the logistic hypothesis $y = g(z) \geq 0.5$ whenever $z = \theta^T x \geq 0$

As stated earlier, the purpose of the logistic regression algorithm is to identify the parameters of the model, θ^T that minimize a proposed definition of error. The loss function this approach minimizes is:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)} \cdot log(h(x^{(i)})) - (1 - y^{(i)}) \cdot log(1 - h(x^{(i)}))]$$

where m is the number of examples. It is easy to understand the way this equation works. When the true label of the i^{th} example, $y^{(i)} = 1$, the second term is close to null. The remaining $-log(h(x))$ tends to zero as our prediction $h(x)$is also close to one. Similarly, in the case when the target label $y = 0$, what remains in the cost function of the i^{th}sample, is only the second term, $-log(1 - h(x^{(i)}))$. As shown in Fig. 4.3 below, the cost function is minimized as our prediction, $h(x)$also tends to zero.

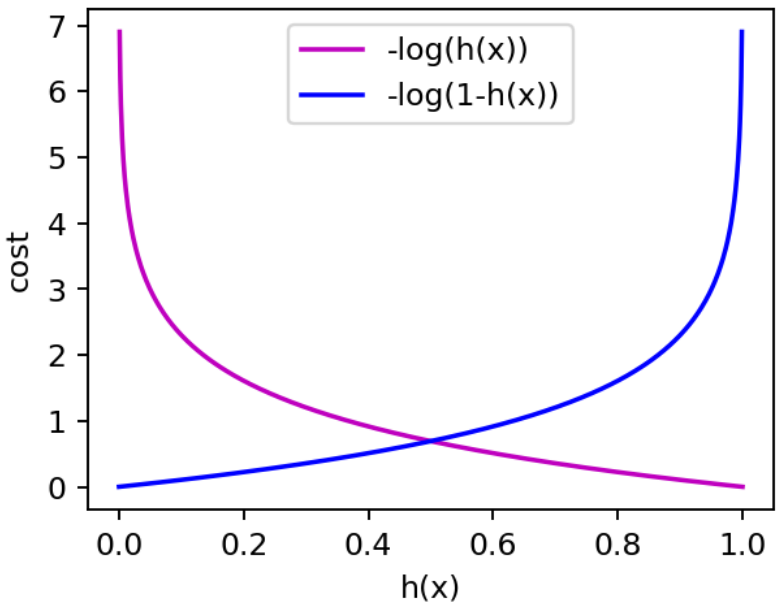

Fig. 4.3 Graph of the two terms of the cost function.

All that remains now is to update the values of the weights θin a way that minimizes the loss function $J(\theta)$. This is achieved with the use of the **Gradient Descent** algorithm which repeatedly updates each parameter by subtracting the partial derivative of the loss function with respect to θ while also taking into account the hyperparameter of the learning rate α.

$$\theta = \theta - \alpha \cdot \frac{\partial}{\partial \theta} J(\theta)$$

$$= \theta - \alpha \cdot \sum_{i=1}^{m} [(h(x^{(i)}) - y^{(i)})x^{(i)}]$$

This update takes place again and again until convergence is reached, meaning the global minimum is found. The parameters that result in this global minimum are put into the hypothesis function $h(x)$and the result is the function that best fits our data. After the logistic regression classifier has been trained we can insert new inputs, x's to be classified.

4.3.1 Regularization

The number of features is very large in NLP problems. The large availability of features can lead to the issue of overfitting when running logistic regression. Overfitting happens in the case when the model is overspecialized to the training set, performing great with $J(\theta) \approx 0$ but cannot generalize effectively to new examples. Usually when overfitting happens we have less or as many rows in our dataset as we have columns.

This problem can be addressed through a process called **regularization**. More elaborately, this adds a term to the loss function, providing a penalty for large coefficients, therefore assisting the model in removing unnecessary features. The tuning parameter, αdetermines how important is the $R(\theta)$function.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h(x), y) - \frac{\alpha}{m} \cdot R(\theta)$$

If the parameter αis selected to have too large a value, the weights θare severely penalized and end up being close to zero. The result is a hypothesis function $h(x) = \theta_0$, a straight line. In this case the model is **underfitting** our data.

The two common choices for the regularization function, $R(\theta)$ are the $L2$ and $L1$ methods. $L2$ applies a penalty as a quadratic function of the sum of the weights:

$$R(\theta) = ||\theta||_2^2 = \sum_{i=1}^{n} \theta_j^2$$

while $L1$applies a linear penalty as a function of the weights:

$$R(\theta) = ||\theta||_2 = \sum_{i=1}^{n} |\theta_j|$$

The coefficients that have too large values, which is an indication of overfitting, end up being thrown out of the model and therefore both regularization methods act as a form of feature selection to some extent. In general, $L1$ is not as smooth as $L2$ and it proves harder to optimize. Because of this, $L2$ is a most common choice of regularization.

4.3.2 Multinomial

Multinomial Logistic Regression (MLR) is a variation of the basic, binary logistic regression algorithm that can classify inputs into more than two categories. The difference between the two is that MLR uses a generalization of the sigmoid function, called the softmax. The softmax compresses all the values within the range $[0, 1]$ and the elements add up to 1. This happens because of the first term which normalizes each element by summing over the C possible values of the labels . The softmax function returns a vector of probabilities, each element of which corresponds to one value for every available class.

Specifically, for an input sample x, the probabilities $\boldsymbol{\mathcal{P}}(y = c|x)$, $c = 1, 2, \ldots C$ are calculated, where c is each one of the different class labels it can be assigned. The hypothesis function in MLR becomes:

$$h(x) = \frac{1}{\sum_{j=1}^{C} e^{\theta_j^T x}} \cdot \begin{bmatrix} e^{\theta_1^T x} \\ e^{\theta_2^T x} \\ e^{\theta_3^T x} \\ \cdots \\ e^{\theta_C^T x} \end{bmatrix}$$

where $\theta_1, \theta_2, \ldots \theta_C \boldsymbol{\epsilon} \mathbb{R}^n$ are the coefficient vectors, methods like gradient descent attempt to find by minimizing the loss function.

Lastly, Logistic Regression is an efficient, easy to train algorithm that finds a linear decision boundary in binary problems but one that can also be extended for multi class classification. The regularization methods help this approach in avoiding overfitting the data. It is also shown to be able to handle vast amounts of data much better than other methods such as Naive Bayes [66].

4.4 Support Vector Machines

A non-probabilistic, binary classifier, Support Vector Machines (SVM) is a highly efficient algorithm in text categorization tasks. The Linear SVM's performance proves to be great, despite the challenge textual input poses due to its structure, especially the sparseness and high dimensionality of the input vectors in NLP classification problems such as emotion recognition [67].

This method creates a hyperplane in a multidimensional space in order to separate distinct classes. The purpose is to discover the maximal margin hyperplane that divides the data that belong to one category from the ones that belong to the other, in such a way that the distance between the decision boundary and the data of each class is maximized.

We wish for this distance to be maximum in order to reduce uncertainty regarding in which class a new point falls. There are multiple planes that divide the data but only one that maximizes the distance between the two classes. The higher this margin is, the lower the misclassification rate is for the SVM.

The data that are located nearest to the optimal hyperplane are called support vectors and they are of the highest significance in the process of determining the most fitting hyperplane for our dataset.

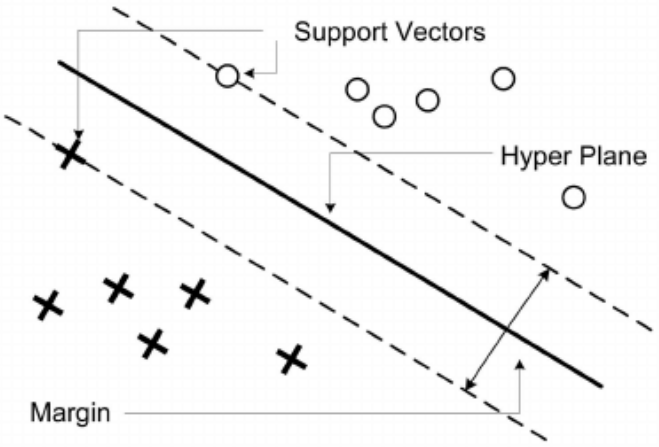

Fig. 4.4 Structure of a typical Linear Support Vector Machine [68]

The large margin problem is translated to the problem of finding the minimum parameters, θ that minimize the following objective function:

$$\underset{\theta}{argmin}\, C \cdot \sum_{i=1}^{m} [-\, y^{(i)} cost_1(\theta^T x) - (1 - y^{(i)}) \cdot cost_0(\theta^T x)] + \sum_{i=1}^{n} \theta_j^2$$

where the graphs of $cost_0$ and $cost_1$ are shown in Fig. 4.5 below. We notice this optimization objective is similar to that of logistic regression. The way the first part works is that when an example document belongs to the positive class, $y = 1$, the cost is nullified when the $\theta^T x \geq 1$, not just $\theta^T x > 0$. Similarly, and as shown in the graph of $cost_0(z)$ we wish for $\theta^T x \leq -1$, instead of just $\theta^T x < 0$as was the case of logistic regression.

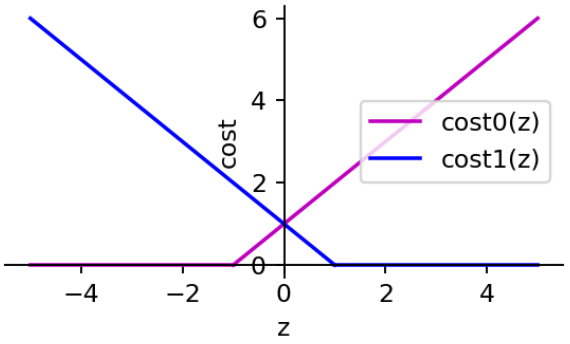

Fig. 4.5 Graphs of the SVM loss function cost terms.

While the purpose of the first sum of the optimization objective is to reduce the error, meaning the number of misclassified data points, the second sum maximizes the margin between the classes.

4.4.1 Soft Margin

When the SVM strictly does not allow for any points inside the margin it is called a **Hard margin classifier**. In this case, the magnitude of the margin is determined exclusively by the support vectors. Moreover, such classifiers perform well only on the rare occasions when the data provided is linearly separable, making the need for some flexibility imperative.

Usually, we wish for the SVM to allow some noisy data to not be taken into account. This happens with **Soft margin classifiers**. They are the SVM's that permit some data points to pass past the margin, making some training mistakes in order to have a larger margin which will, in turn, results in other points being classified correctly. The advantage of a decision boundary that provides a larger margin is that it enables the classifier to generalize better on information that was not included in the training set.

These margin changes are applied by adjusting the hyperparameter C in the SVM's objective function presented earlier. When this regularization constant C is small, significance is given to the second part of the function, which is responsible for maximizing the margin. On the other hand, when giving a large value to$C$, the focus shifts to avoiding misclassification which comes at the cost of having a smaller margin.

The Soft Margin classifier reduces the chance of overfitting by ignoring some margin breaches of data that are inside the margin or that are mistakenly classified by the more flexible decision boundary as we see in the example graph B.

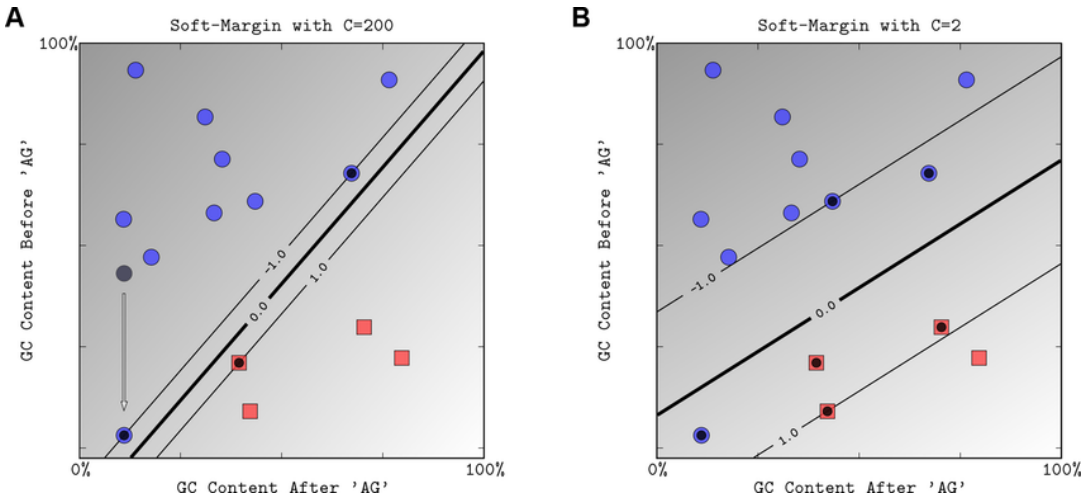

Fig 4.6 Examples of the way a change of the hyperparameter$C$alters the decision boundary of the Support Vector Machines taken from [69] . In (A) the very large value of Cresults in a decision boundary that resembles a Hard Margin classifier. In the more flexible (B) we notice a misclassification and a few tresspases of data into the margin.

In conclusion, SVM's are frequently used for text classification purposes because they work well in high dimensional spaces, finding efficient linear decision boundaries. Moreover, they are memory efficient as they use primarily just a part of the training data in order to separate the classes. Also, through the adjustment of C they allow control over their sensitivity to outliers.

4.5 One-vs-All and One-vs-One

Both Logistic Regression and Support Vector Machines do not inherently support the classification of data into multiple classes. There are two main approaches when it comes

to applying binary classification algorithms to multiple category problems such as emotion detection [57].

The **One-vs-All**, also called **One-vs-Rest**, method divides the multi-class problem into several smaller, binary classification ones. A problem with C different classes is turned into C separate two-class classification tasks. In essence, a new dataset is created where a target category is assigned the positive label, 1 and the rest classes are assigned the negative, 0. For every class of the dataset, i, one hypothesis function $h^{(i)}(x)$ is needed to answer the question of whether a sample belongs to the specific class or whether it belongs to one of the rest. In other words, each classifier we fit estimates what is the probability that y is equal to class i, given x parametrized byθ: $h^{(i)}(x) = \boldsymbol{P}(y = i|x; \theta)$.

Finally, in order to make a prediction for a new input x, we run all the C classifiers on the input and pick whichever one of them is most confident, meaning we select the class i that maximizes the probability$h^{(i)}(x)$.

The **One-vs-One** approach also is a means of utilizing algorithms for binary classification in multiple category problems. Contrary to the one-vs-all approach, for C different labels, according to this method, $\frac{C\cdot(C-1)}{2}$ classifiers have to be constructed. Specifically, one hypothesis function is created for every pair of classes. The training of each classifier takes place on a subset of the original dataset that contains only the two labels.

In the end, given an input x, all the classifiers will make a prediction, returning a label instead of probability scores. The candidate class that was predicted the highest number of times is the final proposed answer of this approach.

Table 4.1 Advantages and disadvantages of each supervised ML algorithm.

	Pros (✓)	Cons (x)
Decision Trees	✓ Not affected by missing values ✓ No heavy data preparation required ✓ Fast predictions	X Slow to train X Oversensitive: changes in data can alter its structure dramatically. X Prone to overfitting
Naive Bayes	✓ Fast ✓ Inherent support for multiclass prediction ✓ Handles large number of features well	X Independence assumption is unrealistic X Unreliable results
Logistic Regression	✓ Simple to implement ✓ Efficient when features are linearly separable ✓ Regularization offers fine-tuning flexibility	X Prone to overfitting when dealing with high dimensional data X Requires many data X Not as powerful as other algorithms
Linear Support Vector	✓ Very efficient when features are linearly separable ✓ Works better in high dimensional spaces ✓ Memory efficient	X Unfit for large datasets X Preprocessing to extract features from data is required X Not easily interpretable results as they are not probabilities

CHAPTER 5

EXPERIMENT METHODOLOGY

5.1 Tools

The experiments conducted in the context of this thesis were written in Python. Specifically, the organizing of the data was executed with the use of the open-source **Pandas** (Python Data Analysis Library) [70]. It was the best option for our problem due to the flexibility the DataFrame object of the library provides to the structuring and handling of big data. Moreover, it proved to be a useful library for easily reading from and writing to the text-emotion files. Its wide set of features includes but is not limited to operations for accessing, printing, deleting and grouping samples as well as concatenating datasets.

For the cleaning and the preparation of the text, the Python **re** (Regular Expressions) [71] module was used in conjunction with **NLTK** (Natural Language Toolkit) [72]. To be more precise, the former was used to find and remove substrings that contained noise and redundant characters within each data sample and remove or replace them and the latter provided the tokenizer and stemming operations performed on our clean data. Furthermore, the NLTK also includes some packages that were used for the removal of stopwords, and also for the tokenization and the stemming of the clean data.

The supervised machine learning algorithms applied in the experiments were the ones implemented in the **Scikit-learn** (a.k.a. Sklearn) [73] open-source framework which is built upon other Python packages such as SciPy and Numpy, the majority of which is

written in C for optimal performance. Scikit-learn also contains a set of packages that was used to transform the text data into vectors via the TF-IDF and Bag-of-Word methods while also supporting the n-gram approach. In addition to this, the calculation of the performance according to the metrics that were chosen and analyzed later is also provided by this framework.

Lastly, the **plotly** and **Matplotlib** plotting graphs libraries were used for data visualization and exploration purposes. More elaborately, the first was employed to demonstrate the emotion distributions in the dataset, the word count distribution in the samples as well as the top unigrams and bigrams before and after cleaning the data. On the other hand, Matplotlib was involved in the building of the normalized confusion matrix for the applied algorithms to visualize better how well they performed on the test data.

5.2 Data preparation

The datasets we picked out of Table 3.1 were the ISEAR, DailyDialog and emotion-stimulus ones. This decision was made due to their lack of noise. More precisely, the first two datasets were created manually by humans so the expectation behind their selection was the need for minimal cleaning and preprocessing. The third dataset is the smallest as shown in the Table 5.1 below and was chosen to be complementary, providing some extra training and testing examples for the last experiment of the final, concatenated dataset.

Table 5.1 Characteristics of the three datasets used in experiments.

	No. Samples	Year	Balanced
emotion-stimulus	1,594	2015	No
ISEAR	7,666	1990	Yes
DailyDialog	102,968	2017	No

Moreover, these three datasets were also selected due to their homogeneity. They all contain the same kind of data, that is dialogues and emotion statements and so it was sensible to group them together, instead of also including other aforementioned datasets that consisted of e.g. Tweets or Facebook posts.

5.2.1 ISEAR

Each dataset required a different approach regarding the extraction of the samples from the original files and the cleaning of the data. More specifically, after turning the original Microsoft Access Database file, that contained the ISEAR dataset, into a comma separated values file (csv), we read it using Pandas and print the first few samples, indicative of its structure. The result follows in Fig. 5.1.

```
print(data.head())
```

```
      ID  CITY  COUN  SUBJ  SEX  AGE  RELI  PRAC  FOCC  MOCC  ...  SELF  RELA
0  11001     1     1     1    1   33     1     2     6     1  ...     3     3
1  11001     1     1     1    1   33     1     2     6     1  ...     2     2
2  11001     1     1     1    1   33     1     2     6     1  ...     2     1
3  11001     1     1     1    1   33     1     2     6     1  ...     1     1
4  11001     1     1     1    1   33     1     2     6     1  ...     0     2

   VERBAL  NEUTRO   Field1  Field3  Field2   MYKEY  \
0       2       0      joy       4       3  110011
1       0       0     fear       3       2  110012
2       0       0    anger       1       3  110013
3       0       2  sadness       4       4  110014
4       0       0  disgust       4       4  110015

                                                 SIT  STATE
0  During the period of falling in love, each tim...      1
1         When I was involved in a traffic accident.      1
2  When I was driving home after  several days of...      1
3  When I lost the person who meant the most to me.       1
4  The time I knocked a deer down - the sight of ...      1

[5 rows x 42 columns]
```

Fig. 5.1 The first five samples of the dataset ISEAR in its original form.

It is obvious that only two out of the 42 columns of the survey are needed for the purposes of our experiment. After extracting the elements of columns SIT and Field1 into a new DataFrame object within the fields Text and Emotion respectively, we wrote a simple function, shown in Fig. 5.2 that removes redundant whitespaces, useless symbols and punctuation marks and expands contractions.

The final form of the DataFrame is shown in Fig. 5.3, while Fig. 5.4 depicts that the samples are almost equally distributed across the emotion classes. After removing the “no response” entries of the survey, each category contains 1057 to 1088 examples.

```python
def cleaning(data, contractions, spec_chars, apostrophes):
    """
    Takes as inputs the DataFrame object, a list of apostrophes,
    special characters and a dictionary and it cleans the rows
    of the Text column and expands the contractions.
    """
    # replace various apostrophes with a universal, single one
    for apostrophe in apostrophes:
        data['Text'] = data['Text'].str.replace(apostrophe, "'")
    # expand contractions according to the input dictionary
    for i,utterance in enumerate(data['Text']):
        for word in utterance.split():
            if word.lower() in contractions:
                data.loc[i, 'Text'] = data.loc[i, 'Text'].replace(word, contractions[word.lower()])
    # remove useless punctuation marks as given in list spec_chars
    for char in spec_chars:
        data['Text'] = data['Text'].str.replace(char, " ")
    # remove whitespaces by splitting and rejoining
    data['Text'] = data['Text'].str.split().str.join(" ")
    # replace caps for coherency
    data['Text'] = data['Text'].str.lower()
```

Fig. 5.2 A short cleaning function used on all three datasets.

	Text	Emotion
0	during the period of falling in love each time that we met and especially when we had not met for a long time	joy
1	when i was involved in a traffic accident	fear
2	when i was driving home after several days of hard work there was a motorist ahead of me who was driving at 50 km hour and refused despite his low speeed to let me overtake	anger
3	when i lost the person who meant the most to me	sadness
4	the time i knocked a deer down the sight of the animal's injuries and helplessness the realization that the animal was so badly hurt that it had to be put down and when the animal screamed at the moment of death	disgust

Fig. 5.3 Final version of the ISEAR data.

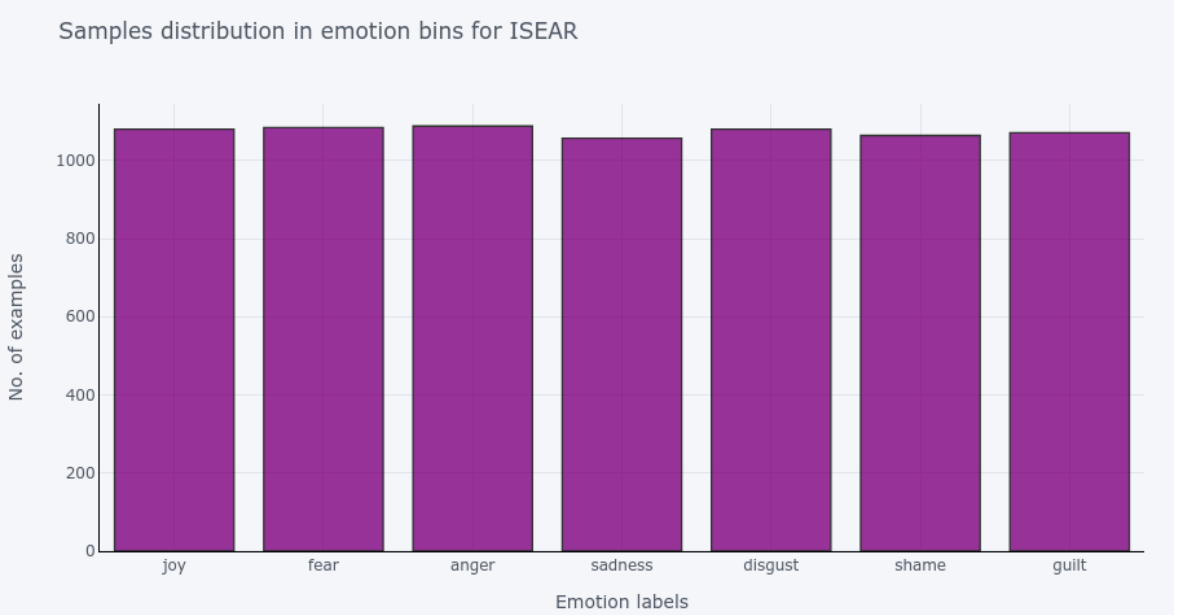

Fig. 5.4 The number of examples for every emotion label is approximately the same for all categories.

5.2.2 DailyDialog

This dataset had a completely different structure. It consisted of two text files, one that contained numerical values that corresponded to the seven emotion categories and another that contained the dialogues the aforementioned labels were assigned to. More elaborately, each line of the latter file contained a dialogue, composed by a number of utterances and separated by the characters "__eou__" that indicated the end of each utterance. At the same time, each line of the labels file contained the same number of numerical values as the number of utterances in the respective line of the dialogues file.

Originally, there were 13,117 dialogues. After their splitting there were 102,968 utterances in total. Just like before, we transferred the data into a DataFrame object. We also replaced the numerical values with the respective emotion textual label so that the labels are coherent across the three datasets we are using. We also removed redundant characters in the utterances texts by using the cleaning function.

```
def extract_data_in_df(PATH, df, type_of_info='utterances'):
    with open(PATH) as text:
            num_df_rows = 0
            data = []
            for line in text:
                # get utterances/labels from each line
                if type_of_info == 'utterances':
                    info = line.split("__eou__")
                elif type_of_info == 'emotions':
                    info = line.split(" ")
                info.remove('\n')
                # insert each utterance/label into list
                for element in info:
                    data.append(element)
                num_df_rows += len(info)

            print(f'Num of {type_of_info}: {num_df_rows}')

    # insert into DataFrame object
    if type_of_info == 'utterances':
        for i in range(len(data)):
            df.loc[i, 'Text'] = data[i]
    elif type_of_info == 'emotions':
        df = df.assign(Emotion=data)
        # turn emotion labels from numerical to text categories
        encodings = {
        '0': 'neutral',
        '1': 'anger',
        '2': 'disgust',
        '3': 'fear',
        '4': 'joy',
        '5': 'sadness',
        '6': 'surprise'}
        df["Emotion"].replace(encodings, inplace=True)
    return df
```

Fig. 5.5 The function used once on the dialogues text and once on the emotions file in order to extract the dataset and insert it into the DataFrame.

After extracting it, the text data is again in a crude form, having many redundant white spaces and special characters, as well as contractions that have to be expanded as shown in Fig. 5.6.

	Text	Emotion
0	The kitchen stinks .	disgust
1	I'll throw out the garbage .	neutral
2	So Dick , how about getting some coffee for tonight ?	joy
3	Coffee ? I don ' t honestly like that kind of stuff .	disgust
4	Come on , you can at least try a little , besides your cigarette .	neutral

Fig. 5.6 The result of df.head() before getting rid of the redundant whitespaces and special.

	Text	Emotion
0	the kitchen stinks	disgust
1	i will throw out the garbage	neutral
2	so dick how about getting some coffee for tonight	joy
3	coffee i do not honestly like that kind of stuff	disgust
4	come on you can at least try a little besides your cigarette	neutral

Fig. 5.7 The same data after having applied the cleaning function.

Due to the nature of this dataset, meaning it consists of everyday dialogues, we expect a lot of duplicate utterances, as parts of the different conversations are likely to be overlapping. Indeed, there are 21,006 duplicate entries, as shown in the screenshot of Fig. 5.8.

After dropping the duplicate values, 81,962 utterances remain. The vast majority of these, as shown in Fig. 5.9, are labeled as neutral and will be dropped when concatenating the datasets in order to build the two larger, balanced and unbalanced ones so that this class does not overshadow the rest in the training phase.

```
len(df[df.duplicated()])
```

```
21006
```

```
df[df.duplicated()].head()
```

	Text	Emotion
136	sure	neutral
205	that is fine	neutral
209	thanks	joy
243	what is it like	neutral
255	yes	neutral

Fig. 5.8 The number of duplicate utterances, as well as the first five elements that appear multiple times in the dataset as new samples.

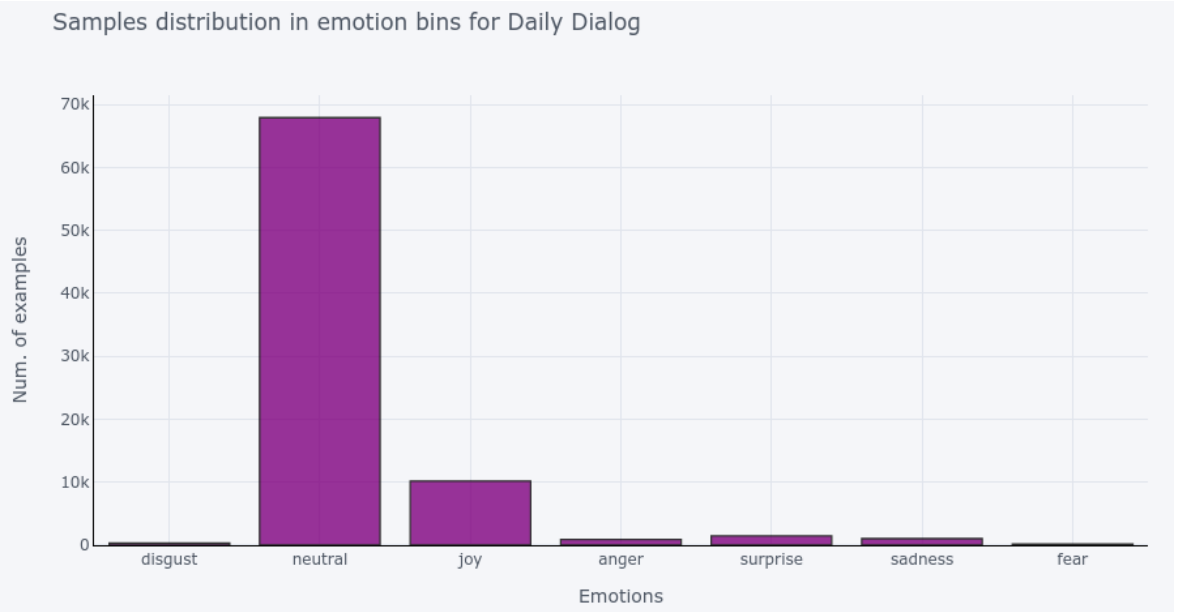

Fig. 5.9 This visualization makes the wide discrepancy of representation for the classes that are not ‘joy’ and ‘neutral’ very obvious.

5.2.3 Emotion-Stimulus

The smallest of our three datasets, Emotion-Stimulus came as a text file, each line of which contained both, the phrase as well as its respective emotion label in the form of tags, as shown below in Fig. 5.10:

	Original file
0	<happy>This did the trick : the boys now have a more distant friendship and David is much happier . <\happy>\n
1	<happy>When Anna left Inspector Aziz , she was much happier . <\happy>\n
2	<happy>And though , as Lachlan had planned , they had expected the attack in the morning , they were quite happy when their lookouts reported the Macleans ' approach in the early evening . <\happy>\n
3	<happy>Honestly , I 'm really happy for you ! <\happy>\n
4	<happy>Lesley was totally happy about it . <\happy>\n

Fig. 5.10 The first five lines of Emotion-Stimulus dataset in its crude form. It contains HTML-like tags that contain the emotion label.

After using the code in Fig. 5.11 in order to extract, clean and adjust the labels so that they are coherent with those of the previous two datasets, we get the result shown in Fig. 5.12. Moreover, as stated previously and as is obvious from Fig 5.13, this dataset is also unbalanced and will be used to supplement the other two in our experiment. The number of samples it contains range from only 57, for the emotion of *disgust*, to 468 for *sadness*.

```
df = pd.DataFrame(columns=['Text', 'Emotion'])
with open(PATH) as f:
    for i,line in enumerate(f):
        # find emotion tag and add it to the respective DataFrame column
        emotion = re.search('<(.+?)>', line).group(1)
        df.loc[i, 'Emotion'] = emotion
        # remove emotion tags from text and add text to the DataFrame
        line = re.sub('<.*?>', '', line)
        line = re.sub('<\.*?>', '', line)
        df.loc[i, 'Text'] = line
# change labels for coherency purposes w/ other dataset labels
df['Emotion'] = df['Emotion'].replace({'happy': 'joy', 'sad':'sadness'})
cleaning(df, contractions, spec_chars, apostrophes)
```

Fig. 5.11 When reading each line from the crude file we use *re.search* and *re.sub* to locate, extract the emotion and remove the tags that contained it from the text.

	Text	Emotion
0	this did the trick the boys now have a more distant friendship and david is much happier	joy
1	when anna left inspector aziz she was much happier	joy
2	and though as lachlan had planned they had expected the attack in the morning they were quite happy when their lookouts reported the macleans approach in the early evening	joy
3	honestly i am really happy for you	joy
4	lesley was totally happy about it	joy

Fig. 5.12 Final form of the structured, cleaned dataset.

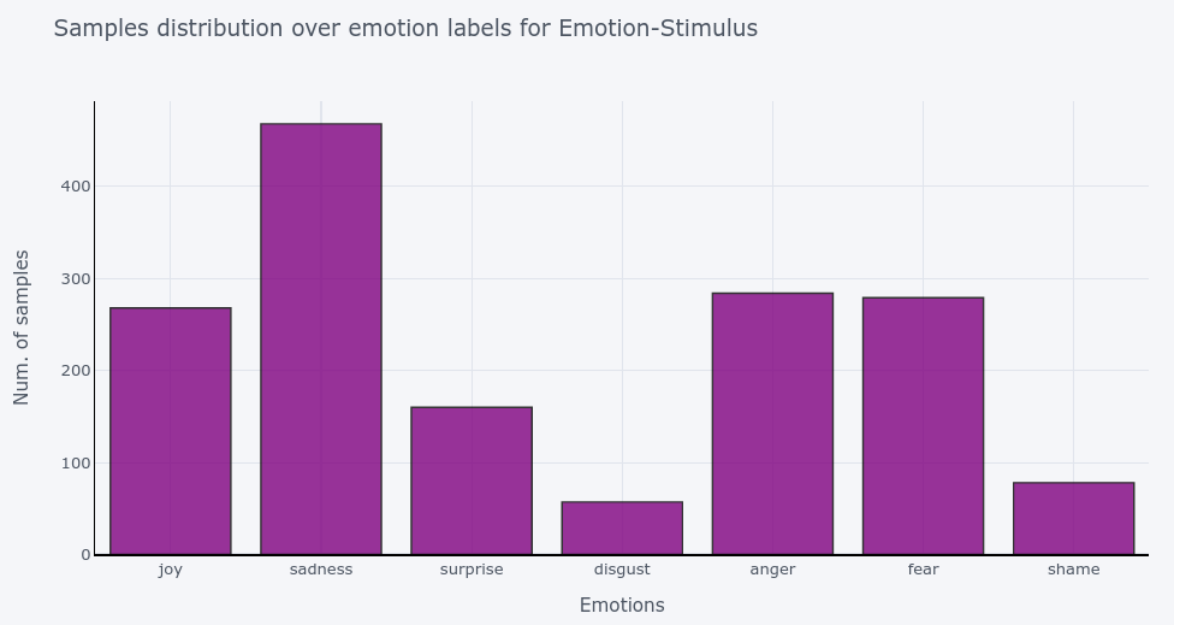

Fig 5.13 We see some classes, such as shame, are underrepresented in this dataset.

5.2.4 Combined datasets

There are two versions of the combination of these datasets we used for the experiments. The first includes examples from all the emotion categories. More specifically, the previously preprocessed data were concatenated exactly as were, with the exception of the emotions of *joy* and *neutral* that originally contained 11,533 and 67,916 examples respectively. Instead, we randomly selected 3,000 from each of these two labels to be included in the unbalanced combined dataset.

We needed our first dataset to be the largest possible for the purposes of the experiment and comparison of the performance of the algorithms, but not so unbalanced that it ended up being trained on only the two aforementioned labels and performing extremely poorly on the rest. The exact number of samples is shown in Table 5.2 while the amount of samples per label is, just like before, demonstrated comparatively in the graph of Fig. 5.14 below.

As is made apparent by Fig 5.14, even though we decreased the number of samples for the *joy* and *neutral* labels, there still is a lot of variation to the amount of examples across the different emotions. Due to this, we expected this dataset to perform worse compared to its balanced version.

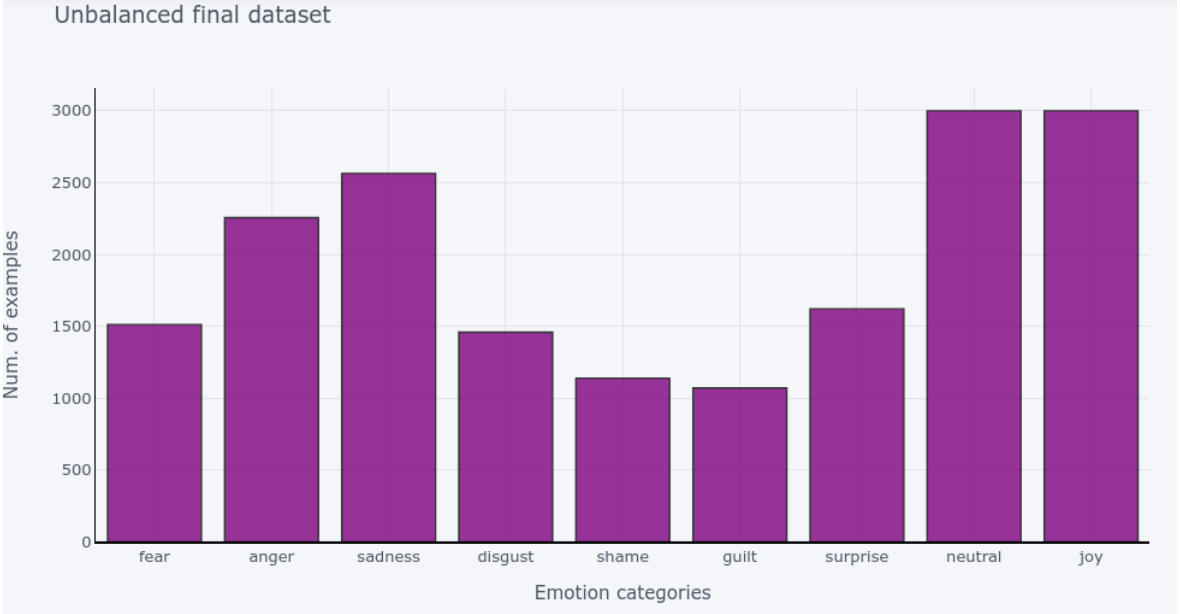

Fig. 5.14 The samples of our final, unbalanced dataset are assigned to nine categories.

Table 5.2 Number of samples per emotion of the combined unbalanced dataset.

Emotion	Num. of samples
joy	3,000
neutral	3,000
sadness	2,564
anger	2,257
surprise	1,622
fear	1,513
disgust	1,460
shame	1,140
guilt	1,071

The second dataset we used for the experiments contained samples almost equally distributed across four emotions. Eventually, we selected the emotions that we had managed to collect the greatest number of samples of. Due to the smaller number of emotion classes, as well as the good amount of samples corresponding to each of the categories, we expect this final, balanced dataset to perform better overall than the previous one.

Again, the following visualizations give us a good idea regarding the features of this dataset. It is apparent from the Fig. 5.15 below that this smaller dataset we created is much more balanced, specifically, as shown in Table 5.3, each category has 1488 to 1600 samples that belong to it.

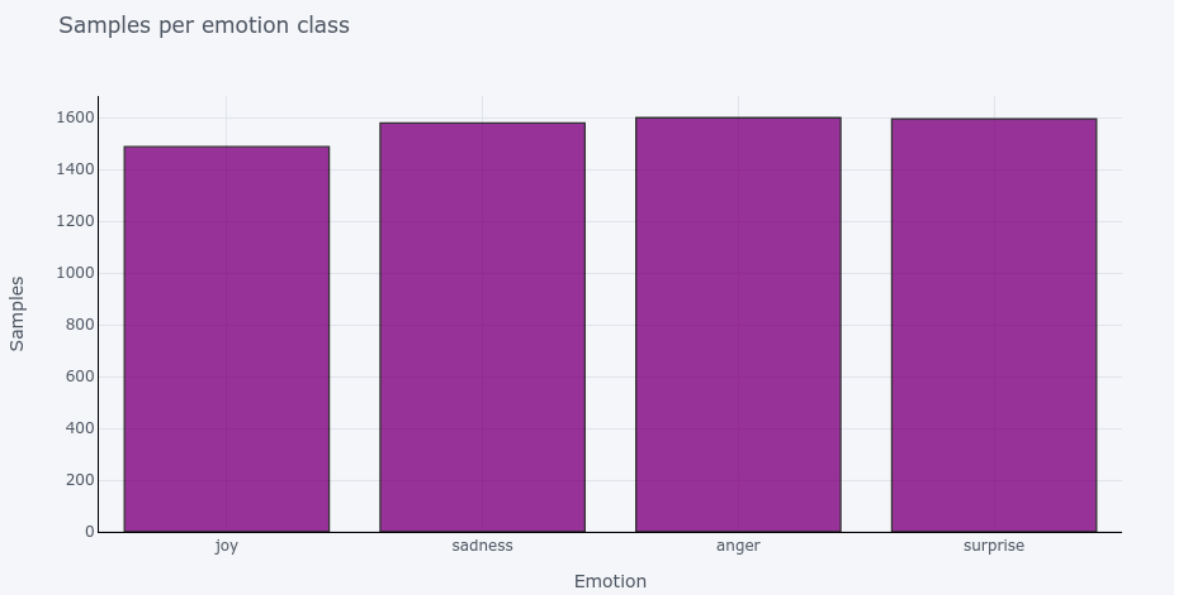

Fig. 5.15 The distribution of our smaller, balanced dataset over the emotion classes.

Table 5.3 Number of samples per emotion of the combined balanced dataset.

Emotion	Samples
anger	1600
surprise	1596
sadness	1580
joy	1488

What follows below are some other interesting features of this dataset that were collected during the visualization and data exploration phase before applying the algorithms. More elaborately, in Fig. 5.16 we see that the vast majority of our text samples contain under 20 words. In only very few cases there are samples that contain whole paragraphs with over 60 words. Keeping this in mind, removing such outliers from our data might prove to be a necessity in order to increase its quality and performance.

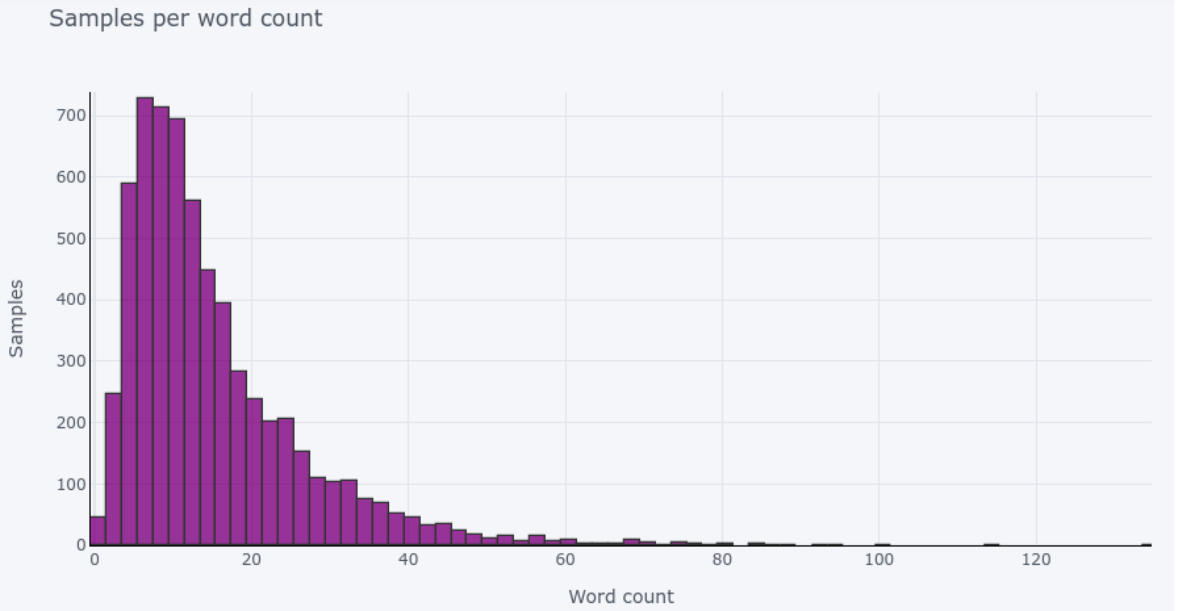

Fig. 5.16 Most samples consist of a small number of words, especially, 3 to 7 words. However, we see there are few exceptions to this with some samples containing more than 60 words.

In addition to this, Fig. 5.17 and Fig. 5.18 show which are the most common unigrams and bigrams in our dataset before deleting the usual english stopwords. As one would expect, personal pronouns, articles and conjunctions occupy the first places of the word frequency graphs. Interestingly, almost all the words that appear most often carry no useful, emotion-related information whatsoever. This is only slightly less the case for the bigrams of Fig. 5.18 where we see a little more information being contained in the bigram "*am sorry*".

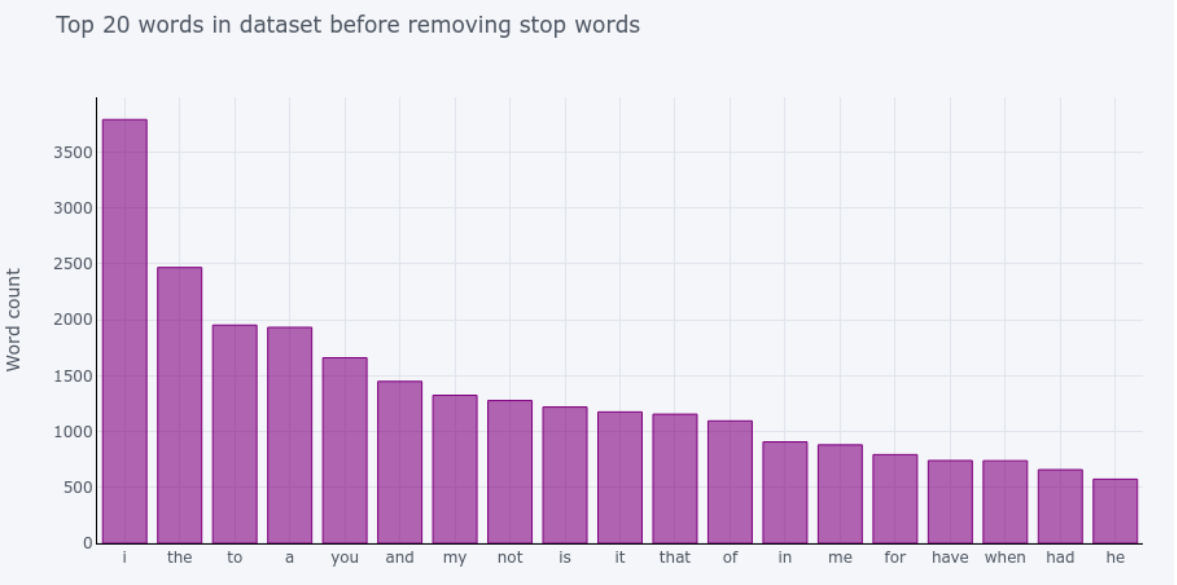

Fig. 5.17 Almost all the most frequent words in our dataset are stopwords that will be removed.

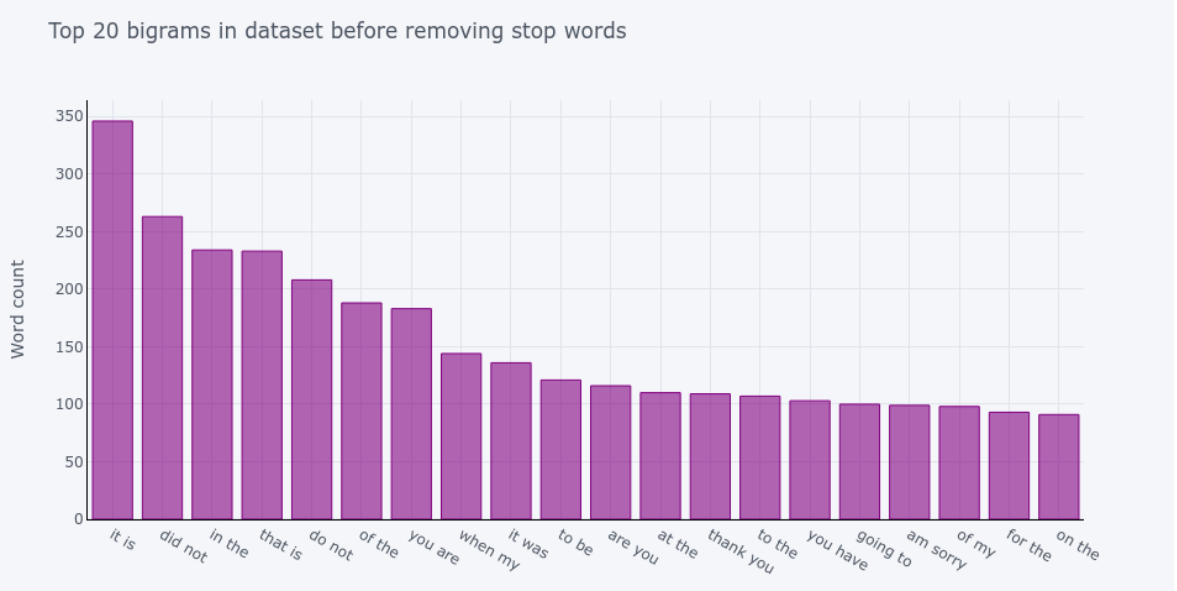

Fig. 5.18 The bigrams that appear most frequently in our dataset before deleting the stopwords. It is noteworthy that only the phrases "*thank you*" and "*am sorry*" contain valuable information.

On the other hand, after deleting the stopwords, even the unigrams of the dataset contain some value for our classification purposes. As we notice in Fig. 5.19 now some of the most frequent words are emotion related (e.g. *felt*) or even emotionally charged (e.g *angry*, *sad*)

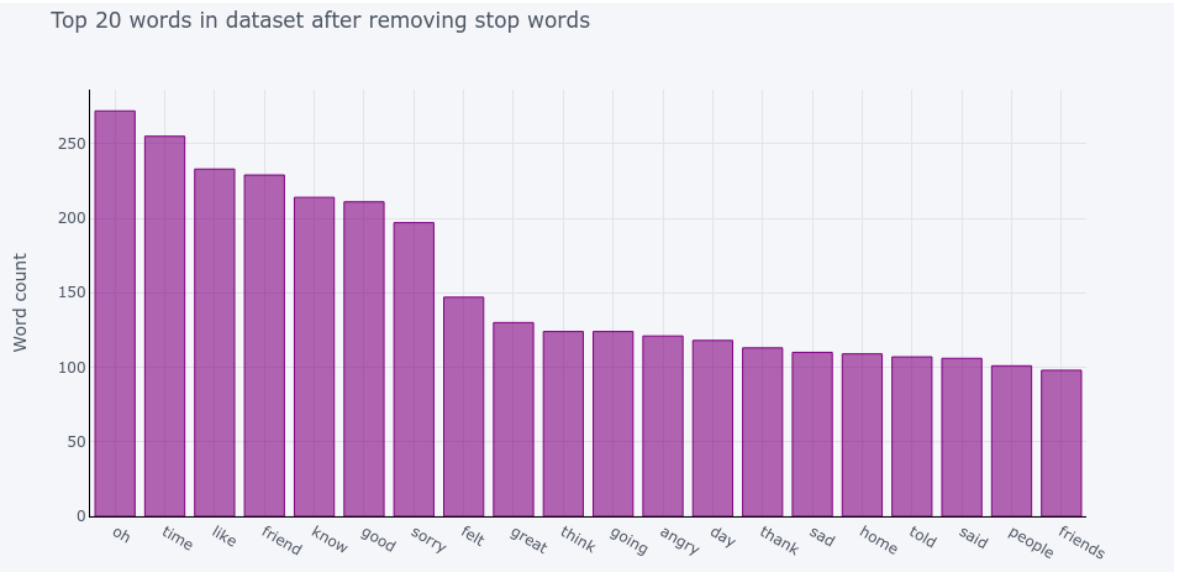

Fig. 5.19 The most frequent words now contain more information.

In a similar vein and unsurprisingly, as Fig. 5.20 shows, the most frequent bigrams after the stopword removal are the most meaningful elements we have come across so far. Every single one of these token sequences contains significant information, as well as strong emotions. Incidentally, it is worth noting that the exclamation "*oh*" that was the most frequent unigram after cleaning, has additional value in its bigram alternatives that helps discern further the underlying emotions.

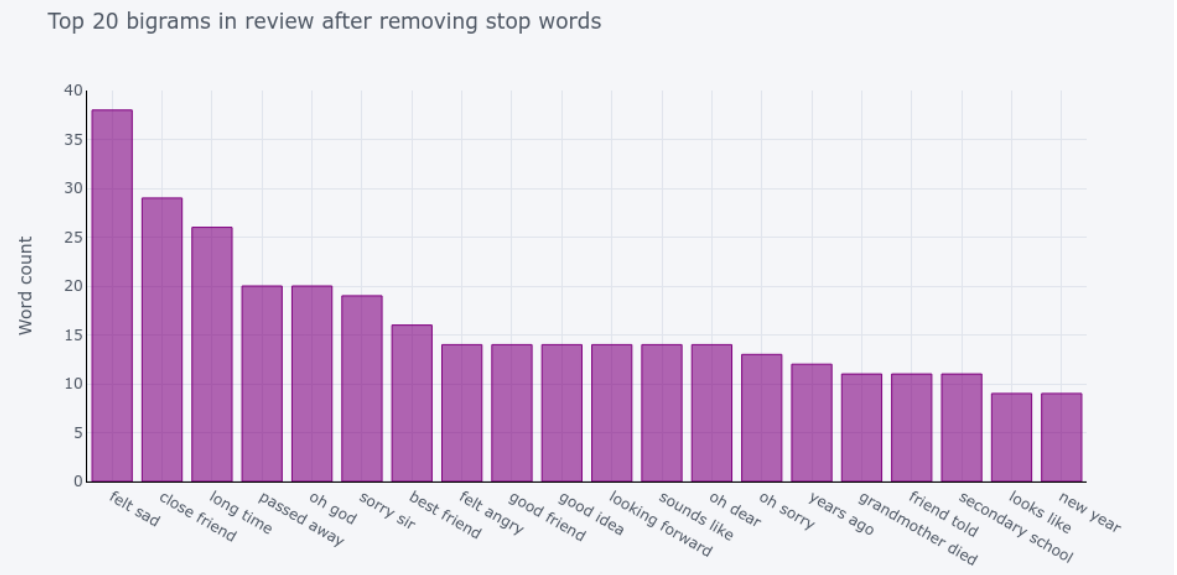

Fig. 5.20 The most frequent, clean bigrams carry apparent emotional information. Therefore, it is sensible that, as we see here, they appear less often than their stopword counterparts in the previous bigram graph.

5.3 Metrics

Accuracy is the total number of correctly classified samples divided by the total number of samples. It works well only on balanced data. When our datasets are unbalanced, the accuracy metric can be misleading when drawing conclusions on model performance.

A more appropriate metric in this case is the F_{β} **score**. It takes into account two other metrics: precision and recall. **Precision** is the accuracy of the positive predictions of our model or, to put it differently, this metric is the information of how many samples were actually positive out of the total number of positively predicted outcomes by our model. Thus, in our case of multi-class classification it is given across all classes by the following formula:

$$Precision = \frac{\sum_{c \epsilon C} TP_c}{\sum_{c \epsilon C} (TP_c + FP_c)}$$

where TP_c stands for True Positive and describes the instance of samples that belong to the class c, being correctly classified as belonging to class c by the model. Similarly, FP_c stands for False Positive and contains, for a given class c, the number of samples incorrectly assigned to this class.

Recall, on the other hand, shows us how good the classifier is at giving a positive prediction in the case that indeed our true label is positive. In other words, recall is the information of how many values were correctly predicted as positive by our classifier, out of the total actually positive values.

$$Recall = \frac{\sum_{c \epsilon C} TP_c}{\sum_{c \epsilon C} (TP_c + FN_c)}$$

where FN_c stands for False Negative and describes the number of samples belonging to class c that were incorrectly labeled as belonging to one of the other $C - 1$ classes by the classifier.

In problems where false negatives play a more important role, recall is the best metric, while in cases where false positives are more the focus of the problem, precision is the metric most fitting to use. In our multiclass problem of emotion detection, we want to consider both recall and precision equally so the $F1$ score, the harmonic mean of the two, will be used:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

$F1$is simply the specific case of F_β for $\beta = 1$.

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{\beta^2 \times (precision + recall)}$$

The weight βis adjustable according to the needs of the problem. When it holds the value of $\beta = 1$, recall and precision are both given equal importance. However, by reducing its

value, more emphasis is given to the precision aspect of the F_{β}. Likewise, increasing the value of the weight results in the recall term having a greater impact on our metric. The maximum value $F1$can take is 1 and it represents a model having optimal recall and precision while the minimum value is 0 [68].

5.4 Results

We splitted our two datasets into training and test sets by randomly drawing *75%* of the total data and saving it to a file called *training.csv* and also saving the rest *25%* to a *test.csv* file. Overall, the larger, unbalanced dataset performed noticeably worse than the balanced one. After running the different classifiers we observed that by removing the samples that contained over 60 words, the performance increased by about *0.5%* on both datasets so the results after this change are the ones presented below. The algorithm that returned the worst scores was in both cases the Decision Trees. On the other hand, the Linear Support Vectors outperformed Logistic Regression, as well as Naive Bayes in both data collections.

5.4.1 Decision Trees

More precisely, we tried the Decision Tree classifier on our datasets, using both the *Gini* and *Entropy* criteria. The smaller, balanced dataset performed steadily with an F1 score that was above *50*. The combination of Bag-of-Words and Gini impurity returned the optimal for this classifier, F1 score of *57.69%*. However, on the large dataset of 9 classes our classifier did not reach a higher F1 score than that of *39.36%*. This result was also acquired by the combination of BoW for vectorization and Gini impurity and it is still better than random guessing. In both cases, the most underperforming scenario was that of TF-IDF paired with entropy as shown in more detail in Table 5.4.

Table 5.4 Performance results of the Decision Trees classifier.

Decision Trees (%)	BoW+Gini	BoW+Entropy	TF-IDF+Gini	TF-IDF+Entropy
Small + Balanced	57.69	55.04	54.77	52.69
Large + Unbalanced	39.36	35.05	37.73	34.55

5.4.2 Naive Bayes

Taking into account the simplicity of this classifier, it performed particularly well on the balanced dataset, reaching an F1 score of *69.05%*. Moreover it also scored approximately *10* points higher than the Decision Trees classifier on the unbalanced one. Since Naive Bayes does not require complicated preprocessing and vectorization methods, we see that even the simpler, BoW representation works almost as well as TF-IDF in both cases.

Table 5.5 Performance results of the Naive Bayes classifier.

Naive Bayes (%)	BoW	TF-IDF
Small + Balanced	69.39	70.05
Large + Unbalanced	48.37	49.14

5.4.3 Logistic Regression

The way we tested Logistic Regression with the One-vs-All approach was by utilizing *sklearn*'s implemented classifier, *LogisticRegression()*, with its *multi_class* parameter set to *'ovr'*. In order to apply the One-vs-One method, we instead used the *OneVsOneClassifier()* with the *estimator* parameter set to *LogisticRegression()*. The top score was given by BoW and OvA and the worst by TF-IDF and OvO in both datasets as shown in Table 5.6.

Table 5.6 Performance results of Logistic Regression.

Logistic Regression (%)	BoW+OvA	BoW+OvO	TF-IDF+OvA	TF-IDF+OvO
Small + Balanced	71.30	70.17	68.18	67.23
Large + Unbalanced	53.89	52.59	51.10	47.32

Unsurprisingly, in all the combinations for the large dataset, the LR classifier failed more often during testing to classify the least represented overall classes in our training data. This is obvious when we take a look at the confusion matrix of Fig. 5.21 below. The diagonal represents the percentage of the total correctly classified test samples for the specific label, out of the whole number of samples for this emotion. It is worth mentioning that the LR classifier was trained noticeably better to recognize labels such as *joy* and *sadness*, for which it contained the most samples, as was shown previously in Fig. 5.14.

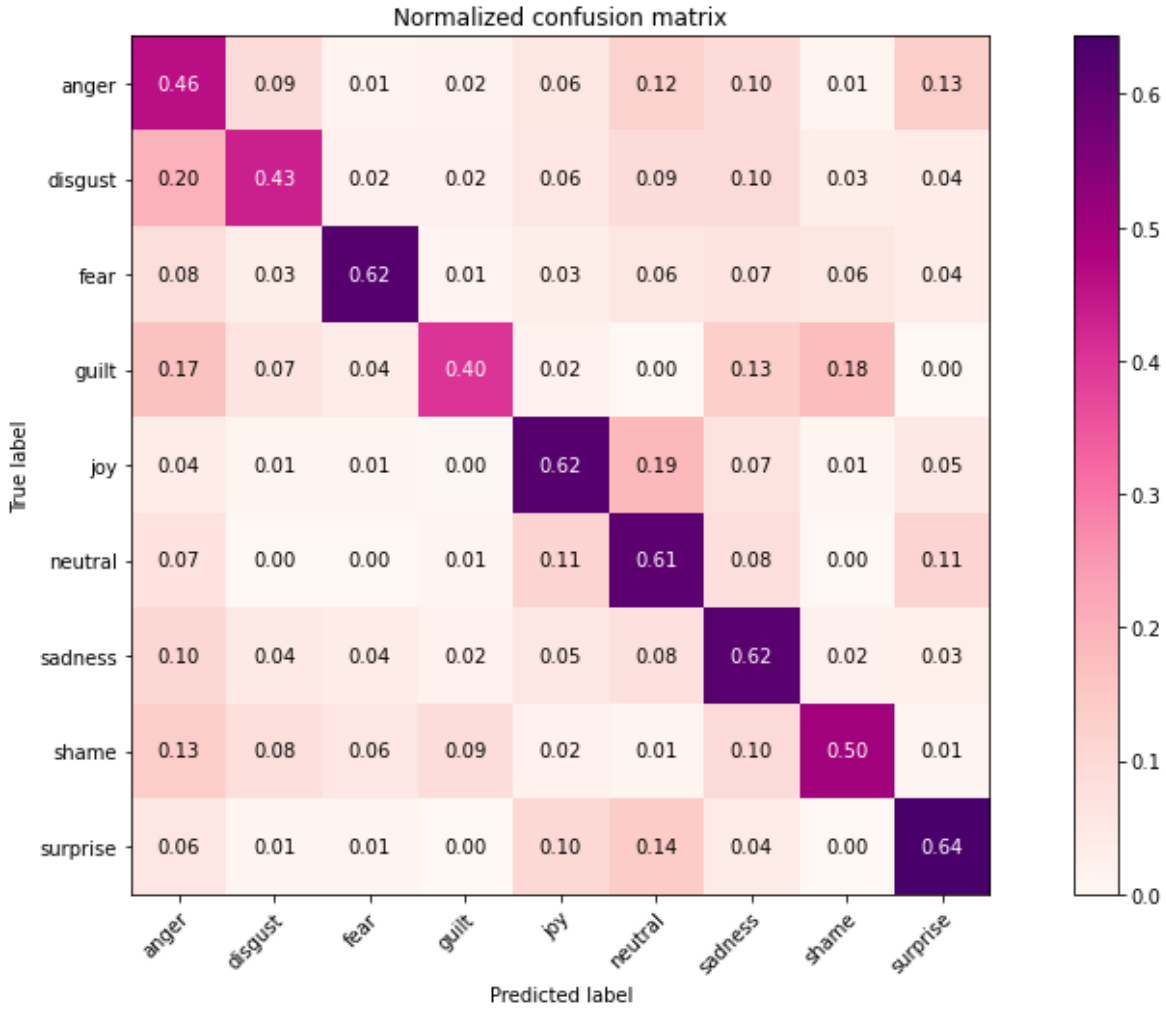

Fig. 5.21 The confusion matrix of the Logistic Regression classifier using the BoW representation and OvA approach. We notice that *disgust*, *guilt* and *shame*, the most underrepresented classes in the large dataset, are also the most oftenly misclassified by LR.

5.4.4 Linear Support Vector

Last but far from least, in order to test the Support Vector Machine algorithm on our datasets, we used the *LinearSVC()* classifier of the *sklearn* framework which inherently

uses the OvA approach. The more complex feature extraction of TF-IDF is indeed utilized by the SVM as it performs better than everything else tried on the datasets, reaching an $F1$ score of *73.31%* as shown in Table 5.7. The OvO method was tested by fitting the training data to the *SVC()* classifier with the kernel parameter set to *linear*. Despite this method being noticeably slower than OvA, when paired with TF-IDF vectorization it performed well, giving a score of *70.03%*.

Table 5.7 Performance results of the Linear Support Vector.

Support Vector Machine (%)	BoW+OvA	BoW+OvO	TF-IDF+OvA	TF-IDF+OvO
Small + Balanced	68.70	68.76	73.31	70.03
Large + Unbalanced	51.74	50.71	54.65	53.76

5.5 Future work

Our experiments showed great variation in performance, depending on the machine learning algorithm and vectorization method used. The other aspects that determined the quality of the results were the dataset used, as well as the preprocessing and vectorization methods applied.

Even though Decision Trees performed far from great on their own, it would be worth looking into ensemble learning methods that utilize them, such as Random Forests. Moreover, another technique that has gained wide acceptance in recent years but has yet to be applied for emotion classification purposes in text is gradient boosting which also usually involves weak models like Decision Trees.

Another compelling path for our future work in emotion classification would be that of trying out different Neural Network architectures. Based on the fact that our simple Logistic Regression experiments provided us with some of the best results in the context of this thesis, the logical next step is to apply neural networks, a combination of logistic regression nodes (as well as that of other functions).

Lastly, a common disadvantage of all the aforementioned approaches, regardless of the high F-scores acquired from some of them, is the fact that they extract and retain very little contextual and semantic information. This is why we consider the employment of word embeddings for our text representation from the words, together with Deep Neural Networks, a set of promising techniques as the deep layers of the latter are able to extract the intrinsic, hidden details that natural language might contain.

BIBLIOGRAPHY

[1] J. McCarthy and National Physical Laboratory (Great Britain), *Programs with Common Sense*. 1958.

[2] R. W. Picard, "Affective Computing." 2000 [Online]. Available: http://dx.doi.org/10.7551/mitpress/1140.001.0001 [Accessed: March, 2021]

[3] D. Schuller and B. W. Schuller, "The Age of Artificial Emotional Intelligence," *Computer*, vol. 51, no. 9. pp. 38–46, 2018 [Online]. Available: http://dx.doi.org/10.1109/mc.2018.3620963 [Accessed: March, 2021]

[4] T. Bincy, P. Vinod, and K. A. Dhanya, "Multiclass Emotion Extraction from Sentences," *International Journal of Scientific & Engineering Research*, vol. 5, no. 2, pp. 12–15, 2014 [Online]. Available: https://www.ijser.org/researchpaper/Multiclass-Emotion-Extraction-from-Sentences.pdf. [Accessed: March, 2021]

[5] M. Z. Asghar *et al.*, "Performance evaluation of supervised machine learning techniques for efficient detection of emotions from online content," *Preprints*, 02-Aug-2019. Available at: http://www.preprints.org/manuscript/201908.0019/v1 [Accessed: April, 2021]

[6] Z. Yuan and M. Purver, "Predicting emotion labels for Chinese microblog texts," in *Advances in Social Media Analysis*, Cham: Springer International Publishing, 2015, pp. 129–149. Available: http://link.springer.com/10.1007/978-3-319-18458-6_7 [Accessed: March, 2021]

[7] M. A. Azim and M. H. Bhuiyan, "Text to emotion extraction using supervised machine learning techniques," *TELKOMNIKA*, vol. 16, no. 3, p. 1394, Jun. 2018, doi: 10.12928/telkomnika.v16i3.8387.

[8] R. C. Solomon, "Emotions, philosophy of," *Routledge Encyclopedia of Philosophy*. Available: http://dx.doi.org/10.4324/9780415249126-n016-1

[9] D. K. Lapsley and P. C. Stey, "Id, Ego, and Superego," *Encyclopedia of Human Behavior*. pp. 393–399, 2012. Available: http://dx.doi.org/10.1016/b978-0-12-375000-6.00199-3

[10] C. Darwin, "The expression of the emotions in man and animals / by Charles Darwin."

1916 [Online]. Available: http://dx.doi.org/10.5962/bhl.title.4820

[11] W. James, “What is an emotion?,” *The emotions, Vol. 1*. pp. 11–30 [Online]. Available: http://dx.doi.org/10.1037/10735-001

[12] A. E. Coleman and J. Snarey, “James-Lange Theory of Emotion,” in *Encyclopedia of Child Behavior and Development*, Boston, MA: Springer US, 2011, pp. 844–846 [Online]. Available: http://link.springer.com/10.1007/978-0-387-79061-9_3146

[13] A. R. Damasio, “The somatic marker hypothesis and the possible functions of the prefrontal cortex,”*The Prefrontal CortexExecutive and Cognitive Functions*.pp. 36–50, 1998.

[14] W. B. Cannon, “Again the James-Lange and the thalamic theories of emotion,” *Psychol. Rev.*, vol. 38, no. 4, pp. 281–295, 1931, doi: 10.1037/h0072957. Available: http://doi.apa.org/getdoi.cfm?doi=10.1037/h0072957 [Accessed: March, 2021]

[15] P. Bard, “On emotional expression after decortication with some remarks on certain theoretical views: Part I,” *Psychol. Rev.*, vol. 41, no. 4, pp. 309–329, 1934, doi: 10.1037/h0070765

[16] S. Schachter and J. E. Singer, “Cognitive, social, and physiological determinants of emotional state,” *Psychol. Rev.*, vol. 69, pp. 379–399, Sep. 1962, doi: 10.1037/h0046234.

[17] J. Ledoux, *The Emotional Brain: The Mysterious Underpinnings of Emotional Life*. Simon and Schuster, 1998.

[18] R. S. Lazarus and S. Folkman, *Stress, Appraisal, and Coping*. Springer Publishing Company, 1984.

[19] D. G. Myers and C. Nathan DeWall, *Psychology in Everyday Life*. Worth Publishers, 2016.

[20] D. Sander, “Models of Emotion,” in *The Cambridge Handbook of Human Affective Neuroscience*, J. Armony and P. Vuilleumier, Eds. Cambridge: Cambridge University Press, 2013, pp. 5–54. Available: http://ebooks.cambridge.org/ref/id/CBO9780511843716A009

[21] R. A. Calvo and S. Mac Kim, “Emotions in text: Dimensional and categorical models,” *Comput. Intell.*, vol. 29, no. 3, pp. 527–543, Aug. 2013, doi: 10.1111/j.1467-8640.2012.00456.x.Available: http://doi.wiley.com/10.1111/j.1467-8640.2012.00456.x [Accessed: March, 2021]

[22] J. A. Russell, “A circumplex model of affect,” *J. Pers. Soc. Psychol.*, vol. 39, no. 6, pp. 1161–1178, 1980, doi: 10.1037/h0077714. [Online]. Available: http://content.apa.org/journals/psp/39/6/1161 [Accessed: March, 2021]

[23] R. Plutchik, “A GENERAL PSYCHOEVOLUTIONARY THEORY OF EMOTION,” in *Theories of Emotion*, Elsevier, 1980, pp. 3–33.

[24] Valence-arousal graph: https://github.com/katedukhnai/valence-arousal-recognition [Accessed: April, 2021]

[25] S. Gu, F. Wang, N. P. Patel, J. A. Bourgeois, and J. H. Huang, “A Model for Basic Emotions Using Observations of Behavior in,” *Front. Psychol.*, vol. 10, p. 781, Apr. 2019, doi: 10.3389/fpsyg.2019.00781.

[26] P. Ekman, “An argument for basic emotions,” *Cogn. Emot.*, vol. 6, no. 3–4, pp. 169–200, May 1992, doi: 10.1080/02699939208411068.

[27] P. Ekman, “Basic Emotions,” in *Handbook of Cognition and Emotion*, Chichester, UK: John Wiley & Sons, Ltd, 2005, pp. 45–60.

[28] M. Kowalska and M. Wróbel, “Basic Emotions,” in *Encyclopedia of Personality and Individual Differences*, Cham: Springer International Publishing, 2017, pp. 1–6.

[29] A. Ortony, G. L. Clore, and A. Collins, “The Cognitive Structure of Emotions.” 1988 [Online]. Available: http://dx.doi.org/10.1017/cbo9780511571299 [Accessed: March, 2021]

[30] M. A. M. Shaikh, H. Prendinger, and M. Ishizuka, “A linguistic interpretation of the OCC emotion model for affect sensing from text,” in *Affective Information Processing*, London: Springer London, 2008, pp. 45–73

[31] L. Huangfu, W. Mao, D. Zeng, and L. Wang, “OCC model-based emotion extraction from online reviews,” *2013 IEEE International Conference on Intelligence and Security Informatics*. 2013 [Online]. Available: http://dx.doi.org/10.1109/isi.2013.657

8799 [Accessed: June, 2021]

[32] E. Cambria, D. Das, S. Bandyopadhyay, and A. Feraco, *A Practical Guide to Sentiment Analysis*. Springer, 2017.

[33] W. Wang, L. Chen, K. Thirunarayan, and A. P. Sheth, "Harnessing Twitter 'Big Data' for Automatic Emotion Identification," *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*. 2012 [Online]. Available: http://dx.doi.org/10.1109/socialcom-passat.2012.119

[34] K. R. Scherer and H. G. Wallbott, "Evidence for universality and cultural variation of differential emotion response patterning," *Journal of Personality and Social Psychology*, vol. 66, no. 2. pp. 310–328, 1994 [Online]. Available: http://dx.doi.org/10.1037/0022-3514.66.2.310 [Accessed: March, 2021]

[35] S. Rosenthal, N. Farra, and P. Nakov, "SemEval-2017 Task 4: Sentiment Analysis in Twitter," *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 2017 [Online]. Available: http://dx.doi.org/10.18653/v1/s17-2088

[36] S. Buechel and U. Hahn, "EmoBank: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis," *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 2017 [Online]. Available: http://dx.doi.org/10.18653/v1/e17-2092 [Accessed: March, 2021]

[37] S. Mohammad and F. Bravo-Marquez, "WASSA-2017 Shared Task on Emotion Intensity," *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 2017 [Online]. Available: http://dx.doi.org/10.18653/v1/w17-5205 [Accessed: March, 2021]

[38] B. Wang, A. Tsakalidis, M. Liakata, A. Zubiaga, R. Procter, and E. Jensen, "SMILE Twitter Emotion dataset." figshare, 21-Apr-2016 [Online]. Available: https://figshare.com/articles/dataset/smile_annotations_final_csv/3187909/2 [Accessed: March, 2021]

[39] S. M. Mohammad and S. Kiritchenko, "Understanding Emotions: A Dataset of Tweets to Study Interactions between Affect Categories," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, 2018 [Online].

Available: https://www.aclweb.org/anthology/L18-1030.pdf. [Accessed: May, 2021]

[40] L. Yanran, S. Hui, S. Xiaoyu, L. Wenjie, C. Ziqiang, and N. Shuzi, “DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset,” 2017 [Online]. Available: https://arxiv.org/pdf/1710.03957. [Accessed: June, 2021]

[41] D. Preoţiuc-Pietro *et al.*, “Modelling valence and arousal in Facebook posts,” in *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, San Diego, California, 2016, doi: 10.18653/v1/w16-0404.

[42] V. Liu, C. Banea, and R. Mihalcea, “Grounded emotions,” in *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*, San Antonio, TX, 2017, doi: 10.1109/acii.2017.8273642.

[43] S. Poria, D. Hazarika, N. Majumder, G. Naik, E. Cambria, and R. Mihalcea, “MELD: A multimodal multi-party dataset for emotion recognition in conversations,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019, doi: 10.18653/v1/p19-1050.

[44] C. Sheng-Yeh, H. Chao-Chun, K. Chuan-Chun, T.-H. K. Huang, and K. Lun-Wei, “EmotionLines: An Emotion Corpus of Multi-Party Conversations,” 2018 [Online]. Available: https://arxiv.org/pdf/1802.08379v1.pdf. [Accessed: June, 2021]

[45] D. Ghazi, D. Inkpen, and S. Szpakowicz, “Detecting emotion stimuli in emotion-bearing sentences,” in *Computational Linguistics and Intelligent Text Processing*, Cham: Springer International Publishing, 2015, pp. 152–165.

[46] I. Gupta and N. Joshi, “Tweet normalization: A knowledge based approach,” in *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, Dubai, United Arab Emirates, 2017, doi: 10.1109/ictus.2017.8285996.

[47] M. F. Porter, “An algorithm for suffix stripping,”, *Program: electronic library and information systems,* vol. 39, p. 88, Mar. 1980, doi: 10.1108/eb046814.

[48] Q. McCallum, *Bad Data Handbook*. “O’Reilly Media, Inc.,” 2012.

[49] Y. Goldberg, *Neural Network Methods in Natural Language Processing*. Morgan &

Claypool Publishers, 2017.

[50] S. Themeli, “Hate Speech Detection using different text representations in online user comments.” 2018. Available: http://rgdoi.net/10.13140/RG.2.2.12991.25764 [Accessed: June, 2021]

[51] “Deep Learning.” [Online]. Available: https://cedar.buffalo.edu/~srihari/CSE676/. [Accessed: June, 2021]

[52] K. Sparck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *J. Doc.*, vol. 28, no. 1, pp. 11–21, Jan. 1972, doi: 10.1108/eb026526.

[53] J. R. Firth, *A Synopsis of Linguistic Theory, 1930-1955*. 1957.

[54] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, “A neural probabilistic language model,” *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003 [Online]. Available: https://dl.acm.org/doi/10.5555/944919.944966. [Accessed: May, 2021]

[55] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” 16-Jan-2013 [Online]. Available: http://arxiv.org/abs/1301.3781. [Accessed: May, 2021]

[56] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014, doi: 10.3115/v1/d14-1162.

[57] S. Raschka, “Naive Bayes and text classification I - introduction and theory.” Unpublished, 2014 [Online]. Available: http://rgdoi.net/10.13140/2.1.2018.3049 [Accessed: June, 2021]

[58] Y. An, S. Sun, and S. Wang, “Naive Bayes classifiers for music emotion classification based on lyrics,” in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, Wuhan, China, 2017, doi: 10.1109/icis.2017.7960070

[59] L. Wikarsa and S. N. Thahir, “A text mining application of emotion classifications of Twitter’s users using Naïve Bayes method,” in *2015 1st International Conference on Wireless and Telematics (ICWT)*, Manado, Indonesia, 2015, doi:

10.1109/icwt.2015.7449218

[60] T. LeCompte and J. Chen, “Sentiment analysis of tweets including emoji data,” in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA, 2017, doi: 10.1109/csci.2017.137

[61] CS246: Mining Massive Data Sets https://web.stanford.edu/class/cs246/. [Accessed: May, 2021]

[62] M. Kuhn and K. Johnson, *Applied predictive modeling*, 1st ed. New York, NY: Springer, 2013.

[63] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[64] M. Allouch, A. Azaria, R. Azoulay, E. Ben-Izchak, M. Zwilling, and D. A. Zachor, “Automatic detection of insulting sentences in conversation,” in *2018 IEEE International Conference on the Science of Electrical Engineering in Israel (ICSEE)*, Eilat, Israel, 2018, doi: 10.1109/icsee.2018.8646165

[65] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. 2015.

[66] A. Prabhat and V. Khullar, “Sentiment classification on big data using Naïve bayes and logistic regression,” *2017 International Conference on Computer Communication and Informatics (ICCCI)*. 2017.

[67] T. Joachims, “Text categorization with Support Vector Machines: Learning with many relevant features,” *Machine Learning: ECML-98*. pp. 137–142, 1998 [Online]. Available: http://dx.doi.org/10.1007/bfb0026683 [Accessed: June, 2021]

[68] H. Alshamsi, V. Kepuska, H. Alshamsi, and H. Meng, “Automated facial expression and speech emotion recognition app development on smart phones using cloud computing,” in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, 2018, doi: 10.1109/iemcon.2018.8614831.

[69] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch, “Support vector machines and kernels for computational biology,” *PLoS Comput. Biol.*, vol. 4, no. 10, p. e1000173, Oct. 2008, doi: 10.1371/journal.pcbi.1000173

[70] W. McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, Austin, Texas, 2010, doi: 10.25080/majora-92bf1922-00a

[71] python, “python/cpython.” [Online]. Available: https://github.com/python/cpython. [Accessed: May, 2021]

[72] E. Loper and S. Bird, “NLTK,” in *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics* -, Philadelphia, Pennsylvania, 2002, doi: 10.3115/1118108.1118117

[73] “Getting Started with Scikit-learn for Machine Learning,” *Python® Machine Learning*. pp. 93–117, 2019.