GHENT UNIVERSITY

FACULTY OF SCIENCES

Fuzzy Rough Set Methods for Emotion Detection and Sentiment Analysis

Olha Kaminska

Dissertation submitted in fulfilment of the requirements for the degree of Doctor of Science: Computer Science

Academic year 2022–2023

Department of Applied Mathematics,
Computer Science and Statistics
LT3 Language and Translation Technology Team
Faculty of Sciences
Ghent University

Supervisors

Prof. dr. Chris Cornelis

Department of Applied Mathematics, Computer Science and Statistics,
Ghent University, Belgium

Prof. dr. Veronique Hoste

LT3 Language and Translation Technology Team,
Ghent University, Belgium

Other members of the examination board

Prof. dr. Kris Coolsaet (chair)

Department of Applied Mathematics, Computer Science and Statistics,
Ghent University, Belgium

Prof. dr. Guy De Tré

Department of Telecommunications and Information Processing,
Ghent University, Belgium

Prof. dr. Els Lefever

Department of Translation, Interpreting and Communication,
Ghent University, Belgium

Dr. Enislay Ramentol

Fraunhofer-Institut für Techno- und Wirtschaftsmathematik ITWM
Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung, Germany

Prof. dr. Walter Daelemans

Department of Linguistics,
University of Antwerp, Belgium

Dr. Germán Hurtado Martín

Data as a Service, Search & Match Team,
The Stepstone Group, Belgium

This work was supported by
Research Foundation Flanders (grant no. G0H9118N)

Contents

List of Tables

List of Figures

List of Acronyms

Summary

In this thesis, we consider several challenges from the Natural Language Processing area connected to the fields of emotion detection and sentiment analysis. We approached these tasks with strategies based on fuzzy rough set models, since they are simple yet effective methods from the classic machine learning family that also can present a local explainability for the results. The goal of this work is to provide an overview of fuzzy rough set based models' performance for emotion detection-related tasks, compare the obtained results with state-of-the-art solutions, and investigate the explainability of our approach.

Particularly, we work with ordinal multiclassification tasks for emotion intensity categorization and binary classification for offensive language, hate speech, irony, and sarcasm detection. As prediction models, we use the following instance-based methods: weighted kNN, fuzzy-rough nearest neighbour (FRNN) classification, and FRNN regression model. We apply several embedding techniques for each of them to transform the text into vector form, and we improve our models' performance by means of parameter tuning, confidence scores and ensembles. In our results, for all emotion detection-related tasks, we demonstrate that simple nearest neighbour-based approaches obtain comparable results to advanced deep learning methods.

The second task that we tackle with the fuzzy-rough-based approach is aspect-based sentiment analysis (ABSA). We consider a dataset of product reviews, where each includes several sentences with predefined aspects, labelled with three classes: category of the product, sentiment, and emotion. For this challenge, we suggest three pipelines involving FRNN classification and fuzzy rough one-versus-one combination (FROVOCO) methods that can provide high accuracy for all three steps of aspect, sentiment, and emotion classification. Our results for the English data perform on the same level as approaches without the usage of fuzzy rough set theory done for the Dutch version of the same dataset.

In the end, another important characteristic of our approach is its explainability. We can classify our approach as a local, self-explaining and example-driven method. We show through error analysis that our methods can extract useful patterns from the data on the local level, in other words, for any test instance individually. In future, this approach can be extended and automatized to provide improvement of the methods described in this thesis.

Nederlandstalige samenvatting

In dit proefschrift richten we ons op diverse uitdagingen uit het domein van natuurlijke taalverwerking, gelinkt aan de gebieden van emotiedetectie en sentimentanalyse. We lossen deze taken op met behulp van fuzzy rough set gebaseerde modellen, aangezien het eenvoudige maar effectieve methoden zijn uit de familie van klassieke machine learning technieken die ook hun resultaten lokaal kunnen verklaren. Het doel van deze thesis bestaat erin om een overzicht te geven van de performantie van fuzzy rough set gebaseerde modellen voor taken gerelateerd aan emotiedetectie, de bekomen resultaten te vergelijken met state-of-the-art aanpakken, en de uitlegbaarheid van onze methode te onderzoeken.

In het bijzonder werken we met ordinale multiklasse-classificatie voor het categoriseren van emotie-intensiteit; binaire classificatie voor het detecteren van aanstootgevende taal, het aanzetten tot haat, ironie en sarcasme; en aspectgebaseerde sentimentanalyse (ABSA) voor gebruikersrecensies. Als predictiemodellen gebruiken we de volgende instantiegebaseerde methoden: gewogen kNN, fuzzy-rough nearest neighbour (FRNN) classificatie en regressie. We gebruiken verschillende woordinbeddingstechnieken voor elk model om tekst om te zetten in vectorvorm, en we verbeteren de prestaties van onze modellen door middel van parameter-tuning, betrouwbaarheidsscores en ensembles. Voor alle emotiedetectietaken laten we zien dat eenvoudige, op naaste buren gebaseerde benaderingen vergelijkbare resultaten opleveren als geavanceerde deep learning-methoden.

De tweede taak die we hebben aangepakt met fuzzy rough set gebaseerde methoden is aspectgebaseerde sentimentanalyse (ABSA). We beschouwen een dataset met productbeoordelingen, die elk bestaan uit meerdere zinnen met daarin voorgedefinieerde aspecten, gelabeld met drie klassen: categorie van het product, sentiment, en emotie. Voor de ABSA-taak stellen we drie pijplijnen voor met methoden gebaseerd op FRNN en op fuzzy rough one-versus-one combination (FROVOCO) die een hoge nauwkeurigheid opleveren voor alle drie de stappen: aspect-, sentiment- en emotieclassificatie. Onze resultaten voor de Engelstalige data liggen op hetzelfde niveau als aanpakken zonder fuzzy rough set theorie uitgevoerd voor de Nederlandstalige versie van dezelfde dataset.

Tot slot is een ander belangrijk kenmerk van onze aanpak haar uitlegbaarheid. We kunnen onze aanpak typeren als een lokale, zelfverklarende en voorbeeldgestuurde methode. We laten door foutanalyse zien dat onze methoden bruikbare patronen uit de gegevens kunnen halen op lokaal niveau, met

andere woorden, voor elk testvoorbeeld afzonderlijk. In de toekomst kan deze aanpak worden uitgebreid en geautomatiseerd om de methdoden beschreven in dit proefschrift te verbeteren.

Acknowledgements

These four years of my PhD journey felt like a totally different life and changed a lot of who I was and what I had. I perceive it as a long journey, full of ups and downs, with plenty of life lessons. Nevertheless, here I am, at the end of this way, holding this book in my hands. This is possible thanks to great people that I was lucky enough to have by my side. And this section is a crucial part of my work, in which I will thank every one of them.

First of all, I want to say the greatest "thank you!" to my dear supervisors, Dr. Chris Cornelis and Dr. Veronique Hoste. Chris was a person who believed in me so much that offered me this great opportunity, relying only on my documents and one online meeting! He also helped me the most during my move to Ghent, and I will always be grateful for this. I greatly appreciate all the academic and research work we did together, and I can say that I learned a lot from him. Veronique was a great example and a huge inspiration for me. Her input in our work and academic suggestions for our publications cannot be overestimated. I truly believe that supervisors are a crucial part of any PhD experience, and from this perspective, I consider myself a very lucky PhD student. Heel hartelijk bedankt!

I also want to express my gratefulness to the members of the examination boards: Dr. Kris Coolsaet, Dr. Guy De Tré, Dr. Els Lefever, Dr. Enislay Ramentol, Dr. Walter Daelemans, and Dr. Germán Hurtado Martín. Your feedback was so kind and helpful that it makes me a lot more proud looking at my thesis and knowing how greatly it was improved thanks to you. Thank you for your attention to my work. It was a great honour to present it to you all. Bedankt! ¡Muchas gracias!

A huge part of my PhD experience were my dear colleagues, who made my life much brighter. Especially, my favourite "board game" group: Camila, Marko, Slađa, Paweł, and Oliver L.! During the most challenging times, every our meeting, every board game, every birthday party, and every small trip kept my spirit and gave me strength to go on. I want to say a separate "thank you" to Camila for our great talks, adventures, and your kind words. To Marko and Slađa for every great trip and kitchen talk, and for sharing your culture and fun moments with me. To Paweł for every "7 Wonders" game session and every borsch and sushi we shared. To Oliver L. for being a huge example for me and for all your help with my work. I also want to say thank you to Martina for

joining our team and bringing so much joy to you. I want to mention all my colleagues from the Statistics department. It is probably impossible, but I will try my best, sending my regards to Adnan, Henri, Hege, J., Hans, Oliver D., Kelly, Wout, Georgi, Stijn, Fatemeh, Joris, and anyone I could unintentionally miss!

Separately, I want to mention my second dear lab named LT3. It is a great community full of brilliant people, and was so grateful for the opportunity to join it (maybe much less often than it actually deserves). I'm grateful to Bram and Sofie for being my first co-authors in our SemEval paper! I want to express my appreciation to Luna, for working together on Digital Humanities seminars and for her adorable thesis cover, which inspired me a lot. Thanks to Gilles for the great Belgian-style housewarming party in Leuven. Thanks to Cynthia for huge help during my moving and support during the loss of phone situation. Also, I'm very grateful to Ellen for her help with SentEmo data and for her cute pets' stories! Particularly, I want to express my great gratefulness to Margo, with whom we started our LT3 journey almost at the same time and shared a lot of coffee cups, yoga lessons, and warm talks! I also want to mention Orphee, Claudia, Aaron, Arda, Jasper, Joke, Lieve, Loic, Michael, Paola, Pranay, Toon, Ayla, Thierry, Camiel, Seza, Colin, Johnatan, Serafina, and anyone I could unintentionally miss!

In the end, I saved my appreciation to my lovely friends and family. These words I want to shape in our beautiful native language - Ukrainian.

Перш за все, я хочу подякувати захисникам і захисницям України, які боронять наш дім і наші родини, даючи іншим змогу працювати та надсилати донати.

Також я дуже вдячна своїм друзям, яких я зустріла вже в Генті - Руслані та Лесі. Дякую Руслані за чудовий час разом за настолками та пивом, а Лесі за довгі прогулянки, душевні бесіди та смачні коктейлі. Кожна зустріч з кожною з вас дуже допомагала мені справлятися з тугою за домом!

Хочу подякувати всім своїм дорогим друзям, які знаходились так далеко від мене, але завжди поруч морально. Дякую Михайлу, Віктору, Катерині та Анні Міт. за наш прекрасний книжковий клуб, походи по театрах і моральну підтримку одне одного в такі складні часи. Дякую Маргариті, Артему, Катерині П., Євгенії, Аліні, Владиславу, Діані, Олегу, Антону, що ви є і за всі ті короткі зустрічі, які ми мали відтоді як я поїхала з Естонії. Дякую Олександрі та Наталі за те що залишилися в моєму житті та за всі чудові зустрічі, що ми мали після випуску з ІПСА. Дякую Юлії та Олександру за ваш академічний приклад в моєму житті, поїздки разом та короткі зустрічі за кавою. Дякую Анні К. та Володимиру за гостинність і ту величезну роботу що ви робите. Дякую Анні Л. за час в Генті та Нансі та розділення захоплення модерном.

Окремо хочу згадати українську художню спільноту, з якою мені випала честь співпрацювати. Дякую за можливість винести моє захоплення малюванням довжиною в життя на зовсім інший рівень і підтримку!

Ближче до кінця цієї глави, хочу сказати про найважливіших людей у

моєму світі. Перш за все, за моїх найкращих у світі подружок, про яких я могла б тільки мріяти - Олену, Катерину Б., Єлизавету та Дарину. Мені не вистачить всієї цієї книги, щоб описати наскільки я вас люблю і за що я вам вдячна... Але я спробую! Дякую Лені за години аудіо, розмови про роботу, життя та все на світі та наші пригоди в Дрездені. Дякую Каті за те що ти є в моєму житті, за всі наші пригоди та інтереси які ми ділимо вже стільки років. Дякую Лізі за всі ночівлі в Кракові та те, що я завжди знаю що ти мене підтримаєш і зрозумієш. Дякую Даші за той позитив, що ти завжди вносиш в моє життя з собою і за неоціненний досвід зі Спарклом! Також хочу окремо подякувати Анні Маз. за те що повернулась в моє життя.

Під кінець, хочу подякувати своїй родині. Дякую бабусі Валентині та діду Володимиру за підтримку і віру в мене. Дякую бабусі Лідії, діду Василю і всім родичам зі сторони мого чоловіка - Людмилі, Ігорю, Валентині, Михайлу та Віталію за вашу доброту. Найголовніше, хочу подякувати своїм дорогим батькам Ярославі та Миколі. Мені важко описати наскільки для мене важливо все, що ви для мене зробили. Дякую вам за підтримку, години в Скайпі та віру в мене. Я б не впоралась без вас і всі ці роки я щодня за вами сумую. Люблю вас!

Якщо ви дочитали до цього місця, ви напевно знаєте хто лишився. Найкраща людина у всесвіті та мій коханий чоловік - Вячеслав. Ти значиш світ для мене і кожна секунда з тобою дає мені сенс йти далі. Я маю цю книгу завдяки тобі, як і все щастя у своєму житті. Кохаю тебе.

Chapter 1

Introduction

In this chapter, we provide a general introduction to this dissertation. In Section 1.1, we discuss emotion detection challenges in the area of Natural Language Processing (NLP), particularly, sentiment analysis, emotion intensity classification, and others. We also list tasks and datasets considered in our work. Section 1.2 provides an overview of Machine Learning (ML) methods for emotion detection, beginning from state-of-the-art approaches to Fuzzy Rough Set (FRS) based techniques that we considered in our experiments. Finally, Section 1.3 contains an overview of the thesis with a short description of each chapter's content.

1.1 Emotion detection challenges in Natural Language Processing

The exponential growth of social media has created various novel ways of communication. A significant part of online content is formed by textual information, which gives rise to diverse tasks within the data science branch of NLP [57]. This field is represented by various tasks, including customer feedback analysis [103], sentiment interpretation [118], topic detection [60], and many others.

The process of revealing emotions in written or spoken language is known as emotion detection [122], which is a major task of the NLP field of study. It can offer important insights into the underlying mood, attitudes, and intentions of people or organizations in different areas. Previously, studies on emotion identification have been conducted in the field of human-computer interaction in a variety of ways, such as the analysis of facial expressions [42] or emotion recognition using a range of sensors [102]. However, the complexity and ambiguity of human language, as well as the delicacy of emotional expression, make it difficult to recognize emotions, especially in NLP-related tasks, where it should be done from the standalone text without concomitant facial expressions, gestures and the tone of a person's voice.

From an application standpoint, automatic emotion recognition in texts is

becoming more and more significant in computational linguistics. It can be used in various fields, including opinion mining of users, market analysis, usage of natural language in entertaining games, or online learning platforms.

1.1.1 Overview of emotion detection tasks

Emotion detection could be categorized under the more general and widely-investigated area of sentiment analysis, where the purpose is to define the attitude conveyed in a text. The sentiment is usually either positive or negative but can also have a third neutral meaning, see e.g. [115].

The sentiment and emotion detection area includes a lot of NLP challenges, such as the identification of specific emotions, as well as their intensity or valence. It can also include the recognition of the speaker's underlying attitudes or objectives (for example, irony, sarcasm, or persuasion). Below, we provide an overview of the main tasks connected to this field of subjective language detection and emotion detection:

- Stance detection, which identifies the author's stance of the considered part of the text toward the specific target. Stance usually is classified into one of three classes: "in favour", "against", or "neither" [71].

- Classification of emotion presented in a text, for example, customer comments, based on whether they express an angry, happy, or disappointed feeling, see e.g. [151]. This challenge can be shaped in different forms, including binary classification, where the goal is to identify if the targeted emotion is present in a piece of text or not, as well as multi-classification tasks, where one emotion can have several types. As a related example of the task, we can consider the Offensive Language Identification Dataset (OLID) by [141], where for each offensive text, the authors provided additional categories, for example: is the offensive speech targeted, and in case it is, is it targeted against individuals, groups, or others.

- Emotion intensity classification, where ordinal labels represent different levels of a given emotion. For example, in [49], the authors labelled a dataset for sentiment with scores from 0 (very negative) to 1 (very positive). They also labelled the same data for various emotions (fear, anger, happiness, etc.), ranging from 0 (absence of the emotion) to 1 (extreme intensity of the emotion).

- Aspect-Based Sentiment Analysis (ABSA), which is a method for examining how people feel about particular qualities or elements of an item, service, or event, see e.g. [106] and [22]. ABSA aims to determine the sentiment connected with specific characteristics of a product or service, unlike traditional sentiment analysis mentioned above, which assigns an overall sentiment score for the whole text. To illustrate the ABSA task, let us take a look at a customer's feedback "the battery of the phone works well", which expresses a positive opinion (satisfaction) about the aspect of

"battery". At the same time, the same customer could also complain that "the memory is too limited", expressing a negative opinion about the aspect of "memory" and showing disappointment.

- Inter-domain tasks, which include the intersection of emotion detection with other fields of data analysis; for example, recommending material (such as movies, songs, or purchase products) based on user interactions with a system (streaming service or online shop) is at the intersection with the recommender systems area and is named "emotion-based content recommendation", see e.g. [72].

In this dissertation, we considered several tasks mentioned above, including tasks that are a bit different but still close to emotion detection. Particularly, as an example of binary classification, we approached two important subjective language classification tasks:

- Hate speech classification. The term "hate speech" is a broad concept that includes all kinds of negative comments targeted to insult someone based on some aspect (gender, race, religion, political beliefs, etc.) Most social networks provide automated hate speech detection and cleaning tools, and research in this area is very active, see e.g. [84]. In our study, we consider a binary classification task, where for each text sample, we should determine if it is hateful or not.

- Irony (or sarcasm) detection. Irony is often identified as a trope or figurative language use where the actual meaning is different from what is literally enunciated [19]. Modelling irony has a lot of potential for various NLP applications, such as automatic sentiment and emotion analysis, but also for online hate speech detection, etc. Unfortunately, detecting irony is complicated, even from a human perspective, as it can be expressed in a variety of ways, using metaphoric language or humour, and very often contextual information or facial expression information is needed to distinguish between ironic and non-ironic utterances. It makes irony detection a very challenging task, both to label such datasets and to train classification models, see e.g. [45]. Here we also tackle a binary classification task with two classes: ironic and non-ironic.

In the following section we discuss these tasks in more details and present specific datasets that we used in our work.

1.1.2 Emotion detection tasks and datasets presented in our work

We apply our methods to the different classification tasks introduced above, particularly: the emotion intensity task, binary emotion classification for hate speech and irony, and the ABSA. For each task, we used one or several datasets,

where the majority of them were provided by the online Semantic Evaluation (SemEval)[1] competitions:

1. For emotion intensity classification, we use the data provided by the SemEval 2018 Task 1 "EI-oc: Affect in Tweets for English"[2], where for four emotions (anger, joy, sadness, and fear), the organisers provided a collection of tweets with intensity labels (ranging from 0, which corresponds to "no emotion can be inferred", to 3, "a high amount of emotion can be inferred").

2. For hate speech detection, we consider two different datasets. The first dataset originates from SemEval 2019 Task 5, "Shared Task on Multilingual Detection of Hate"[3]. We consider English tweets from subtask A, "Hate Speech Detection against Immigrants and Women", a binary classification task with 9,000 training and 1,000 development instances. It refers to an important problem in modern social media communication, such as racism and sexism detection. The second dataset was released in the context of SemEval 2019 Task 6: "OffensEval: Identifying and Categorizing Offensive Language in Social Media"[4]. In subtask A, "Offensive language identification", the authors presented a dataset of more than 13,000 English tweets labelled as offensive or not. Offensive language was defined as language aimed to hurt someone's feelings, increase the level of anger and start arguing[5]. This definition illustrates that the concepts of offensive language and hate speech are very similar. Both of them identify problematic and harmful content, but they still have a different focus. Hate speech detection mostly targets discrimination and violence towards individuals or groups based on race, gender, sexuality and others. Meanwhile, offensive language recognition identifies inappropriate language in general, including profanity, insults, vulgarities, and so on.

3. For irony detection, we use two datasets as well, where the first is the dataset from SemEval 2018 Task 3, "Irony detection in English tweets"[6] and solve subtask A, which is a binary classification issue: is a given tweet ironic or not? The authors gathered the dataset of nearly 4,000 English tweets using three hashtags: #irony, #sarcasm, and #not. After manually labelling the data, the authors gathered a similar number of non-ironic tweets to obtain a balanced dataset. The second dataset originates from SemEval 2022 Task 6 called "iSarcasmEval", which considers sarcasm detection in two languages: English and Arabic [1]. We tackled subtask A for English with approximately 4,800 instances, where for a given text, we should determine whether it is sarcastic. As described by the authors of

[1] `https://semeval.github.io/`
[2] `https://competitions.codalab.org/competitions/17751`
[3] `https://competitions.codalab.org/competitions/19935`
[4] `https://competitions.codalab.org/competitions/20011`
[5] `https://aclawgroup.com.au/criminal-law/offences/offensive-language/`
[6] `https://competitions.codalab.org/competitions/17468`

the dataset, text writers provided the labels by themselves to exclude subjective labelling. As we mentioned, irony detection is quite a challenging task because the concept of irony itself is vague even for human beings, and it requires manual annotation to obtain a high-quality dataset. Meanwhile the concept of irony portrays a contrast between reality and human expectations; sarcasm employs mocking language for neglect or amusement.

4. For the ABSA, we used Fast Moving Consumer Goods (FMCG) reviews which were collected and manually labelled within SentEMO project[7] and contain about 1,300 product reviews. Each review is represented by one or multiple sentences and contains several or no "aspect terms", which are words or collocations which have been assigned three labels, namely a category class, sentiment class and emotion class. In further experiments, we will divide each of these annotations into three separate tasks, each of which generates a set of classification labels.

1.2 Machine learning and fuzzy rough set based methods for emotion detection

The NLP field offers a variety of techniques for emotion recognition challenges, including rule-based systems, ML, and Deep Learning (DL) approaches. Rule-based systems offer different solutions, for example, based on the usage of lexicons or knowledge bases, which link words or phrases with particular emotions. ML-based algorithms learn models to recognize the patterns and attributes hidden in the data to detect them in new, unseen instances. Such techniques can be divided into three main groups, including supervised learning (where the model learns from labelled data with corresponding instance-class examples), unsupervised learning (where data is unlabelled and the model learns to identify hidden patterns and clusters), and reinforcement learning (where a so-called "agent" learns in a dynamical environment based on feedback). In this thesis, our focus will be on supervised classification approaches. The description of DL-based methods is provided in the following section.

Before we dive into those methods, it is important to mention that k Nearest Neighbours (kNN)-based methods were used for NLP-based tasks previously [31]. The main motivation for their usage in such challenges was their non-parametric characteristic, which allows diverse representation learning methods. The kNN-based methods can also provide the handling of exceptions and generalizations, as well as allow similarity-based nearest neighbour explanations.

1.2.1 State-of-the-art methods

Cutting-edge solutions to the NLP-based problems are typically based on DL (see e.g. [56]), where the dominant methods for a long time were Neural Network (NN) approaches, including Convolutional Neural Networks (CNN) [14,

[7] `http://sentemo.org`

121], Long Short Term Memory (LSTM) [8, 113], and similar. During the last years, transformers were introduced. Transformers have a neural network architecture and deserve a more detailed description. The encoder and decoder are the main parts of transformers. To create contextualized representations for each token, the encoder goes through the input sequence token by token. By paying attention to every other token in the sequence, each token's representation is improved, capturing both local and global context. While the encoder concentrates on comprehending the input sequence, the decoder pays attention to the encoder's representations to produce the output sequence. Another fundamental element of transformers is the attention mechanism, which enables the model to concentrate on important sections of the input sequence. Self-attention is a form of attention in which the input sequence is viewed as a collection of questions, keys, and values. This technique enables the model to recognize dependencies between various input sequence positions. One of the transformers' advantages, compared to the older approaches, is that they enable parallel processing and capture distant dependencies, which reduces the drawbacks of sequential processing. Transformers have demonstrated state-of-the-art performance on various benchmarks, surpassing previous techniques.

One of the first presented transformer-based models was Bidirectional Encoder Representations from Transformers (BERT) by [37]. BERT works by pre-training a language representation model with a transformer-based architecture on massive volumes of text. BERT is pre-trained using unsupervised learning to pick up general language representations from a big corpus. During pre-training, it makes use of challenges for Masked Language Modeling (MLM) and next-sentence prediction. This model can afterwards be adjusted for subsequent tasks by fine-tuning on additional data. So during the following tuning, the model is refined using task-specific labelled data for downstream tasks like text categorization or named entity recognition. By including task-specific layers and training on the labelled data, fine-tuning adjusts the previously trained BERT model to particular tasks This model was a significant advance in the NLP area and produced state-of-the-art outcomes on a variety of challenges, including sentiment analysis, Named Entity Recognition (NER) and question answering.

Since the introduction of BERT, several novel models were presented, which are, in fact, modifications of this first model. One of the most notable of such models is Robustly Optimized BERT Pre-training Approach (RoBERTa) by [82]. The RoBERTa model is an improvement and expansion of the BERT approach. Although the two models are comparable, RoBERTa incorporates adjustments and training techniques that enhance performance. Its pre-training procedure is similar to BERT's but with some changes, particularly, RoBERTa uses a higher percentage of masked input tokens and completely skips the next-sentence prediction task since it was proven to be less useful for downstream tasks. Both BERT and RoBERTa's architectures use the transformer model, which consists of encoder layers with self-attention mechanisms. However, in comparison to the original BERT, RoBERTa is trained with larger model sizes

and more parameters, enabling it to recognize more intricate linguistic patterns.

Other modifications of BERT are Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA) by [26] (a more sample-efficient version of BERT), A Lite Bidirectional Encoder Representations from Transformers (ALBERT) by [74] (a smaller version of BERT with parameters reduction and increased speed of performance), and more. In contrast to these models, which often were created for specialized tasks, the most recent state-of-the-art transformers were created to be a general-purpose NLP model that can be applied to any task, for example, Text-to-Text Transfer Transformer (T5) by [109], or transformer-based text generative models, such as Generative Pre-trained Transformer (GPT) models, especially GPT-3 [17] with 175 billion parameters and the newest GPT-4 [98], which is even bigger.

Meanwhile, transformers-based models remain state-of-the-art language models for various emotion detection-based tasks, including the ones we consider in this thesis: sentiment analysis and emotion intensity recognition [23], hate speech detection [95], and irony classification [108]. While such solutions are generally able to reach high prediction accuracy, a downside to their use is that they are black-box solutions and hence suffer from a lack of explainability regarding the way the predictions are obtained. Therefore, a growing need emerges for explainable models that can identify e.g. why a particular text was labelled with a particular emotion intensity level and which patterns can be identified. For these reasons, we consider kNN-based solutions, not as a replacement, but as an alternative and extension of DL-based methods.

1.2.2 Fuzzy rough set based methods

Inspired by the fuzzy nature of textual data, in our research, we consider the usage of FRS-based methods. Firstly proposed by [140], fuzzy logic and fuzzy set theory are normally used to model the partial truth of logical propositions. It assumes that an expression has a degree of truth represented by a value from 0, which denotes a statement that is entirely false, to 1, which corresponds to an entirely true statement. Fuzzy relations and fuzzy membership degrees in decision classes are two main applications of fuzzy logic in the data analysis field. The first ones are used to measure the relationships among vector-represented instances, including text embedding vectors. The second one assumes that one instance can be part of various classes on different levels, where such an approach can be used in tasks like a recommendation. It also can be applied to the emotion detection task, with the assumption that some emotions are presented in multiple text pieces with different degrees.

One of the first approaches based on fuzzy rough methods and nearest neighbours was proposed in [61]. Such FRS approaches have been used successfully in various ML applications [127], including rule-based classifiers [149], fuzzy decision trees [143], learning from imbalanced data [128], feature and instance selection [146], fuzzy rough neural networks [147], etc. As an added value of fuzziness we can name gradual similarity (elements can have varying degrees of membership as opposed to being simply in or out) since rough sets do not have

it.

An advantage of instance-based methods like those based on FRS is that they provide a more understandable architecture with interpretable results by virtue of the concept of graded similarity. Concretely, new text fragments are paired with similar extracts from training data, providing immediate and intuitive clues as to why a given decision was made. While FRS-based methods initially provide less accurate predictions than DL approaches, we show in our study that with proper feature engineering and ensemble construction, we can obtain results on the same level as state-of-the-art DL approaches.

Finally, we would like to highlight that FRS-based methods have previously rarely been applied to text analysis tasks, and their explainability advantage was not investigated. Several fields of application of methods based on FRS to the emotion-related NLP tasks and exploration of the obtained results to improve our model provide the novelty of our research and give a lot of opportunities for future studies.

1.3 Overview of the dissertation

In this thesis, we explore an application of FRS-based prediction models to emotion detection tasks in the NLP area. Particularly, we investigate the explainability of such an approach and how it can be used to improve the model. These ideas were not analysed in detail before; this investigation is therefore a novel contribution of this dissertation.

This dissertation has the following structure:

- Chapter 2 provides an overview of related studies. Particularly, in the first part, we take a look at the SemEval competitions from the authors whose datasets we used and the winning teams. It gives us an opportunity to explore novel solutions that were used and compare the performance of our methods to that of theirs. In the second part, we discuss papers in the field of ABSA, from the first works to the corresponding SemEval competitions and the SentEmo project, which provided us with the dataset we considered in our work. Finally, we analyse studies in the area of interpretability of methods for text analysis. There we provide several classifications of such approaches; in particular, we talk about post-hoc and self-explanatory methods. In the end, we describe our solution as local, self-explaining, and example-driven.

- In Chapter 3, we provide a list of methods that we used for data preprocessing and transformation. In the first part, we discuss all text cleaning techniques that we considered for the datasets we used, such as emoji modification, stop word cleaning, and others. In the second, we describe all text vectorization or embedding methods that we used in our work, for example, Word2Vec, DeepMoji, BERT-based models, and so on. Finally, we review the concept of lexicons and list several of them that we were using.

- Chapter 4 provides the main theoretical background behind our experiments, describing the classification and regression model that we considered, alongside the similarity relation we used. It also depicts the idea of an ensemble of methods and gives the list of metrics that we applied for result evaluation.

- Chapter 5 provides the results of our experiments for different emotion recognition tasks and is divided into two parts. In the first one, we dive into the topic of emotion intensity recognition, where we tackle the multi-class classification task with two classification models and one regression model, comparing their outcomes. In the second part, we consider a binary classification task, particularly hate speech and irony detection, using the kNN-based and FRS models.

- In Chapter 6, we show our outcomes for the ABSA task, starting with the dataset and task interpretation, following an overview of the three pipeline approaches we suggested. There, beside the methodology description, we also review the model tuning steps and illustrate how we combine them into pipelines. In the end, we discuss the obtained results and evaluate the performance of our pipelines.

- Chapter 7 uses the results of the previous two chapters to perform an error analysis. For each dataset, we illustrate how the explainability of our approach on the local level works and what kind of patterns we can observe.

- In Chapter 8, we recapitulate the main conclusions of the dissertation, and discuss different opportunities for future work.

- At the end of the thesis, we provide Appendix A with an overview of the software that we produced during the work on the dissertation and Appendix B with the list of publications.

Chapter 2

Related work

In this chapter, we provide an overview of the main sources that we considered during the work on this dissertation. Since we were researching several aspects of our methodology by using various datasets and methods and investigating explainability techniques, we split this chapter into several parts. In Section 2.1, we describe emotion-based datasets from different SemEval competitions with which we were working. In Section 2.2, we provide an overview of ABSA related works through the history of its development and the specific setup that we considered. In the last Section 2.3, we explore the topic of interpretability in text analysis methods and classify our solution regarding provided criteria.

2.1 SemEval competitions: tasks and winning solutions

To investigate the performance of our proposal based on fuzzy-rough methodologies for the emotion detection area, we apply our methods to different classification problems originating from the SemEval[1] competition. The SemEval competition is an annual event that provides a set of challenges for researchers in different aspects of the NLP field. Each year, the SemEval competition presents around twelve tasks divided into groups based on a common topic. During the last several years, the most repeated topics were lexical semantics, emotions detection or sentiment analysis, information extraction, and application of NLP techniques to textual datasets from other fields such as law or medicine.

In our experiments, we considered datasets for emotion intensity (Section 2.1.1), hate speech classification (Section 2.1.2), and sarcasm and irony detection (Section 2.1.3). Below we will describe the competitions' tasks and winners' solutions, meanwhile, our results and obtained places are provided in Chapter 5.

[1] `https://semeval.github.io/`

2.1.1 Emotion intensity

We considered SemEval 2018 Task 1 [93] for the task of assessing emotion intensity. It has several subtasks for emotion detection, such as an emotion intensity regression task, an emotion intensity ordinal classification task, a sentiment intensity regression task, a sentiment analysis (ordinal classification) task, and an emotion classification task. We focused on an emotion intensity ordinal classification task, where contestants had to categorize a particular tweet into one of four ordinal classes of one of four given emotions that best reflected the tweeter's mental state. As for any SemEval competition, the organisers used a leaderboard, where all solutions were ranked according to some specific evaluation method, thus allowing the participating research teams to compare their solutions with those of their competitors. As the evaluation metric, we used the one suggested by organisers - Pearson Correlation Coefficient (PCC), see also Section 4.5.1 - evaluating on a set of unseen test tweets.

The winning solutions described in [93] are mainly based on DL methods. The first-placed team [41] proposed an ensemble of Random Forest and XGBoost based on embedding vectors. To make a final prediction, their system does domain adaptation for four different models to create an ensemble. The authors also included similar preprocessing as we considered in our study, particularly the transformation of emojis to their textual descriptions. For the feature extraction, they considered several options, including DeepMoji. The second-placed team [44] presented a solution with LSTM-based models and transfer learning techniques. Particularly, the authors considered as their first model a bidirectional LSTM and as their second one, an LSTM with an attention mechanism. To use transfer learning in their approach, the authors first pre-train the mentioned LSTM models using the sentiment data, and then extract a single vector out of them to use it as input to new dense layers. The third-placed team [116] used an ensemble of models with Gated Recurrent Units (GRU) and CNN. Particularly, the suggested system has three main parts: first, word embeddings, tuned for the specific task, secondly, model training (GRU with a CNN attention mechanism), and lastly, for each subtask it trains stacking-based ensembles.

2.1.2 Hate speech detection

Binary hate speech classification consists of categorizing text as hateful or non-hateful. For this task, we considered several datasets. The first one was presented by [142] in the form of the SemEval 2019 Task 6. It involves several classification problems related to offensive language detection. This competition is based on the OLID dataset from [141] that was created specifically for it. The competition has three subtasks that can be performed one by one: offensive language identification, automatic categorization of offence types, and offence target identification. In our work, we primarily focus on the first step of the binary classification subtask to decide whether a tweet is offensive or not. The authors provided an overview of the winning solutions, where they reported that 70% of the winning solutions were based on DL, where the three best ones for the binary

classification subtask [81, 97, 150] propose a fine-tuned BERT-based approach. Particularly, the first place [81] also used in their preprocessing a transformation of emojis to texts and they trained BERT-Base for three epochs. The second place [97] considered a lot of different models, including CNN, Recurrent Neural Network (RNN), and others, but the best-performing one for the binary offensive speech detection subtask was BERT-Large uncased (Base model has twelve layers in the encoder stack, when Large - twenty-four layers). For the same subtask, the third place [150] fine-tuned BERT-Base uncased model using a linear layer with a text sequence classification on top.

While SemEval 2019 Task 6 is focused on the broader topic of offensive language detection, Task 5 from the same SemEval 2019 competition, proposed by [12], considers a more specific type of hate speech. In the first subtask, contestants should classify if the text has hate speech detection against immigrants and women. In the second subtask, the target is to detect aggressive behaviour and classify the target. In this thesis, we consider the first binary classification subtask, for which the authors provided two baselines: one assigns the most frequent training label to all test instances, and the second is based on an Support Vector Machine (SVM) model with Term Frequency-Inverse Document Frequency (TF-IDF) features. In [58], the first-place team obtained the best result using an SVM model with pre-trained Universal Sentence Encoder (USE) embeddings [18]. The authors also noticed that, however, the mentioned setup was the best for the test dataset, for the development one, the most efficient was the usage of pre-trained Infersent embeddings [27] with XGBoost algorithm. While the second team did not publish their solution, the third-placed team [38] used a capsule network [117] with training stacked Bidirectional Gated Recurrent Units (BiGRUs) [24] including FastText word embeddings [63]. Specifically, text embeddings obtained with FastText were used in stacked BiGRUs models. Then, their output was considered as the input to the capsule network. Finally, the final prediction was calculated by the softmax function.

2.1.3 Irony and sarcasm detection

In order to detect irony (or sarcasm), we need to classify if the particular tweet is ironic (sarcastic) or not. For this task, we also considered a couple of datasets. Particularly, [126] released a dataset of tweets for irony detection in the framework of SemEval 2018 Task 3. Participants had to decide if a tweet is ironic or not for the first subtask, whereas, for the second subtask, they had to distinguish between non-ironic and ironic tweets – the latter of which are separated into various categories. The authors provided two baselines: one with random labels assigned and the second based on a SVM model with TF-IDF features. We participated in the binary irony classification, where the best solution provided by [134] proposed densely connected LSTM models that uses all outputs of the previous layer as inputs of the next one. The authors used different features, such as text embeddings, sentiment, and syntactic features. In contrast, the runner-up [13] used RNN-based solutions such as Bi-directional Long Short Term Memory (BiLSTM) on both word and character levels in order to get

both the syntactic and the semantic information hidden in the text. The authors also implemented a self-attention mechanism into the system to identify in the tweets the most valuable words, in this way, adding more explainability to their solution. The third place system [114] combined SVM and Logistic Regression (LR) into an ensemble with averaged tweet embeddings as features. Particularly, they considered a combination of different types of features, such as sentiment, distributional semantic, and text surface.

We also experimented with the dataset released in the framework of SemEval 2022 Task 6 called "iSarcasmEval", which considers sarcasm detection in two languages: English and Arabic [2]. It has three subtasks: in the first, participants must decide if a text is sarcastic or not; in the second, they must decide which category of ironic speech the text, if any, falls into. Lastly, they must determine which of two texts that have the same meaning but different expressions is sarcastic in the third task for a given sarcastic text and its non-sarcastic paraphrase. We tackled the first binary classification task for English. The top solutions for this subtask considered transformer-based models. The approach which ranked first [139] used an ensemble of RoBERTa [82], Decoding-enhanced BERT with disentangled attention (DeBERTa) [52], and XLMRoBERTa [28]. The authors used the external data for the model tuning, including the Irony dataset from SemEval 2018 [126] described above. Additionally, they applied several types of lexical-based and statistical features to the data, which were applied to irony detection in related works. The second place system [50] used DeBERTa and Multilingual Enhanced Representation through kNowledge IntEgration (ERNIEM) [99]. The authors finetuned those pre-trained language models to recognise sarcasm for both Arabic and English languages with multilingual settings. In the end, they applied an ensemble approach by taking the average of all test sets' outputs with optimal models. The third place system [6] applied a standalone BERT model using BERTweet checkpoints that were fine-tuned on another sarcasm-detection dataset. Particularly, the authors used the intended sarcasm subset of the SPIRS dataset [120] as the second pre-training phase after pre-training on sentiment and emotion BERTweet. They finalized the pre-trained model on the targeted Sarcasm dataset.

2.1.4 Summary

As can be concluded from this overview, the recent boost of transformers in NLP significantly impacted the solutions for the considered tasks. The majority of the systems use deep NN models or BERT transformers. However, despite their higher performance compared to more traditional feature-based methodologies, they also often remain black boxes that are unable to justify their predictions. Our proposed approach, therefore, aims to provide more explanation for the predicted labels. We do not avoid DL altogether since we still use text embedding techniques that were pre-trained using NN techniques or transformers, but as classification methods, we consider more interpretable nearest neighbour-based approaches.

2.2 Aspect-Based Sentiment Analysis

In this section, we review the general ABSA task. We describe how it evolved into its current form and briefly review the three SemEval competitions that were devoted to this task. Finally, we introduce the specific task we are working on.

2.2.1 ABSA studies summary

One of the first studies that presented a task similar to ABSA was [54] in 2004, where the authors called this task "feature-based summary" or "feature-based opinion mining". They formulated a sentiment analysis task containing three subtasks: (1) identifying the specific product features about which customers left their opinion (referred to as product features), (2) identifying review sentences that give positive or negative opinions for each feature and (3) constructing a summary using the information that has been discovered.

Some years later, [80] formulated ABSA as a task with two steps: aspect extraction and aspect sentiment classification. While initially, lexicon-based and feature-based supervised learning approaches were applied to the ABSA task, the introduction of transformer-based approaches in NLP also led to their application in ABSA. In [79], the authors presented a solution where a BERT-based architecture outperformed all previous approaches with a superficial linear classification layer. Later, in [70], the authors used adversarial training on a general BERT and a domain-specific post-trained BERT for the two ABSA tasks mentioned before, i.e., aspect extraction and aspect sentiment classification. Meanwhile, in [145], a knowledge-enabled language representation BERT-based model was introduced for the ABSA task. This model could provide explainable results by leveraging extra information obtained from a sentiment knowledge graph to navigate the input embedding of a sentence with a BERT language representation model.

Recent studies targeting ABSA are primarily based on transformer-based models [70, 145, 137]. Particularly, in [137], the authors presented a transformer-based multi-task learning framework. They called this solution the Cross-Modal Multitask Transformer, whose task is to deal with Multimodal Aspect-Based Sentiment Analysis (MABSA), where aspect-sentiment tandems were extracted from pairs of sentences and images.

2.2.2 ABSA as a SemEval task

ABSA was presented for the first time in the format of a shared task at SemEval-2014 Task 4 by [104]. The organisers presented several datasets of reviews for different business types, where aspect terms and their corresponding polarities were annotated for each sentence. They expanded their work in SemEval 2015 Task 12 [105], where all the recognised components of the expressed opinions (i.e., aspects, opinion target expressions, and sentiment polarities) were linked

within sentence-level tuples in a framework, which combined previously introduced subtasks into a single task. The organisers extended this task once again using text-level ABSA annotations in the following SemEval 2016 Task 5 [107]. They also expanded tasks to new domains and presented seven more languages besides English. After that year, there were no more ABSA-related challenges presented by this team. However, this task is still in demand nowadays. For example, at the domain `www.kaggle.com`, new datasets related to ABSA task are published each year. We decided to use in our experiments a newer labelled dataset gathered by the platform described in the following Section 2.2.3.

2.2.3 SentEMO project

The specific task that we worked on was provided by [35]. In their paper, the authors presented the SentEMO platform, a tool which performs ABSA, but also Aspect-Based Emotion Detection (ABED), after which the results are visualised by means of different dashboards. For both sentiment and emotion detection, they trained a model established on transformer-based text embeddings and SVM classifiers for six domains of the Dutch language. Moreover, the authors introduced a pipeline structure similar to the one we will design in our study, where the output of each step serves as the input for the next one. Their pipeline consists of four steps: (1) extraction of an aspect term, (2) aspect category classification, (3) polarity classification and (4) emotion classification, where each aspect is thus assigned a sentiment and an emotion. In our experiment, we will use the English version of one of the authors' datasets, described further in Section 6.1.

2.3 Interpretability in text analysis

To examine the explainability of methods available for emotion detection and ABSA tasks, as well as to compare our solution with them, in this section, we provide an overview of several methods' explainability classification criteria. We can discern two main aspects in model interpretability for text analysis [32, 48, 3]. The first one defines the "level" of explainability: local methods deliver an explanation for a single prediction, and global ones provide an explanation for the whole prediction model. For the second main aspect in model interpretability and the one we will be focusing on mostly in this section, we can define two types: post-hoc interpretation (Section 2.3.1) and self-explanatory models (Section 2.3.2). For the second approach, which is more recent and widely used, we will describe its application for emotion-detection-related tasks (Section 2.3.2). Finally, in Section 2.3.3, we discuss five main explainability techniques; and in Section 2.3.4, we will classify our solution by the interpretability type it can provide.

2.3.1 Post-hoc methods

As an example of post-hoc interpretation, we can take a look at Perturbed Masking [135], which uses MLM to calculate a particular word's impact on the prediction results for another word. The concept underlying MLM is to replace or mask part of the words in a sentence, train a model to predict the missing or replaced words based on the context that the remaining words offer, and then use that information to estimate the missing or masking words. Correspondingly, Perturbed Masking's core principle is to vary or perturb the input text by masking various groups of words and then track how the model's predictions change as a result. We may determine which words or phrases in the input text are most crucial for the model's prediction by contrasting the predictions of the original text to the predictions of the perturbed texts. In particular, a word or phrase is likely to be significant for the model's prediction if masking it results in a significant difference in prediction scores.

Yet another example of a post-hoc method is Local Interpretable Model-agnostic Explanations (LIME) [112], which aims to approximate the behaviour of the black-box model in a local region surrounding the instance in order to provide interpretable reasons for specific prediction. In order to accomplish this, LIME first chooses a "neighbourhood" around the target instance, which is a collection of related examples used to train the local interpretable model. This model is trained using a collection of created by LIME "perturbations", which are changed replicas of the instances from the neighbourhood. Then, LIME uses the black-box model's predictions as labels to train a simple interpretable model on the perturbed data. The prediction is then explained using the newly trained model and determining which features are most crucial for the output.

Another such method is SampleShapley [123], which uses primary concepts of coalitional game theory, where contributions of individual feature values explain the predictions by evaluating how much each feature contributes to the prediction of the model. To be more precise, this method accomplishes this evaluation by randomly selecting subsets of features and calculating the difference between predictions made with and without them. The relevance of each feature can be calculated by averaging over all potential feature subsets. SampleShapley can be used with any black-box model, including ones based on embeddings, where it can be employed to determine the relative weights of different embedding dimensions.

2.3.2 Self-explanatory methods

Most current explainable models in NLP belong to the self-explanatory category. For example, Variational Word Masks (VMASK) [21] make the model focus on the most important words during the prediction-making process. We can also use attention weights from the model to analyze its predictions, as was done e.g. by [138], who propose a Hierarchical Attention Network (HAN) for a document classification task. Their solution includes an attention mechanism with two levels (word and sentence level) and provides a clear visualization for the human

eye, where the most meaningful words and parts of a sentence are highlighted.

Example of usage for irony and hate speech detection

In [4], the authors aimed to develop an interpretable DL model for sarcasm detection for English social media data. They used preprocessed data and BERT embeddings in a NN based solution, including a multi-head self-attention module and GRU. The self-attention part was added for interpretability purposes, and the authors demonstrated that this module improved the results. They built an attention map that provides a picture of the per-word attention weights for the sentences. So for sentences with sarcasm, this map showed words that have more attention than others (for example, "just", "again", "totally") in order to give the researcher more insight about which words attribute a high sarcasm level to the text.

Another self-explanatory method was investigated for hate speech detection in [136]. The authors examined hate speech in social media data at the span level (a span is an ordered sequence of tokens, in other words, a slice of the text) and interpreted the model based on the assumption that the toxicity of a text corresponds to its most toxic span. They trained transformer-based models (BERT and ELECTRA [26]) for hate speech detection on two levels: the whole text and the span. By interpretability of the model, the authors mean that it should provide a set of words to justify its prediction; they also listed the twenty most frequent toxic words.

As an example of a self-explanatory solution for irony detection, [46] used an architecture similar to BERT but with in-domain embeddings. They addressed the explainability of their approach by investigating multi-head self-attention mechanisms. The authors evaluated their approach to English and Spanish irony datasets showing promising results.

2.3.3 Main explainability techniques

Besides distinguishing between "local" vs. "global" and "post-hoc" vs. "self-explanatory" explanations, [32] also identified the following five main explainability techniques:

1. Feature importance: an explanation is obtained by investigating the importance scores of features that were used to generate predictions.

2. Surrogate models: predictions of the model are explained by learning another, more explainable model.

3. Example-driven techniques: they explain the prediction for the input instance by connecting it to other labelled instances.

4. Provenance-based: we can explain some of the prediction-making process steps, which are usually represented by an intuitive approach, so the result is obtained as a series of logical transformations.

5. Declarative induction: by means of human-level readable explanations, for example, a set of rules.

Two of the most widely used techniques are the attention mechanism [9] and first-derivative saliency [78], which are part of feature importance-based explanation methods. Such approaches can consider different types of features; however, they all share the same idea, i.e., to explain model predictions by examining the importance of different features.

We can illustrate this by considering another paper for hate speech detection [86]. The authors tried to extend DL-based solutions with more understandable features and performed experiments to evaluate the influence of emotion-based features and stylometric features (representing the style of the text, for example, the unique style of a particular author). They also extracted other features (character N-grams, words, etc.) for comparison and used them in an SVM classifier. The authors compared their results with models based on CNN, LSTM, and BERT. They found that, although stylometric and emotion-based features were not the most informative ones, those features can be helpful in combination with other methods, especially in ensembles with NN models. They may provide some cross-domain rules to identify hate speech.

Another popular type is example-driven solutions. These techniques relate the input instance to other labelled examples in the dataset to explain the prediction for the target instance. As we can see from the definition, example-driven type of approaches is typical for local-level explainability models, which provide an explanation for a single prediction. Particularly, [30] provides an example of such a local self-explanatory solution based on Layer-wise Relevance Propagation (LRP). The LRP works by using a set of specifically created propagation rules to the NN to propagate the prediction backwards. Meanwhile, [62] presents a local post-hoc approach that uses explainability-aware architecture.

2.3.4 Our choice for a local, self-explaining example-driven method

Considering that we represent tweets as high-dimensional text embedding vectors, feature importance-based methods are not the best choice since the individual components of an embedding vector are difficult to interpret. By contrast, in our approach, we provide explanations for the predicted label of the test instance by looking at its neighbouring labelled instances; thus, it can be categorized as a local, self-explaining example-driven method. It gives a novelty to our approach since interpretability was not investigated much for the considered emotion-detection tasks before, nor was the usage of DL-based and BERT-based embedding techniques with the nearest neighbour fuzzy rough-based approaches.

In Section 7, we will illustrate our method as part of the error analysis process and show that it may provide a more informative picture than an attention-based map. Particularly, besides detecting the keywords that influenced the prediction-making process, we can identify topics related to a given tweet. We can also identify mistakes caused by similar tweet topics and detect confusing

tweets that affect our results. In this way, we can correct our model to obtain better results in future.

It is also worth mentioning that our best solution can provide insights into the dissimilarity between instances (Section 4.2.2). In other words, it is able to define not only how the instance is similar to every class, but also, how it differs from each of them.

Chapter 3

Data and resources

This chapter presents the theoretical background of our data preparation steps and the resources we used for this. Section 3.1 describes the different text preprocessing steps we considered. Then, Section 3.2 presents a description of text embedding methods and explains how we applied them in our pipeline. Finally, in Section 3.3, we describe one of the techniques we used to improve the quality of our embeddings using emotion lexicons.

3.1 Text cleaning techniques

Besides pure text, every tweet usually also contains some of the following information: name tags, hashtags, emojis, links, etc. Some of them can be a source of helpful information, while others should be cleaned to improve the quality of the text embedding process, described in Section 3.2.

In general, we tried three different text preprocessing approaches for each dataset and each embedding method to identify the best setups:

1. Using raw tweets without any preprocessing (Figure 3.1.a).
2. Using tweet cleaning (Figure 3.1.b): masking numbers, special symbols, links, user tags (Section 3.1.3), and deleting the "#" symbol (Section 3.1.2), as well as replacing emojis with their textual descriptions (Section 3.1.1).
3. Similar to the previous setup, but with additional stop word removal (Section 3.1.4, Figure 3.1.c).

3.1.1 Emojis

We decided to retain emojis because they may provide hints about the writer's emotional state (as illustrated by [15, 133]). We tried direct and indirect approaches to adapt the emoji Unicode format for text embedding. In the direct

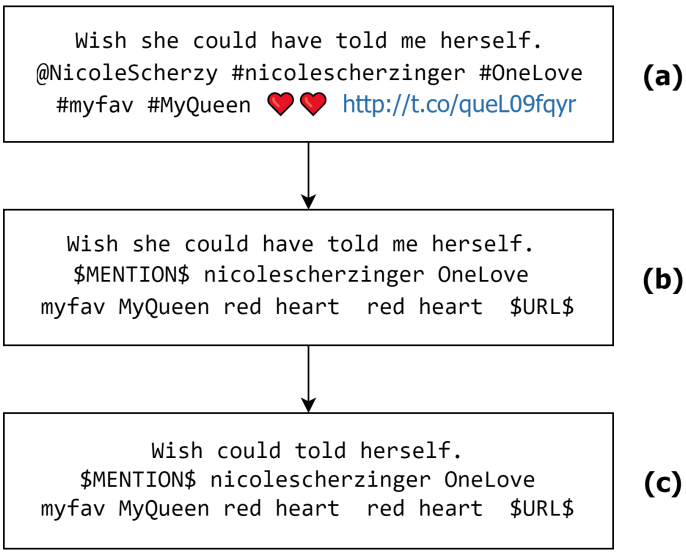


Figure 3.1: Example of the tweet preprocessing steps, for training instance from Irony dataset [126].

approach, we used an embedding method called Emoji2Vec[1], which represents a dictionary and transforms each emoji into a vector. However, this solution is limited to a particular set of emojis and does not consider the text's context. In the indirect approach, we used the Emoji package[2] which transforms each emoji to its textual description, such that it becomes part of the textual content of the tweet and is embedded with it. We performed experiments for the emotion detection dataset with the same classification setup, Cross-Validation (CV), and evaluation measurement for both emoji preprocessing methods. The results showed that the indirect approach provides higher evaluation scores, so we kept this method for our future experiments.

3.1.2 Hashtags

Another specific feature of tweets is the use of hashtags. Hashtags are an instance of textual information, but they usually contain more important keywords than the main part of the tweet. Often, hashtags are used to collect data (e.g. [93] used a list of hashtags to parse tweets and to gather emotion datasets). Initially, we tried to delete "#" symbols before hashtags and keep them as part of the text. However, since hashtags could be part of a tweet message (*"I feel so #angry, can't believe that really happened to me!"*) or be listed at the end of the tweet (*"Well, I guess my day is ruined... #sad #angst #sadness"*), we also tried

[1] `https://github.com/uclnlp/emoji2vec`
[2] `https://pypi.org/project/emoji/`

to take this into account by transforming these latter hashtags into separate sentences (*"Well, I guess my day is ruined... Sad. Angst. Sadness."*). We also considered providing more weights to hashtags, by calculating the separate embedding vector for each hashtag word and combining them with the tweet's vector. However, both described approaches did not affect our results much, so we kept the initial approach - deleting the "#" symbol and maintaining hashtags as they are.

3.1.3 Non-textual parts of text

Regarding the rest of the non-textual parts of texts, such as numbers, punctuation, special symbols (like *"/n"*), links, and user tags, we tried to delete, mask or keep them to improve our results. "Masking" means that we replace the actual number with the tag "number" in the text, the link with the tag "URL", and so on. This transforming can be done with custom functions using the Regex package[3] or using existing packages for tweet parsing, like Tweet-preprocessor[4]. It is worth mentioning that we did not consider link parsing to extract more text since a majority of links had expired and was not working.

After experiments on the different datasets, we concluded that the most effective approach is to delete non-textual tweet parts for emotion detection datasets and mask them for hate speech and irony ones (Figure 3.1). As for punctuation marks, we kept them for all cases because they could be useful during the data embedding step to save the tweet's context.

3.1.4 Stop words cleaning

A final possibility to clean tweets consists in deleting stop words from the main text of the tweet. Stop words are frequently used words that do not have any important meaning (*"the", "a", "any", etc.*). A list of such words for the English language is provided in the NLTK package[5]. As we will show in the experimental part, this solution proves useful for some embedding techniques but is mostly inferior when the embedding algorithm considers the whole context of the text.

3.2 Text vectorization with embedding methods

To use tweets in classification methods, we shall represent them as vectors in an N-dimensional space, so tweets with similar meanings will be represented by neighbouring vectors. This is achieved by embedding algorithms: they transform the text into a numerical shape while maintaining their similarities. Tweet embeddings operate on different text levels - at the level of individual symbols, words, collocations, sentences, paragraphs, or the whole text. Also, the method itself can take various shapes, ranging from simple dictionaries to context-based

[3] `https://docs.python.org/3/library/re.html`
[4] `https://pypi.org/project/tweet-preprocessor/`
[5] `https://www.nltk.org/`

language models. In this work, we will consider different types of text embedding techniques. Before we will dive into each of them, we will discuss the basic method for representing text in a numerical format, which we used as a baseline.

3.2.1 Bag of Words and N-grams

Bag of Words (BOW) is the name of the fundamental approach for the transformation of textual data into numerical format. It disregards grammar and word order but takes into account multiplicity, representing text as a collection (or bag) of words. BOW is typically used to generate features based on a text corpus. Although it is a straightforward method, BOW has one significant drawback: when encoding, it ignores word order and semantics.

To improve the basic approach of BOW, instead of each word separately, we can consider N-grams. An N-gram represents a connecting sequence of N elements, which are usually words, as we will consider for our experiments as well. In this way, with a Bag of N-grams, we pay more attention to collocations since they can contain more context than a single word. However, this approach creates a much bigger dictionary than BOW, which takes a lot of memory and creates long sparse vectors (the ones that have a relatively small number of non-zero elements). We will elaborate on those ideas with our experiments in Chapter 5.

3.2.2 Word2Vec and its variants

The first and earliest type of DL-based embedding method is Word2Vec, initially presented in [89] and [88]. Word2Vec has two architecture options: Continuous Bag of Words (CBOW) and Skip-gram. The CBOW model uses context representations to predict a missing word, whereas the Skip-gram model uses a representation of the word to predict the context. After model training, specific weights for every word are extracted as embedding vectors. As such, the Word2Vec model used in this work represents each word as a 300-dimensional vector. This dimensionality was used as the standard one for the Word2Vec model presented in the Gensim package[6]. It provides a Word2Vec model pre-trained on a Google News dataset containing nearly 100 billion words. Word2Vec has the form of a dictionary of almost 3 million words and phrases and cannot be fine-tuned.

After the Word2Vec papers were published, many similar approaches appeared, for example, the earlier mentioned Emoji2Vec and the Doc2Vec[7] packages. The last one works similarly to Word2Vec, but for whole text paragraphs rather than individual words. We also tried this approach but got very unsatisfying results for our datasets. Therefore, we will only use the Gensim pre-trained Word2Vec model. We take the mean of all its words' vectors to obtain a vector for a whole tweet.

[6] `https://radimrehurek.com/gensim/models/word2vec.html`
[7] `https://radimrehurek.com/gensim/models/doc2vec.html`

3.2.3 DeepMoji

Embedding methods can also take into account the sentiment hidden in a sentence. For example, the DeepMoji model presented in [43] is an LSTM-based model trained on over one million tweets containing one out of sixty-four different emojis with the purpose of recognizing emotions in text. DeepMoji, unlike Word2Vec, provides a vector representation for the full tweet. For our experiments, we used the PyTorch implementation of the DeepMoji model provided by Huggingface[8], which encodes text as a 2,304-dimensional vector.

3.2.4 Universal Sentence Encoder

We also used more advanced recent methods, for example, the USE developed by TensorFlow[9] and described in [18]. This model was pre-trained on various datasets for different tasks, from text classification to sentence similarity. USE works at the level of paragraphs and provides a 512-dimensional (standard size of the pre-trained USE embedding) vector representation for the full tweet. USE has two available architectures; one is based on a Deep Averaging Network (DAN), and the second uses a transformer encoder. After experimenting on the emotion detection dataset with the same classification method, text preprocessing steps and evaluation, we chose the second option as it gave better results.

3.2.5 Transformer-based encoders

The transformer encoders represent the current state-of-the-art approach in the NLP area. The very first such model was BERT, introduced in [37]. It was developed by the Google AI Language Team with the idea of pre-training deep bidirectional representations based on unlabelled text. BERT was pre-trained on datasets for two tasks: language modelling and next-sentence prediction. However, this model can be easily fine-tuned for different tasks with extra output layers without architecture modifications. We used the PyTorch[10] BERT realisation to extract 768-dimensional vectors for each subword identified by the model and again used the mean to obtain the whole tweet's vector. The size of 768 dimensions is standard for the base-BERT model; however, it increases for larger models.

We also considered other models based on BERT, for example, Sentence-Bidirectional Encoder Representations from Transformers (SBERT) by [111], which is a variation of the original BERT, tuned for sentence-level vector extraction. SBERT was trained on a collection of sentence pairs, and it can process two tweets simultaneously because it uses twin network structures, providing a 768-dimensional vector for the whole tweet.

[8] https://github.com/huggingface/torchMoji

[9] https://www.tensorflow.org/hub/tutorials/semantic_similarity_with_tf_hub_universal_encoder

[10] https://pytorch.org/

Another model is the Twitter-RoBERTa-based model by [11]. The authors presented several models for different tasks: emotion detection, hate speech, irony classification, and more. These models are fine-tuned on similar SemEval data versions of the RoBERTa, analogous to the BERT model, but with some minor changes in the training procedure and architecture.

For the ABSA task, we also considered several BERT-based models, including the BERT[11] and ALBERT[12] models, fine-tuned using the Text-Attack (TA) package [94] and the YELP polarity dataset [144] for the sequence classification task. The original ALBERT model was presented by [74] and consumes less time and memory for training compared to the regular BERT model. The authors also showed that this model has 89% parameters reduction, in contrast to BERT, with equal performance. However, the best results for our ABSA experiments were obtained with the DistilBERT Yelp Review Sentiment model[13], which was fine-tuned on one million reviews from the YELP dataset [144] for the sentiment analysis task. The DistilBERT architecture, initially presented in [119], is a lighter and faster version of BERT that takes less time to fine-tune. The mentioned BERT-based models were fine-tuned on the YELP dataset that contains user reviews from `yelp.com`. We chose such models due to the similar nature of the data used for their fine-tuning and ours. Although these models were used as classification/regression models for the sequence classification and sentiment analysis tasks, we will use them as embedding models by extracting the encoded vector representations of text.

In general, all described transformers-based models can perform classification directly. For example, consider text classification with the BERT model. In the first step, it tokenizes input text as subword tokens and then converts them into embedding representations. Secondly, the BERT model is pre-trained on unlabeled data, from which it learns contextual word representations using MLM. The following step of fine-tuning includes an adaptation of the BERT model to specific tasks. It is performed by replacing the model's last layer with a classification head. Next, during the training of the model, it uses labelled data to update weights. Meanwhile, for the inference step, the input text goes through the model, so it computes class probabilities, which are based on the task-specific classification layer. Finally, by aggregation of information from the specific [CLS] token (or by the usage of other pooling strategies), the bidirectional attention of BERT enables it to provide high results for classification tasks.

In our experiments, we used transformer-based models to extract texts' embedding vectors. They were calculated as weights from the inner model's layer.

3.2.6 Other embedding methods

Besides the previously described methods, we also investigated several other embedding approaches that did not perform well and therefore were not included

[11] `https://huggingface.co/textattack/bert-base-uncased-yelp-polarity`
[12] `https://huggingface.co/textattack/albert-base-v2-yelp-polarity`
[13] `https://huggingface.co/spentaur/yelp`

in our main experiments. Mainly, we tried FastText pre-trained word vectors by [87], which is a more advanced Word2Vec solution based on CBOW, spaCy embedding models[14] by [53] which are based on CNN models and available for English in two sizes (small and large, both considered for our experiments), and BERTweet by [96], which is a BERT-based model with a RoBERTa pre-training procedure executed on 850 million English tweets.

3.3 Lexicons

Finally, we also investigated the idea of using emotion lexicons to improve the quality of the obtained embedding vectors and add more emotion-related information. An emotion lexicon is a word vocabulary where each word is assigned an emotional intensity score. In our experiments, we used the following English lexicons:

- Valence Arousal Dominance (NRC VAD) lexicon (20,007 words) [90] – each word has a score (float number between 0 and 1) for Valence, Arousal, and Dominance.
- Emotional Lexicon (EMOLEX) (14,182 words) lexicon [91] – each word has ten scores (0 or 1), one per emotion: anger, anticipation, disgust, fear, joy, negative, positive, sadness, surprise, and trust.
- Affect Intensity (AI) lexicon (nearly 6,000 terms) [92] – each word has four scores (float number from 0 to 1), one per emotion: anger, fear, sadness, and joy.
- Affective norms for English words (ANEW) lexicon (1034 words) [16] – each word has six scores (float number between 0 and 10): Mean and SD for Valence, Arousal and Dominance.
- Warriner's lexicon (13,915 lemmas) [131] – each word has 63 scores (float number between 0 and 1000), reflecting different statistical characteristics of Valence, Arousal, and Dominance.

We considered the following two methods of combining word embeddings with lexicon vocabulary scores:

- For each word during the embedding process, lexicon scores were appended to the end of the tweet vector. The size of the obtained vector is the word embedding size plus the number of lexicon scores.
- We construct a separate feature for each lexicon. These models are then combined with the embedding vectors in an ensemble classifier, as described in Section 4.4.

[14] `https://spacy.io/models`

As will be shown further in Chapter 5, lexicons did not significantly improve the prediction results. For this reason, we performed experiments for only the emotion datasets (Anger, Joy, Sadness, Fear) with one or several lexicons for each.

Chapter 4

Prediction methods

In this chapter, we discuss all methods that we use for label prediction. Firstly, in Section 4.1, we describe which similarity relation is used for our methods. Then, Sections 4.2 and 4.3 provide an overview of the corresponding classification and regression models we used. In Section 4.4, we discuss an ensemble approach that we applied to the prediction models. Finally, Section 4.5 considers the evaluation metrics that we used to compare the obtained results.

4.1 Similarity relation

Before describing the classification methods we researched, it is important to discuss the similarity metric we used to measure how similar the considered vectors (in other words, tweet embeddings) are.

In [55], the authors compared different metrics for related NLP tasks: Euclidean distance, Jaccard coefficient, PCC, averaged Kullback-Leibler divergence, and cosine similarity. In the end, Euclidean distance performed the worst, whereas cosine and Jaccard coefficient were the most efficient ones. Cosine similarity also showed the best results for the imbalanced datasets.

Cosine similarity also has an additional interesting characteristic for our setup. Particularly, the cosine applies to the angle between two embedding vectors, which is advantageous for textual data since it considers the orientation of the vectors rather than magnitudes. It allows the comparison of texts based on their semantic similarity regardless of length.

Based on these arguments, we chose the cosine metric for our experiments. The cosine distance can be defined by Eq.(4.1):

$$\cos(A, B) = \frac{A \cdot B}{||A|| \times ||B||} \tag{4.1}$$

In Eq.(4.1), A and B correspond to elements from the same vector space (text embeddings), $A \cdot B$ denotes their scalar product, and $||x||$ is the vector norm of x.

To fit NN-based methods, we need a similarity relation that provides values between 0 (vectors are totally different) and 1 (vectors are identical). In contrast, the cosine distance's outputs are between -1 (perfectly dissimilar vectors) and 1 (perfectly similar vectors). Hence, we apply the following transformation to make similarity out of distance:

$$cos\_similarity(A, B) = \frac{1 + cos(A, B)}{2}. \tag{4.2}$$

Finally, we should note that we also examined Hausdorff's distance [51, 39], which measures how far two subsets are from each other in a metric space. We used it for comparing texts, considering that each text can be seen as a set of individual words. Hausdorff's metric is a meta-metric based on another distance metric that, in our case, was cosine. We decided to investigate this metric since it differs from cosine similarity and others suggested in [55] by using sets of vectors. However, this approach did not provide satisfactory results, so we dismissed it.

4.2 Classification models

This section presents the nearest neighbour-based classification methods that we investigated for our experiments.

4.2.1 Weighted kNN

The weighted k Nearest Neighbours (wkNN) classification method [40] is a refinement of the regular kNN [29].

The kNN algorithm is a commonly used non-parametric classification and regression technique for ML tasks. It can be applied to both supervised and unsupervised learning problems. kNN is a lazy learning algorithm, which means that during the training phase, no model is built; instead, the full training dataset is stored in memory. This method categorizes new instances based on their proximity to the k nearest neighbours from the training dataset. A new instance's class label is chosen by a majority vote or by averaging the labels of its k closest neighbours.

The primary distinction between wkNN and kNN is how neighbours are taken into account while making predictions. Each of the k nearest neighbours participates equally in the decision-making process in the conventional kNN algorithm. However, with wkNN, each neighbour's impact is assessed by allocating weights based on how close they are to the query instance. For this method, weights are typically determined based on distance, with closer neighbours having higher weights and having a greater impact on the prediction. In other words, the wkNN approach aims to assign a more significant weight to the closest instances and a smaller weight to the ones that are further away. The class label of the new test instance is decided by a weighted total of the labels of the k nearest neighbours rather than by a simple majority vote or average.

wkNN has two main parameters: the used metric or similarity relation and the number k of considered neighbours. The choice of distance metric has a big influence on measuring the similarity between instances. For our experiments, we use the previously described cosine similarity as a relation function.

Regarding the parameter k, it determines the number of neighbours to take into account while classifying a new instance. In order to prevent either overfitting or underfitting, it must be properly chosen. However, there is no one-fits-all rule to determine it. To examine the impact of k, we will use various numbers of neighbours for each emotion dataset for the best-performing methods in our experiments.

We use wKNN both as a standalone method as well as inside of an ensemble of methods (Section 4.4).

4.2.2 FRNN-OWA

Our work considers methods based on FRS theory, mainly the Fuzzy-Rough Nearest Neighbour (FRNN) classification model proposed in [61].

FRNN is an instance-based algorithm that performs classification using lower (L) and upper (U) approximations from FRS theory. The authors of [61] showed that the method outperformed classical NN approaches and that it is competitive with the leading classification algorithms.

To make the approach more robust and noise-tolerant, [129] proposed an FRNN extension with Ordered Weighted Average (OWA) operators. The authors used OWA operators to determine membership to the lower and upper approximation by means of an aggregation process. In [76], the authors presented an approximate FRNN-OWA solution, which modifies the approximations' calculation process. They managed to keep the accuracy of the original approach while improving the execution speed. They also developed a Python package[1] for these methods in [75], which will be used in our experiments.

We will denote the OWA aggregation of value set V ($v_{(i)}$ is the i^{th} largest element in V) with weight vector $\overrightarrow{W} = \langle w_1, w_2, ..., w_{|V|} \rangle$, where $(\forall i)(w_i \in [0,1])$ and $\sum_{i=1}^{|V|} w_i = 1$, with:

$$OWA_{\overrightarrow{W}}(V) = \sum_{i=1}^{|V|} (w_i v_{(i)}) \tag{4.3}$$

We experimented with several types of OWA operators and concluded that the additive weight type [129] clearly performed best for our data.

Additive weights are linearly increasing for lower approximation (Formula (4.4), where p ($p > 1$) denotes the vector's length) and decreasing for upper approximation (Formula (4.5)).

$$\overrightarrow{W}_L^{add} = \langle \frac{2}{p(p+1)}, \frac{4}{p(p+1)}, ..., \frac{2(p-1)}{p(p+1)}, \frac{2}{p+1} \rangle \tag{4.4}$$

[1] `https://github.com/oulenz/fuzzy-rough-learn`

$$\overrightarrow{W}_{U}^{add} = \langle \frac{2}{p+1}, \frac{2(p-1)}{p(p+1)}, ..., \frac{4}{p(p+1)}, \frac{2}{p(p+1)} \rangle \tag{4.5}$$

Next, during the classification of a test vector y, the FRNN-OWA method calculates the membership degree of y to the lower (Formula (4.6)) and upper (Formula (4.7)) approximation of each decision class C. In those formulas, by $R(x, y)$, we refer to the similarity relation, which in our case is cosine similarity described in Section 4.1.

$$\underline{C}(y) = OWA_{\overrightarrow{W}_{L}^{add}}\{1 - R(x, y) \mid x \in X \setminus C\}) \tag{4.6}$$

$$\overline{C}(y) = OWA_{\overrightarrow{W}_{U}^{add}}\{R(x, y) \mid x \in C\}) \tag{4.7}$$

Then, the method assigns y to the class C that has the highest sum $\underline{C}(y) + \overline{C}(y)$. For efficiency purposes, calculations are limited by a parameter k (amount of nearest neighbours of test instance y used to construct the approximations). In Eq. (4.6), k refers to the number of neighbours of y from classes other than C and in Eq. (4.7) to the number of neighbours of y from C. There are no general rules on how to set k. Hence, we will tune this parameter for each dataset in our experiments.

After deeper inspection of both methods, we can highlight an important difference between wkNN and FRNN, which motivated us to consider the latter one for the NLP-related tasks. In general, while the wkNN method can only introduce the concept of "similarity", FRNN also provides a "dissimilarity" relation. In this way, we can define how a particular instance is similar to all suggested classes and, at the same time, how it differs from each of them.

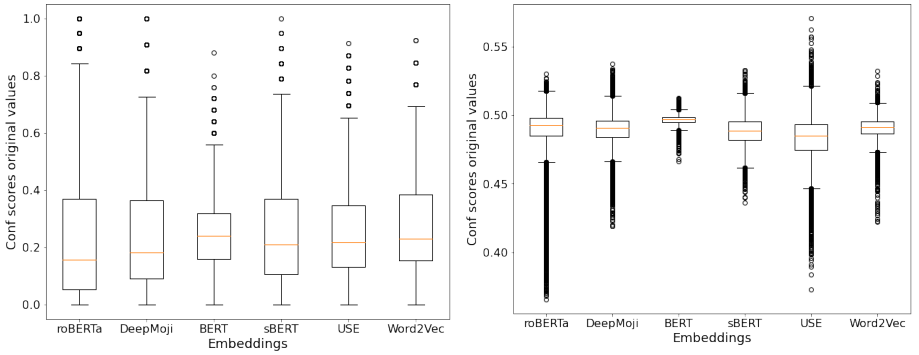
Confidence scores

FRNN-OWA also provides a natural way to derive confidence scores for its predictions. In particular, for each class, C and test instance y, the confidence score can be calculated as

$$Conf(C, y) = \frac{\underline{C}(y) + \overline{C}(y)}{\sum_{C' \in \mathcal{C}} \underline{C'}(y) + \overline{C'}(y)}, \tag{4.8}$$

where $\mathcal{C}$ represents the set of all classes.

An analysis of the confidence scores we obtained in our experiments (Section 5) revealed that they are often close to each other (Figure 4.1). Particularly, when we apply the wkNN model with different embedding methods to the same dataset, we can see that all scores are between 0.1 and 0.4, with a median of 0.2 (Figure 4.1a). Meanwhile, for the FRNN-OWA method, confidence scores have an even smaller range from roughly 0.47 to 0.51 with a median of around 0.49. The BERT embedding provides the smallest range of confidence scores for both classification models (Figure 4.1b). In general, we hypothesize that this



(a) For wkNN.

(b) For FRNN-OWA.

Figure 4.1: Confidence scores sensitivity (before rescaling) for two classification methods and different embeddings on the Anger dataset.

may be due to the high dimensionality of text embeddings, causing the upper approximation memberships to be close to 1 and the lower ones to 0.

To mend the described issue, we propose rescaling the original confidence scores in order to amplify their differences. To use these scores in ensembles (Section 4.4), we proceed as follows. Denote by $Conf_i(C_j, y)$ the confidence score of the i-th member of the ensemble for test instance y to belong to class C_j, calculated as in Eq. (4.8). We subtract 0.5 from $Conf_i(C_j, y)$ and divide the result by a small value α ($0 < \alpha < 1$). Next, we compute the sum of the scores for each class. Since the obtained values may be negative, we use the softmax transformation to turn them into values between 0 and 1. The steps of this rescaling process are summarized as follows:

$$w_i = \frac{\exp(\sum_j (Conf_i(C_j, y) - 0.5)/\alpha)}{\sum_k \exp(\sum_j (Conf_k(C_j, y) - 0.5)/\alpha)}, \tag{4.9}$$

4.2.3 FROVOCO

Fuzzy Rough One-Versus-One COmbination (FROVOCO) is also an instance-based algorithm, introduced in [128]. FROVOCO is developed for multi-class classification and especially for imbalanced tasks, which could be suitable for our purposes given the imbalance present in our data. It decomposes several classes into separate One-Vs.-One (OVO) and One-Vs.-All (OVA) tasks. In our experiments, we used a weighted average of three components: OVA (as was performed in FRNN, where it represents a mean of lower and upper components), OVO (in this case, we perform FRNN for each pair of classes and then aggregate results for each class), and negative affinity (it compares whether the target instance is similar to the other classes in the same way as the predicted class).

FROVOCO considers the constructed classes' Imbalance Ratio (IR) for

each pair to use specific weights. To determine these weights, FROVOCO uses the Imbalanced Fuzzy RoughOrdered Weighted Average Nearest Neighbour (IFROWANN) scheme described in [110]. This technique computes the approximations by considering all samples belonging to the opposite class, not only the closest ones, and by giving each sample a decreasing weight based on how similar it is to the test sample. IFROWANN consists of two steps, the first of which involves comparing the weight vectors for the majority and minority classes while taking into consideration the fact that the former includes far fewer items than the latter. In the second stage, we use the OWA model to combine the contributions of the training data. Such a solution allows us to work with imbalanced datasets.

For FROVOCO, we also used the cosine similarity metric and additive weights. Additive weights represent a vector with a length corresponding to the number of neighbours with linearly increasing and decreasing weights, Formulas (4.4) and (4.5). We used the FROVOCO implementation from [75].

4.3 Regression models

As an alternative to classification approaches for the emotion detection issue, which can be considered as an ordinal classification task, we also investigated FRNN regression [61].

For the test instance y, this algorithm predicts a value based on the classes of the k nearest neighbours of instance y, similar to kNN regression. The main feature of FRNN regression is that it calculates the output for y as a weighted mean, where the weights are represented with the upper and lower approximation membership degrees of the k neighbours' output values. We fine-tuned the parameter k for each emotion dataset in our experiments.

The FRNN regression algorithm returns a float number for each test instance. We use standard rounding for each output value to adjust this algorithm to obtain a class prediction for our classification task.

4.4 Ensembles

Apart from performing our experiments for standalone classification methods, we will also consider their combination in an ensemble, where each model will be based on a separate embedding method. We followed standard practices of combining different classifiers (in our case, they are based on different embedding methods) into an ensemble. The motivation behind this is to combine the advantages of different embedding methods in one approach.

To this aim, we tune parameters for each combination of a dataset and an embedding method to identify optimal setups. To obtain the final output, a voting function is needed. After experiments with various options (median, majority, maximum, etc.), the best-performing voting function was the average. For FRNN-OWA, we also used a weighted average, where the found to be weights

are derived from the confidence scores that each model generates. Meanwhile, we used the classical average as a voting function for FRNN regression, which does not generate any confidence scores.

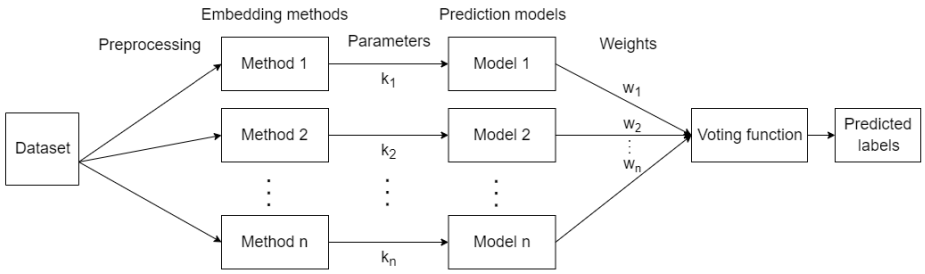


Figure 4.2: Scheme of our approach with an ensemble of prediction models.

Noteworthy is that the mean function can provide a float value between 0 and 3 instead of the required intensity labels 0, 1, 2, or 3. This was not a problem during training because our labels are not different classes but ordinal intensity labels. For the test datasets, the obtained values were rounded to submit our predictions.

The full architecture of our approach is shown in Fig. 4.2, where n is the number of embedding methods, k_i is the number of neighbours (parameter k) for the i-th ensemble model, and w_i is the weight for the i-th model's output in the voting function.

4.5 Evaluation

To measure and compare the obtained prediction results, we first applied them to the training data using 5-fold CV. This technique divides the training dataset into five subsets, utilizing four for training and one for testing in each of its five iterations to ensure a more robust evaluation. For datasets where both training and development datasets were provided, we merged them. Also, to allow repeatability of our results, we fixed the random seed value as 3 (for the function that generates random instances' index numbers to create a training dataset for each fold of CV). We also noticed that differences between folds are minor for all datasets and methods, so we will not discuss it further.

We used several metrics for the evaluation of the results. Some of them were proposed by the organisers of the respective SemEval competitions, whereas others were added to obtain a more complete picture of our methods' performance.

4.5.1 Pearson Correlation Coefficient

PCC was used in the SemEval competition on emotion detection. Let y be the vector of predicted values and x that of correct values, with x_i and y_i as the i^{th}

elements of x and y, and denote their means by $\bar{x}$ and $\bar{y}$. The PCC is given by Formula (4.10).

$$PCC = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}. \tag{4.10}$$

PCC scores vary between -1 (a total negative linear correlation) and 1 (a total positive linear correlation), where 0 corresponds to no linear correlation. As a consequence, during model comparison, we will seek the model with the highest PCC.

4.5.2 Mean Absolute Error

An additional metric that we consider for the emotion detection task is Mean Absolute Error (MAE) (4.11). This choice of metric is inspired by the fact that for the emotion intensity task, we are dealing with an ordinal classification. The MAE formula is:

$$MAE = \frac{\sum_i |y_i - \bar{x_i}|}{n}, \tag{4.11}$$

where n is the size of vectors x and y. Lower MAE values mean better prediction.

4.5.3 F1-score

Another metric is the F1-score (4.12) that was used for the binary classification tasks. The formula for the F1-score is as follows:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}, \tag{4.12}$$

where *Precision* is the fraction of correctly predicted instances out of all predicted labels, and *Recall* is the fraction of correctly predicted labels out of all ground-truth labels. For the F1-score, we are looking for the highest value to choose the best model.

For the Offensive Language, Hate Speech, and Irony detection datasets, we used the macro-averaged F1-score, which is calculated by averaging the F1-scores for each class. By doing this, it is ensured that the classifier's performance for each class contributes equally to the final result. Meanwhile, for the Sarcasm dataset, we considered the F1-score calculated for the sarcastic class. These metrics were used by corresponding SemEval competitions to compare participants' results.

For the ABSA task experiments, we calculated the weighted F1-score. It computes F1-score for each class, but unlike the macro-averaged version, the weighted F1-score assign bigger weights to the classes with a larger number of instances. In other words, the weighted F1-score represents the average of F1-scores for all classes, where each such score is weighted by the number of samples in the corresponding class. We used F1-score from Scikit-Learn library[2].

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`

4.5.4 Accuracy and CCA

Another evaluation metric that we use in our ABSA experiments is accuracy. It corresponds to the number of correctly predicted instances out of all instances in the dataset. It is not the most suitable metric for imbalanced datasets. Still, in our case, it reflects how many data instances are actually left after all the steps of the considered pipelines, described further in Section 6.3.

As an alternative to accuracy, we also calculated the Cost Corrected Accuracy (CCA) metric, introduced by [33]. This metric applies to ordinal classification tasks only and is similar to accuracy. However, CCA takes into account the cost of prediction. In other words, for ordinal classes, the prediction of a class close to the actual one (i.e., to the gold label) should have a lower cost than a prediction of a class which is further removed. As a simple example, we can consider a polarity classification task with ordered labels "positive", "neutral", and "negative". Then, we can set up costs in a way that, for instance, with a gold label "positive", a correctly predicted label will have a cost of 0; a prediction of "neutral" has an associated cost of 1/2, while predicting the "negative" label will correspond to the full cost of 1. Based on this information, we can construct a cost matrix, which is symmetrical with classes on the rows and columns, 0 on the diagonal and the cost of each "prediction mistake" on other positions.

Then, having the confusion matrix Cf (a performance evaluation tool that displays the counts of true positive, true negative, false positive, and false negative predictions of the model while summarizing the classification results) and a cost matrix Ct, we can calculate CCA with Formula (4.13):

$$CCA = 1 - Cf \cdot Ct \tag{4.13}$$

In this way, higher CCA scores will correspond to better methods in the same way as accuracy, which can be observed as its special case with the cost of each prediction mistake equal to 1.

In our experiments, we can apply CCA to emotion detection tasks because it is an ordinal classification. All four emotion detection datasets from [93] have the same emotion intensity classes, for which we suggested our option for the cost matrices. It is provided in Table 4.1, where in labels by "emotion" we mean one of the following: anger, joy, sadness, and fear - for each one out of four corresponding datasets, where rows and columns decode the same emotion intensity levels.

Table 4.1: Emotion intensity cost matrix for the emotion classification task [93].

	0	**1**	**2**	**3**
0: no emotion can be inferred	0	0.33	0.66	1
1: low amount of emotion can be inferred	0.33	0	0.33	0.66
2: moderate amount of emotion can be inferred	0.66	0.33	0	0.33
3: high amount of emotion can be inferred	1	0.66	0.33	0

We can also use the CCA score for the ABSA task, but only for the sentiment and emotion classification since, for aspects, we do not have ordered classes. For both those tasks, we provided the cost matrices based on the ones suggested in [35] for the same tasks with the Dutch version of the considered dataset. We present our cost matrices for sentiment and emotion classification in Table 4.2 (with "very_neg" corresponding to "very_negative" class and "very_pos" - to "very_positive") and Table 4.3, respectively.

Table 4.2: Sentiment cost matrix for the ABSA task.

	Negative	**Neutral**	**Positive**	**Very_Neg**	**Very_Pos**
Negative	0	0.5	1	0.5	1
Neutral	0.5	0	0.5	0.5	0.5
Positive	1	0.5	0	1	0.5
Very_Neg	0.5	0.5	1	0	1
Very_Pos	1	0.5	0.5	1	0

Table 4.3: Emotion cost matrix for the ABSA task.

	Anger	**Anticipation**	**Disgust**	**Dissatisfaction**	**Distrust**	**Fear**	**Joy**	**Neutral**	**Sadness**	**Satisfaction**	**Surprise**	**Trust**
Anger	0	1	0.25	0.5	0.25	0.25	1	0.75	0.25	1	0.25	1
Anticipation	1	0	1	1	1	1	0.25	0.75	1	0.5	0.25	0.25
Disgust	0.25	1	0	0.5	0.25	0.25	1	0.75	0.25	1	0.25	1
Dissatisfaction	0.5	1	0.5	0	0.5	0.5	1	0.5	0.5	1	0.25	1
Distrust	0.25	1	0.25	0.5	0	0.25	1	0.75	0.25	1	0.25	1
Fear	0.25	1	0.25	0.5	0.25	0	1	0.75	0.25	1	0.25	1
Joy	1	0.25	1	1	1	1	0	0.75	1	0.5	0.25	0.25
Neutral	0.75	0.75	0.75	0.5	0.75	0.75	0.75	0	0.75	0.5	0.75	0.75
Sadness	0.25	1	0.25	0.5	0.25	0.25	1	0.75	0	1	0.25	1
Satisfaction	1	0.5	1	1	1	1	0.5	0.5	1	0	0.25	0.5
Surprise	0.25	0.25	0.25	0.5	0.25	0.25	0.25	0.75	0.25	0.5	0	0.25
Trust	1	0.25	1	1	1	1	0.25	0.75	1	0.5	0.25	0

Chapter 5

Application 1: emotion recognition

In this chapter, we present the results of the methods described in Chapter 4 on the datasets mentioned in Section 2.1. Section 5.1 considers the task of recognising emotion intensity, which we split into three parts. Firstly, we detect the best setup for each dataset and each embedding method on the training dataset. Secondly, we tune ensembles of models to obtain the most efficient approach per dataset. Finally, we apply the best setup on the test datasets. In Section 5.2, we perform the same steps on binary classification tasks - hate speech, irony, and sarcasm detection.

5.1 Emotion intensity detection

In this section, we provide more details about the structure of the emotion detection datasets and the outcomes of our experiments. Particularly, we show results for several embedding methods and three prediction models. For each model, we tune the parameters to detect the best setup for each dataset. In the end, we compare all considered models and define the most suitable one based on test datasets.

5.1.1 Datasets and task

As was mentioned in Section 2.1.1, we used SemEval 2018 Task 1: Affect in Tweets for English by [93] as an example of an emotion detection task. The data for this competition was collected using the Twitter API with a vocabulary of keyword hashtags related to different emotions (for example, “annoyed”, “panic”, “happy”, etc.).

We considered subtask EI-oc, which represents an emotion intensity ordinal classification problem for English. Tweets for this task are labelled with four

emotion intensity scores for four different emotions: anger, sadness, joy, and fear. Each intensity score (or simply a class) represents a level of emotional intensity: 0 stands for "no emotion can be inferred", 1 corresponds to "low amount of emotion can be inferred", 2 means "moderate amount of emotion can be inferred", and 3 - "high amount of emotion can be inferred". For each emotion, the training, development, and test datasets were provided in the framework of the SemEval 2018 competition. As was mentioned in Section 4.5, we merged the training and development datasets to train our model with 5-fold CV.

We also explored some characteristics of the datasets and presented them in Table 5.1. One of the characteristics is the class imbalance. It is quantified by the IR, which is equal to the ratio of the sizes of the largest and the smallest classes in the dataset.

Table 5.1: Characteristics of the training data for the four emotion datasets.

Characteristic	**Anger**	**Joy**	**Sadness**	**Fear**
IR	1.677	1.47	2.2	8.04
Total number of instances	2,089	1,906	1,930	2,641

Table 5.1 shows that the Fear dataset is the biggest and the most imbalanced. Meanwhile, the three other datasets (Anger, Joy, and Sadness) have similar sizes and IR scores, with Joy being the most balanced dataset among them.

5.1.2 The baseline

Before proceeding with more advanced embedding methods and FRNN-based methods, we first conducted baseline experiments in which we used BOW and Bag of N-grams with weighted kNN for each dataset.

From our experiments with tuning such a baseline, we can draw several conclusions. First of all, as we discussed in Section 3.2.1, for each dataset, we obtain a huge corpus of unique words and unique N-grams, which leads to a sparse embedding vector representation for each instance, takes a lot of the system's memory, and slows down the experiments. As we saw from our calculation, there is no sense to go deeper than 3-grams, which already provides pretty low scores. Hence, we will consider 1-gram (all unique words), 2-grams (unique collocation of two words), and 3-grams (unique collocation of three words).

Another observation is that the considered setups provided such low scores, that fine-tuning of the text preprocessing technique or the number of neighbours k did not change the situation. Because of that, we will provide scores for the same setup for each dataset, where the number of neighbours k is equal to 23, and the standard text cleaning (Section 3.1) with lower-casing was performed. Our results for the emotion datasets are presented in Table 5.2. To calculate the Bag of N-grams, we used the function CountVectorizer from the Scikit-Learn[1].

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`

Table 5.2: PCC scores for a Bag of N-grams (with N=1,2,3) and the wkNN model applied to the four emotion datasets.

N-gram size	Anger	Joy	Sadness	Fear
1	0.0230	0.1826	0.1482	0.1258
2	-0.0106	-0.0187	0.0677	0.0677
3	-0.0091	-0.0047	0.0239	0.0239

As we can see from Table 5.2, such approach provides very weak results for each dataset. As a pattern for all of them, we can notice that the 1-gram solution, in other words, BOW, performed better than any Bag of N-grams. Hence, we will consider it as our baseline.

5.1.3 Model tuning

First, we select the best text preprocessing approach for each dataset-embedding combination by performing 5-fold CV on the training data with the same k values for each preprocessing option (raw, cleaned, and without stop words), using the PCC metric. We also investigate how lexicons perform on their own and in combination with embedding models. As RoBERTa-based model, we used Twitter-RoBERTa-base for Emotion Recognition[2] (further "RoBERTa") by [11].

Before we dive into each model, let us take a look at the time performance for each embedding method. As an example, we take the Anger dataset and apply each embedding approach to its original tweets without spending time on preprocessing. As we mentioned in Table 5.1, the size of the Anger dataset is 2,089 instances. In Table 5.3, we show how much time in seconds it takes for each embedding method to encode this amount of tweets. We also provide the average time performance for one tweet and the size of the produced embedding vector for each instance. We ran our experiments in Jupyter Notebook with the Python 3 environment. For time measurements, we used the *time*[3] Python package.

In Table 5.3, we listed the embedding approaches by the size of the output vector for one tweet, from the biggest to the smallest one. As we can see, the three BERT-based methods (RoBERTa, BERT-Base, and SBERT) have the same output size, with DeepMoji being the biggest and Word2Vec the smallest. Meanwhile, Word2Vec is also the fastest embedding approach; however, we cannot see the same pattern of time-size connection for other methods. Particularly, BERT-based methods have quite different speeds, while DeepMoji has a vector size three times bigger and is much faster than any of them. Hence, we can suggest that the real reason behind the performance speed is the size and nature of the embedding model itself.

[2] `https://huggingface.co/cardiffnlp/twitter-roberta-base-emotion`
[3] `https://docs.python.org/3/library/time.html`

For example, while the fastest method, Word2Vec, is basically a pre-loaded dictionary of words and their encodings, the slowest model RoBERTa is a fine-tuned transformer, which we run and intercept to extract encoding vectors.

Table 5.3: Time of Anger dataset encoding with various embedding methods.

Embedding	Time (sec), dataset	Avg time (sec), instance	Size of embedding
DeepMoji	38.53	0.018	2304
RoBERTa	1285.63	0.615	768
BERT	149.05	0.071	768
SBERT	80.48	0.038	768
USE	232.52	0.111	512
Word2Vec	8.37	0.004	300

Regarding the data preprocessing techniques that we will apply to the datasets in the next step, it actually takes a minor amount of time. Particularly, applying our basic text cleaning function (Figure 3.1.b) to the whole Anger dataset will take approximately 0.001 seconds. In case we add a stop-words removal step (Figure 3.1.c), this process will take 0.0015 seconds. Compared to the time needed for each embedding to encode the whole dataset, the difference is huge.

In general, we observe the same pattern for all datasets. Hence, we will not provide analogous measurements for the datasets in Section 5.2 and Section 6.

Classification models: the weighted kNN

For the wkNN, at the first step, we calculated the PCC for different preprocessing versions for each emotion dataset and each embedding with different amounts of neighbours. These values and the resulting PCC for the optimal setup are shown in Table 5.4.

We can observe that stop word cleaning only improved results for the Word2Vec embedding and that for the RoBERTa-based model, it mostly makes sense to use the raw tweets. Also, among the different embeddings, RoBERTa obtains the highest results for all four datasets. DeepMoji provides the second-best results for all datasets as well. This can be explained by the fact that both RoBERTa-based and DeepMoji embeddings are explicitly trained on emotion datasets. The next best results are provided by SBERT and USE models, while the two remaining embeddings (basic BERT and Word2Vec) lag considerably. We also can notice that Fear received the lowest scores for all models, which we conjecture is related to the imbalanced nature of this dataset.

In the second step, we discuss our experiments of joining the previously identified best setups of the embedding methods with emotion lexicons, using the different combination strategies outlined in Section 3.3. Initially, we evaluated models based purely on lexicons. The goal here is to check the intrinsic classification strength of each lexicon and of the lexicon-based approach as a whole.

Table 5.4: PCC scores for the best setup for each emotion dataset for different embeddings with the wkNN model.

Setup	Anger	Joy	Sadness	Fear
Twitter-RoBERTa-base				
Tweet preprocessing	No	No	No	Yes
Stop word cleaning	No	No	No	No
Number of neighbours	19	13	9	11
PCC	**0.6544**	**0.6855**	**0.6966**	**0.5811**
DeepMoji				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	No	No	No	No
Number of neighbours	11	21	13	13
PCC	0.5605	0.6157	0.6456	0.5492
BERT				
Tweet preprocessing	No	Yes	Yes	Yes
Stop word cleaning	No	No	No	Yes
Number of neighbours	25	21	21	9
PCC	0.4011	0.4861	0.4269	0.2842
SBERT				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	Yes	No	No	No
Number of neighbours	19	9	21	13
PCC	0.5046	0.5218	0.5201	0.4396
USE				
Tweet preprocessing	No	Yes	Yes	No
Stop word cleaning	No	No	No	No
Number of neighbours	23	21	19	11
PCC	0.5028	0.5421	0.5929	0.5236
Word2Vec				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	Yes	Yes	Yes	Yes
The number of neighbours	13	23	21	13
PCC	0.3311	0.3550	0.3764	0.3805

It is important to note that we use lexicons without handling negation or intensifier techniques. While those methods are meaningful for polarity prediction, they do not make much sense for our experiments since, for the target word, each considered lexicon presents a scalar value in different ranges for multiple emotions, so the negation or intensifiers process is not clear for this case. As was mentioned before, each lexicon works as a dictionary: if a word is present in the lexicon, it receives a particular score; in the other case, it is assigned a zero. For lexicons with several scores per word, we take all of them.

To obtain the lexicon score for a full tweet, we compute the mean of its words' scores. For each of the five lexicons, the output is saved as a separate vector. The sixth vector is constructed by combining all lexicons' scores. For each vector, we applied a wkNN classification model, with initially the same

number of neighbours for all datasets k=23. Results are presented in Table 5.5.

Table 5.5: PCC scored for the lexicon-based features for the wkNN.

Lexicon	**Anger**	**Joy**	**Sadness**	**Fear**
NRC VAD	**0.1172**	**0.1626**	0.0646	0.0650
EMOLEX	0.1142	0.1314	0.1221	**0.1116**
AI	0.0946	0.1587	**0.2081**	0.0938
ANEW	0.0701	0.0907	0.1021	0.0051
Warriner's	0.0460	0.1306	0.0885	0.0804
Combined	0.0789	0.1302	0.0844	0.0684

We can observe that the NRC VAD lexicon is the best-performing lexicon for two out of the four datasets. Also, in general, the scores obtained by the wkNN model using lexicon-based features are much lower than the ones obtained with embedding models. That could be predictable since the vector produced by a lexicon is rather short and contains less information, which is based on particular words, than an embedding method that takes into account the full text.

Next, for each emotion dataset and its best-performing lexicon, the best k value was detected. Results are presented in Table 5.6. As we can see, for different datasets, different values of k perform better and obtained scores are lower than the ones obtained by embedding models.

Table 5.6: PCC scores for the best setup for each emotion dataset for different lexicon-based feature vectors with the wkNN model.

Dataset	**Lexicon**	**k value**	**PCC**
Anger	NRC VAD	29	0.1412
Joy	NRC VAD	31	0.1815
Sadness	AI	27	0.2178
Fear	EMOLEX	9	0.1732

In the third step, we combine embedding and lexicon features by merging their vectors into one. The embedding and lexicon scores are normalized to values between 0 and 1 to account for ranges differences. To obtain the vector of a tweet, we take the average of all vectors of its words. Table 5.7 presents obtained results, and the previous ones using none of the lexicons to check the appending strategy's added value.

As can be seen, for half of the experiments, the use of lexicons does not improve the PCC value. In the cases where scores were improved, we can notice that the difference with the standalone embedding score is minor. We also can notice that one of the weakest models (BERT) benefited most from the added lexicon information, although the improvement is still marginal. In general,

Table 5.7: PCC scores for the lexicon combination approach with wkNN.

Lexicon	Anger	Joy	Sadness	Fear
Twitter-RoBERTa-base				
NRC VAD	**0.6559**	**0.6879**	0.6969	0.5444
EMOLEX	0.6555	0.6864	**0.6971**	0.5475
AI	0.6556	0.6854	0.6966	0.5483
ANEW	0.6539	0.6851	0.6963	0.5459
Warriner	0.6552	0.6855	0.6968	0.5471
None	0.6544	0.6855	0.6966	**0.5811**
DeepMoji				
NRC VAD	0.5514	0.6018	0.5286	0.4409
EMOLEX	0.5586	0.6086	0.5467	0.4519
AI	0.5562	0.6169	0.5364	0.4586
ANEW	0.5533	**0.6187**	0.5418	0.4562
Warriner	0.5520	0.6178	0.5335	0.4526
None	**0.5605**	0.6157	**0.6456**	**0.5492**
BERT				
NRC VAD	0.4007	0.5044	0.4292	0.3780
EMOLEX	0.4033	0.5038	0.4243	0.3785
AI	0.3993	**0.5069**	**0.4346**	0.3797
ANEW	**0.4061**	0.5043	0.4255	0.3825
Warriner	0.4011	0.5041	0.4281	**0.3843**
None	0.4011	0.4861	0.4269	0.2842
SBERT				
NRC VAD	0.4738	0.5197	0.5352	0.4005
EMOLEX	0.4736	0.5178	0.5331	0.4013
AI	0.4741	0.5203	0.5373	0.3996
ANEW	0.4728	0.5215	0.5373	0.3997
Warriner	0.4736	0.5217	**0.5386**	0.3994
None	**0.5046**	**0.5218**	0.5201	**0.4396**
USE				
NRC VAD	**0.5178**	0.5306	0.6074	0.5052
EMOLEX	0.5141	0.5175	0.6043	0.5238
AI	0.5125	0.5284	**0.6185**	**0.5319**
ANEW	0.5063	0.5300	0.6065	0.5131
Warriner	0.5042	0.5331	0.6075	0.5234
None	0.5028	**0.5421**	0.5929	0.5236
Word2Vec				
NRC VAD	0.2135	0.3214	0.2109	0.3284
EMOLEX	0.2152	0.3443	0.2345	0.3233
AI	0.2038	**0.3556**	0.2182	0.3379
ANEW	0.1799	0.3232	0.2020	0.3353
Warriner	0.1823	0.3378	0.2040	0.3315
None	**0.3311**	0.3550	**0.3764**	**0.3805**

when some lexicons improved results, they were different for different datasets, with no noticeable pattern to detect the best lexicon for all models. Our hypothesis why adding the emotion lexicon information to the embedding vectors does not improve the classification performance is that the overlap of the lexicon dictionary and the tweets dataset is too small, which leads to a sparse lexicon

embedding representation. If we compare the best lexicons from Table 5.7 with the best ones from Table 5.6, we can see that they are different.

Hence, we can say that role of lexicons, in this case, is rather weak, and the main contribution to the scores is made by embedding models.

Classification models: FRNN-OWA

For the FRNN-OWA approach, we perform the same steps as for the wkNN.

Table 5.8: Optimal FRNN-OWA classification setup (preprocessing, number of neighbours k) and corresponding PCC score per embedding for the emotion datasets.

Setup	Anger	Joy	Sadness	Fear
Twitter-RoBERTa-base				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	No	No	No	No
k value	19	9	23	9
PCC	**0.6749**	**0.6845**	**0.6917**	**0.6037**
DeepMoji				
Tweet preprocessing	No	No	Yes	Yes
Stop word cleaning	No	No	No	No
k value	23	19	23	21
PCC	0.5701	0.6342	0.6699	0.6005
BERT				
Tweet preprocessing	No	No	No	No
Stop word cleaning	No	No	No	No
k value	19	17	23	7
PCC	0.4405	0.5243	0.4563	0.4306
SBERT				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	No	No	No	No
k value	19	15	23	11
PCC	0.5124	0.5530	0.5372	0.5095
USE				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	No	No	No	No
k value	23	23	23	21
PCC	0.4973	0.5573	0.5988	0.5531
Word2Vec				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	Yes	Yes	Yes	Yes
k value	27	23	23	7
PCC	0.4093	0.4497	0.4601	0.4073

First, we evaluate the best setup for each embedding-dataset combination and provide results in Table 5.8, where we can immediately observe that the RoBERTa-based embedding method provides superior results to the others on all datasets. DeepMoji, again is the second-best embedding for all datasets, followed by SBERT and USE. The lowest scores are obtained for Word2Vec and BERT. Also, PCC scores for Fear are often the lowest among the emotions,

which, as in the previous approach, may be due to the fact that this dataset is the most imbalanced. We should note that some embeddings mostly do not require any preprocessing, like DeepMoji and BERT. For the other methods, tweet cleaning generally performed better, while only Word2Vec benefits from stop word removal. The optimal value of k varies between methods and datasets, while for Fear, lower values of k are more often an optimal choice, which may again be linked to the imbalance of this dataset. In other words, when for a Fear test instance we take too many neighbours from the training data to make a prediction, at some point, we will receive too many instances from the dominant class. However, this is not because those samples are really close to the target test instance, but rather because there will not be enough proper neighbours from other classes. In this way, the small number k keeps only the most relevant neighbours for the final prediction making. In general, scores of FRNN-OWA and wkNN seem close; however, FRNN-OWA obtained higher results for the most imbalanced dataset - Fear.

Next, for each dataset, we performed an evaluation on the single lexicons (Table 5.9) and with a fine-tuned k for the best one (Table 5.10).

Table 5.9: PCC scored for the lexicon-based features for the FRNN-OWA.

Lexicon	**Anger**	**Joy**	**Sadness**	**Fear**
NRC VAD	0.1595	0.1657	0.1513	0.0859
EMOLEX	0.1357	0.1155	0.1113	**0.1594**
AI	0.1411	0.1686	**0.2073**	0.0404
ANEW	0.1093	0.1421	0.1136	0.1105
Warriner's	0.1896	0.2092	0.1708	0.1170
Combined	**0.1914**	**0.2387**	0.1830	0.0995

Table 5.10: PCC scores for the best setup for each emotion dataset for different lexicon-based feature vectors with the FRNN-OWA model.

Dataset	**Lexicon**	k **value**	**PCC**
Anger	Combined	9	0.2317
Joy	Combined	25	0.2387
Sadness	AI	23	0.2073
Fear	EMOLEX	7	0.1948

The combined vector performed better for the two datasets. These scores are lower than those obtained with standalone embeddings. Comparing with the setups obtained with wkNN (Table 5.6), we can see that we received higher scores for Anger, Joy, and Fear with FRNN-OWA. Also, for Sadness and Fear, the same lexicon became the best one.

Table 5.11: PCC scores for the lexicon combination with FRNN-OWA.

Lexicon	Anger	Joy	Sadness	Fear
Twitter-RoBERTa-base				
NRC VAD	0.6618	0.6826	**0.6989**	0.5718
EMOLEX	0.6631	0.6830	0.6986	0.5701
AI	0.6627	0.6831	0.6986	0.5698
ANEW	0.6627	0.6831	0.6983	0.5715
Warriner	0.6627	0.6831	0.6986	0.5701
None	**0.6749**	**0.6845**	0.6917	**0.6037**
DeepMoji				
NRC VAD	0.5473	0.6185	0.5434	0.4872
EMOLEX	0.5468	0.6061	0.5473	0.4916
AI	0.5585	0.6123	0.5392	0.4782
ANEW	0.5544	0.6119	0.5422	0.4792
Warriner	0.5495	0.6143	0.5466	0.4764
None	**0.5701**	**0.6342**	**0.6699**	**0.6005**
BERT				
NRC VAD	0.4159	0.5104	0.4326	0.4313
EMOLEX	0.4163	0.5106	0.4353	**0.4373**
AI	0.4169	0.5115	0.4325	0.4340
ANEW	0.4178	0.5108	0.4292	0.4344
Warriner	0.4168	0.5099	0.4302	0.4340
None	**0.4405**	**0.5243**	**0.4563**	0.4306
SBERT				
NRC VAD	0.4806	0.5474	0.5148	0.4792
EMOLEX	0.4814	0.5484	0.5167	0.4792
AI	0.4814	0.5484	0.5163	0.4769
ANEW	0.4822	0.5485	0.5159	0.4746
Warriner	0.4821	0.5484	0.5166	0.4746
None	**0.5124**	**0.5530**	**0.5372**	**0.5095**
USE				
NRC VAD	0.4740	0.5102	0.5441	0.5118
EMOLEX	0.4798	0.5237	0.5617	0.5282
AI	0.4859	0.5138	0.5608	0.5251
ANEW	0.4862	0.5280	0.5532	0.5146
Warriner	0.4876	0.5094	0.5476	0.5218
None	**0.4973**	**0.5573**	**0.5988**	**0.5531**
Word2Vec				
NRC VAD	0.2887	0.4232	0.2735	0.3343
EMOLEX	0.3015	0.4467	0.2873	0.3409
AI	0.2956	0.4479	0.2923	0.3452
ANEW	0.2654	0.4330	0.2614	0.3375
Warriner	0.2822	0.4288	0.2714	0.3319
None	**0.4093**	**0.4497**	**0.4601**	**0.4073**

In the next step, we combined embeddings with lexicons and presented these results in Table 5.11. We can see that for the majority of cases, lexicons did not improve the PCC scores, especially for one of the best embeddings, DeepMoji, which does not have a single setup that was improved. This conclusion is also true for SBERT, USE, and Word2Vec.

In a couple of cases that benefited from lexicons, the scores' increases were minor.

Regression models

For the FRNN regression, we perform the same steps. Firstly, we evaluate the best setup for each dataset and each embedding (Table 5.12).

Table 5.12: Optimal FRNN regression setup (preprocessing, number of neighbours k) and corresponding PCC score per embedding for the emotion datasets.

Setup	**Anger**	**Joy**	**Sadness**	**Fear**
Twitter-RoBERTa-base				
Tweet preprocessing	Yes	Yes	No	Yes
Stop word cleaning	No	No	No	No
k value	23	15	19	7
PCC	**0.6945**	**0.7059**	**0.7207**	**0.6179**
DeepMoji				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	No	No	No	No
k value	27	23	23	17
PCC	0.6280	0.6369	0.6582	0.6134
BERT				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	No	No	No	No
k value	13	19	19	5
PCC	0.4468	0.5228	0.4640	0.4321
SBERT				
Tweet preprocessing	Yes	No	Yes	Yes
Stop word cleaning	Yes	No	No	No
k value	11	17	7	19
PCC	0.5058	0.5356	0.5332	0.4773
USE				
Tweet preprocessing	No	Yes	No	Yes
Stop word cleaning	No	No	No	No
k value	29	13	29	11
PCC	0.5367	0.5749	0.6378	0.5633
Word2Vec				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	Yes	No	Yes	Yes
k value	21	9	29	5
PCC	0.4155	0.4589	0.4650	0.4207

We can notice that for most embeddings, the results slightly improve those obtained with wkNN and FRNN-OWA methods. Concerning the differences between embeddings, we observe similar trends as for wkNN and FRNN-OWA classifications: the best results are obtained for RoBERTa, with DeepMoji as the second, followed by SBERT and USE. BERT and Word2Vec are again underperforming by comparison. The optimal preprocessing options also vary a bit, with tweet cleaning now proving beneficial in the majority of cases.

We also evaluated the lexicons' performance on their own (Table 5.13) to define the best lexicon and its best parameter k for each dataset (Table 5.14).

Table 5.13: PCC scored for the lexicon-based features for the FRNN regression.

Lexicon	**Anger**	**Joy**	**Sadness**	**Fear**
NRC VAD	0.0949	0.1971	0.1112	0.0972
EMOLEX	0.1217	0.1296	0.0851	0.0979
AI	0.0819	0.1443	**0.1778**	**0.1437**
ANEW	0.0375	0.1321	0.1040	0.0410
Warriner's	0.1221	**0.2107**	0.1401	0.0600
Combined	**0.1478**	0.1724	0.1169	0.0336

We can observe that the AI lexicon is the best for two datasets out of four, with no other significant pattern to notice. If we compare the scores from Table 5.14 with similar ones from Tables 5.6 and 5.10, we can see that the regression approach mostly performs worse than both classification methods. Another observation is that for the Sadness dataset, for all three models, the best-performing lexicon was the same, particularly, the AI lexicon.

Table 5.14: PCC scores for the best setup for each emotion dataset for different lexicon-based feature vectors with the FRNN regression model.

Dataset	**Lexicon**	k **value**	**PCC**
Anger	Combined	21	0.1494
Joy	Warriner's	5	0.2492
Sadness	AI	19	0.1861
Fear	AI	15	0.1644

Finally, we combined embeddings with lexicons for each dataset (Table 5.15). Lexicons improve PCC scores only for one dataset-embedding pair out of twenty-four; even there, the increase is minor.

The biggest drop in scores for lexicon usage was observed for the Word2Vec and DeepMoji embeddings, while for others, the difference is not significant. Regarding the comparison with wkNN and FRNN-OWA classification results, we can see that theFRNN regression best scores for each dataset-embedding pair from Table 5.15 are a bit better than the corresponding scores from Tables 5.7 and 5.15 for the majority of cases.

As we can see from the presented results, so far, lexicons have provided quite low results for all classification and regression approaches.

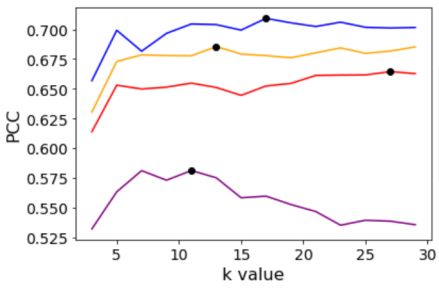
Table 5.15: PCC for the lexicon combination with FRNN regression.

Lexicon	Anger	Joy	Sadness	Fear
Twitter-RoBERTa-base				
NRC VAD	0.6801	0.6939	0.7191	0.5881
EMOLEX	0.6802	0.6951	0.7197	0.5885
AI	0.6805	0.6952	0.7187	0.5886
ANEW	0.6798	0.6953	0.7185	0.5888
Warriner	0.6801	0.6952	0.7181	0.5876
None	**0.6945**	**0.7059**	**0.7207**	**0.6179**
DeepMoji				
NRC VAD	0.5556	0.5921	0.5513	0.5032
EMOLEX	0.5687	0.5964	0.5471	0.4975
AI	0.5648	0.5947	0.5513	0.4928
ANEW	0.5636	0.5996	0.5440	0.4886
Warriner	0.5623	0.5948	0.5473	0.4889
None	**0.6280**	**0.6369**	**0.6582**	**0.6134**
BERT				
NRC VAD	0.3897	0.5169	0.4275	0.4363
EMOLEX	0.3958	0.5182	0.4316	0.4390
AI	0.3969	0.5193	0.4332	0.4405
ANEW	0.3964	0.5208	0.4310	0.4386
Warriner	0.3925	0.5216	0.4328	**0.4409**
None	**0.4468**	**0.5228**	**0.4640**	0.4321
SBERT				
NRC VAD	0.4606	0.5230	0.5195	0.4679
EMOLEX	0.4596	0.5236	0.5172	0.4709
AI	0.4578	0.5213	0.5184	0.4715
ANEW	0.4577	0.5217	0.5187	0.4714
Warriner	0.4577	0.5204	0.5185	0.4708
None	**0.5058**	**0.5356**	**0.5332**	**0.4773**
USE				
NRC VAD	0.4988	0.5210	0.6220	0.5239
EMOLEX	0.4966	0.5314	0.6194	0.5415
AI	0.5104	0.5382	0.6339	0.5394
ANEW	0.5023	0.5088	0.6210	0.5337
Warriner	0.5109	0.5235	0.6278	0.5395
None	**0.5367**	**0.5749**	**0.6378**	**0.5633**
Word2Vec				
NRC VAD	0.1909	0.3314	0.2663	0.2265
EMOLEX	0.1984	0.3493	0.2302	0.2584
AI	0.2002	0.3615	0.2474	0.2564
ANEW	0.1800	0.3285	0.2318	0.2555
Warriner	0.1832	0.3605	0.2334	0.2607
None	**0.4155**	**0.4589**	**0.4650**	**0.4207**

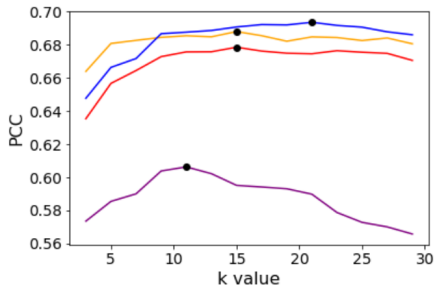
Sensitivity of parameter k

It is interesting to examine the sensitivity of parameter k, which represents the number of neighbours that will be considered in the similarity relationship, as there is no universal rule for choosing the best value.

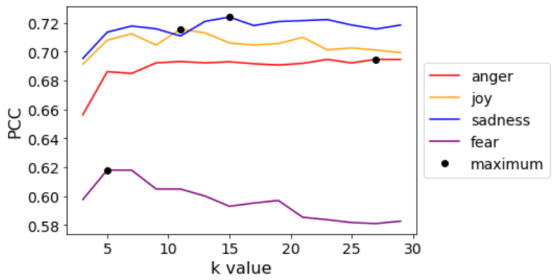
Hence, we performed such exploration for the RoBERTa embedding for three prediction models we consider and all four emotion datasets (Figure 5.1).



(a) For wkNN.



(b) For FRNN-OWA.



(c) For FRNN regression.

Figure 5.1: Sensitivity analysis of k parameter for three different models and four datasets with RoBERTa embedding.

From Figure 5.1, we can see that tuning of the parameter k can improve PCC scores for all suggested cases but not drastically. We also notice that there is no significant difference among predicting methods. However, for the Fear dataset, a smaller amount of k usually performs better, as was observed previously. Meanwhile, for the Anger dataset, it is often the opposite case.

5.1.4 Ensemble of models

In the next step, we consider an ensemble of several best models tuned in the previous step. To define the best ensemble for each prediction model, we evaluate the usefulness of confidence scores, the best subset of embeddings to use, and whether lexicons can improve these results.

Classification models: weighted kNN

The first ensemble that we tried for the wkNN model combines all classifiers based on embedding methods that we considered in Table 5.4. We trained a model for each vector separately with the best k value and the best tweet

preprocessing pipeline described in Table 5.4. Table 5.16 shows the results, using first the standard mean as a voting function (in other words, attributing equal importance to each classifier) and then the weighted average involving rescaled confidence scores as defined in Eq. (4.9). For the latter approach, the α parameter was tuned by a grid search and its optimal values for each data set are also included in the table.

Table 5.16: PCC scores for an ensemble of six wkNN methods with different embeddings, using two different voting functions.

Setup	**Anger**	**Joy**	**Sadness**	**Fear**
Standard mean	**0.6574**	**0.6968**	**0.7187**	**0.6329**
Conf. scores	0.6293	0.6716	0.7018	0.5341
tuned α	$\alpha = 0.9$	$\alpha = 0.9$	$\alpha = 0.9$	$\alpha = 0.1$

We can see that a regular label-based solution with the standard mean voting function works best for all datasets. Next, we aim to improve this ensemble of all six embeddings with the standard mean that appears to be the best setup (marked as #1) by combining it with lexicon vectors in Table 5.17. In this table, we list the setup of each ensemble (all models that are included), its size (number of models that we are using), and PCC scores for all four emotion datasets. In the first setup, for each dataset, we took the best-performing lexicon from Table 5.6 and added it as a separate classifier to the baseline ensemble (#2). For comparison, we also consider a setup where all five lexicons and their combination are added as six more classifiers to check how each of them influences the output scores (#3). The obtained results, shown in the second and third lines of Table 5.17, illustrate that, in general, the lexicons are unable to improve the baseline and that adding all lexicons takes the scores down.

Given that the RoBERTa-based method performs the best among all embeddings (Table 5.4) and that it benefits from the lexicon appending strategy for two datasets out of four (Table 5.7), we also consider two additional setups. One extends the baseline (#1) with the lexicon-appended RoBERTa classifier (#4). Another one adds the best lexicon to the previous ensemble (#5).

These approaches' results are presented in the fourth and fifth lines of Table 5.17. We can see that these adjustments improve the scores noticeably. For all emotion datasets, the #4 setup performs best, where we combine all embeddings with RoBERTa merged with the best lexicon (different for each dataset).

To explore the success of the setup #4, we aimed to perform analogical experiments for each dataset. In setup #6, we keep all embedding models but replace the last model, which is based on the RoBERTa vector merged with the best lexicon, with the regular RoBERTa model. Basically, in this setup, we are considering the RoBERTa model twice and all other embeddings one time and do not consider any lexicon at all. All discussed results are provided in Table 5.18.

Table 5.17: PCC scores for the ensemble approach with different feature combinations for all emotion datasets with wkNN method.

#	Setup of ensemble	Size	Anger	Joy	Sadness	Fear
1	All embedding vectors	6	0.6574	0.6968	0.7187	0.6329
2	#1 plus the best lexicon	7	0.6353	0.6817	0.7055	0.6203
3	#1 plus all five lexicons and their combination	12	0.5432	0.6099	0.6165	0.5333
4	#1 plus the best embedding merged with the best lexicon	7	**0.6811**	**0.7171**	**0.7252**	**0.6521**
5	#1 plus the best lexicon and the best embedding merged with the best lexicon	8	0.6483	0.7046	0.7213	0.6425

From Table 5.18 we can see that for all datasets, setup #4 outperformed #6, but the difference is imperceptible. Meanwhile, for the Anger and Sadness datasets, the compared scores are basically the same.

Table 5.18: PCC scores for the best wkNN setup and its modification without the usage of lexicons.

#	Setup of ensemble	Anger	Joy	Sadness	Fear
#4	#1 plus roBERTa merged with the best lexicon	**0.6811**	**0.7171**	**0.7252**	**0.6521**
#6	#1 plus roBERTa with the same setup	**0.6811**	0.7146	**0.7252**	0.6514

For these results, we can conclude that setup #4 outperforms all other solutions for wkNN that includes lexicons from Table 5.17 because it uses the RoBERTa model twice (one regular and a second one in combination with lexicons). Those outcomes confirm one more time that lexicons do not seem to bring any additional information to the information already present in the embedding vectors and that RoBERTa remains the strongest embedding model. We will not use triple or more weights to the results of the RoBERTa model to avoid overfitting.

We also used grid search to identify the best subset of embeddings, with the hypothesis that by reducing the amount of weaker embeddings in the ensemble, we can improve its outcomes. We calculate the PCC score for all combinations of embeddings with the size of each subset between two and five models. The results for the best subsets for each emotion dataset are provided in Table 5.19. As we can see, each best subset includes RoBERTa and DeepMoji models.

The obtained for the best subsets scores are higher than for ensembles of all six embeddings (setup #1 from Table 5.17), but also they are lower than setup #4.

Table 5.19: PCC scores of the best subsets of embedding models for wkNN classification ensemble for the emotion datasets.

Dataset	Setup	PCC
Anger	RoBERTa, DeepMoji, SBERT, USE	0.6808
Joy	RoBERTa, DeepMoji, SBERT	0.7167
Sadness	RoBERTa, DeepMoji, BERT, USE, Word2Vec	0.7051
Fear	RoBERTa, DeepMoji, BERT, USE	0.6246

Therefore, we consider setup #4 from Table 5.17 the best solution for each emotion dataset for the wkNN classification model, which will later be applied to the test dataset.

Classification models: FRNN-OWA

Similarly to wkNN, we used six FRNN-OWA classifiers corresponding to the six embedding methods outlined before with the best setups in Table 5.8. Table 5.20 shows the results, where we performed an evaluation for both label-based (marked as "standard mean") and confidence scores-based (with corresponding α parameter) approaches.

Table 5.20: PCC scores for an ensemble of six FRNN-OWA methods with different embeddings using two different voting functions.

Setup	Anger	Joy	Sadness	Fear
Standard mean	**0.6475**	**0.7126**	**0.7152**	**0.6448**
Conf. scores	0.6381	0.6823	0.6931	0.5776
tuned α	$\alpha = 0.3$	$\alpha = 0.1$	$\alpha = 0.1$	$\alpha = 0.3$

From Table 5.20, we can see that for all datasets, similarly to the wkNN, the weighted average voting function with confident scores performed worse than the standard mean. We also notice for all emotion datasets, the ensemble outperforms the use of a single classifier.

To improve the ensemble of all six embedding models (#1), we again considered the usage of lexicons. Results are provided in Table 5.21. Similarly to the wkNN, in Table 5.21 we tried to expand the baseline ensemble of six embeddings (#1) with the best lexicon vector from Table 5.10 (#2), with all lexicons and their combination (#3), with the best embedding merged with the best lexicon (#4), and with a combination of #2 and #4 (#5). We can see that approaches #1 and #2 are very close, showing another time that the addition of the single best lexicon does not influence the score much. However, it did help Sadness

Table 5.21: PCC scores for the ensemble approach with different feature combinations for all emotion datasets with FRNN-OWA method.

#	Setup of ensemble	Size	Anger	Joy	Sadness	Fear
1	All embedding vectors	6	0.6475	0.7126	0.7152	0.6448
2	#1 plus the best lexicon	7	0.6415	0.7118	0.7214	0.6307
3	#1 plus all five lexicons and their combination	12	0.6079	0.6724	0.6569	0.6333
4	#1 plus the best embedding merged with the best lexicon	7	**0.7088**	**0.7685**	0.7515	0.6467
5	#1 plus the best lexicon and the best embedding merged with the best lexicon	8	0.6938	0.7603	**0.7516**	**0.6623**

and Fear datasets obtain slightly better results with the #5 approach, compared with the #4 method, which was the best for Anger and Joy.

To improve these results, we also consider ensembles constructed with a subset of embeddings, using a grid search to identify the optimal setup. In contrast to the results of wkNN, for the FRNN-OWA, we received higher scores with some subsets than for the full ensembles for each dataset, and we provide the best subsets in Table 5.22. Noticeably, RoBERTa, DeepMoji, and USE are common embeddings for each dataset's best setup, while the other one-two embeddings left can vary.

Table 5.22: PCC scores of the best subsets of embedding models for FRNN-OWA classification ensemble for the emotion datasets.

Dataset	Setup	PCC
Anger	RoBERTa, DeepMoji, BERT, USE, Word2Vec	0.7241
Joy	RoBERTa, DeepMoji, USE, SBERT, BERT	0.7788
Sadness	RoBERTa, DeepMoji, USE, SBERT	0.7719
Fear	RoBERTa, DeepMoji, USE, Word2Vec, SBERT	0.6930

We can see that subsets' scores in Table 5.22 are higher than the best setups #4 and #5 from Table 5.21. To evaluate how the addition of lexicons can or cannot improve the subset-based setups, we performed experiments similar to the ones done for wkNN in Table 5.18, as well as those in Table 5.19, i.e., trying to remove some of the weakest performing embeddings. Hence, for each ensemble in Table 5.22, we considered the analogous setup, where RoBERTa was replaced by RoBERTa merged the best lexicon. The obtained scores were very close to the original ones, with the same minimal difference as was presented in Table 5.18. Hence, for the FRNN-OWA method, lexicons did not improve the best

setups.

We can conclude that the best setups for the emotions datasets with FRNN-OWA require four to five embeddings to provide the best results and that the usage of lexicons does not improve it.

Regression models

Since we do not have confidence scores for the regression models, we skip the step with α parameter tuning and proceed directly to the combination of an ensemble of all embedding with lexicon vectors.

We use here the same four improvements of the baseline approach (#1) as we did for the wkNN and FRNN-OWA methods: with the best lexicon (#2), with all lexicons (#3), with the best embedding merged with the best lexicon alone (#4) or with the additional best lexicon vector (#5). The obtained results are listed in Table 5.23.

Table 5.23: PCC scores of the best subsets of embedding models for FRNN regression ensemble for the emotion datasets.

#	Setup of ensemble	Size	Anger	Joy	Sadness	Fear
1	All embedding vectors	6	0.6318	0.6814	0.6940	0.5743
2	#1 plus the best lexicon	7	0.6188	0.6853	0.6853	0.5572
3	#1 plus all five lexicons and their combination	12	0.6061	0.6222	0.6295	0.3804
4	#1 plus the best embedding merged with the best lexicon	7	**0.6425**	0.6826	**0.7033**	**0.6002**
5	#1 plus the best lexicon and the best embedding merged with the best lexicon	8	0.6394	**0.6871**	0.6920	0.5815

Here we can see a similar pattern to the wkNN and FRNN-OWA classification models, where approaches #1 and #2 perform almost equally, solution #3 shows the worst results and the best scores are obtained with approach #4 (for Anger, Sadness, and Fear datasets) or #5 (Joy dataset). However, in comparison with the PCC scores from Table 5.12, we can see that RoBERTa provides better results than the discussed solutions for all datasets.

We also include the grid search of the best embedding setup, with results for each emotion dataset provided in Table 5.24. We can see that RoBERTa and DeepMoji models are common for all best subsets. Also, these subsets provided higher scores than the ensembles of all embedding methods (#1 setup from Table 5.23). However, in comparison with standalone RoBERTa scores for Anger, Joy, and Sadness datasets from Table 5.12, the subsets' scores from Table 5.24 are lower, except for the Fear dataset. Hence, we will consider the

standalone RoBERTa model for Anger, Joy, and Sadness as the best setup with the FRNN regression model, while for the Fear dataset, we use the setup from Table 5.12 with the ensemble of four embeddings.

Table 5.24: Optimal FRNN regression ensemble setup and corresponding PCC score for the emotion datasets.

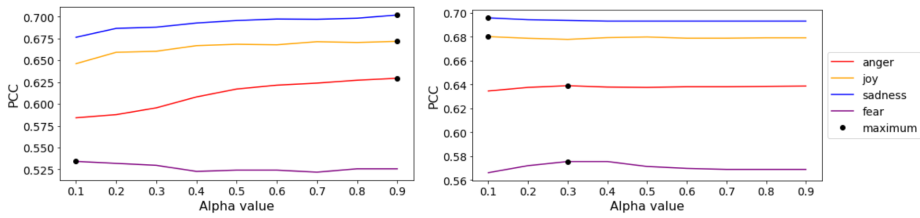
Dataset	**Setup**	**PCC**
Anger	RoBERTa, DeepMoji, SBERT, USE	0.6615
Joy	RoBERTa, DeepMoji, BERT, SBERT	0.6983
Sadness	RoBERTa, DeepMoji, USE	0.7134
Fear	RoBERTa, DeepMoji, SBERT, USE	0.6565

For the FRNN regression model, in the same way, as we did for wkNN and FRNN-OWA methods, we questioned if lexicons can improve the best setup for each dataset, by replacing the RoBERTa embedding with RoBERTa vector merged with the best lexicon's scores.

The described results were analogous to those for the classification models, showing from none to minor changes in the scores, proving our suggestion that lexicons, in our case, are not that useful to improve the results.

Sensitivity of parameter α

In a similar way as for the value of k, we want to explore the sensitivity of the α parameter, which we use to rescale confidence scores for classification models. Figure 5.2 illustrates its behaviour for both wkNN and FRNN-OWA models for all four emotion datasets.



(a) For wkNN. (b) For FRNN-OWA.

Figure 5.2: Sensitivity analysis of α parameter for two classification models and four datasets with RoBERTa embedding.

As we can see from Figure 5.2, tuning of the α parameter does improve the PCC score but not dramatically. For the Fear dataset, it shows the best performance with a small α value for both classification models, which could be explained by its imbalanced nature.

Meanwhile, for the three other datasets, a small α fits only the FRNN-OWA model, while wkNN requires the maximum value of the parameter.

5.1.5 Test data results

In this section, we provide and discuss the results of the best settings described in the previous section (Tables 5.18, 5.22, and 5.24) applied to the test datasets. Since our main approaches clearly outperformed the suggested baseline, we will not include it in the final table with results.

To determine the strength of the obtained best approaches and compare the results of the wkNN, FRNN-OWA classification, and FRNN regression methods, we combined their averaged scores on four emotion test datasets. Evaluation scores were calculated for each emotion dataset and obtained results were averaged. Apart from the PCC used by the competition organisers, we also included MAE and CCA scores.

Test results for each evaluation metric, together with the CV and averaged scores for all four datasets, are provided in a separate table for each metric. We calculated the mean scores of all four datasets because the averaged score is used in the original SemEval competition [93] to compare the solutions.

The PCC scores are shown in Table 5.25, where, on average, PCC scores for the test data drop several points compared to the training data (the biggest difference is for the FRNN-OWA approach), but, in general, our approach appears to generalize well to new data.

Table 5.25: PCC scores of the best approach for the wkNN, FRNN-OWA, and FRNN regression on the cross-validation and test data for the emotion datasets.

Dataset	**wkNN**		**FRNN-OWA**		**FRNN reg**	
	CV	Test	CV	Test	CV	Test
Anger	0.6811	0.6753	0.7241	0.6388	0.6945	0.6671
Joy	0.7171	0.6978	0.7788	0.7115	0.7059	0.6738
Sadness	0.7252	0.6756	0.7719	0.6967	0.7207	0.6865
Fear	0.6521	0.5620	0.6930	0.5705	0.6565	0.5724
Averaged	0.7120	0.6526	**0.7419**	**0.6544**	0.6957	0.6499
SemEval place	3^{rd}		2^{nd}		3^{rd}	
Winners	TOP-1: **0.695** - Random Forest & XGBoost [41] TOP-2: **0.653** - LSTM & transfer learning [44] TOP-3: **0.646** - GRU & CNN [116]					

We also included the results of the competition's top-3 solutions, which were discussed in Section 2.1, where we showed that they are primarily based on state-of-the-art methods such as XGBoost, LSTM, and CNN. By considering these PCC scores, we can compare the performance of our approach with DL-

based methods and determine our place on the leaderboard for the English EI-oc subtask[4]. As we can see from Table 5.25, the FRNN-OWA method performs better for both the CV and test data, leading us to the second place in the leaderboard, while wkNN and FRNN regression performs not much worse on the test data, but still reach only the third place. Noticeably, for all methods, for the test data, we received the lowest scores for the Fear dataset.

The MAE scores are provided in Table 5.26. Unlike PCC and CCA, for MAE, lower scores identify a better solution. Here, FRNN regression provides the best scores for CV data and the wkNN - for the test data, with the second-best score for CV.

Table 5.26: MAE scores of the best approach for the wkNN, FRNN-OWA, and FRNN regression on the cross-validation and test data for the emotion datasets.

Dataset	**wkNN**		**FRNN-OWA**		**FRNN reg**	
	CV	Test	CV	Test	CV	Test
Anger	0.6299	0.6786	0.6282	0.7305	0.5615	0.7327
Joy	0.5440	0.5466	0.5727	0.4914	0.5755	0.6321
Sadness	0.5435	0.5630	0.5571	0.5671	0.5476	0.6440
Fear	0.4354	0.4949	0.4228	0.5375	0.4498	0.5818
Averaged	0.5382	**0.5707**	0.5452	0.5816	**0.5336**	0.6476

From the CCA scores listed in Table 5.27, we can see that all scores are quite similar and close to the value of 0.8. However, FRNN-OWA provides the highest score for CV data and the lowest for the test data, for which the best solution with a minor margin is FRNN regression.

Table 5.27: CCA scores of the best approach for the wkNN, FRNN-OWA, and FRNN regression on the cross-validation and test data for the emotion datasets.

Dataset	**wkNN**		**FRNN-OWA**		**FRNN reg**	
	CV	Test	CV	Test	CV	Test
Anger	0.7907	0.7759	0.8050	0.7589	0.8104	0.8036
Joy	0.8193	0.8195	0.8346	0.8378	0.8065	0.8097
Sadness	0.8243	0.8141	0.8229	0.8128	0.8188	0.8080
Fear	0.8542	0.8365	0.8592	0.8225	0.8396	0.8318
Averaged	0.8221	0.8115	**0.8304**	0.8080	0.8188	**0.8132**

As we can notice, the FRNN-OWA model is not the best one for all evaluation metrics and all datasets. It has the best CV scores for PCC and CCA, however,

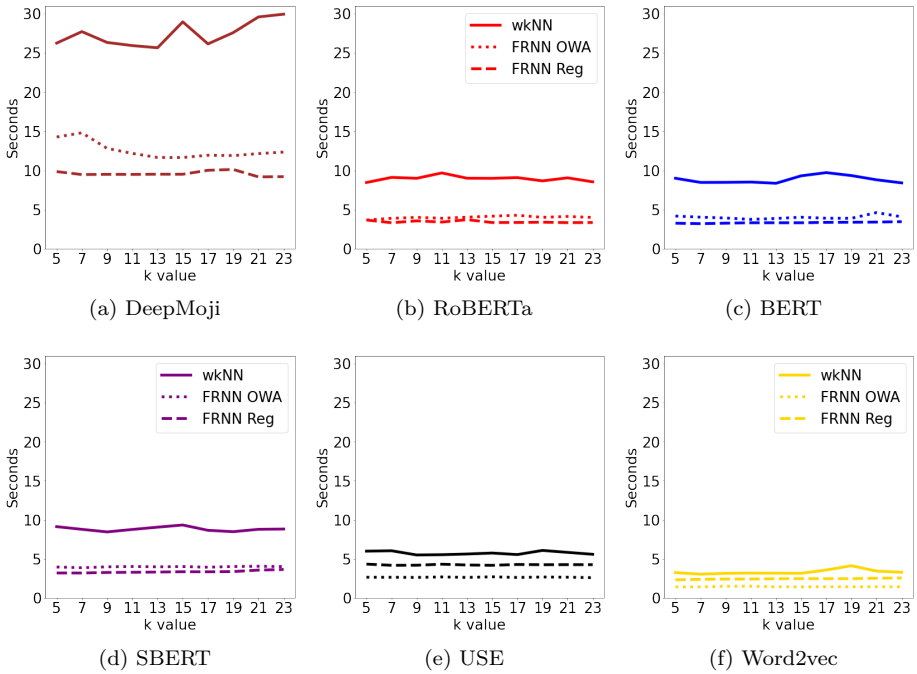
[4] `https://competitions.codalab.org/competitions/17751#results`

the FRNN-OWA got the last place for MAE CV value. We can assume that the reason for such a difference is the CV process itself, where we were tuning several model hyperparameters (text preprocessing, number of neighbours, and so on) based on the PCC metric. In this way, solutions presented in this section are based on setups that work the best for the PCC evaluation metric; meanwhile, for MAE, it could be not the most optimal choice.

In other words, models are fitted to receive the best setup for PCC score, which could result in a weaker MAE result. In order to receive a better test set MAE score, hyperparameters of FRNN-OWA model should be tuned on CV with MAE as an evaluation metric.

From another point of view, we can mention that MAE is less sensitive to outliers compared to PCC. Meanwhile, PCC, being a correlation measure, can be influenced by a few extreme data points. Taking into account this statement and confidence scores' values for both classification models, we can assume that kNN produces more outliers than FRNN-OWA.

Finally, we want to take a look at the performance time for each prediction model. We will measure it in the same way as we did for embedding methods' time measurement, again for the Anger dataset (Figure 5.3).



(a) DeepMoji (b) RoBERTa (c) BERT

(d) SBERT (e) USE (f) Word2vec

Figure 5.3: Time performance for wkNN, FRNN OWA, and FRNN regression models on the Anger dataset for different embedding methods.

As we can see from Figure 5.3, for all embeddings, the wkNN model takes the

longest time to compute. Also, for four cases out of six, FRNN-OWA was a bit slower than FRNN regression, but in general, their performance is quite close. Probably the only case where we can see some difference is the performance of DeepMoji (Figure 5.3a). This embedding is also the only one where we can see some fluctuation of time for different sizes of neighbours k for each method. Meanwhile, for other embedding methods and prediction models, variations of k are not that visible.

Another observation that we can make is a comparison of different embedding methods' performance times among each other. For this purpose, we presented all plots in the same time range (from zero to thirty seconds). As we can see, DeepMoji (Figure 5.3a) is definitely the slowest one for any prediction model. Then we have RoBERTa, BERT, and SBERT, which have almost the same performance time for all models. The USE embedding is a bit faster for the wkNN and FRNN-OWA models, and finally, the Word2Vec is the fastest for any prediction model.

We can also compare those observations from Figure 5.3 with the sizes of vectors produced by these embeddings (Table 5.3). We can see a logical pattern where bigger embeddings take more time to perform for any prediction model and also where embeddings with the same size take nearly the same time.

5.1.6 Summary

To sum up, we can conclude that based on the best PCC scores as a key evaluation metric, for this emotion intensity task, the best solution was achieved with an ensemble of FRNN-OWA classification models using four to five different text embedding techniques for each emotion dataset.

However, in each case, the RoBERTa-based model was the best embedding. This makes sense, given that it was tuned on data related to the considered dataset. We can also refer to the work of [34], where a fine-tuned RoBERTa-based model was the best solution for the Dutch emotion detection task (compared to a fine-tuned BERT-based model). It did not benefit from the usage of lexicons, as was shown in our results as well.

Also, we can mention that the FRNN-OWA classification model used in our final best approach for each emotion dataset appears to be one of the fastest considered prediction models. It makes our solution more efficient than it could be with the analogous wkNN-based approach.

Regarding the publication of results, the majority of them were published in four different studies. The first experiments with the wkNN model for the emotional datasets were provided in [68], while the current chapter has improved results. The main results for the FRNN-OWA model for the same datasets were published in [66], while they were extended with FRNN regression in [65].

5.2 Hate speech and irony recognition

In this section, we investigate our methods' performance for the Hate Speech, Offensive Language, Irony, and Sarcasm datasets.

5.2.1 Dataset and task

As we described in Section 2.1.2, we considered SemEval 2019 Task 5 by [12] ("Hate Speech") and SemEval 2019 Task 6 by [142] ("Offens") as a source of data for our hate speech detection experiments. Also, as mentioned in Section 2.1.3, we used data from two competitions related to irony detection - SemEval 2018 Task 3 by [126] ("Irony") and SemEval 2022 Task 6 by [2] ("Sarcasm"). For all four competitions, we considered binary classification tasks, where we should determine whether a provided text fragment is hateful (ironic) or not.

Regarding data collection, the Hate Speech dataset is based on Twitter data gathered with different techniques. It includes the monitoring of accounts of potential hate speech targets and identified haters, as well as filtering all tweets with specific keywords [12]. Meanwhile, the offensive speech detection task is presented by the OLID dataset [141], where a specific hierarchical three-level annotation system that considers both the aim and the sort of offensive content was used for annotation [142]. Regarding the Irony dataset, it was created by browsing the hashtags #irony, #not, and #sarcasm on Twitter. These hashtags were deleted from the dataset later, with manual labelling of all instances [126]. Finally, for the Sarcasm dataset (English version), the authors were working via the Prolific Academic platform[5] space with users of Twitter who are native English speakers. They provided sarcastic and non-sarcastic tweets together with the labels created by themselves to exclude subjective labelling [2].

We explored some characteristics (such as the IR) of all four datasets and listed them in Table 5.28. Notably, for each dataset, the size of the non-hateful (non-ironic) class is bigger than the hateful (ironic) one.

Table 5.28: Characteristics of the training data for Hate Speech, Offensive Language, Irony, and Sarcasm datasets.

Characteristic	**Hate Speech**	**Offens**	**Irony**	**Sarcasm**
IR	1.37	2.009	1.007	3
Total number of instances	10,000	13,240	3,817	3,468

From Table 5.28, we can see that for the Sarcasm dataset, the size of the non-ironic class is three times bigger than the size of the ironic class, which makes it the most imbalanced dataset. Meanwhile, for Irony, the IR score is close to 1, so it is the most balanced dataset considered in this work. Also, Hate Speech and Offensive datasets have similar sizes and are almost three times bigger than Irony and Sarcasm.

[5] `https://prolific.co`

5.2.2 The baseline

In the same way as for the emotional datasets, for the hate speech and irony-based datasets, we start with a baseline. We again consider the usage of BOW and Bag of N-grams with weighted kNN classification for each dataset. Similarly to Section 5.1.2, we will provide scores for all dataset-N-gram combinations, since the scores are quite low and tuning does not make much difference. So, we will again consider 1-gram, 2-grams, and 3-grams (unique collocation of three words), and the same parameters (k equals 23, and preprocessing is the standard text cleaning with lower-casing). The results for the hate speech and irony-based datasets are presented in Table 5.29. For the Hate Speech, Offensive, and Irony datasets, we calculated macro-averaged F1-score, while for the Sarcasm dataset – F1-score for sarcastic class was calculated, as was requested by SemEval organisers.

Table 5.29: F1-scores for a Bag of N-grams (with N=1,2,3) and the wkNN model applied to Hate Speech, Offensive, Irony, and Sarcasm datasets.

N-gram size	Hate Speech	Offens	Irony	Sarcasm
1	0.6462	0.4627	0.4537	0.0244
2	0.3814	0.4099	0.3341	0.0
3	0.3665	0.4005	0.3324	0.0

From Table 5.29, we can see that 1-gram (BOW) was the best for each dataset, so it remains the main baseline. The Sarcasm dataset receives the lowest scores according to the specific F1-score it uses, with results for 2-gram and 3-gram so insignificant that they are almost equal to zero.

5.2.3 Model tuning

In the same way as we did for emotion datasets, in the beginning, we tuned the parameters for each embedding model for the hate speech and irony-based datasets. Since these tasks are binary classification challenges, we did not apply regression methods to them, investigating only classification models. As an evaluation metric, we used the F1-score (macro-averaged for Hate Speech, Offensive, and Irony datasets and "sarcastic" for the Sarcasm dataset).

Regarding RoBERTa-based models, we used a separate model from [11] for each dataset. Each model was trained on nearly 58M tweets and fine-tuned on the specific dataset. Particularly, for the Hate Speech dataset, we used Twitter-RoBERTa-base for Hate Speech Detection[6], for the offensive dataset - Twitter-RoBERTa-base for Offensive Language Identification[7], meanwhile, for both Irony and Sarcasm datasets we considered Twitter-RoBERTa-base for Irony Detection[8]. Further on, we will refer to these models simply as "RoBERTa".

[6] https://huggingface.co/cardiffnlp/twitter-roberta-base-hate

[7] https://huggingface.co/cardiffnlp/twitter-roberta-base-offensive

[8] https://huggingface.co/cardiffnlp/twitter-roberta-base-irony

Classification models: the weighted kNN

At first, we considered the weighted kNN classification model and tuned the best setup for each pair dataset-embedding (Table 5.30).

Table 5.30: F1-scores for optimal weighted kNN classification setup (preprocessing, number of neighbours k) for all embeddings for the Hate Speech, Offensive, Irony, and Sarcasm datasets.

Setup	Hate Speech	Offens	Irony	Sarcasm
Twitter-RoBERTa-base				
Tweet preprocessing	No	No	No	No
Stop word cleaning	No	No	No	No
k value	29	23	11	5
F1 score	**0.8829**	**0.8343**	**0.9373**	**0.3553**
DeepMoji				
Tweet preprocessing	Yes	No	No	Yes
Stop word cleaning	Yes	No	No	No
k value	19	11	25	5
F1 score	0.7019	0.6422	0.6546	0.2571
BERT				
Tweet preprocessing	Yes	No	No	Yes
Stop word cleaning	No	No	No	Yes
k value	15	27	25	5
F1 score	0.7406	0.6797	0.6582	0.2134
SBERT				
Tweet preprocessing	No	No	No	No
Stop word cleaning	No	No	No	No
k value	15	27	29	7
F1 score	0.7400	0.6946	0.6284	0.1257
USE				
Tweet preprocessing	Yes	No	No	No
Stop word cleaning	No	No	No	No
k value	13	25	25	5
F1 score	0.7333	0.6823	0.6439	0.2496
Word2Vec				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	Yes	Yes	Yes	Yes
k value	13	29	25	5
F1 score	0.6648	0.6778	0.5297	0.1803

We can see that the RoBERTa embedding by far outperforms other methods. This was more or less expected since the used RoBERTa models were tuned on the related datasets for three out of four of the considered tasks out (except the Sarcasm dataset). Meanwhile, the lowest results for the Hate Speech and Offensive datasets were obtained by the DeepMoji embedding, for Irony by Word2Vec, and for Sarcasm - by SBERT.

The results also indicate that for most of the embedding methods, preprocessing is not helpful (with Word2Vec as the only exception). We can also notice that for the Sarcasm dataset, the lower amount of neighbours is more beneficial. The Sarcasm dataset is the most imbalanced compared to others, so this obser-

vation is analogical to the one we did for the emotional datasets, where the Fear dataset was the most imbalanced and required the lowest k as well.

We also can notice that results for sarcasm are lower than for other datasets, but it actually uses a slightly different metric - F1-score calculated for the sarcastic class and not a macro-averaged value like for the other datasets. As was mentioned in Section 4.5.3, we used different metrics for the Sarcasm dataset since it was the metric suggested by the competition's organisers.

Classification models: FRNN-OWA

Results of FRNN-OWA model tuned for each embedding for all four datasets are presented in Table 5.31, which reveals that for all datasets, the obtained results are very close to the ones obtained by the wkNN method in Table 5.30.

Table 5.31: F1-scores for optimal FRNN-OWA classification setup (preprocessing, number of neighbours k) for all embeddings for the Hate Speech, Offensive, Irony, and Sarcasm datasets.

Setup	Hate Speech	Offens	Irony	Sarcasm
Twitter-RoBERTa-base				
Tweet preprocessing	No	No	No	No
Stop word cleaning	No	No	No	No
k value	25	45	27	5
F1 score	**0.8765**	**0.8377**	**0.9365**	**0.3722**
DeepMoji				
Tweet preprocessing	No	No	No	Yes
Stop word cleaning	No	No	No	No
k value	19	39	15	5
F1 score	0.6223	0.6567	0.6774	0.3157
BERT				
Tweet preprocessing	Yes	No	No	Yes
Stop word cleaning	No	No	No	Yes
k value	15	47	23	5
F1 score	0.7172	0.6847	0.6563	0.2351
SBERT				
Tweet preprocessing	No	No	No	No
Stop word cleaning	No	No	No	No
k value	15	47	29	7
F1 score	0.7063	0.7063	0.6504	0.1618
USE				
Tweet preprocessing	Yes	No	No	No
Stop word cleaning	No	No	No	No
k value	13	35	25	5
F1 score	0.7037	0.6898	0.6650	0.2808
Word2Vec				
Tweet preprocessing	Yes	Yes	Yes	Yes
Stop word cleaning	Yes	Yes	Yes	Yes
k value	13	39	27	5
F1 score	0.6700	0.6622	0.5966	0.2050

For the majority of embedding methods, wkNN performed better for the Hate Speech dataset; meanwhile, FRNN-OWA was better for all other datasets. We can also observe similar patterns - the strongest model being RoBERTa and Sarcasm requiring the lowest k value.

5.2.4 Ensemble of models

After obtaining the best setup for each embedding, we combine them all in an ensemble setup for each dataset.

Classification models: the weighted kNN

Scores for the wkNN model are presented in Table 5.32, where it is clear that usage of confidence scores outperforms the standard mean. Also, that ensemble provides quite low results for the Sarcasm dataset, in the same way as single embeddings for this dataset performed worse than for others. However, using the standalone RoBERTa-based embedding for all datasets is still better than any of the ensemble strategies. We also performed a grid search of a subset of embeddings which could outperform obtained scores, but it did not work for any of the datasets.

Table 5.32: F1-score values for an ensemble of six weighted kNN methods with different embeddings, using two different voting functions.

Setup	Hate Speech	Offens	Irony	Sarcasm
Standard mean	0.8072	0.7183	0.7158	0.0509
Conf. scores	**0.8341**	**0.7732**	**0.8627**	**0.0966**
tuned α	$\alpha = 0.1$	$\alpha = 0.8$	$\alpha = 0.5$	$\alpha = 0.8$

Classification models: FRNN-OWA

Results for the FRNN-OWA are presented in Table 5.33 with F1-scores for two voting functions.

Table 5.33: F1-score values for an ensemble of six FRNN-OWA methods with different embeddings, using two different voting functions.

Setup	Hate Speech	Offens	Irony	Sarcasm
Standard mean	0.7500	0.6796	0.7331	0.0995
Conf. scores	**0.8116**	**0.7119**	**0.8350**	**0.1754**
tuned α	$\alpha = 0.1$	$\alpha = 0.8$	$\alpha = 0.5$	$\alpha = 0.8$

From Table 5.33, we again can see similar patterns as in Table 5.32, where for the same best α scores, the second function performed better than the first, which

is a standard mean. If we compare the best scores from those tables, we can see that FRNN-OWA performs best for the Sarcasm dataset, while wkNN is better for others. But the standalone RoBERTa performs better than any embedding combination for FRNN-OWA. Hence, for both classification methods and all datasets, we will consider single RoBERTa-based as the best setup.

5.2.5 Test data results

Similarly as for the emotion intensity datasets, we will not include the baseline in the final results, since our main approaches significantly outperformed it. For all hate speech and irony-based datasets, we applied the best wkNN and FRNN-OWA models (with a standalone RoBERTa) to the test data (Table 5.34).

Table 5.34: Test F1-scores for the best wkNN and FRNN-OWA setups with RoBERTa embedding for Hate Speech, Offensive, Irony, and Sarcasm datasets.

Dataset	**Test F1-score wkNN**	**Test F1-score FRNN-OWA**	**SemEval place FRNN-OWA**	**SemEval winners**
Hate	0.5273	**0.5351**	5^{th}	#1: **0.651** SVM & USE [58] #2: **0.571** [-] #3: **0.546** BiGRUs & FastText [38]
Offens	0.7893	**0.8109**	4^{th}	#1: **0.829** tuned BERT [81] #2: **0.815** BERT-Large [97] #3: **0.814** tuned BERT [150]
Irony	0.5911	**0.6515**	3^{rd}	#1: **0.705** LSTM [134] #2: **0.672** BiLSTM [13] #3: **0.650** SVM & LR [114]
Sarcasm	0.3295	**0.4242**	9^{th}	#1: **0.605** DeBERTa & XML-RoBERTa [139] #2: **0.569** DeBERTa & ERNIEM [50] #3: **0.530** tuned BERT-Base [6]

Although the leaderboard is private for the Hate Speech and Offensive Language competition (we requested them separately), it is available for the Irony[9] and Sarcasm [2] competitions. Hence, we added the results of the top-3 teams for each SemEval competition, which are mainly based on DL and transformer methods (BERT, RoBERTa, BiGRUs, LSTM, BiLSTM), to Table 5.34.

As can be observed from Table 5.34, test scores for the FRNN-OWA method outperformed wkNN for each dataset. Hence, we would provide our SemEval positions for the FRNN-OWA results as the best ones. We can see that we obtained top-9 positions for all these competitions.[10] Compared to Table 5.31, we can also notice that for the Hate Speech, Irony, and Sarcasm datasets, the gap between CV and test scores is much bigger than for the offensive language detection task. Also, the test F1-score for sarcasm is even higher than its CV F1-score. After some additional experiments for those datasets, we can assume that it was caused neither by the RoBERTa embedding method nor by the number of neighbours. We suggest that probably the reason is that training and test data came from different distributions.

We also want to mention that we didn't measure the time performance of the different embedding methods and prediction models for the datasets considered in this section, since we used the same ones as in the previous Section 5.1, where we already measured their implementation.

5.2.6 Summary

To sum up, we can conclude that a proper fine-tuned embedding method, in our case, RoBERTa, is able to provide strong results. Also, the FRNN-OWA classifier, implemented without an ensemble approach, was better than the corresponding wkNN model. It is worth highlighting once again, that the Sarcasm dataset was the only one that used the RoBERTa model fine-tuned on a different dataset and that performance on the Sarcasm dataset was F1-score calculated for the sarcastic class. It is also the most imbalanced dataset, compared to the other three, which altogether might explain the lower scores for this dataset.

The results for Hate Speech, Offensive Language, and Irony datasets with FRNN-OWA model were published in [65]. Meanwhile, for the Sarcasm dataset for the same prediction model presented results are provided in [67]. The baseline and wkNN results, were not published before.

[9] `https://competitions.codalab.org/competitions/17468#results`

[10] Due to CodaLab restrictions, it was not possible to submit our labels for Hate Speech, Offense, and Irony leaderboards, so we calculated our places on our own with the provided validation scripts.

Chapter 6

Application 2: Aspect Based Sentiment Analysis

In this chapter, we demonstrate that our interpretable approach using FRS also obtains promising results for the ABSA, correctly predicting all three classification tasks (category, sentiment, and emotions) for the majority of test instances.

6.1 Dataset and task description

As data for the experiments, we used FMCG reviews which were collected and manually labelled in the framework of the multilingual SentEMO project[1] [35]. The dataset consists of product reviews, i.e., almost 900 reviews in the training set and nearly 400 in the test set. Each review comprises one or more sentences, while each may contain several or no "aspect terms", which are words or collocations which have been assigned three labels, namely an category class, sentiment class and emotion class (we will refer to them as "gold labels").

This annotation is exemplified in Figure 6.1 for the term "breakfast", which is assigned the category class "Food&Drinks_general", and for which a positive sentiment is expressed, as well as the emotion, "joy".

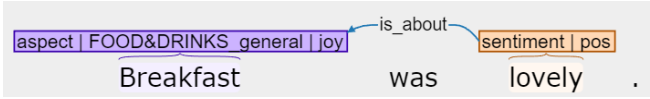
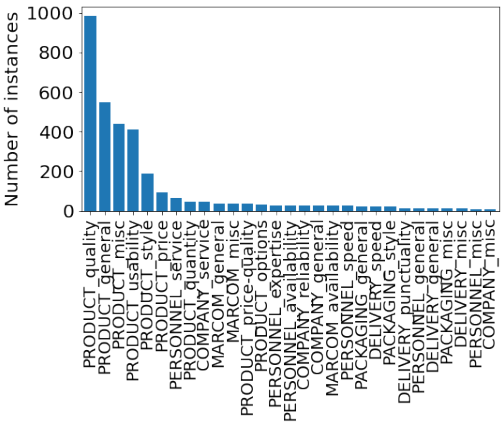


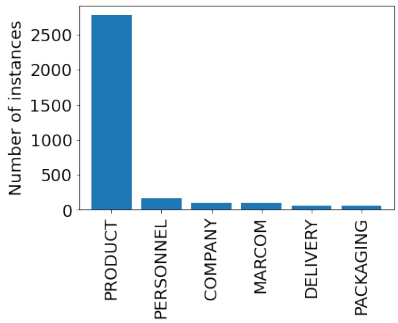
Figure 6.1: Annotation example of the user review with a demonstration of the defined term's category, sentiment, and emotion classes.

Each of these annotations results in a set of classification labels, which we will consider as three separate tasks in the classification experiments:

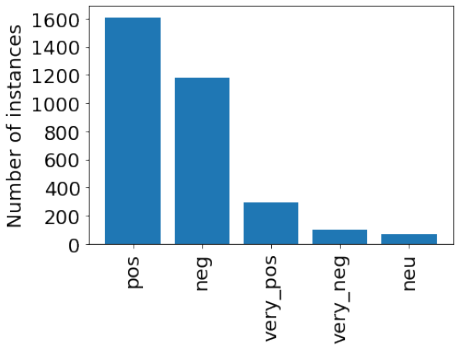
[1] http://sentemo.org



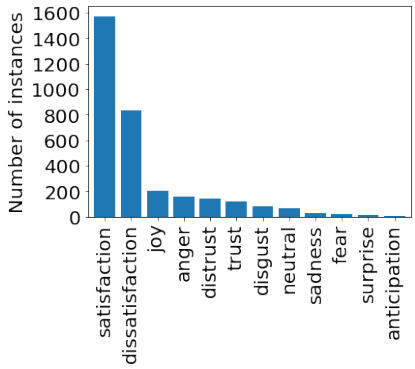
(a) Categories classes distribution.



(b) Categories main classes distribution



(c) Sentiment classes distribution.



(d) Emotion classes distribution.

Figure 6.2: Histograms depicting class distribution for training data for each classification task.

- For the aspect's category classification, we consider six main categories ("product", "personnel", "company", "marcom", "delivery", and "packaging"; Figure 6.2b), and each category is further divided into subcategories ("product_quality", "product_general", etc.; Figure 6.2a), which results in a total of 29 classes.
- For sentiment classification, we distinguish five ordinal labels: "very positive", "positive", "neutral", "negative", and "very negative" (Figure 6.2c).
- For emotions, there are 12 associated labels: "anger", "neutral", "disgust", "surprise", "trust", "distrust", "dissatisfaction", "fear", "joy", "satisfaction", "anticipation", and "sadness" (Figure 6.2d).

For each of the three tasks, the class distribution is skewed, as is also clearly visualised in Figure 6.2. While the large majority of instances fall within the "product" category, the sentiment and emotion annotations are also primarily situated in 2 out of the available 5 (sentiment) or 12 (emotion) classes. The main statistical characteristics of the ABSA dataset for all three tasks are listed in Table 6.1, where we can see how big the imbalances for each task are.

Table 6.1: Characteristics of the training data for the different ABSA tasks.

Characteristic	**Category**	**Category_main**	**Sentiment**	**Emotions**
IR	163.6	52.3	22.9	196.6
Total number of instances	3,253	3,253	3,253	3,253

For our experiments, we selected each term as a separate data instance with all corresponding information, such as the ordinal number of the review, the original full sentence, and the individual classification task labels. We decided not to use any additional text preprocessing to the text before its usage in the embedding methods in order not to dismiss any potentially helpful information.

On the other hand, we tried to provide different text spans to the embedding method (described in Section 3.2) and, more specifically, considered four options:

1. The target term (it can be one word or a collocation).
2. The sentence that contains our target term (this could be repeated for different instances because one sentence can contain several terms).
3. The combination of the previous two vectors, or the so-called "merged" vector of term and sentence vectors (in this way, this embedding vector's length will be double the previous one).
4. A window of terms, which means that we take into account the words around the target term. We considered windows with sizes of three and five; if a sentence has fewer words before or after the target term, we take as much as it has. We also tried two approaches to apply the embedding: for the first approach, we compute the vector embedding for each word in the window separately and then calculate its mean, while for the second approach, we take the embedding of the whole piece of text.

6.2 The baseline

Before describing our pipeline approach and FRS-based models, we want to introduce a baseline solution. Similarly to Sections 5.1.2 and 5.2.2, we applied to the ABSA dataset a wkNN model, based on BOW and Bag of N-grams. We will again use 1-gram, 2-grams, and 3-grams, with the k parameter equal to 23 and standard text cleaning with lower-casing. Regarding the three subtasks

of the ABSA challenge, we treat category, sentiment, and emotion detection separately to assess the efficiency of different N-grams. We also consider three evaluation metrics: weighted F1-scores, accuracy, and cost-corrected accuracy (which can not be calculated for the category classification task, since it is not ordinal). Our results are presented in Table 6.2.

Table 6.2: The weighted F1-scores, accuracy, and cost-corrected accuracy for all classification tasks for the baseline, based on the wkNN and Bag of N-grams.

N	**Category**			**Sentiment**			**Emotion**		
	F1	**Acc**	**CCA**	**F1**	**Acc**	**CCA**	**F1**	**Acc**	**CCA**
1	0.0453	0.3107	-	0.2408	0.5554	0.6095	0.0897	0.5069	0.5815
2	0.0137	0.1789	-	0.1675	0.5109	0.5673	0.0963	0.5059	0.5709
3	0.0099	0.1690	-	0.1437	0.5032	0.5591	0.0744	0.4949	0.5591

From Table 6.2, we can observe that a bigger size of N-gram leads to worse scores for all tasks and all evaluation metrics. Hence, we considered the 1-gram or BOW setup as our baseline.

6.3 Three pipeline approaches

While others have already investigated tackling some of the ABSA subtasks jointly [20, 85, 130], the predominant methodology in ABSA is still a pipeline approach in which first aspect categorisation is performed, and afterwards, the sentiment and emotion labels are predicted for the predicted categories. These pipeline approaches are known to be sensitive to error propagation, which might be even more the case for the imbalanced datasets we are working with. In our experiments, we want to investigate different pipelines and their errors. More precisely, we want to study how fuzzy rough-set-based methods such as FRNN-OWA and FROVOCO (which is specifically designed to handle imbalanced datasets) behave in the suggested pipeline setup.

It is worth noting that we use gold labels at the first step for tuning the best models setups for each subtask; further, we use these subtasks in a pipeline structure, where one task is based on the output of the previous one. The only step that we omit from the classical ABSA pipeline is the first one of term recognition.

6.3.1 Methodology description

Before describing the actual pipeline for the ABSA task, we should mention some preparatory steps that we performed. First of all, we aimed to detect the best classification model setup for each task separately (i.e., category, sentiment, and

emotion classification) based on gold labels and CV evaluation. For this purpose, for each task, we computed predictions using various classification models and different text spans for the text embeddings and tuned each model's parameters. Through this, using CV, we were able to define the best approach for each classification task.

These best models were then applied one by one, forming the "pipeline" for the ABSA task. First, the best "category model" was applied to predict aspects' categories classes for all test set instances. Then, the best "sentiment model" was evaluated on the instances with correctly predicted category labels, after which the "emotion model" was applied to the instances which were correctly predicted in the previous step. Contrary to a normal test scenario, in which we evidently do not have access to gold standard annotations, this approach of only taking into account the correctly predicted instances for the next step primarily enabled us to assess the error propagation throughout the classification pipeline.

We improved this baseline approach with two modifications. First of all, as was showcased in Figure 6.2a, the class distribution of the aspects' categories is highly imbalanced. Since there was simply too little training data for the large majority of categories, we generalised the 29 category classes into 6 main categories as discussed in Section 6.1. A second modification we considered is splitting the emotion models into positive and negative ones. We thus divided the gold emotion labels into two groups; some of them we joined into one emotion class due to the similar nature of the emotions and the small size of their classes:

1. Three positive emotions: joy combined with anticipation and "positive" surprise (instances that have emotion gold label "surprise" and sentiment gold labels "positive" or "very positive"); satisfaction; and trust.

2. Five negative emotions: anger; disgust; dissatisfaction; distrust merged with fear; and sadness combined with "negative" surprise (instances that have emotion gold label "surprise" and sentiment gold labels "negative" or "very negative").

For the model tuning on the gold standard, we divided the training instances into two groups based on their gold emotion labels (positive and negative). However, for the pipeline approach, the approach had to be different. Since we base our emotion prediction step on sentiment classification results, we use the predicted sentiment labels to divide instances into three groups. We combine all instances with "positive" and "very positive" sentiment labels and apply a positive emotion model setup. A similar procedure is conducted for the "negative" and "very negative" sentiment labels. The remaining instances have a "neutral" sentiment label and are automatically assigned the "neutral" emotion. We call the described pipeline "System 1", which is shown in Figure 6.3.

To illustrate how System 1 works, we can take a look at the example:

Example 6.3.1. The staff was very friendly, but the breakfast could have been better.

In Example 6.3.1, the term "staff" has the gold labels "personnel" as an aspect's category, "positive" as sentiment and "joy" as emotion. Let us assume that System 1 predicts "staff" as the "personnel" category on the first step, so it can continue to the second one. But then it predicts sentiment as "very positive", which is wrong, and because of that, this instance will not be taken into account anymore for emotion prediction.

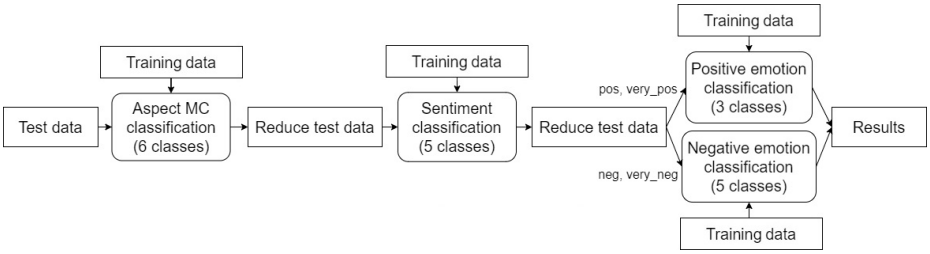


Figure 6.3: System 1: pipeline of three classification tasks in sequence with modifications in the form of aspects' main categories prediction and two emotion models.

We also considered a more relaxed pipeline. For "System 2" (Figure 6.4), we perform data reduction with the cost matrices described in Section 4.5. Since the first step (category classification) is not an ordinal task, and we do not have a cost matrix for it, this approach is only relevant for the latter two tasks. While in System 1, we only kept instances for further processing when they had a cost of 0, we now also keep instances with a cost of 0.5. This modification allows us to bring more instances to the emotion detection step. However, the scores for category and sentiment detection evidently will remain the same.

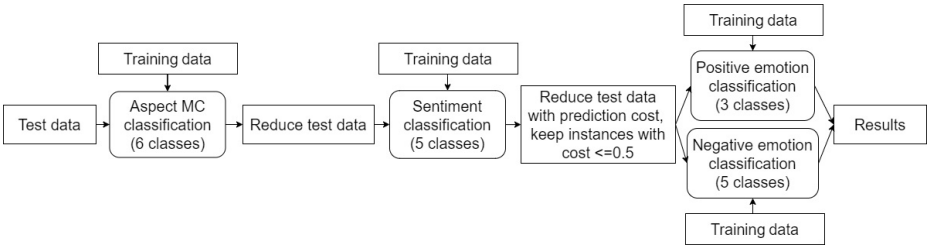


Figure 6.4: System 2: a modification of System 1, where after the sentiment prediction step, data reduction is performed based on misclassification cost.

To illustrate the work of System 2, we can consider the same Example 6.3.1. In the same way, as it was done by System 1, System 2 will successfully predict "staff" as the "personnel" class. In the following step, a "very positive" prediction for the sentiment in the case of System 2 is acceptable since it has a cost of 0.5, compared to the gold label "positive", so it can move on. In the third step, the system gives its prediction for the emotion, which can be correctly equal to the

gold label "joy" and will influence the final score.

Finally, in "System 3", we make our three classification steps independent (Figure 6.5) of each other, giving us an idea of the performance of each of the classifiers on the complete test data. In fact, this third system can be observed as the upper bound for the previous two pipeline systems, in which a test instance is only counted correctly if all three classification predictions are correct.

In the case of Example 6.3.1, all three predictions ("personnel", "very positive", and "joy") will be calculated without depending on each other, and each of them will influence their own evaluation score (for category, sentiment, and emotion detection) separately.

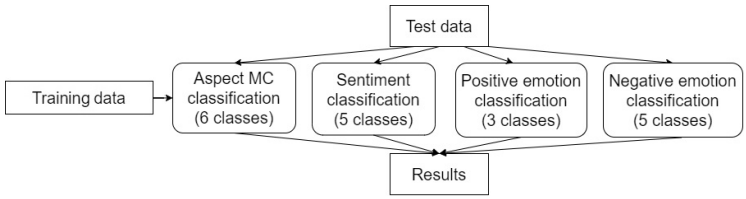


Figure 6.5: System 3: three independent classification tasks on the full test set with no data reduction.

To sum up, we can conclude that with described setups, aspects' categories scores will be the same for all three systems; System 1 and System 2 will have the same sentiment scores, while for System 3, it will differ; and for emotion detection, all three systems will have nonidentical results.

6.3.2 Model tuning

To achieve the highest possible scores in the pipeline, we should be sure that we receive the best results on each classification step. With this idea, we tuned the best model for each task of the three classification steps: aspect's category, sentiment, and emotion classification.

In Section 3.2.5, we described three transformer-based embedding models that we considered for this task: BERT and ALBERT from TA and DistilBERT Yelp Review Sentiment model (DBERT YRS or DistilBERT further), all fine-tuned on the YELP dataset. As we mentioned before, this particular DistilBERT model outperformed both BERT TA and ALBERT TA models. To illustrate this claim, we provide the results of our experiments involving TextAttack models.

Particularly, in Table 6.3, we show the weighted F1-scores for sentiment classification, which is the second step of our ABSA task, based on the gold labels. We illustrate the performance of all three embedding methods for both classification methods (FRNN-OWA and FROVOCO) for different text spans (where "sent" stands for "sentence", "merged" is the merged vector of term and sentence, and "w5" - a window of text size 5 that was chosen as it was the best window option). To make a fair comparison, we used the same amount of neighbours, k equal to 7, as a default value that usually showed good results.

Table 6.3: The weighted F1-scores of sentiment classification task for three BERT-based embedding methods (BERT AT, ALBERT AT, and DBERT YRS) with four text spans and number of neighbours k=7.

Method	**FRNN-OWA**				**FROVOCO**			
	term	**sent**	**merged**	**w5**	**term**	**sent**	**merged**	**w5**
BERT	0.5540	0.6690	0.6582	0.6656	0.5389	0.6665	0.6601	0.6537
ALBERT	0.5523	0.6804	0.6680	0.6576	0.5643	0.6794	0.6780	0.6474
DBERT	**0.5742**	**0.6979**	**0.7171**	**0.7233**	**0.5807**	**0.7060**	**0.7103**	**0.7142**

We highlighted with bold text the best score for each embedding-text span pair, and as we can see from Table 6.3, DistilBERT indeed outperformed both BERT TA and ALBERT TA embedding models for each setup. We can also notice that FRNN-OWA and FROVOCO have similar performances, and that term is usually the weakest text span.

We also compared the time performance for those embedding methods. We applied them to the standalone sentences of each instance and measured the time for the whole dataset and the average time for one instance, given 3,253 instances in this dataset in total. Results are provided in Table 6.4.

Table 6.4: Time of ABSA dataset encoding with various embedding methods.

Embedding	**Time (sec), dataset**	**Avg time (sec), instance**	**Size of embedding**
BERT	241.25	0.074	768
ALBERT	285.68	0.087	768
DBERT	997.09	0.306	768

From Table 6.4, we can see that all considered embeddings provide vectors of the same length for each instance (since they all are BERT-based). Regarding their performance time, we can see that it is almost identical for BERT and ALBERT models and almost four times bigger for DBERT. A possible explanation for this observation could be given by the data types in which the vectors we extract are encoded. Particularly, the BERT and ALBERT models work using the PyTorch[2] package, while DBERT uses TensorFlow[3], which apparently takes more time to process. Despite their speed, in the following experiments, we provide the results only for the DistilBERT model as the best embedding method.

To detect the best setup for each task that we have (category, sentiment, and emotion classification separated into positive and negative emotions), we com-

[2] `https://pytorch.org/`

[3] `https://www.tensorflow.org/`

pared two classification models (FRNN-OWA and FROVOCO), four text spans (term, sentence, window of words and combined vectors of term and sentence) with DistilBERT embedding model, and finally tuned the model parameter k (the number of neighbours). We used the weighted F1-score and the 5-fold CV to evaluate the results based on gold labels. The results for all tasks with their best setups and corresponding F1-scores are provided in Table 6.5.

Table 6.5: The weighted F1-scores for best setups for each individual classification task: category, sentiment and emotion prediction with DistilBERT embedding.

Task	**Model**	**k**	**Text Span**	**F1 CV**
Aspect Main Categories	FRNN-OWA	3	merged	0.9036
Sentiment	FROVOCO	9	w5 whole	0.7289
Positive Emotions	FRNN-OWA	9	merged	0.8273
Negative Emotions	FROVOCO	5	merged	0.7025

In Table 6.5, "Aspect Main Categories" corresponds to the category classification task with six main classes, while "Positive Emotions" and "Negative Emotions" stand for the positive and negative emotion models explained in Section 6.3.1. When we consider the text spans, "merged" means the merged vector of term and sentence, and "w5 whole" stands for the usage of an embedding vector generated for the text span obtained with a window size of five around the target term.

From Table 6.5, we can observe that we obtained the highest F1-score for the category classification task and the lowest for the negative emotion prediction. Regarding the classification model of choice, FRNN-OWA and FROVOCO are both selected as the best classifiers for two of the four classification tasks. During the experiments, we noticed that both classification methods provided quite close results without the clear dominance of one of them. Notably, FRNN-OWA was the best model for the category classification task, which is the most imbalanced one. Meanwhile, FROVOCO performs the best for the sentiment classification, which is represented by a more balanced dataset. Due to the nature of FROVOCO, we would expect opposite results; however, it is hard to make conclusions from these observations since the performances of both methods were quite close.

As for the text spans, we can say that the option of the "merged" vector was always better than the terms and the sentence vectors. A window-of-terms approach only outperformed the merged vector setup once, namely for the sentiment classification task. A window with size five and an embedding taking into account the whole text span was always the strongest approach compared to the other windows' setups.

If we compare the obtained results with baseline scores for wkNN and BOW from Table 6.2, we can see that the F1-scores from Table 6.5 are much better.

Hence, we will not consider the baseline in the pipeline format.

6.3.3 Systems' results

Once we obtained the best setup for each classification task, we combined them in the three systems described in Section 6.3.1. For each system, we measured the F1-score, accuracy, and CCA for each of the three classification tasks: category, sentiment and emotion classification. The results for all three systems are provided in Table 6.6, where # corresponds to the number of the System (1, 2, or 3). Notably, we do not have the results of CCA scores for the category detection task since it is not an ordinal classification task, and we can form a corresponding cost matrix.

Table 6.6: The weighted F1-scores, accuracy, and cost-corrected accuracy for all classification tasks for the three pipeline systems, based on the best individual-task performances.

#	Category			Sentiment			Emotion		
	F1	**Acc**	**CCA**	**F1**	**Acc**	**CCA**	**F1**	**Acc**	**CCA**
1	0.8406	0.8627	-	0.7147	0.6756	0.7268	0.5647	0.4872	0.5697
2	0.8406	0.8627	-	0.7147	0.6756	0.7268	0.6155	0.5598	0.6564
3	0.8406	0.8627	-	0.7740	0.7846	0.8458	0.6851	0.7012	0.8142

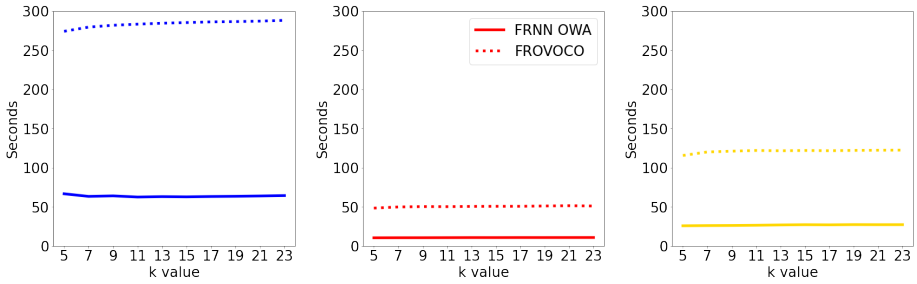
As we can observe from Table 6.6, we received the same scores for category detection for all systems and the same metrics for System 1 and System 2, just like we expected. We can see that System 3, with the separate systems approach, provided the best scores and yielded performances of 86.3% for aspects' categorisation (accuracy), 84.6% for sentiment analysis (CCA) and 81.4% for emotion detection (CCA). When we consider the first two pipeline systems, the results evidently drop because an error in the previous step negatively impacts the following step. Furthermore, the cost-driven filtering of the instances seems to pay off.

We do not compare the obtained results directly with the alternative solution provided in [35] since their scores were provided for the Dutch version of the dataset, which makes direct comparison inaccurate.

There is one important note regarding the statistical significance between the obtained scores. Normally, we would use statistical tests for experiments performed with the same setup on a number of different datasets. However, in our experiments, we deal with one particular dataset, where such tests will not make much sense.

Finally, we wanted to measure the performance time for both considered classification models, FRNN-OWA and FROVOCO. We compared their performance separately for each classification subtask (category, sentiment, and

emotions) since they all have different amounts of classes, with various numbers of neighbours k (Figure 6.6).



(a) Category (b) Sentiment (c) Emotions

Figure 6.6: Time performance for FRNN OWA and FROVOCO models on ABSA dataset for three classification subtasks.

As we can see from Figure 6.6, FROVOCO is nearly five times slower than FRNN-OWA for any classification task. It makes sense, considering how FROVOCO works, performing one-vs.-one splitting of the multiclassification tasks. Both prediction methods take more time for challenges with a larger amount of classes; correspondingly, category classification is the slowest, and sentiment is the fastest task. Also, we again can notice that fluctuations of parameter k do not influence performance speed much.

6.4 Summary

Taking a closer look at Table 6.6, we can confirm our expectations of obtaining higher scores for category prediction, lower for sentiment, and the lowest for emotion detection for System 1 and System 2, since both of them include data reduction steps. With each such step, we eliminate more instances that will allow us to make correct predictions for less amount of data that will reduce our scores. From this perspective, we also can notice that all metrics increase from System 1 to System 3 because each next model keeps more instances for the next step, so the last system provides the highest scores for all tasks.

However, even for System 3, emotion detection performs worse than sentiment and category. It still makes sense because, for categories, we use only the main ones, reducing the number of classes to six. These classes are also very imbalanced, with one huge class “product” dominating all others, so we can assume that predicting this label for the majority of instances will lead to a high score. The dividing of the other five small classes is still a challenge and an area for future work. As for the sentiment classification, we were working with five classes, where two of them have close sizes (“positive” and “negative”) and are much bigger than the three others. It gives us a high score, but it is

still lower than for categories. Finally, for emotions, we are dealing with twelve classes with the biggest "satisfaction", twice smaller "dissatisfaction", and ten more tiny classes. It provides the lower scores, but with System 3, we would say results for emotion detection are sufficient.

Due to the absence of publications at this moment, we can not compare our results with other models on the same dataset yet. However, we can estimate our performance in comparison with scores obtained in [35] for the Dutch version of the dataset. The authors use the same configuration of the F1-score for the test data, particularly the FMCG dataset that we considered. For the main aspect categories classification, where our system achieved an F1-score of 0.8406, Table 2 from [35] provides lower results of 0.798. For sentiment classification, Table 3 from [35] shows an F1-score equal to 0.809, when we received a lower score of 0.7740 with the best setup with System 3. Finally, our system achieves an F1-score of 0.6851 and CCA equal to 0.8142 for emotion detection when Table 4 from [35] - lower F1-score of 0.612 and higher CCA score of 0.848. As we can see, even since these results are provided for the different versions of the same dataset, we still can say that our results were in line (classification of sentiment and emotions) or even slightly higher (classification of main categories).

All provided results were published in [64].

Chapter 7

Interpretability and error analysis

To examine the performance of our proposed approach in more detail and to illustrate the solution's explainability of a fuzzy-rough-based kNN approach, we explore correct and wrong samples of test instances. To detect neighbouring training tweets for the test instance, we, in some sense, went back into our pipeline.

We calculated the cosine similarity between the test tweet and all training tweets for each embedding separately since they provide different locations of instances in the multi-dimensional space. We took into account all embedding approaches used in models of the best ensemble and took the top k closest neighbours for each. The parameter k was previously tuned for each classification model based on a particular embedding method. In this way, we find k neighbours for each of the best embedding methods so that we can check their intersection.

Below, when we present "training neighbours of the test instance", we mean tweets in the intersection of the selected top-k neighbourhoods.

7.1 Emotion datasets

First of all, we computed the confusion matrices for all test datasets (Table 7.1). These scores are the results of the best-selected setups for each dataset from Section 5.1. Particularly as was described in Section 5.1.5, for all emotion datasets, we used the FRNN-OWA model, which provided the highest PCC scores compared to the wkNN and FRNN regression models. The best solutions for all four emotion datasets are presented as the ensemble of four or five models based on embeddings, where RoBERTa, DeepMoji, and USE are used for all datasets, BERT for Anger and Joy, SBERT for Joy, Sadness, and Fear, and Word2Vec - for Anger and Fear (Table 5.22).

Table 7.1: Confusion matrices for emotion test datasets.

True class	**Predicted class**							
	0	1	2	3	0	1	2	3
	Anger				**Joy**			
0	94	267	104	0	87	86	21	0
1	1	50	96	1	26	142	157	8
2	0	40	193	10	1	49	241	69
3	0	4	100	42	0	3	90	125
	Sadness				**Fear**			
0	178	176	44	0	398	230	5	0
1	13	80	93	7	24	94	6	0
2	4	53	172	26	20	112	20	6
3	0	5	72	52	4	27	36	4

In Table 7.1, classes correspond to the original labels from [93] dataset, where '0' means no emotion can be inferred, '1' means that a low amount of emotion can be inferred, '2' means a moderate amount of emotion can be inferred, and '3' - high amount of emotion can be inferred (by emotion we mean one of the four emotions: anger, joy, sadness, or fear). As we can see from obtained confusion matrices, the true class is confused with one of the neighbouring classes in most of the erroneous predictions. For example, in the Anger dataset, the real class '2' is mainly predicted as '2', with minor cases predicted as adjacent classes '1' or '3' and never as '0'.

Remarkably, opposite classes '0' and '3' are rarely confused. The only exception to this pattern is the Fear dataset, where four test samples with true class '3' are labelled as '0'. We examined these four mislabelled samples to understand the nature of the mistake. One of these test tweets, provided at Example 7.1.1, has the majority of closest neighbours from training tweets with topics related to social media and the label '0'.

Example 7.1.1. Things that terrify me: remembering my bf follows me on Twitter.

For example, a neighbour of Example 7.1.1: *"The way I'm always on Twitter at work is a little alarming :woman_facepalming:"* has class '0' (no fear was detected).

Another wrongly predicted Fear test example illustrated at Example 7.1.2 with class '3' was classified with class '0'.

Example 7.1.2. @USER My dad ordered my tickets for the show in Hamburg, his name is now printed on the tickets, is the same surname enough? #panic

Neighbours of Example 7.1.2 are about entertainment-related topics, like tickets: *"@USER why does the ticket website never work? Trying to buy Palace tickets and it's impossible and says there's an error #awful"* (class '0') or the

name: *"thank you for your concern, computer, but my last name isn't misspelled, it's just weird"* (class '0').

For comparison, we also take a closer look at one of the four correctly predicted Fear test samples Example 7.1.3 with the highest level of Fear (class '3').

Example 7.1.3. Ugh going to college tm, so nervous. #college #life #collegelife #newyearnewme.

The closest training neighbours for this sample are mainly related to school and often consist of the word *"nervous"*, for example, *"I have another test tonight #nervous"* with class '3'. For the other three correctly predicted cases with class '3', the situation is similar, where the majority of neighbours share the same word *"nervous"* or *"anxiety"*.

As we can see from the presented samples, having a common topic is a strong feature for neighbour detection. To get a broader picture, we explore more datasets. As a sample of correctly predicted tweets from the Joy dataset with class '3', we can consider the test instance Example 7.1.4.

Example 7.1.4. @USER Happy #blissful birthday.

The majority of Example 7.1.4 neighbours are about birthdays as well, for example, *"@USER happy birthday :) have a blessed day, love from Toronto :) #bday"* with class '3'. In this case, we can consider *"birthday"* not only as a common topic but also as a strong keyword for neighbour detection.

A similar situation we can see for the wrongly predicted Joy test tweet Example 7.1.5 with correct class '1' that was predicted as '3'.

Example 7.1.5. Good Night everyone... #goodnight #sleep #nice #great #night #music #day

Neighbours of Example 7.1.5 are mainly about good night-morning-afternoon wishes, with classes '3' or '2', such as *"Good night, Twitter world! Wish you all good sleep / productive jovial days! :)"* with class '3'.

Another interesting example from the Anger dataset, where the specific keyword plays a bigger role than the common topic, is Example 7.1.6.

Example 7.1.6. @USER I know you mean well but I'm offended. Prick.

It has true class '2', which our system predicted correctly. If we take a look at its neighbours, we can observe an interesting pattern, where the majority of them have the word "offended", such as *"@USER, I'm honestly offended"* (class '2') or *"@USER1 @USER2 @USER3 no! Kinda offended that you had to ask"* (class '1'). The word "offended" is emotionally coloured, which, as we can see, is a strong keyword, which leads to neighbours with proper classes.

So far, we can notice that similar topics have a big influence on neighbours' selection results. This makes sense, taking into account the logic behind the text embedding methods that we are using. Besides a lot of examples being topic-based, we also observed some based on the emotional keyword, which is rare. However, if we take into account related words, such as [125], we can see that these topics (e.g. a topic about "dentists") were an important key to irony recognition.

7.2 Hate speech and irony-based datasets

For the Hate Speech, Offensive Language, Irony, and Sarcasm datasets, the confusion matrices are shown in Table 7.2, where '0' corresponds to the non-hateful (non-ironic) class and '1' - to the hateful (ironic) one. Provided results are based on the solutions described in Section 5.2. Specifically, as was shown in Section 5.2.5, for each of Hate Speech and Irony datasets, the best setup is the FRNN-OWA method based on a standalone RoBERTa embedding.

Table 7.2: Confusion matrices for Hate Speech, Offensive Language, and Irony test datasets.

	Hate Speech		**Offensive**		**Irony**		**Sarcasm**	
True classes	**Predicted classes**							
	0	1	0	1	0	1	0	1
0	486	1254	574	46	455	18	1036	164
1	62	1198	77	163	202	109	102	98

This case looks different because of binary classification. It also seems easier to define and analyze mistakes.

Particularly, the Offensive Language dataset is the only one where the number of correct predictions for both classes is higher than the false positive and false negative predictions. However, the amount of false negative predictions is bigger than false positive. Hence we will take a look at a test instance with gold class '1' and predicted class '0'. Let us consider Example 7.2.1, which is marked as offensive.

Example 7.2.1. #Antifa: Take note of how to protest with civility. This is why socialism (aka communism) will never win. It is inherently evil and unsustainable. URL

We considered the closest neighbours of this instance from the training dataset to find some patterns. We noticed that almost all of them are about politics and movements, using terms such as "authoritarianism", "liberal ideology", "capitalist", "conservatives", and others. Meanwhile, the majority of neighbours are marked as non-offensive, even if they contain some criticism. From this perspective, we also can assume that the fixed test example is rather a criticism than a direct offence, but labellers decided otherwise. This illustrates another time that the hate speech topic is complicated to label due to its subjectivity.

As an example of correctly predicted offensive test instances, we can take a look at Example 7.2.2, which is labelled as non-offensive.

Example 7.2.2. 5 Tips to Enhance Audience Connection on Facebook URL @USER #socialmedia #smm URL

Almost all its neighbours, which were considered by our approach, are non-offensive, so it led to the correct prediction. Although, the pattern that connects

them is not that obvious. We saw a lot of tweets about media, news, and the tech sector, alongside politics. However, almost every neighbour has the tag "URL", when it was not the case for other examples that we saw.

Taking a look at the Hate Speech dataset, we can notice that many tweets are similar and concern hate speech towards immigrants or women, as mentioned in the dataset's description. For example, Example 7.2.3 is a correctly classified hateful test tweet with class '1'.

Example 7.2.3. WAKE UP AMERICA. We cannot continue to allow illegal aliens to stay in County. They are a real and present danger to LEGAL AMERICAN CITIZENS. #BuildThatWall #EndCatchAndReleash #DefundSantuaryCities

The majority of its neighbours are hateful (class '1') and share the hashtag *"#BuildThatWall"*, such as *"Illegal Criminals EVERYWHERE #BuildThatWall !!"*. Hence, this hashtag can be considered a strong keyword.

A possible reason for wrong classifications could be the use of similar topics or words in a different context. For example, the test instance Example 7.2.4 from the Hate Speech dataset has class '1' but was predicted as '0'.

Example 7.2.4. The Last Refuge has a fantastic collection of reports on a business model that profits from illegal immigration. #UniParty #RobbingUsBlind #EndChainMigration #tcot #ccot #pjnet #qanon

The majority of training neighbours of Example 7.2.4 have class '0' and contain words like *"migration"* or *"immigration"*, which are used in an informative rather than hateful sense, such as *"The Truth about #Immigration LINK"* with class '0'.

For the Irony dataset, we can take a look at Example 7.2.5 with true class '1'.

Example 7.2.5. Christmas alone :smiling_face_with_smiling_eyes: how nice #not

Its neighbours are mainly about Christmas, gifts, or winter and are not ironic (class '0'). A sample: *"Yay for days off. #coffee #HarryPotter #christmasbreak #morning LINK"* with class '0'. Hence, we can see that the classifier is misled by Christmas as a strong topic or even a keyword. On the other hand, the hashtag "#not", which for humans is considered a strong indicator of ironic speech, was probably not taken into account because of its generic content.

For the Sarcasm dataset, we first traced back several correctly predicted test tweets. For example, for the sarcastic test tweet Example 7.2.6, we got four sarcastic training neighbours out of five.

Example 7.2.6. So the Scottish Government want people to get their booster shots so badly that the website doesn't even work

These four neighbours were connected to the health topic and contained collocations such as *"mental health"*, *"health insurance"*, *"covid vaccine"*, and *"healthcare"*. The fifth neighbour was about emails that could be connected to

the word *"website"* from the test tweet. From this sample, we could conclude that having a common topic is an important feature for neighbours detection, and our model deals well with it, as we also noticed from exploring other test samples.

As for wrong predictions, we also found an illustrative example for the sarcastic test tweet Example 7.2.7 that was predicted as non-sarcastic.

Example 7.2.7. Sometimes I lay in bed and think about how today will be the day I make my life better. Exercise, drinking water, eating healthy. Then I wake up.

It has four training neighbours about daily routine and lifestyle with mostly non-sarcastic labels, leading to the wrong prediction. For example, the closest training neighbour *"me: I'm gonna wash my hair and shave my legs! Me instead: I'm gonna dissociate in the shower for 45 minutes"* looks pretty similar to the test sample but has a non-sarcastic label. Here, we could highlight again the difficulty of Sarcasm dataset labelling and how subjective it could be.

In conclusion, similar topics and common keywords are strong neighbour detection features on which our approach is based. A similar pattern was observed in related works, such as [125], where similar topics were key to irony recognition. However, we can observe errors with similar words used in a different context. This can also be a result of incorrect annotations or general uncertainty in the author's message.

7.3 ABSA dataset

In order to gain more insights in the errors for the ABSA task, we can take a look at the confusion matrix that we receive at the last step of the first pipeline approach, where we filter out all wrong predictions. We thus consider the confusion matrix for emotion prediction subtask of the ABSA challenge using system #1 from Table 6.6, for which we received an accuracy near 0.49. The described confusion matrix is provided in Table 7.3, where the columns (predicted classes) and rows (true classes) have names that correspond to 12 emotions, *None* represents all incorrectly classified instances from the previous two steps (category and sentiment prediction), and the diagonal consists of a number of correctly predicted instances for each class.

As we can see in Table 7.3, we obtain a lot of correct predictions, mostly for satisfaction (the biggest class). There is also a number of instances predicted wrongly during the previous steps (column *None*). Regarding wrongly predicted emotions, besides the *None* column, we can see that the biggest number of mistakes were made for dissatisfaction predicted as satisfaction (18 instances). If we check the cost of such a mistake in the cost matrix for emotion in Table 4.3, it is 1, which is the highest cost, and that makes sense. Another mistake is anger, classified as dissatisfaction, where we have 13 instances. This cost is 0.5 and, in theory, can be reduced to 0.25, since anger and dissatisfaction could be perceived as close emotions. The same amount of wrong predictions were

Table 7.3: Confusion matrices for emotion detection task of ABSA datasets with the system #1, where we filter out all wrong predictions.

Emotion	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.
1. Anger	9	0	1	13	4	0	0	0	1	1	0	0	10
2. Anticipation	0	0	0	0	0	0	1	0	0	0	0	0	1
3. Disgust	0	0	3	2	0	0	0	1	0	0	0	0	1
4. Dissatisfaction	8	0	3	72	2	1	0	1	1	18	0	0	57
5. Distrust	3	0	0	9	31	0	0	0	1	0	0	0	19
6. Fear	0	0	0	2	2	1	0	0	0	0	0	0	0
7. Joy	0	0	0	1	0	0	14	0	0	12	0	3	23
8. Neutral	0	0	0	0	0	0	0	0	0	0	0	0	14
9. Sadness	3	0	0	4	1	0	0	0	4	1	0	0	3
10. Satisfaction	0	0	0	7	0	0	9	3	0	223	0	0	98
11. Surprise	0	0	1	0	0	0	0	0	0	3	0	0	1
12. Trust	0	0	0	1	0	0	3	0	0	13	0	5	14
13. None	0	0	0	0	0	0	0	0	0	0	0	0	0

made for trust classified as satisfaction, which are both positive emotions with the same cost of 0.5. Fewer mistakes (12 instances) were made for joy, classified as satisfaction. Those emotions are even closer but still have a cost of 0.5. To sum up, our mistakes mostly occur for similar emotions. For some of them, we can reduce the cost from 0.5 to 0.25 due to their similarity to improve the cost-corrected accuracy value.

We also took a closer look at particular test instances. First, we considered a test sample that obtained the wrong sentiment prediction for all systems:

Example 7.3.1. In fact, I will likely buy a second.

From here on, for each example, we will label a term with an underline. For Example 7.3.1, its gold aspect main label "product" was correctly predicted by all three systems. However, the gold sentiment label "positive" was misclassified by all systems as "negative". It leads to no emotion prediction for Systems 1 and 2, while System 3 predicted "dissatisfaction", which is the opposite of the gold label "trust", but was in line with the predicted negative sentiment.

Looking at the training neighbours for the sentiment task (Table 7.4), we can observe that most of them represent negative feedback. These examples demonstrate that they all have a common topic - the impression of the product that influences the user's decision to repurchase it or to recommend this product to others. We can suggest that due to the high number of examples where users were dissatisfied with their products, we can have a lot of negative neighbours for our test example, leading to a wrong prediction. Hence, the same topic can be a substantial similarity feature for our approach.

Second, we take a look at a test sample, where the sentiment was wrong with a low cost:

Example 7.3.2. We use these in our bathrooms and kitchen and believe they work very well.

Table 7.4: Training neighbour instances for Example 7.3.1.

The full sentence	**Sentiment**
I plan to return this item and look for a higher quality air purifier.	Negative
I will probably not choose Cottonelle Ultra next time around.	Negative
I would recommend it for sure, but now I do not have this equipment.	Positive
I would HIGHLY suggest you choose another fan, as this one seems to be nothing but one disaster after another.	Very negative

All systems predicted the aspect class "product" correctly, while the gold sentiment label "very positive" was predicted as "positive". Because of that,

Table 7.5: Training neighbour instances for Example 7.3.2.

The full sentence	**Sentiment**
I have worked with them and I think they work well.	Positive
They are very responsive, very professional and very present when we need them.	Very positive
We are very happy with the product, the machines are reliable and perform.	Very positive
They work well and the one we have mounted in our small bathroom helps cut down on the heat in that room, which really builds up as it does not receive air conditioning like the rest of our home.	Positive

System 1 cannot predict any emotion; however, System 2 can because the cost of this mistake is 0.5, and we allowed it. Due to this, System 2 predicts the "satisfaction" emotion, which corresponds to the prediction of System 3 and the gold label.

If we take a look at some of the neighbouring training instances (Table 7.5), we can observe that all neighbours are either "positive" or "very positive", so it is easy to confuse. Meanwhile, we can also notice an interesting pattern – the common thing among those neighbours is not a topic but rather positive words, such as "work well" (the same collocation as in the test example), "very professional", "very happy", and others. We can conclude that words with high

emotional colouring could be a trigger for our similarity algorithm.

Third, we considered a test example where all systems guessed the sentiment correctly:

Example 7.3.3. It has NO SENSOR for odor detection, therefore, it will not automatically change the fan setting if unwelcomed odors were to invade the space.

Table 7.6: Training neighbour instances for Example 7.3.3.

The full sentence	**Emotion**
Unfortunately, upon installing and turning on the air purifier, the output air had a chemical odor smell that is similar to what other reviewers have been describing since 2018.	Satisfaction
The error messages with calibrators are annoying though, because they always show up, and it does not say which error it is.	Dissatisfaction
Due to the characteristics of immunoassay designs, especially in the free ideas in the measurement of two-step methods, interference and transport disturbances of thyroid hormones are reduced.	Dissatisfaction
Your AB and screen choice can be set to activate when you raise your arm, but there is an irritating delay, even when set to the arm raise's sensitive setting.	Anger

While the gold aspect main label "product" and gold sentiment label "negative" were correctly predicted by all three systems, for the emotion label, all systems made a mistake, as instead of the gold emotion label "anger", for each system "dissatisfaction" was predicted. To investigate that, we can take a look at the neighbouring training instances to the corresponding test one in Table 7.6.

While the majority of neighbours have the label "dissatisfaction" (or "anger"), actually, the closest sentence by meaning is the first sentence that is labelled with "satisfaction". However, the content of the first neighbour seems rather disappointing and negatively toned. In this way, we can confirm our preliminary conclusions that the common topic seems to be a strong feature that marks the neighbours. Moreover, we can notice that some training instances can have confusing or even unsuitable labels. We cannot dismiss the fact that similar emotions, such as anger and dissatisfaction, could be easily confused due to the subjectivity of emotions, which again shows the usefulness of the CCA metric.

Finally, we take a look at some test examples that were predicted with a wrong aspect label (gold aspect classes are provided between brackets):

Example 7.3.4. 1. This shampoo did not come spilled, packaged to perfection. ("packaging")

2. There are good sales people. ("personnel")

3. Dishonest company and seller. ("company")

All these texts were predicted as "product", which can be expected due to the huge imbalance of the data. For this reason, we have no predictions for the sentiment and emotion labels from Systems 1 and 2; meanwhile, System 3 predicted them correctly for each sample ("positive" and "satisfaction" for Example 7.3.4(1) and Example 7.3.4(2), and "negative" and "distrust" for Example 7.3.4(3)). What is curious about these examples is that they all are quite short and still have some emotionally strong words that can appear in their neighbours: "perfect" for the first, "good" for the second, and "dishonest" ("horrible", "awful") for the third.

To conclude, we can say that human emotions are very subjective concepts that can lead to contradictory instance labelling and unexpected patterns chosen by the system. By a manual analysis of the output and the nearest neighbours of both the FRNN-OWA and FROVOCO systems, we can inspect which instances lead to a given classification decision, gaining more insights into the underlying data. This not only enables us to find some explanations for classification decisions but can also aid us in pinpointing errors, shortcomings or even biases in the underlying data as a basis for improving our future work. As another potential direction, we can consider giving more weight to the keywords of each tweet, instead of reducing their size. Also, we can check dependencies between specific keywords and correct or incorrect predictions we receive, to construct a rule for usage in rule-based systems. In any case, the first step of keyword extraction should be automatized with, for example, BERT-based models, such as topic-BERT or tBERT [101].

Chapter 8

Conclusion

In this chapter, we will provide a summary of our thesis (Section 8.1) alongside thoughts on how it can be extended in future (Section 8.2). For future work, we consider several main directions of our study, including additional data manipulations (Section 8.2.1) and exploration of explainability (Section 8.2.2).

8.1 Summary

In this manuscript, we have evaluated the potential of interpretable ML methods based on FRS for different subjective language classification tasks and demonstrated that they are competitive with more complex state-of-the-art NN-based approaches. We also dedicated part of our study to the exploration of the interpretability of our FRS-based approach.

In the four introductory chapters, we discussed the development of the emotion detection task in the NLP field alongside related works (Chapters 1 and 2) and the theoretical background of the thesis (Chapters 3 and 4), to lay the foundation for our experimental work from Chapter 5 onwards. The main contributions of our dissertation are the following:

- In Chapter 5, we tuned and optimized weighted ensembles of wkNN models, FRNN-OWA classification and FRNN regression for emotion intensity, hate speech and irony detection tasks. Our solution uses feature vectors obtained from different word embeddings, which are mostly sentiment-oriented and applied at the sentence level. For all these tasks, FRNN-OWA showed the highest results, providing us with the second place for the emotion intensity task and TOP-9 places for the hate speech and irony detection competitions. These leaderboards also gave us an opportunity to compare our performance with the best solution to these challenges.
- In Chapter 6, we have considered FRS based techniques for the task of ABSA, which in our setup consists of three subtasks: aspect, sentiment and

emotion classification. We tackled them using a pipeline approach, where for each of these subtasks, we implemented the FRS based methods FRNN-OWA and FROVOCO using transformers-based text embeddings, where our solution obtained good results on the test data. To compare our results, we considered the paper [35], where the authors used the Dutch version of the same dataset. We used the same evaluation metrics, such as F1-score and CCA, so we could observe that our results are in line (for sentiment and emotions) or even slightly higher (main aspects classification).

- In Chapter 7, we dived into the topic of the interpretability of our method. Our approach is explainable in a way that we can trace back the test instance and find the training instances that determined the predicted class to explore some patterns. The error analysis revealed that our methods are capable of identifying useful patterns that can explain their predictions. For example, we observed the significance of the tweet topic and even of particular keywords.

To highlight the motivation of our study, we can take another look at the explainable solutions discussed in Section 2.3, all of them having their weaknesses. Particularly, the post-hoc approaches that typically detect relations between changes in the model's inputs and outputs may also ignore non-obvious interactions among the first ones. The self-explanatory methods that develop explanations in the process of model training also need prior expert knowledge or at least annotated data that will be used to guide the learning procedure. Taking into account described disadvantages of these explainable approaches, we can conclude that there is room for improvement and simplification of such methods. The fuzzy rough based methodology proposed in this thesis aims to fill this gap with a simple and explainable solution.

To summarize, our main contribution consists in the investigation of interpretable instance-based fuzzy rough methods to NLP-related tasks, and more particularly to emotion, hate speech, irony detection and ABSA. This area, as well as the usage of DL and BERT embedding techniques with the fuzzy-rough approaches, were not investigated much before.

8.2 Future work

The results described in this thesis still offer room for improvement, and below we will highlight several aspects that can be investigated and boosted to obtain better results.

8.2.1 Data manipulations

As an initial idea, we can assume that the provided solution may be improved by additional text preprocessing techniques. In our experiments, we saw how text cleaning and stop word removal helped to improve the score of models in some setups. It naturally leads to the assumption that more advanced text

preprocessing techniques can improve model performance even more. For future work, we can suggest the following text preprocessing approaches:

- Part-Of-Speech (POS) tagging (see e.g. [36]), which consists of an identification for each token in a sentence of its grammatical structure (noun, verb, adverb, and so on). For example, we can identify adjectives in the text and give them more weight during the vectorization process, with the assumption that they will highlight the mood hidden in it.
- NER (for example, used in [148]), which classifies nouns in text into one of several categories, such as organizations, person names, or locations. Similarly to POS tagging, with NER, we can identify and highlight the more important terms in the text, which in this case happen to be nouns that could be useful for, in particular, aspect categorization in the ABSA task.
- Chunking (see e.g. [59]), is assembling in a text all the consecutive words that make a meaningful phrase, for example, "good breakfast". This technique could be especially useful for the ABSA challenge to group an aspect term with an adjective that corresponds to the hidden sentiment or emotion.
- Synonym replacement (as was done in [132]), which represents a process of replacing some words in the text with their synonyms. Usually, it is performed to improve the coherence of text and reduce the sparsity of data in general. In our case, we can use it to generate better-representing embedding vectors. For example, to replace adjectives with more emotional-coloured synonyms to highlight the mood of the text. Additionally, we can replace verbs and nouns with their synonyms as well.

In the next step, we can take a closer look at the embedding methods we explored. Particularly, for all emotion detection tasks, pre-trained RoBERTa showed outstanding results, as well as DistilBERT for the ABSA task. We can assume that fine-tuning some BERT-based model, for example, variants of RoBERTa, can provide us with a new strong text vectorization method to use. Fine-tuning (see e.g. [100]) can be done on training samples from the datasets considered in this thesis, or also include the usage of additional datasets. In the latter case, for better results, they should be close to the ones we used.

One more aspect that can be considered during the model training is the boundary region of predicted classes. This region is one of the areas defined by rough set theory, and it contains objects whose membership in the class is vague and has the greatest degree of uncertainty. In general, we can say that a close lower and upper approximation membership for some instances shows that the model is confident about their belonging to the particular class and vice versa. There could be some tweets in this boundary region whose classification is unclear, so we can suggest a special way to treat them differently. For example, we can check membership degrees of training tweets neighbours of targeted test text to see how far they are and how sure our prediction is. Those

scores for training tweets can be calculated during the training process and saved to be inspected afterwards. For test instances with wrong predictions, such information could contribute to the explainability process significantly.

Another important dataset characteristic that influences our results is its imbalance. We were able to observe it for the Fear dataset, which is the most imbalanced one among all emotion intensity detection datasets. One more noticeable example is aspect categorization for the ABSA challenge, where one class dominates all others. In our work, we considered the FROVOCO method, which was designed for multi-class imbalanced datasets, to tackle the imbalance issue. As we observed in Section 6.3.2, FROVOCO performed on a similar level with FRNN-OWA for the ABSA task. However, the subtasks where FROVOCO was dominant were the most imbalanced ones. It leads us to the idea of FROVOCO model tuning and improvement in future work to use its imbalance-related characteristic on the full scale.

For more experiments with imbalanced datasets, we can consider the usage of imbalanced ML classification methods such as:

- Oversampling (as, for example, was done in [69]), which is the generation of more training instances for the smaller class. It can be performed on the vector level by adding various linear combinations of already existing text representation vectors. Alternatively, and preferably, we can change not the vectors but text and generate new tweets. To modify the original tweets, we can, for example, replace several words with their synonyms or use a rephrasing tool, including ones based on the GPT model. In this way, by changing the text itself, we can manually check the quality of transformation on the intrinsic level (is the meaning saved) and on the extrinsic level (is the performance of the model improved). Hence, the new text will make sense during error analysis since we can always refer to the original tweet on which it was based.

- Anomaly detection (see e.g. [10]), which entails employing unsupervised learning techniques to identify the minority class as an abnormality or outlier and treating it as such.

- Cost-sensitive learning (for example, performed in [77]), which includes altering the cost function of the classification method to punish minority class misclassification mistakes more severely than majority class classification errors.

In general, dealing with imbalanced data is a challenging task that was not solved yet for the emotion detection and sentiment analysis domain and has a lot of room for improvement.

8.2.2 Further exploration of explainability

As one of the main future challenges, we consider a more systematic study of our approach's explainability. For example, the authors of [32] provide several hints on how to do this:

- First-derivative saliency (see e.g. [7]), which works with gradient-based explanations. It calculates the partial derivative of the acquired output with respect to the input to determine the contribution of each input instance to the final prediction. This approach can be utilized to provide feature importance explainability, particularly on word- and token-level features.
- Input perturbations (for example, in [5]), suggested along with LIME in [112]. By creating random input perturbations and training an explainable model (often a linear one), this method may explain the output given an input example.
- Attention mechanisms (see e.g. [83]), which can be added to most NN designs and, because of their appeal to human intuition, can assist in revealing on what the model is "focusing" by acting less as an operation and more as a technique to enable the NN to explain predictions. In our case, we can consider this solution for embedding models, especially if we will fine-tune a new one.

Moreover, we can examine the application of fuzzy rule-based methods [25, 73] on top of the set of nearest neighbours using high-level features. For example, the Fuzzy Rule-Based k Nearest Neighbours (FRKNN) classifier from [73] contains a set of fuzzy rules that generates particular output for each input instance with a magnitude that determines the class membership of the targeted instance. The authors provide thoughts regarding the main questions that should be addressed for rule-based systems: the number of rules, their source, and usage of rules for the class decision. In general, such methods may further enhance explainability and provide us with new insights from the introduced rules.

It is important to say that neither global nor local methods outperform each other since they both have different strengths. In future, such methods can be combined to supplement each other. For example, the kNN-based local method could discern some pattern in the data, which could be confirmed or rejected with some global method.

Another approach we consider for future work is an application of the extracted patterns from our models to improve the next ones. As was shown in Section 7, for a majority of provided test samples, the neighbouring training texts contain similar keywords or belong to the same topic. However, such characteristics of neighbours cannot guarantee the correct prediction since they can belong to the same topic but have different intensities of the emotion (or be part of another class). This observation made us think about the importance of correct weights for all neighbours considered for the final prediction.

To investigate this direction, we decided to concentrate on the emotion detection task and considered one of the test instances (Example 8.2.1) from the Anger dataset.

Example 8.2.1. Shitty is the worst feeling ever #depressed #anxiety

This instance has the label '3' (a high amount of anger can be inferred); meanwhile, our solution for the Anger dataset (based on FRNN-OWA and dis-

cussed in Section 5.1.5) predicted the label '0'. Considering the number of nearest neighbours k equal to five, we take a closer look at neighbours of Example 8.2.1 from the training dataset provided in Table 8.1. The first neighbours in the table are the closest ones to the targeted test sample Example 8.2.1.

We can notice that all provided examples are about the intense emotions of users or about expressing these emotions. Their classes are below the gold class '3', although we have two neighbours with class '2', which, preferably, should be given larger weights in the prediction process. In other words, the instances with class '2' should be assigned the highest weight for Example 8.2.1 and the instances with class '0' should receive the lowest weight. But with our current approach, the first neighbour with class '0' receives the highest weight. The weights produced by our model and provided in Table 8.1 are based on normalized cosine similarity scores, which, as we can notice, are quite similar. Our idea is to use the discovered patterns, in this case, the emotions of users, to transform the weights for the original training neighbours to obtain a more accurate prediction for the test instance.

Table 8.1: Training neighbour instances for Example 8.2.1.

The full sentence	**Weights**	**Class**
@USER > huff louder	0.2306	0
I miss doing nothing someone I care about and attacking their face with kisses #foreveralone #bitter :weary_face:	0.1968	2
Yay bmth canceled Melbourne show fanfuckingtastic just lost a days pay and hotel fees not happy atm #sad	0.1937	2
When you burst out crying alone and u realize that no one truly knows how unhappy you really are because you don't want anyone to know	0.1934	1
@USER No idea, just exhausted and I was tidying my room and just burst out crying ... fs	0.1853	0

To apply the common topic concept, we want to see how the updated neighbours' texts can improve their weights for a prediction. Our idea is to shorten the original text of the training instance by keeping the collocations of tokens with more relevant words to the topic. In Table 8.2, we provide a reduced version of the same neighbours from Table 8.1, where for each instance, we manually select from two to four words that, in our opinion, express the key idea of the text - the emotional state of the author, which is the common topic we are aiming for. Further, we will discuss how this text reduction procedure can be automated.

After the reduction of the text, we update our embeddings for these new

texts to obtain new vectors and recalculate the distances between neighbours and the target test instance Example 8.2.1. The updated neighbours' texts and weights are listed in Table 8.2, ranked by the normalized similarity score to the test instance. As we can see, now we have neighbours with classes '2' at the top and instances with class '0' at the bottom of the list that makes sense regarding the gold label '3' of the test instance.

Table 8.2: Updated training neighbour instances for Example 8.2.1.

The full sentence	**Weights**	**Class**
not happy atm #sad	0.2469	2
#foreveralone #bitter :weary_face:	0.2465	2
crying alone unhappy	0.2448	1
exhausted burst out crying	0.1909	0
huff louder	0.0709	0

When we use updated distances to calculate a new prediction for Example 8.2.1, we receive a class '1'. This result is still incorrect if we compare it with a gold class '3'; however, given the closest neighbours that we have in this case, it would be quite difficult or even impossible to receive a correct prediction. Also, we should have in mind the subjectivity of emotion dataset labelling. From this perspective, the new prediction of class '1' is closer to the truth than the previous prediction of class '0'. Such an update, especially for not one but for a bunch of incorrectly predicted instances, can improve the evaluation score of our model. This difference can be clearly visible when using CCA as an evaluation metric, where the cost of misclassification of gold class '3' with predicted class '0' is equal to 1 while such cost for predicted class '1' is lower - 0.66 (Table 4.1).

To conclude this example, we can say that usage of the patterns discovered between neighbouring training instances and the test instance can be useful on the local level. Specifically, when we are able to determine the common topic among them and filtrate neighbours' texts to keep only the most important for this topic words. This procedure can influence the vector representation of neighbours, their distance to the targeted test text, their weights in the prediction model, and so - it can change the prediction and improve it. Meanwhile, to take this improvement from the local level to the global and update prediction to the larger part of incorrectly predicted instances, this solution should be scaled. Particularly, the process of neighbours' text reduction should be automated, which could be a challenging task. Also, the strongest embedding methods considered in our experiments did not require much text preprocessing. For example, the standalone RoBERTa model did not use any preprocessing for

the emotion detection with wkNN classification method (Table 5.4) and only the basic text preprocessing for FRNN-OWA (Table 5.8) and FRNN regression (Table 4.3) methods. Still, the suggested technique of text reduction can be seen not as preprocessing but rather as a highlight of the most relevant to the detected topic words in the tweet. Since such models as RoBERTa are context-based, we can perceive text reduction as removing the irrelevant to the main context parts and focusing on the key ideas. However, this suggestion should be verified on a larger amount of examples.

In future work, we can consider for automated text reduction the usage of topic modelling techniques, for example, transformer-based models such as BERTopic [47] or tBERT [101] or keyword extraction (see e.g. [124]) for the test instance and its neighbours. Once keywords will be defined, we can keep them with some window of tokens around from the original neighbouring text and neglect the rest of the words. As was mentioned before, this idea should be experimentally validated as part of our future work.

Appendices

Appendix A

Software overview

In this appendix, we describe the software that was used to generate experiments for this thesis. Since this work is based on five papers published in conferences and journals, we have GitHub repositories for each of them. Three papers out of five, particularly [66], [67], and [65], are located on the same GitHub repository but on different branches. It was done in this way because they all investigate similar datasets using variations of the FRNN-OWA techniques. Meanwhile, [68] and [64] have separate repositories, where the first describes the wkNN approach and the second - the ABSA task, which is different from others.

The purpose of these repositories is to show how exactly our experiments were performed and give readers the opportunity to repeat the proposed setup. In the following Sections, we present an overview of each repository separately.

A.1 wknn_emotion_detection

This repository[1] contains the code for the paper "Nearest neighbour approaches for Emotion Detection in Tweets" [68].

This repository contains the following parts:

- The **code** folder contains *.py* files with different functions:
  - *preprocessing.py* - functions for data uploading and preparation;
  - *embeddings_and_lexicons.py* - functions for tweets embeddings with different methods and lexicons;
  - *wknn_eval.py* - functions for wkNN approach and cross-validation evaluation.
- The **data** folder contains *README_data_download.md* file with instructions on uploading necessary dataset files that should be saved in the **data** folder.

[1] `https://github.com/olha-kaminska/wknn_emotion_detection`

- The **lexica** folder contains *README_lexicons_download.md* file with instructions on uploading necessary lexicons files that should be saved in the **lexica** folder.
- The **model** directory contains *README_model_download.md* file with instructions on uploading necessary models that should be saved in the **model** folder.
- The file *Example.ipynb* provides an overview of all functions and their usage on the example of the Anger dataset. It is built as a pipeline described in the paper with corresponding results.
- The file *requirements.txt* contains the list of all necessary packages and versions used with the Python 3.7.4 environment.
- The file *WASSA2021_poster_Olha_Kaminska.pdf* contains a poster that was presented for this paper at WASSA 2021.

A.2 frnn_emotion_detection

This repository[2] contains three branches with the code for three papers:

- The **main** branch[3]: "Fuzzy-Rough Nearest Neighbour Approaches for Emotion Detection in Tweets" [66];
- The **iSarcasmEval** branch[4]: "LT3 at SemEval-2022 Task 6: Fuzzy-Rough Nearest neighbour Classification for Sarcasm Detection" [67];
- The **emotions_irony_hatespeech** branch[5]: "Fuzzy Rough Nearest Neighbour Methods for Detecting Emotions, Hate Speech and Irony" [65].

The **main** branch contains the following parts:

- The **code** folder contains *.py* files with different functions:
  - *data_preprocessing.py* - functions for data uploading and data preperation;
  - *frnn_owa_eval.py* - functions for FRNN-OWA approach and cross-validation;
  - *tweets_embedding.py* - functions for tweets embeddings with different methods.
- The **data** folder contains *README_data_download.md* file with instructions on uploading necessary datasets.

[2] https://github.com/olha-kaminska/frnn_emotion_detection
[3] https://github.com/olha-kaminska/frnn_emotion_detection/tree/main
[4] https://github.com/olha-kaminska/frnn_emotion_detection/tree/iSarcasmEval
[5] https://github.com/olha-kaminska/frnn_emotion_detection/tree/emotions_irony_hatespeech

- The **model** directory contains *README_model_download.md* file with instructions on uploading necessary models that should be saved in the **model** folder.
- The file *Test.ipynb* provides an overview of all function and their usage. It is built as a pipeline described in the paper with corresponding results.
- The file *requirements.txt* contains the list of all necessary packages and versions used with the Python 3.7.4 environment.

The **iSarcasmEval** branch contains the following parts:

- The **code** folder contains *.py* files with different functions:
  - *data_preprocessing.py* - functions for data uploading and data preperation;
  - *frnn_owa_eval.py* - functions for FRNN-OWA approach and cross-validation;
  - *tweets_embedding.py* - functions for tweets embeddings with different methods.
- The **data** folder contains *README_data_download.md* file with instructions on uploading necessary datasets.
- The **model** directory contains *README_model_download.md* file with instructions on uploading necessary models that should be saved in the **model** folder.
- The file *iSarcasmEval.ipynb* provides an overview of our pipeline during problem-solving.
- The file *test_labels.txt* is the final file with test labels that we provided to the organisers.
- The file *requirements.txt* contains the list of all necessary packages and versions used with the Python 3.7.4 environment.

The **emotions_irony_hatespeech** branch contains the following parts:

- The **code** folder contains *.py* files with different functions:
  - *data_preprocessing.py* - functions for data uploading and data preperation;
  - *frnn_owa_eval.py* - functions for FRNN-OWA approach and cross-validation;
  - *tweets_embedding.py* - functions for tweets embeddings with different methods.
- The **data** folder contains *README_data_download.md* file with instructions on uploading necessary datasets.

- The **model** directory contains *README_model_download.md* file with instructions on uploading necessary models that should be saved in the **model** folder.
- The file *Example.ipynb* provides an overview of all function and their usage. It is built as a pipeline described in the paper with corresponding results. As an example, it uses one out of seven datasets that we considered the Hate Speech dataset.
- The file *requirements.txt* contains the list of all necessary packages and versions used with the Python 3.7.4 environment.

A.3 frnn_absa

This repositoryh[6] contains the code for the paper "Fuzzy Rough Nearest Neighbour Methods for Aspect-Based Sentiment Analysis" [64].

The following components make up this repository:

- The **code** directory contains *.py* files with different functions:
  - *preprocessing.py* - functions for data uploading and data preparation and for tweets embeddings;
  - *fuzzy_eval.py* - functions for fuzzy-rough-based approach and cross-validation evaluation;
  - *systems.py* - function to perform pipeline-based solutions.
- The **data** directory contains:
  - *README_data_download.md* file with instructions on uploading necessary dataset files;
  - *sentiment_cost.json* file with cost matrix for the sentiment dataset;
  - *emotion_cost.json* file with cost matrix for the emotion dataset.
- The file *Example.ipynb* provides an overview of all functions and their usage on the example of our dataset. It is built as a pipeline described in the paper with corresponding results.
- The file *requirements.txt* contains the list of all necessary packages and versions used with the Python 3.7.4 environment.

[6] `https://github.com/olha-kaminska/frnn_absa`

Appendix B

List of publications

Papers in international journals listed in the Science Citation Index

- Olha Kaminska, Chris Cornelis, and Veronique Hoste. "Fuzzy Rough Nearest Neighbour Methods for Aspect-Based Sentiment Analysis". In: Electronics 12.5 (2023). issn: 2079-9292. doi: 10.3390/electronics12051088. url: https://www.mdpi.com/2079-9292/12/5/1088.
- Olha Kaminska, Chris Cornelis, and Veronique Hoste. "Fuzzy rough nearest neighbour methods for detecting emotions, hate speech and irony". In: Information Sciences 625 (2023), pp. 521–535. issn: 0020-0255. doi: https://doi.org/10.1016/j.ins.2023.01.054. url: https://www.sciencedirect.com/science/article/pii/S0020025523000543.

Conference proceedings

- Olha Kaminska, Chris Cornelis, and Veronique Hoste. "Fuzzy-Rough Nearest Neighbour Approaches for Emotion Detection in Tweets". In: International Joint Conference on Rough Sets. Springer. 2021, pp. 231–246
- Olha Kaminska, Chris Cornelis, and Veronique Hoste. "LT3 at SemEval-2022 Task 6: Fuzzy-Rough Nearest Neighbor Classification for Sarcasm Detection". In: Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022). 2022, pp. 987–992
- Olha Kaminska, Chris Cornelis, and Veronique Hoste. "Nearest neighbour approaches for Emotion Detection in Tweets". In: Proc. 11th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis. 2021, pp. 203–212

Bibliography

[1] Ibrahim Abu Farha et al. “SemEval 2022: iSarcasmEval - Intended Sarcasm Detection in English and Arabic”. In: *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. July 2022.

[2] Ibrahim Abu Farha et al. “SemEval-2022 Task 6: iSarcasmEval, Intended Sarcasm Detection in English and Arabic”. In: *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 802–814. DOI: 10.18653/v1/2022.semeval-1.111. URL: https://aclanthology.org/2022.semeval-1.111.

[3] Amina Adadi and Mohammed Berrada. “Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)”. In: *IEEE access* 6 (2018), pp. 52138–52160.

[4] Ramya Akula and Ivan Garibay. “Explainable Detection of Sarcasm in Social Media”. In: *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Apr. 2021, pp. 34–39.

[5] David Alvarez-Melis and Tommi S Jaakkola. “A causal framework for explaining the predictions of black-box sequence-to-sequence models”. In: *arXiv preprint arXiv:1707.01943* (2017).

[6] Jason Angel, Segun Aroyehun, and Alexander Gelbukh. “TUG-CIC at SemEval-2021 Task 6: Two-stage Fine-tuning for Intended Sarcasm Detection”. In: *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. 2022, pp. 951–955.

[7] Malika Aubakirova and Mohit Bansal. “Interpreting neural networks to improve politeness comprehension”. In: *arXiv preprint arXiv:1610.02683* (2016).

[8] Muhammad Awais et al. “LSTM-based emotion detection using physiological signals: IoT framework for healthcare and distance learning in COVID-19”. In: *IEEE Internet of Things Journal* 8.23 (2020), pp. 16863–16871.

[9] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

[10] VS Bakkialakshmi and T Sudalaimuthu. "Anomaly Detection in Social Media Using Text-Mining and Emotion Classification with Emotion Detection". In: *Cognition and Recognition: 8th International Conference, ICCR 2021, Mandya, India, December 30–31, 2021, Revised Selected Papers*. Springer. 2023, pp. 67–78.

[11] Francesco Barbieri et al. "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Nov. 2020, pp. 1644–1650.

[12] Valerio Basile et al. "Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter". In: *13th International Workshop on Semantic Evaluation*. 2019, pp. 54–63.

[13] Christos Baziotis et al. "NTUA-SLP at SemEval-2018 Task 3: Tracking Ironic Tweets using Ensembles of Word and Character Level Attentive RNNs". In: *Proc. 12th International Workshop on Semantic Evaluation*. June 2018, pp. 613–621.

[14] Dario Bertero and Pascale Fung. "A first look into a convolutional neural network for speech emotion detection". In: *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2017, pp. 5115–5119.

[15] Susann Boy, Dana Ruiter, and Dietrich Klakow. "Emoji-Based Transfer Learning for Sentiment Tasks". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*. Apr. 2021, pp. 103–110.

[16] Margaret M Bradley and Peter J Lang. *Affective norms for English words (ANEW): Instruction manual and affective ratings*. Tech. rep. Technical report C-1, the center for research in psychophysiology, 1999.

[17] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[18] Daniel Cer et al. "Universal Sentence Encoder for English". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Nov. 2018, pp. 169–174. DOI: 10.18653/v1/D18-2029. URL: https://www.aclweb.org/anthology/D18-2029.

[19] Daniel Chandler and Rod Munday. *A dictionary of media and communication*. OUP Oxford, 2011.

[20] Guimin Chen, Yuanhe Tian, and Yan Song. "Joint Aspect Extraction and Sentiment Analysis with Directional Graph Convolutional Networks". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 272–279. DOI: 10.18653/v1/2020.coling-main.24. URL: https://aclanthology.org/2020.coling-main.24.

[21] Hanjie Chen and Yangfeng Ji. "Learning variational word masks to improve the interpretability of neural text classifiers". In: *arXiv preprint arXiv:2010.00667* (2020).

[22] Anandan Chinnalagu and Ashok Kumar Durairaj. "Context-based sentiment analysis on customer reviews using machine learning linear models". In: *PeerJ Computer Science* 7 (2021), e813.

[23] Andrea Chiorrini et al. "Emotion and sentiment analysis of tweets using BERT". In: *EDBT/ICDT Workshops*. 2021.

[24] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[25] TeckWee Chua and WoeiWan Tan. "A new fuzzy rule-based initialization method for K-Nearest neighbor classifier". In: *2009 IEEE International Conference on Fuzzy Systems*. 2009, pp. 415–420. DOI: 10.1109/FUZZY.2009.5277215.

[26] Kevin Clark et al. "Electra: Pre-training text encoders as discriminators rather than generators". In: *arXiv preprint arXiv:2003.10555* (2020).

[27] Alexis Conneau et al. "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 670–680. URL: https://www.aclweb.org/anthology/D17-1070.

[28] Alexis Conneau et al. "Unsupervised cross-lingual representation learning at scale". In: *arXiv preprint arXiv:1911.02116* (2019).

[29] Thomas Cover and Peter Hart. "Nearest neighbor pattern classification". In: *IEEE transactions on information theory* 13.1 (1967), pp. 21–27.

[30] Danilo Croce, Daniele Rossini, and Roberto Basili. "Auditing deep learning processes through kernel-based explanatory models". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 4037–4046.

[31] W Daelemans. "Van den Bosch. 2005. Memory-Based Language Processing". In: *Cambridge: Cambndge UniversityPress* (85).

[32] Marina Danilevsky et al. “A Survey of the State of Explainable AI for Natural Language Processing”. In: *Proc. 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*. Dec. 2020, pp. 447–459.

[33] Luna De Bruyne and Orphée De Clercq. “Prospects for Dutch Emotion Detection: Insights from the New EmotioNL Dataset”. In: *Computational Linguistics in the Netherlands Journal* 11 (2022), pp. 231–255.

[34] Luna De Bruyne, Orphée De Clercq, and Véronique Hoste. “Emotional RobBERT and insensitive BERTje: combining transformers and affect lexica for Dutch emotion detection”. In: *Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA), held in conjunction with EACL 2021*. Association for Computational Linguistics. 2021, pp. 257–263.

[35] Ellen De Geyndt et al. “SentEMO: A Multilingual Adaptive Platform for Aspect-based Sentiment and Emotion Analysis”. In: *12th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis, collocated with ACL 2022*. Association for Computational Linguistics. 2022, pp. 51–61.

[36] Bart Desmet and Véronique Hoste. “Emotion detection in suicide notes”. In: *Expert Systems with Applications* 40.16 (2013), pp. 6351–6358.

[37] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[38] Yunxia Ding, Xiaobing Zhou, and Xuejie Zhang. “YNU_DYX at SemEval-2019 Task 5: A Stacked BiGRU Model Based on Capsule Network in Detection of Hate”. In: *Proc. 13th International Workshop on Semantic Evaluation*. June 2019, pp. 535–539.

[39] MP Dubuisson and AK Jain. “A modified Hausdorff distance for object matching”. In: *International Conference on Pattern Recognition*. Vol. 1. IEEE. Jerusalem, Israel, 1994, pp. 566–568.

[40] Sahibsingh A Dudani. “The distance-weighted k-nearest-neighbor rule”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 4 (1976), pp. 325–327.

[41] Venkatesh Duppada, Royal Jain, and Sushant Hiray. “SeerNet at SemEval-2018 Task 1: Domain Adaptation for Affect in Tweets”. In: *Proc. 12th International Workshop on Semantic Evaluation*. June 2018, pp. 18–23.

[42] Paul Ekman. “Biological and cultural contributions to body and facial movement”. In: *The anthropology of the body* (1977), pp. 39–84.

[43] Bjarke Felbo et al. "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm". In: *Proc. 2017 Conference on Empirical Methods in Natural Language Processing* (2017).

[44] Grace Gee and Eugene Wang. "psyML at SemEval-2018 Task 1: Transfer learning for sentiment and emotion analysis". In: *Proc. 12th International Workshop on Semantic Evaluation*. 2018, pp. 369–376.

[45] Bilal Ghanem et al. "Irony detection in a multilingual context". In: *Advances in Information Retrieval* 12036 (2020), pp. 141–149.

[46] José Ángel González, Lluís-F. Hurtado, and Ferran Pla. "Transformer based contextualization of pre-trained word embeddings for irony detection in Twitter". In: *Information Processing & Management* 57.4 (2020), p. 102262. ISSN: 0306-4573.

[47] Maarten Grootendorst. "BERTopic: Neural topic modeling with a class-based TF-IDF procedure". In: *arXiv preprint arXiv:2203.05794* (2022).

[48] Riccardo Guidotti et al. "A survey of methods for explaining black box models". In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42.

[49] Raj Kumar Gupta, Ajay Vishwanath, and Yinping Yang. "COVID-19 Twitter dataset with latent topics, sentiments and emotions attributes". In: (2021). DOI: `https://doi.org/10.3886/E120321V11`.

[50] Yaqian Han et al. "X-pudu at semeval-2022 task 6: Multilingual learning for english and arabic sarcasm detection". In: *arXiv preprint arXiv:2211.16883* (2022).

[51] Felix Hausdorff. *Grundzüge der Mengenlehre*. German. Leipzig: Veit & Comp, 1914.

[52] Pengcheng He et al. "Deberta: Decoding-enhanced bert with disentangled attention". In: *arXiv preprint arXiv:2006.03654* (2020).

[53] Matthew Honnibal and Ines Montani. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing". In: *To appear* 7.1 (2017), pp. 411–420.

[54] Minqing Hu and Bing Liu. "Mining and summarizing customer reviews". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 168–177.

[55] Anna Huang. "Similarity measures for text document clustering". In: *Proc. 6th New Zealand computer science research student conference (NZCSRSC2008)*. Vol. 4. 2008, pp. 9–56.

[56] Ali Shariq Imran et al. "Cross-cultural polarity and emotion detection using sentiment analysis and deep learning on COVID-19 related tweets". In: *Ieee Access* 8 (2020), pp. 181074–181090.

[57] Nitin Indurkhya and Fred J Damerau. *Handbook of natural language processing*. Vol. 2. CRC Press, 2010.

[58] Vijayasaradhi Indurthi et al. “FERMI at SemEval-2019 Task 5: Using Sentence embeddings to Identify Hate Speech Against Immigrants and Women in Twitter”. In: *Proc. 13th International Workshop on Semantic Evaluation*. June 2019, pp. 70–74.

[59] Hayeon Jang and Hyopil Shin. “Language-specific sentiment analysis in morphologically rich languages”. In: *Coling 2010: Posters*. 2010, pp. 498–506.

[60] Hamed Jelodar et al. “Deep Sentiment Classification and Topic Discovery on Novel Coronavirus or COVID-19 Online Discussions: NLP Using LSTM Recurrent Neural Network Approach”. In: *IEEE Journal of Biomedical and Health Informatics* 24.10 (2020), pp. 2733–2742.

[61] Richard Jensen and Chris Cornelis. “Fuzzy-rough nearest neighbour classification and prediction”. In: *Theoretical Computer Science* 412.42 (2011), pp. 5871–5884.

[62] Yichen Jiang et al. “Explore, Propose, and Assemble: An Interpretable Model for Multi-Hop Reading Comprehension”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2714–2725.

[63] Armand Joulin et al. “Bag of Tricks for Efficient Text Classification”. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 2017, pp. 427–431.

[64] Olha Kaminska, Chris Cornelis, and Veronique Hoste. “Fuzzy Rough Nearest Neighbour Methods for Aspect-Based Sentiment Analysis”. In: *Electronics* 12.5 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12051088. URL: https://www.mdpi.com/2079-9292/12/5/1088.

[65] Olha Kaminska, Chris Cornelis, and Veronique Hoste. “Fuzzy rough nearest neighbour methods for detecting emotions, hate speech and irony”. In: *Information Sciences* 625 (2023), pp. 521–535. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2023.01.054. URL: https://www.sciencedirect.com/science/article/pii/S0020025523000543.

[66] Olha Kaminska, Chris Cornelis, and Veronique Hoste. “Fuzzy-Rough Nearest Neighbour Approaches for Emotion Detection in Tweets”. In: *International Joint Conference on Rough Sets*. Springer. 2021, pp. 231–246.

[67] Olha Kaminska, Chris Cornelis, and Veronique Hoste. “LT3 at SemEval-2022 Task 6: Fuzzy-Rough Nearest Neighbor Classification for Sarcasm Detection”. In: *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. 2022, pp. 987–992.

[68] Olha Kaminska, Chris Cornelis, and Veronique Hoste. "Nearest neighbour approaches for Emotion Detection in Tweets". In: *Proc. 11th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 2021, pp. 203–212.

[69] Aditya Kane et al. "Transformer based ensemble for emotion detection". In: *arXiv preprint arXiv:2203.11899* (2022).

[70] Akbar Karimi, Leonardo Rossi, and Andrea Prati. "Adversarial training for aspect-based sentiment analysis with bert". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 8797–8803.

[71] Dilek Küçük and Fazli Can. "Stance detection: A survey". In: *ACM Computing Surveys (CSUR)* 53.1 (2020), pp. 1–37.

[72] Fang-Fei Kuo et al. "Emotion-based music recommendation by association discovery from film music". In: *Proceedings of the 13th annual ACM international conference on Multimedia*. 2005, pp. 507–510.

[73] Arijit Laha. "Building contextual classifiers by integrating fuzzy rule based classification technique and k-nn method for credit scoring". In: *Advanced Engineering Informatics* 21.3 (2007). Applications Eligible for Data Mining, pp. 281–291. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2006.12.004.

[74] Zhenzhong Lan et al. "Albert: A lite bert for self-supervised learning of language representations". In: *arXiv preprint arXiv:1909.11942* (2019).

[75] Oliver Urs Lenz, Daniel Peralta, and Chris Cornelis. "fuzzy-rough-learn 0.1: a Python library for machine learning with fuzzy rough sets". In: *IJCRS 2020: Proc. International Joint Conference on Rough Sets*. Vol. 12179. Lecture Notes in Artificial Intelligence. 2020, pp. 491–499.

[76] Oliver Urs Lenz, Daniel Peralta, and Chris Cornelis. "Scalable approximate FRNN-OWA classification". In: *IEEE Transactions on Fuzzy Systems* 28.5 (2019), pp. 929–938.

[77] Dongdong Li, Yingchun Yang, and Weihui Dai. "Cost-sensitive learning for emotion robust speaker recognition". In: *The Scientific World Journal* 2014 (2014).

[78] Jiwei Li et al. "Visualizing and Understanding Neural Models in NLP". In: *Proceedings of NAACL-HLT*. 2016, pp. 681–691.

[79] Xin Li et al. *Exploiting BERT for End-to-End Aspect-based Sentiment Analysis*. 2019.

[80] Bing Liu. "Sentiment analysis and opinion mining". In: *Synthesis lectures on human language technologies* 5.1 (2012), pp. 1–167.

[81] Ping Liu, Wen Li, and Liang Zou. "NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers". In: *Proc. 13th international workshop on semantic evaluation*. 2019, pp. 87–91.

[82] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[83] Ling Luo et al. “Beyond Polarity: Interpretable Financial Sentiment Analysis with Hierarchical Query-driven Attention.” In: *IJCAI*. 2018, pp. 4244–4250.

[84] Zafer Al-Makhadmeh and Amr Tolba. “Automatic hate speech detection using killer natural language processing optimizing ensemble deep learning approach”. In: *Computing* 102.2 (2020), pp. 501–522.

[85] Yue Mao et al. “A joint training dual-mrc framework for aspect based sentiment analysis”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 2021, pp. 13543–13551.

[86] Ilia Markov et al. “Exploring Stylometric and Emotion-Based Features for Multilingual Cross-Domain Hate Speech Detection”. In: *Proc. 11th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Apr. 2021.

[87] Tomas Mikolov et al. “Advances in Pre-Training Distributed Word Representations”. In: *Proc. International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[88] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. 2013, pp. 3111–3119.

[89] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR* abs/1301.3781 (2013).

[90] Saif Mohammad. “Obtaining Reliable Human Ratings of Valence, Arousal, and Dominance for 20,000 English Words”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 174–184. DOI: 10.18653/v1/P18-1017. URL: https://www.aclweb.org/anthology/P18-1017.

[91] Saif M Mohammad and Peter D Turney. “Crowdsourcing a word–emotion association lexicon”. In: *Computational intelligence* 29.3 (2013), pp. 436–465.

[92] Saif M. Mohammad. “Word Affect Intensities”. In: *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference (LREC-2018)*. Miyazaki, Japan, 2018.

[93] Saif M. Mohammad et al. “SemEval-2018 Task 1: Affect in Tweets”. In: *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*. New Orleans, LA, USA, 2018.

[94] John Morris et al. "TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 119–126.

[95] Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. "A BERT-based transfer learning approach for hate speech detection in online social media". In: *International Conference on Complex Networks and Their Applications*. Springer. 2019, pp. 928–940.

[96] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. "BERTweet: A pre-trained language model for English Tweets". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 9–14. DOI: 10.18653/v1/2020.emnlp-demos.2.

[97] Alex Nikolov and Victor Radivchev. "Nikolov-Radivchev at SemEval-2019 Task 6: Offensive Tweet Classification with BERT and Ensembles". In: *Proc. 13th International Workshop on Semantic Evaluation*. June 2019, pp. 691–695.

[98] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[99] Xuan Ouyang et al. "ERNIE-M: Enhanced multilingual representation by aligning cross-lingual semantics with monolingual corpora". In: *arXiv preprint arXiv:2012.15674* (2020).

[100] Sungjoon Park et al. "Dimensional emotion detection from categorical emotion". In: *arXiv preprint arXiv:1911.02499* (2019).

[101] Nicole Peinelt, Dong Nguyen, and Maria Liakata. "tBERT: Topic models and BERT joining forces for semantic similarity detection". In: *Proceedings of the 58th annual meeting of the association for computational linguistics*. 2020, pp. 7047–7055.

[102] Rosalind W Picard. "Affective computing MIT press". In: *Cambridge, Massachsusetts* (1997), p. 2.

[103] Yolande Piris and Anne-Cécile Gay. "Customer satisfaction and natural language processing". In: *Journal of Business Research* 124 (2021), pp. 264–271. ISSN: 0148-2963.

[104] Maria Pontiki et al. "SemEval-2014 Task 4: Aspect Based Sentiment Analysis". In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 27–35. DOI: 10.3115/v1/S14-2004. URL: https://aclanthology.org/S14-2004.

[105] Maria Pontiki et al. "SemEval-2015 Task 12: Aspect Based Sentiment Analysis". In: *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, June 2015, pp. 486–495. DOI: 10.18653/v1/S15-2082. URL: https://aclanthology.org/S15-2082.

[106] Maria Pontiki et al. "SemEval-2015 task 12: Aspect based sentiment analysis". In: *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*. 2015, pp. 486–495.

[107] Maria Pontiki et al. "SemEval-2016 Task 5: Aspect Based Sentiment Analysis". In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 19–30. DOI: 10.18653/v1/S16-1002. URL: https://aclanthology.org/S16-1002.

[108] Rolandos Alexandros Potamias, Georgios Siolas, and Andreas-Georgios Stafylopatis. "A transformer-based approach to irony and sarcasm detection". In: *Neural Computing and Applications* 32.23 (2020), pp. 17309–17320.

[109] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551.

[110] Enislay Ramentol et al. "IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification". In: *IEEE Transactions on Fuzzy Systems* 23.5 (2014), pp. 1622–1637.

[111] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proc. 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Nov. 2019, pp. 3982–3992.

[112] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should I trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[113] M Alfa Riza and Novrido Charibaldi. "Emotion Detection in Twitter Social Media Using Long Short-Term Memory (LSTM) and Fast Text". In: *International Journal of Artificial Intelligence & Robotics (IJAIR)* 3.1 (2021), pp. 15–26.

[114] Omid Rohanian et al. "WLV at SemEval-2018 Task 3: Dissecting Tweets in Search of Irony". In: *Proc. 12th International Workshop on Semantic Evaluation*. June 2018, pp. 553–559.

[115] Sara Rosenthal, Preslav Nakov, and Svetlana Kiritchenko. "SemEval-2017 Task 4: Sentiment analysis in Twitter". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics. Vancouver, Canada, 2017, pp. 502–518.

[116] Alon Rozental and Daniel Fleischer. "Amobee at SemEval-2018 Task 1: GRU Neural Network with a CNN Attention Mechanism for Sentiment Classification". In: *Proc. 12th International Workshop on Semantic Evaluation*. June 2018, pp. 218–225.

[117] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. "Dynamic routing between capsules". In: *Proceeding of 31st Conference on Neural Information Processing Systems (NIPS 2017)*. 2017.

[118] Mehmet Umut Salur and Ilhan Aydin. "A Novel Hybrid Deep Learning Model for Sentiment Classification". In: *IEEE Access* 8 (2020), pp. 58080–58093.

[119] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2019. DOI: 10.48550/ARXIV.1910.01108. URL: https://arxiv.org/abs/1910.01108.

[120] Boaz Shmueli, Lun-Wei Ku, and Soumya Ray. "Reactive Supervision: A New Method for Collecting Sarcasm Data". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2553–2559. DOI: 10.18653/v1/2020.emnlp-main.201.

[121] Kush Shrivastava, Shishir Kumar, and Deepak Kumar Jain. "An effective approach for emotion detection in multimedia text data using sequence based convolutional neural network". In: *Multimedia tools and applications* 78 (2019), pp. 29607–29639.

[122] Carlo Strapparava and Rada Mihalcea. "Learning to identify emotions in text". In: *Proceedings of the 2008 ACM symposium on Applied computing*. 2008, pp. 1556–1560.

[123] Erik Strumbelj and Igor Kononenko. "An efficient explanation of individual classifications using game theory". In: *The Journal of Machine Learning Research* 11 (2010), pp. 1–18.

[124] Matthew Tang et al. "Progress notes classification and keyword extraction using attention-based deep learning models with BERT". In: *arXiv preprint arXiv:1910.05786* (2019).

[125] Cynthia Van Hee. "Can machines sense irony?: exploring automatic irony detection on social media". PhD thesis. Ghent University, 2017.

[126] Cynthia Van Hee, Els Lefever, and Véronique Hoste. "SemEval-2018 Task 3: Irony Detection in English Tweets". In: *Proc. 12th International Workshop on Semantic Evaluation*. June 2018, pp. 39–50.

[127] Sarah Vluymans et al. "Applications of fuzzy rough set theory in machine learning: a survey". In: *Fundamenta Informaticae* 142.1-4 (2015), pp. 53–86.

[128] Sarah Vluymans et al. "Dynamic affinity-based classification of multi-class imbalanced data with one-versus-one decomposition: a fuzzy rough set approach". In: *Knowledge and Information Systems* 56.1 (2018), pp. 55–84.

[129] Sarah Vluymans et al. "Weight selection strategies for ordered weighted average based fuzzy rough sets". In: *Information Sciences* 501 (2019), pp. 155–171.

[130] Hai Wan et al. “Target-aspect-sentiment joint detection for aspect-based sentiment analysis”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34. 2020, pp. 9122–9129.

[131] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. “Norms of valence, arousal, and dominance for 13,915 English lemmas”. In: *Behavior research methods* 45.4 (2013), pp. 1191–1207.

[132] Jason Wei and Kai Zou. “Eda: Easy data augmentation techniques for boosting performance on text classification tasks”. In: *arXiv preprint arXiv:1901.11196* (2019).

[133] Michael Wiegand and Josef Ruppenhofer. “Exploiting Emojis for Abusive Language Detection”. In: *Proc. 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Apr. 2021, pp. 369–380.

[134] Chuhan Wu et al. “THU_NGN at SemEval-2018 Task 3: Tweet Irony Detection with Densely connected LSTM and Multi-task Learning”. In: *Proc. 12th International Workshop on Semantic Evaluation*. June 2018, pp. 51–56.

[135] Zhiyong Wu et al. “Perturbed masking: Parameter-free probing for analyzing and interpreting BERT”. In: *arXiv preprint arXiv:2004.14786* (2020).

[136] Tong Xiang et al. “ToxCCIn: Toxic Content Classification with Interpretability”. In: *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Apr. 2021, pp. 1–12.

[137] Li Yang, Jin-Cheon Na, and Jianfei Yu. “Cross-modal multitask transformer for end-to-end multimodal aspect-based sentiment analysis”. In: *Information Processing & Management* 59.5 (2022), p. 103038.

[138] Zichao Yang et al. “Hierarchical attention networks for document classification”. In: *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2016, pp. 1480–1489.

[139] Mengfei Yuan et al. “stce at semeval-2022 task 6: Sarcasm detection in english tweets”. In: *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. 2022, pp. 820–826.

[140] Lotfi A Zadeh. “Fuzzy sets”. In: *Information and control* 8.3 (1965), pp. 338–353.

[141] Marcos Zampieri et al. “Predicting the Type and Target of Offensive Posts in Social Media”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 1415–1420. DOI: 10.18653/v1/N19-1144.

[142] Marcos Zampieri et al. "SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval)". In: *Proc. 13th International Workshop on Semantic Evaluation*. June 2019, pp. 75–86.

[143] Jun-hai Zhai. "Fuzzy decision tree based on fuzzy-rough technique". In: *Soft Computing* 15.6 (2011), pp. 1087–1096.

[144] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-Level Convolutional Networks for Text Classification". In: *arXiv:1509.01626 [cs]* (Sept. 2015). arXiv: 1509.01626 [cs].

[145] Anping Zhao and Yu Yu. "Knowledge-enabled BERT for aspect-based sentiment analysis". In: *Knowledge-Based Systems* 227 (2021), p. 107220.

[146] Hong Zhao et al. "Fuzzy Rough Set Based Feature Selection for Large-Scale Hierarchical Classification". In: *IEEE Transactions on Fuzzy Systems* 27.10 (2019), pp. 1891–1903.

[147] Jun Y. Zhao and Zhi L. Zhang. "Fuzzy rough neural network and its application to feature selection". In: *Proc. 4th International Workshop on Advanced Computational Intelligence*. 2011, pp. 684–687.

[148] Lingyun Zhao et al. "A BERT based sentiment analysis and key entity detection approach for online financial texts". In: *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE. 2021, pp. 1233–1238.

[149] Suyun Zhao et al. "Building a Rule-Based Classifier—A Fuzzy-Rough Set Approach". In: *IEEE Transactions on Knowledge and Data Engineering* 22.5 (2010), pp. 624–638. DOI: 10.1109/TKDE.2009.118.

[150] Jian Zhu, Zuoyu Tian, and Sandra Kübler. "UM-IU@LING at SemEval-2019 Task 6: Identifying Offensive Tweets Using BERT and SVMs". In: *Proc. 13th International Workshop on Semantic Evaluation*. June 2019, pp. 788–795.

[151] John Jianjun Zhu et al. "Online critical review classification in response strategy and service provider rating: Algorithms from heuristic processing, sentiment analysis to deep learning". In: *Journal of Business Research* 129 (2021), pp. 860–877. ISSN: 0148-2963.