Highlights

Fuzzy Rough Nearest Neighbour Methods for Detecting Emotions, Hate Speech and Irony

Olha Kaminska, Chris Cornelis, Veronique Hoste

- We apply fuzzy rough nearest neighbour methods to classify textual information based on characteristics such as emotions, hate speech and irony.
- We show that by careful feature engineering and ensemble construction, our methods can achieve similar accuracy as state-of-the-art methods based on deep learning.
- We argue that our proposed approach provides a more interpretable alternative for black box methods based on deep neural networks.

Fuzzy Rough Nearest Neighbour Methods for Detecting Emotions, Hate Speech and Irony

Olha Kaminska[a], Chris Cornelis[a], Veronique Hoste[b]

[a]*Computational Web Intelligence, Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium*
[b]*LT3 Language and Translation Technology Team, Ghent University, Ghent, Belgium*

Abstract

Due to the ever-expanding volumes of information available on social media, the need for reliable and efficient automated text understanding mechanisms becomes evident. Unfortunately, most current approaches rely on black-box solutions rooted in deep learning technologies. In order to provide a more transparent and interpretable framework for extracting intrinsic text characteristics like emotions, hate speech and irony, we propose to integrate fuzzy rough set techniques and text embeddings. We apply our methods to different classification problems originating from Semantic Evaluation (SemEval) competitions, and demonstrate that their accuracy is on par with leading deep learning solutions.

Keywords: Natural Language Processing, emotion detection, fuzzy rough sets, text embeddings

1. Introduction

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

Natural Language Processing (NLP). They include customer feedback analysis, sentiment interpretation, topic detection, as well as emotion detection, which is one of the main topics considered in this paper, and which presents itself in different shapes:

- Classification of customer comments, based on whether they express an angry, happy, or disappointed emotion, see e.g. [1].
- Aspect-based sentiment or emotion analysis, see e.g. [2]. For example, a customer could claim that "the battery of the phone works well", which expresses a positive opinion (satisfaction) about the aspect of "battery". At the same time, that same customer could also complain that "the memory is too limited", expressing a negative opinion about the aspect of "memory" and showing disappointment.
- Emotion intensity classification, where ordinal labels represent different levels of a given emotion. For example, in [3], 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).

Apart from emotion detection, in this paper we also consider two other important subjective language classification tasks:

- Hate speech detection. 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 [4].

- 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 [5]. 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 [6].

Cutting-edge solutions to the above problems are typically based on deep learning (DL). For example, Bidirectional Encoder Representations from Transformers (BERT) by Devlin et al. [7] is a state-of-the-art language model used for various NLP tasks. 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.

Inspired by the fuzzy nature of textual data, in this paper, we consider the usage of the fuzzy-rough nearest neighbour (FRNN) based methods proposed in [8]. Fuzzy rough set approaches have been used successfully in various machine learning applications [9], including fuzzy decision trees [10], learning from imbalanced data [11], feature and instance selection [12], etc.

An advantage of instance-based methods like those based on FRNN 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 FRNN-based methods initially provide less accurate predictions than DL approaches, we show in this paper that with proper feature engineering and ensemble construction, we can obtain results on the same level as state-of-the-art deep learning approaches. In particular, we apply our methods to the three classification tasks introduced above:

- For emotion intensity classification, we use the data provided by the SemEval-2018 Task 1 "EI-oc: Affect in Tweets for English"[1], where for four emotions (anger, joy, sadness, and fear), the organizers 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").

- For hate speech detection, we consider two different datasets. The first one was released in the context of SemEval 2019 Task 6: "OffensEval: Identifying and Categorizing Offensive Language in Social Media"[2]. 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

[1] `https://competitions.codalab.org/competitions/17751`

[2] `https://competitions.codalab.org/competitions/20011`

feelings, increase the level of anger and start arguing[3]. This definition illustrates that the concepts of offensive language and hate speech are very similar. The second dataset originates from SemEval 2019 Task 5, "Shared Task on Multilingual Detection of Hate"[4]. 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.

- For irony detection, we use the dataset from SemEval-2018 Task 3, "Irony detection in English tweets"[5] 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 labeling the data, the authors gathered a similar number of non-ironic tweets to obtain a balanced dataset.

The remainder of this paper has the following structure: Section 2 recalls current research efforts related to the SemEval tasks and to interpretability for text analysis. In Section 3, we discuss the methodological choices we took regarding data preprocessing; tweet embeddings, classifiers and evaluation measures. In Section 4, we train our models, first detecting the best individual setup for each embedding in Section 4.1, followed by the results for ensembles in Section 4.2. In Section 5, we apply the best performing setups to the test data, discuss the results and provide an error analysis. Finally, in

[3] https://aclawgroup.com.au/criminal-law/offences/offensive-language/

[4] https://competitions.codalab.org/competitions/19935

[5] https://competitions.codalab.org/competitions/17468

Section 6 we formulate the main conclusions of our study and discuss possible directions for future work.

The code for this paper can be found in the GitHub repository[6].

2. Related Work

2.1. SemEval competitions: tasks and winning solutions

As mentioned before, we apply our methods to three different classification problems originating from the Semantic Evaluation (SemEval) competition.

For SemEval-2018 Task 1, Mohammad et al. [13] collected the data using the Twitter API with a vocabulary of keyword hashtags related to different emotions (for example, "annoyed", "panic", "happy", etc.). As for any SemEval competition, the organizers used a leaderboard. For the emotion intensity task, the evaluation metric was Pearson Correlation Coefficient (PCC), evaluated on a set of unseen test tweets. The winning solutions described in [13] are mainly based on DL methods. The first-placed team [14] proposed an ensemble of Random Forest and XGBoost based on embedding vectors, while the second-placed team [15] presented a solution with LSTM neural networks and transfer learning techniques. The third-placed team [16] used an ensemble of models with Gated-Recurrent-Units (GRU) and a CNN as an attention mechanism.

Zampieri et al. [17] formulated the SemEval 2019 Task 6, using the Offensive Language Identification Dataset (OLID) from [18]. In this paper, we

[6] `https://github.com/olha-kaminska/frnn\_emotion\_detection/tree/emotions\_irony\_hatespeech`

focus on the binary classification task to decide whether a tweet is offensive or not. The authors reported that 70% of winning solutions were based on DL and that the three best ones for this subtask [19, 20, 21] propose fine-tuned BERT-based solutions.

Task 5 from the same SemEval 2019 competition, proposed by Basile et al. [22], considered a more specific type of hate speech, targeted mostly against women and immigrants. In this paper, we consider the 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 SVM model with TF-IDF text representation. Indurthi et al. [23] obtained the best result using an SVM model with Universal Sentence Encoder (USE) embeddings. While the second team did not publish their solution, the third-placed team [24] used a capsule network with training stacked Bidirectional Gated Recurrent Units (BiGRUs) including fastText word embeddings.

Finally, for irony detection, Van Hee et al. [25] released a dataset of tweets for irony detection in the framework of SemEval-2018 Task 3. They also provided two baselines: one with random labels assigned and the second based on an SVM model with TF-IDF features. The best solution for binary irony classification [26] proposed densely connected LSTMs that use different features, such as text embeddings, sentiment, and syntactic features, while the runner-up [27] used recurrent neural networks (BiLSTM) on both word and character levels, and the third place system [28] combined SVM and logistic regression (LR) into an ensemble with averaged tweet embeddings as features.

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 neural networks or BERT transformers. As such, they 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 fully, since we still use text embedding techniques that were pre-trained using neural networks or transformers, but as classification methods, we consider more interpretable nearest neighbour-based approaches.

Previously, in [29] we explored the efficiency of the weighted k Nearest Neighbour (wkNN) classifier for the emotion detection task. In this paper, we will replace wkNN by the fuzzy-rough nearest neighbour (FRNN) classifier, which is a more flexible instance-based method that uses the neighbour concept in a more intricate way.

2.2. Interpretability in text analysis

We can discern two main aspects in model interpretability for text analysis [30]. 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. In this paper, we will be concerned with local predictions.

From another point of view, we can define two types of interpretability approaches: post-hoc interpretation and self-explanatory models. As an example of the former, we can consider Perturbed Masking [31] which uses Masked Language Modeling (MLM) to calculate a particular word's impact on the prediction results for another word. Another example of a post-hoc method is LIME (Local Interpretable Model-agnostic Explanations, [32]),

which explains the results of a classifier's predictions by approximating the learning process locally with an interpretable model.

Most current explainable models in NLP belong to the self-explanatory category. For example, Variational Word Masks (VMASK) [33] 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 [34], 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.

In another study, Akula and Garibay [35] aimed to develop an interpretable deep learning model for sarcasm detection for English social media data. They used preprocessed data and BERT embeddings in a Neural Network (NN) based solution, including a multi-head self-attention module and GRU. The self-attention part was added for interpretability purposes, and the authors illustrated that this module improved 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.

Besides distinguishing between "local" vs. "global" and "post-hoc" vs. "self-explanatory" explanations, Danilevsky et al. [30] also identified 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 are able to explain some of the prediction-making process steps, which usually are 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.

Taking into account 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.

In Section 5.2, we will illustrate our method as part of the error analysis process and show that it may provide a wider picture than an attention-based map. Particularly, besides detecting the keywords that influenced the prediction-making process, we can identify whole topics that relate to a given tweet. We can also identify mistakes caused by similar tweet topics and detect confusing tweets that affected our results. In this way, we can correct our model to obtain better results in future experiments.

3. Methodology

In this section, we present the theoretical background of our experiments. In Section 3.1 we describe the different text preprocessing steps we considered. Section 3.2 presents a description of text embedding methods and explains how we applied them in our pipeline. In Sections 3.3 and 3.4, we give an overview of the classification and regression methods that we considered during our experiments and the evaluation metrics to evaluate our results.

3.1. Data cleaning

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. Text embedding methods take the text as input and provide a numerical vector (or a set of vectors) as output. This vector represents the original text in a multi-dimensional space in such a way that similar text fragments are encoded by close vectors. Text embedding methods are described in detail 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.
2. Using tweet cleaning: deleting numbers, special symbols, links, user tags, and the "#" symbol with the *"tweet-preprocessor"* package, and replacing emojis with their textual descriptions with the *"emoji"* package.

3. Similar as the previous step, but with additional stop-word removal using the *"NLTK"* package.

Regarding emojis (which are present in less than 15% of tweets in the emotion detection dataset), we decided to retain them because they may provide hints about the writer's emotional state (as was illustrated by [36]). We tried direct and indirect approaches to adapt the emoji Unicode format for text embedding. In the direct approach, we used an embedding method called Emoji2Vec[7], 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 tweet's context. In the indirect approach, we used the Emoji package[8] 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, and evaluation measurement for both emoji preprocessing methods. As the results showed that the indirect approach provides higher evaluation scores, we kept this method for our future experiments.

Another specific feature of tweets is the use of hashtags, which are present in roughly 40% of the tweets. 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. [13] 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.

[7] `https://github.com/uclnlp/emoji2vec`

[8] `https://pypi.org/project/emoji/`

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 to take this into account by transforming these latter hashtags into separate sentences (*"Well, I guess my day is ruined... Sad. Angst. Sadness."*). Since this did not affect our results much, we kept the initial approach - deleting the "#" symbol and maintaining hashtags as they are.

Regarding the rest of the non-textual parts of tweets, such as numbers, punctuation, special symbols (like *"/n"*), links, and user tags, we tried to delete (clean) or keep them to improve our results. Cleaning can be done with custom functions using the *"regex"* package[9] or using existing packages for tweet parsing, like *"tweet-preprocessor"*[10]. After experiments on the emotion detection dataset, we concluded that the most effective approach is to delete all these tweet parts, except for punctuation which could be useful during the data embedding step to save the tweet's context.

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 don't have any important meaning (*"the", "a", "any", etc.*). A list of such words for the English language is provided in the *"NLTK"* package[11]. As we will show in the experimental part, this solution proves useful for some embedding techniques but is mostly inferior if the embedding algorithm considers the context of the text.

[9] `https://docs.python.org/3/library/re.html`

[10] `https://pypi.org/project/tweet-preprocessor/`

[11] `https://www.nltk.org/`

3.2. Tweet embedding

To use tweets in classification methods, we shall represent them as vectors in an N-dimensional space, so tweets that have a similar meaning 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 language models. In this paper, we will consider different types of text embedding techniques.

The first and earliest type of DL-based embedding method is Word2Vec, first presented in [37] and [38]. 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 paper 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[12]. It provides a Word2Vec model pre-trained on a Google News dataset that contains near to 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 ap-

[12] `https://radimrehurek.com/gensim/models/word2vec.html`

peared, for example the Doc2Vec[13] packages that work similarly to Word2Vec, but for whole paragraphs of a text rather than for individual words. We tried the latter approach as well, but got very unsatisfying results for our data. Therefore, we will only use the Gensim pre-trained Word2Vec model. To obtain a vector for a whole tweet, we take the mean of all its words' vectors.

Embedding methods can also take into account sentiment hidden in a sentence. For example, the DeepMoji model presented in [39] is an LSTM-based model trained on over one million tweets containing one out of 64 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[14].

We also used more advanced recent methods, for example, the Universal Sentence Encoder (USE) developed by TensorFlow[15] and described in [40]. 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 experiments on the emotion detection dataset with the same classification method, text preprocessing steps and evaluation,

[13] `https://radimrehurek.com/gensim/models/doc2vec.html`

[14] `https://github.com/huggingface/torchMoji`

[15] `https://www.tensorflow.org/hub/tutorials/semantic\_similarity\_with\_tf\_hub\_universal\_encoder`

we chose the second option as it gave better results.

The earlier mentioned transformer encoders represent the current state-of-the-art approach in the NLP area. The very first such model was BERT by [7]. 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 modeling 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 BERT realisation[16] to extract 768-dimensional vectors for each tweet's token (part of the word) 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-BERT (SBERT) by Reimers and Gurevych [41] 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. Another model is the Twitter-roBERTa-based model by Barbieri et al. [42]. The authors presented seven models for different tasks, including emotion detection, hate speech, and irony classification. These models are fine-tuned on similar SemEval data versions of the Robustly Optimized BERT Pre-training Approach (roBERTa), analogous to the BERT model but with some minor changes of the training procedure and architec-

[16] `https://github.com/dnanhkhoa/pytorch-pretrained-BERT/blob/master/examples/extract\_features.py`

ture.

Besides the previously described methods, we also investigated several other embedding approaches that did not perform well and therefore were not included in our main experiments. Mainly, we tried FastText pre-trained word vectors by Mikolov et al. [43] which is a more advanced Word2Vec solution based on CBOW, spaCy embedding models[17] by Honnibal and Montani [44] which are based on CNN models and available for English in two sizes: small and large (we considered both models for our experiments), and BERTweet by Nguyen et al. [45], which is a BERT-based model with a RoBERTa pre-training procedure executed on 850 million English tweets.

Finally, we also investigated the idea to use emotion lexicons to improve the quality of the obtained embedding vectors and add more emotion-related information. An emotion lexicon is a vocabulary of words, where each word is assigned an emotional intensity score. In [29], we tried five different lexicons along with different ways of combining them with word embedding vectors, but none of them improved the obtained results. For this reason, emotion lexicons are not considered in the current paper.

3.3. Classification methods

This section presents the fuzzy rough set-based methods we investigated for our experiments and their ensembles.

3.3.1. Similarity relation

Prior to the description of classification methods, it is important to discuss the similarity metric that we used to measure how similar the considered

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

vectors (in other words, tweet embeddings) are. Huang [46] compared different metrics for similar NLP tasks: Euclidean distance, Jaccard coefficient, Pearson Correlation Coefficient, averaged Kullback-Leibler divergence, and cosine similarity. Based on their findings, we chose the latter for our experiments:

$$\cos(A, B) = \frac{A \cdot B}{||A|| \times ||B||} \tag{1}$$

In Eq.(1), A and B correspond to elements from the same vector space (tweets 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:

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

Finally, we note that we also examined Hausdorff distance for comparing tweets, considering that tweets can be seen as sets of individual words. However, this approach did not provide satisfactory results, so we dismissed it.

3.3.2. FRNN-OWA method

In our work, we consider methods based on fuzzy rough set theory, mainly the fuzzy-rough nearest neighbour (FRNN) classification model that was proposed in [8].

FRNN is an instance-based algorithm that performs classification using lower (L) and upper (U) approximations from fuzzy rough set theory. [8]

showed that the method outperformed classical NN approaches and that it is competitive with the leading classification algorithms. To make the solution more robust and noise-tolerant, [47] 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 [48], 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[18] for these methods in [49], 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:

$$OWA_{\overrightarrow{W}}(V) = \sum_{i=1}^{|V|} (w_i v_{(i)}) \tag{3}$$

We experimented with several types of OWA operators and concluded that the additive weight type ([47]) clearly performed best for our data. Additive weights are linearly increasing for lower approximation (Formula (4), where p ($p > 1$) denotes the vector's length) and decreasing for upper approximation (Formula (5)).

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

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

$$\overrightarrow{W}_U^{add} = \langle \frac{2}{p+1}, \frac{2(p-1)}{p(p+1)}, ..., \frac{4}{p(p+1)}, \frac{2}{p(p+1)} \rangle \quad (5)$$

Next, during the classification of a test vector y, the FRNN-OWA method calculates the membership degree of y to the lower (Formula (6)) and upper (Formula (7)) approximation of each decision class C.

$$\underline{C}(y) = OWA_{\overrightarrow{W}_L}(\{1 - R(x,y) \mid x \in X \setminus C\}) \quad (6)$$

$$\overline{C}(y) = OWA_{\overrightarrow{W}_U}(\{R(x,y) \mid x \in C\}) \quad (7)$$

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. (6), k refers to the number of neighbours of y from classes other than C and in Eq. (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.

Finally, 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

$$Conf(C,y) = \frac{\underline{C}(y) + \overline{C}(y)}{\sum_{C' \in \mathcal{C}} \underline{C'}(y) + \overline{C'}(y)} \quad (8)$$

3.3.3. Method based on FRNN-OWA regression

As an alternative solution to FRNN-OWA for the emotion detection issue, which can be considered as an ordinal classification task, we also tried FRNN regression ([8], Algorithm 4).

For the test instance y, this algorithm predicts a value based on the classes of the k nearest neighbours of instance y, similarly 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. In our experiments, we fine-tuned the parameter k for each emotion dataset.

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

3.3.4. 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. 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 turned out to be a weighted average, where the weights are derived from the confidence scores that each FRNN-OWA classifier generates[19].

An analysis of the confidence scores we obtained in our experiments pointed out that they are often close to each other and generally lie within the range $[0.4, 0.6]$. We hypothesize that this may be due to the high dimension-

[19]For FRNN regression, which does not generate any confidence scores, we used the classical average as voting function.

ality of tweet embeddings, causing the upper approximation memberships to be close to 1 and the lower ones to 0. To mend this issue, we propose rescaling the original confidence scores in order to amplify their differences.

In practice, 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. (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:

$$w_i = \frac{\exp(\sum_j (Conf_i(C_j, y) - 0.5)/\alpha)}{\sum_k \exp(\sum_j (Conf_k(C_j, y) - 0.5)/\alpha)}, \tag{9}$$

The full architecture of our solution is shown in Fig. 1, 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.

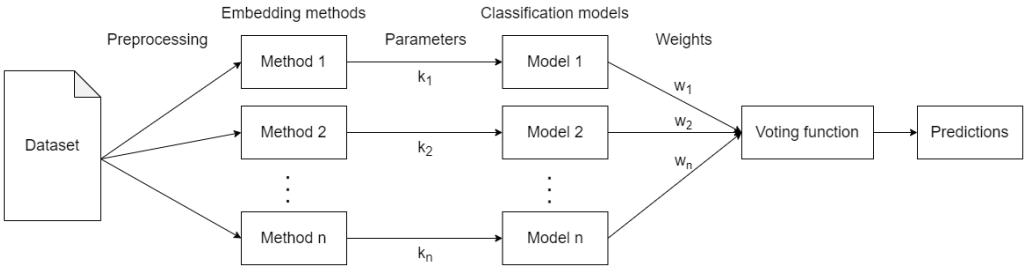

Figure 1: Scheme of our solution with an ensemble of FRNN-OWA classifiers.

3.4. Evaluation methods

To measure and compare the obtained prediction results, we use three different metrics. Two of them were proposed by the organizers of the respective SemEval competitions, whereas the last is added to obtain a more complete picture of our methods' performance.

PCC (10) 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:

$$PCC = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}. \tag{10}$$

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.

An additional metric that we consider for the emotion detection task is mean absolute error (MAE) (11). This choice of metric is inspired by the fact that we are dealing with an ordinal classification task. The MAE formula is:

$$MAE = \frac{\sum_i |y_i - \bar{x_i}|}{n}, \tag{11}$$

where n is the size of vectors x and y. Better predictions correspond to lower MAE values.

A final metric is the F1-score (12) that was used for the offensive language, hate speech, and irony detection SemEval competitions. Particularly, a macro-averaged F1-score is used, where all classes have equal weights. The

formula for the F1-score is:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}, \tag{12}$$

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 score to choose the best model.

4. Experiments

In this section, we present the results of the methods described in Section 3 on the training portion of the datasets mentioned in Section 2.1. In the first step (Section 4.1), we detect the best setup for each dataset and each embedding method, and in the second (Section 4.2), we tune ensembles of models to obtain the most efficient approach per dataset. In Section 5, we provide the results of the best setups on the test data to see how well they generalize to new data.

4.1. Detecting the best setup for embedding methods

For each dataset, we apply the six embedding methods described in Section 3.2. To detect the best setup for each method, we tuned the parameter k (number of neighbours) and prior text cleaning options.

First, we select the best text preprocessing approach for each combination of a dataset and an embedding method. For this purpose, we perform 5-fold cross-validation on the training data with the same list of k values (determined based on the size of each data set) for each preprocessing option: raw

tweets, cleaned tweets, and cleaned tweets without stop words. For the emotion datasets (anger, joy, sadness, and fear) we use PCC, and for the other datasets (hate speech, offensive language, and irony) we use the F1-score.

Table 1: 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**
		roBERTa-based		
Tweet cleaning	Yes	Yes	Yes	Yes
Stop-word cleaning	No	No	No	No
k value	19	9	23	9
PCC	**0.6779**	**0.6956**	**0.7062**	**0.6031**
		DeepMoji		
Tweet cleaning	No	No	No	No
Stop-word cleaning	No	No	No	No
k value	23	19	23	21
PCC	0.5853	0.6520	0.6380	0.5745
		BERT		
Tweet cleaning	No	No	No	No
Stop-word cleaning	No	No	No	No
k value	19	17	23	7
PCC	0.4492	0.5374	0.4391	0.4500
		SBERT		
Tweet cleaning	Yes	Yes	Yes	Yes
Stop-word cleaning	No	No	No	No
k value	19	15	23	11
PCC	0.5016	0.5660	0.5655	0.5192
		USE		
Tweet cleaning	Yes	Yes	Yes	Yes
Stop-word cleaning	No	No	No	No
k value	23	23	23	21
PCC	0.5054	0.5693	0.5961	0.5764
		Word2Vec		
Tweet cleaning	Yes	Yes	Yes	Yes
Stop-word cleaning	Yes	Yes	Yes	Yes
k value	21	23	23	7
PCC	0.5009	0.5099	0.5048	0.4496

Table 1 shows the results on the four emotions datasets for the FRNN-OWA classification method. From this table, we can immediately observe that the roBERTa-based embedding method provides superior results to the others on all datasets. This observation comes as no particular surprise since roBERTa was fine-tuned on data similar to the SemEval competition data and its performance is in line with earlier results for related classification tasks [42]. On the other hand, the lowest scores are obtained for Word2Vec and BERT. Their relatively poor performance may be explained by the fact that they were trained on a too generic corpus. Also, we note that the PCC scores for Fear are often the lowest among the emotions, which may be due to the fact that this dataset is the most unbalanced dataset. On the other hand, the most balanced dataset (Joy) generally receives the best prediction results.

Note that some embeddings do not require any preprocessing at all, like DeepMoji and BERT. Probably this is due to the fact that these embeddings take into account the context of words. 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; we note that for Fear, lower values of k are generally better, which may again be linked to the imbalance of this dataset.

The results for FRNN regression are shown in Table 2. First, we may notice that for most embeddings, the results still slightly improve those of their classification counterparts. A notable exception is the Word2Vec method, which is the only one to perform worse on all four emotions. Concerning the differences between embeddings, we observe similar trends as for FRNN-

Table 2: Optimal FRNN regression setup (preprocessing, number of neighbours k) and corresponding PCC score per embedding for the emotion datasets

Setup	**Anger**	**Joy**	**Sadness**	**Fear**
		roBERTa-based		
Tweet cleaning	Yes	Yes	No	Yes
Stop-word cleaning	No	No	No	No
k value	23	15	19	7
PCC	**0.6930**	**0.7146**	**0.7188**	0.6169
		DeepMoji		
Tweet cleaning	Yes	Yes	Yes	Yes
Stop-word cleaning	No	No	No	No
k value	27	23	23	17
PCC	0.6316	0.6448	0.6772	**0.6271**
		BERT		
Tweet cleaning	Yes	Yes	Yes	Yes
Stop-word cleaning	No	No	No	No
k value	13	19	19	5
PCC	0.4618	0.5469	0.4760	0.4338
		SBERT		
Tweet cleaning	Yes	No	Yes	Yes
Stop-word cleaning	Yes	No	No	No
k value	11	17	7	19
PCC	0.5298	0.5587	0.5527	0.4976
		USE		
Tweet cleaning	No	Yes	No	Yes
Stop-word cleaning	No	No	No	No
k value	29	13	29	11
PCC	0.5494	0.5913	0.6419	0.5798
		Word2Vec		
Tweet cleaning	Yes	Yes	Yes	Yes
Stop-word cleaning	Yes	No	Yes	Yes
k value	21	9	29	5
PCC	0.4273	0.4582	0.4806	0.4295

OWA classification: the best results are obtained for roBERTa, except for Fear, on which DeepMoji performs slightly better. BERT and Word2Vec are underperforming by comparison. The optimal preprocessing options also

vary a bit, with tweet cleaning now proving beneficial in the majority of cases.

Finally, we repeated the above experimental analysis also for the Hate Speech, Offensive language and Irony datasets with the FRNN-OWA method. Our findings, summarized in Table 3, reveal for all of them, that the roBERTa-based embedding by far outperforms the remaining embedding methods. The results also indicate that in this case preprocessing is not helpful.

4.2. Ensembles

In the next step, we consider an ensemble of six FRNN-OWA classifiers, corresponding to the six tweet embeddings methods outlined before. Table 4 shows the results, using first the standard mean as voting function (in other words, attributing equal importance to each classifier) and then the weighted average involving rescaled confidence scores as defined in Eq. (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.

From Table 4, we can see that overall the weighted average voting function is a better option than the standard mean, improving the results of the latter except for Fear, for which it obtains a slightly lower PCC score. We also notice that for all emotion datasets the ensemble outperforms the use of a single classifier. For the Hate Speech, Offensive language and Irony datasets, however, it is clear that using the standalone roBERTa-based embedding (see Table 3) is still better than any of the ensemble strategies, so we will not consider the latter further on.

Given the large performance gap between different individual embeddings in Tables 1 and 2, we also consider ensembles constructed with a subset of

Table 3: Optimal FRNN-OWA classification setup (preprocessing, number of neighbours k) and corresponding F1 score for all embedding for the Hate Speech, Offensive and Irony datasets

Setup	**Hate Speech**	**Offens**	**Irony**
	roBERTa-based		
Tweet cleaning	No	No	No
Stop-word cleaning	No	No	No
k value	25	45	27
F1 score	**0.8765**	**0.8377**	**0.9365**
	DeepMoji		
Tweet cleaning	No	No	No
Stop-word cleaning	No	No	No
k value	19	39	15
F1 score	0.6223	0.6567	0.6774
	BERT		
Tweet cleaning	Yes	No	No
Stop-word cleaning	No	No	No
k value	15	47	23
F1 score	0.7172	0.6847	0.6563
	SBERT		
Tweet cleaning	No	No	No
Stop-word cleaning	No	No	No
k value	15	47	29
F1 score	0.7063	0.7063	0.6504
	USE		
Tweet cleaning	Yes	No	No
Stop-word cleaning	No	No	No
k value	13	35	25
F1 score	0.7037	0.6898	0.6650
	Word2Vec		
Tweet cleaning	Yes	Yes	Yes
Stop-word cleaning	Yes	Yes	Yes
k value	13	39	27
F1 score	0.6700	0.6622	0.5966

Table 4: Results for an ensemble of six FRNN-OWA methods with different embeddings, using two different voting functions.

Setup	Anger	Joy	Sadness	Fear	Hate Speech	Offens	Irony
	PCC				F1-score		
Standard mean	0.6475	0.7126	0.7152	**0.6448**	0.7500	0.6796	0.7331
Conf. scores	**0.6960**	**0.7512**	**0.7455**	0.6430	**0.8116**	**0.7119**	**0.8350**
tuned α	$\alpha = 0.029$	$\alpha = 0.032$	$\alpha = 0.032$	$\alpha = 0.046$	$\alpha = 0.1$	$\alpha = 0.8$	$\alpha = 0.5$

embeddings, using a grid search to identify the optimal setup. The results of this analysis are shown in Table 5.

Table 5: Optimal FRNN-OWA classification and FRNN regression ensemble setup and corresponding PCC score for the emotion datasets.

Dataset	Setup	PCC
	FRNN-OWA	
Anger	roBERTa, DeepMoji, USE, Word2Vec, BERT	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
Hate speech	roBERTa	0.8765
Offensive	roBERTa	0.8377
Irony	roBERTa	0.9365
	FRNN Regression	
Anger	roBERTa	0.6930
Joy	roBERTa	0.7146
Sadness	roBERTa	0.7188
Fear	roBERTa, DeepMoji, USE, sBERT	0.6565

We conclude that the best setups for the emotions datasets with FRNN-OWA require four to five embeddings to provide the best results. Meanwhile, for the other datasets (Hate Speech, Offensive language, and Irony) and FRNN regression, the best results are obtained with the standalone roBERTa embedding, except for FRNN regression on the Fear dataset.

5. Evaluation on the test data

In this section, we provide and discuss the results of the best settings described in the previous section (cfr. Table 5) applied to the test datasets. We also examine several samples of correct and wrong predictions in the error analysis part.

5.1. Test results

Table 6 lists the results of our approaches for the emotion datasets. Apart from the PCC used by the competition organizers, we also included the MAE. We calculated the mean scores of all four datasets because the averaged score is used in the original SemEval competition [13] to compare the solutions. We also included the results of the competition's top-3 solutions , which were discussed in Section 2.1.

As we can see from Table 6, the FRNN-OWA method performs better for the PCC metric and FRNN regression for MAE. It makes sense, taking into account the idea behind the MAE metric. However, since the PCC is the evaluation metric of this SemEval competition, the FRNN-OWA classifier obtains a higher position on the leaderboard for the English EI-oc subtask[20].

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

The PCC and MAE scores are generally slightly worse for the test data than for the training data, with the notable exception of Fear and MAE. Furthermore, similar patterns may be observed as in the training stage: the best performance is obtained for Joy and the worst one for Fear.

Table 6: Comparison of evaluation metrics for the best setup for the emotion test dataset with FRNN-OWA and FRNN regression methods.

Dataset	**FRNN-OWA**		**FRNN regression**	
	Train	Test	Train	Test
		MAE		
Anger	1.4447	1.0698	0.5615	0.6866
Joy	1.3420	1.5447	0.5755	0.5665
Sadness	1.2181	1.1179	0.5476	0.5712
Fear	0.6209	0.6622	0.4498	0.5801
Averaged	1.1564	1.0986	**0.5336**	**0.6011**
		PCC		
Anger	0.7241	0.6388	0.6930	0.6671
Joy	0.7788	0.7115	0.7146	0.6738
Sadness	0.7719	0.6967	0.7188	0.6865
Fear	0.6930	0.5705	0.6565	0.5724
Averaged	**0.7419**	**0.6544**	0.6957	0.6499
Leaderboard	2^{nd} place		4^{rd} place	
TOP-3 places test PCC scores	0.695, 0.653, 0.646 [14], [15], [16]			

For the other datasets (Hate Speech, Offensive language, and Irony), we listed the F1-scores obtained with the FRNN-OWA method in Table 7. For

Table 7: F1-scores for the best FRNN-OWA setup for the hate speech, offensive and irony test datasets.

Setup	**HateSpeech**	**Offens**	**Irony**
Train	0.8765	0.8377	0.9365
Test	0.5351	0.8109	0.6515
Leaderboard	5^{th} place	4^{th} place	3^{rd} place
TOP-3 places	0.651, 0.571, 0.546	0.829, 0.815, 0.814	0.7054, 0.6719, 0.65
test F1-scores	[23], -, [24]	[19], [20], [21]	[26], [27], [28]

all these competitions, we obtained top-5 positions.[21] Although the leaderboard is private for the Hate Speech and Offensive language competition, it is available for the Irony competition.[22] Hence, we added the results of the top-3 teams (see Section 2.1) for each SemEval competition to Table 7. Similarly to Table 6, the winners' approaches are mainly based on deep learning methods (BERT, BiGRUs, LSTM, BiLSTM).

We also can notice that for the Hate Speech and Irony datasets, the gap between cross-validation and test scores is much bigger than for the Offensive language detection task. 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 train and test data came from different distributions.

[21]Due to CodaLab restrictions it was not possible to submit our labels, so we calculated our places on our own with the provided validation scripts.

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

5.2. Error analysis

To examine the performance of our proposed approach in more detail, we explore correct and wrong samples of test instances. This may also serve to illustrate our solution's explainability, where particular patterns could be observed.

To detect neighbouring training tweets for the test instance, we calculated the cosine similarly 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. Below, when we present "training neighbours of the test instance", we mean those tweets in the intersection, or in most, of the selected top-k neighbourhoods.

First of all, we computed the confusion matrices for all test datasets. Results for the emotion datasets are presented in Table 8. We can see that 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, *"things that terrify me: remembering my bf follows me on twitter"*, has the majority of closest neighbours from training tweets with topics related to social media.

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

Another test example *"@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"* with class '3' was classified with class '0'. Its neighbours 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 test samples with the highest level of Fear (class '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 joyful tweets with class '3', we can consider the test instance *"@USER Happy #blissful birthday"*. The majority of its 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 test tweet *"Good Night everyone... #goodnight #sleep #nice #great #night #music #day"* with correct class '1' that was predicted as '3'. Its neighbours 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'.

Table 8: 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

For the other three datasets (Hate Speech, Offensive language, and Irony), the confusion matrices are shown in Table 9. 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. Hence, we will

Table 9: Confusion matrices for Hate Speech, Offensive language, and Irony test datasets.

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

take a closer look at the other two datasets and provide some examples of wrong predictions.

As for the Irony dataset, we can take a look at one sample with true class '1' (irony present): *"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 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.

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, a correctly classified hateful test tweet with class '1' is the following: *"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 #End-*

CatchAndReleash #DefundSantuaryCities". The majority of its neighbours are hateful (class '1') and share the hashtag *"#BuildThatWall"*, such as *"Illegal Criminals EVERYWHERE #BuildThatWall !!"*. This hashtag can also be considered as a strong keyword, but in this case caused a misclassification.

Another possible reason for wrong classifications could be the use of similar topics/words in a different context. For example, the test sample *"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"* has class '1', but the majority of its training neighbours 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'.

In conclusion, similar topics and common keywords are strong neighbour detection features on which our approach is based. However, they may cause errors when the same word is used in a different context.

6. Conclusion and future work

In this paper, we have evaluated the potential of interpretable machine learning methods based on fuzzy rough sets for different subjective language classification tasks and demonstrated that they are competitive with more complex state-of-the-art neural network-based approaches. In particular, we designed and optimized weighted ensembles of FRNN-OWA classification and FRNN regression using feature vectors obtained from different word embeddings, which are mostly sentiment-oriented and applied at the sentence level. Also, our error analysis reveals that our methods are capable of identifying

useful patterns that can explain their predictions.

As one of the main future challenges, we consider a more systematic approach to solution explainability. Danilevsky et al. [30] provide several hints on how to do this. Also, we can examine the application of fuzzy rule-based methods [50] on top of the set of nearest neighbours using high-level features since such methods may further enhance explainability.

Another important characteristic that influences the results is data imbalance, as we observed, for example, for the Fear dataset. For further experiments, we consider the usage of imbalanced machine learning classification methods like those described in [11].

Acknowledgements

Olha Kaminska and Chris Cornelis would like to thank the Odysseus project from Flanders Research Foundation (FWO), grant no. G0H9118N, for funding their research.

References

[1] J. J. Zhu, Y.-C. Chang, C.-H. Ku, S. Y. Li, C.-J. Chen, Online critical review classification in response strategy and service provider rating: Algorithms from heuristic processing, sentiment analysis to deep learning, Journal of Business Research 129 (2021) 860–877.

[2] A. Chinnalagu, A. K. Durairaj, Context-based sentiment analysis on customer reviews using machine learning linear models, PeerJ Computer Science 7 (2021) e813.

[3] R. K. Gupta, A. Vishwanath, Y. Yang, Covid-19 twitter dataset with latent topics, sentiments and emotions attributes (2021-11-04). doi:https://doi.org/10.3886/E120321V11.

[4] Z. Al-Makhadmeh, A. Tolba, Automatic hate speech detection using killer natural language processing optimizing ensemble deep learning approach, Computing 102 (2) (2020) 501–522.

[5] D. Chandler, R. Munday, A dictionary of media and communication, OUP Oxford, 2011.

[6] B. Ghanem, J. Karoui, F. Benamara, P. Rosso, V. Moriceau, Irony detection in a multilingual context, Advances in Information Retrieval 12036 (2020) 141–149.

[7] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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.

[8] R. Jensen, C. Cornelis, Fuzzy-rough nearest neighbour classification and prediction, Theoretical Computer Science 412 (42) (2011) 5871–5884.

[9] S. Vluymans, L. D'eer, Y. Saeys, C. Cornelis, Applications of fuzzy rough set theory in machine learning: a survey, Fundamenta Informaticae 142 (1-4) (2015) 53–86.

[10] J.-h. Zhai, Fuzzy decision tree based on fuzzy-rough technique, Soft Computing 15 (6) (2011) 1087–1096.

[11] S. Vluymans, A. Fernández, Y. Saeys, C. Cornelis, F. Herrera, Dynamic affinity-based classification of multi-class imbalanced data with one-versus-one decomposition: a fuzzy rough set approach, Knowledge and Information Systems 56 (1) (2018) 55–84.

[12] H. Zhao, P. Wang, Q. Hu, P. Zhu, Fuzzy rough set based feature selection for large-scale hierarchical classification, IEEE Transactions on Fuzzy Systems 27 (10) (2019) 1891–1903.

[13] S. M. Mohammad, F. Bravo-Marquez, M. Salameh, S. Kiritchenko, Semeval-2018 Task 1: Affect in tweets, in: Proceedings of International Workshop on Semantic Evaluation (SemEval-2018).

[14] V. Duppada, R. Jain, S. Hiray, SeerNet at SemEval-2018 task 1: Domain adaptation for affect in tweets, in: Proc. 12th International Workshop on Semantic Evaluation, 2018, pp. 18–23.

[15] G. Gee, E. 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.

[16] A. Rozental, D. 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, 2018, pp. 218–225.

[17] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, SemEval-2019 task 6: Identifying and categorizing offensive language

in social media (OffensEval), in: Proc. 13th International Workshop on Semantic Evaluation, 2019, pp. 75–86.

[18] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, R. Kumar, 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.

[19] P. Liu, W. Li, L. 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.

[20] A. Nikolov, V. Radivchev, Nikolov-radivchev at SemEval-2019 task 6: Offensive tweet classification with BERT and ensembles, in: Proc. 13th International Workshop on Semantic Evaluation, 2019, pp. 691–695.

[21] J. Zhu, Z. Tian, S. Kübler, UM-IU@LING at SemEval-2019 task 6: Identifying offensive tweets using BERT and SVMs, in: Proc. 13th International Workshop on Semantic Evaluation, 2019, pp. 788–795.

[22] V. Basile, C. Bosco, E. Fersini, N. Debora, V. Patti, F. M. R. Pardo, P. Rosso, M. Sanguinetti, 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.

[23] V. Indurthi, B. Syed, M. Shrivastava, N. Chakravartula, M. Gupta,

V. Varma, FERMI at SemEval-2019 task 5: Using sentence embeddings to identify hate speech against immigrants and women in Twitter.

[24] Y. Ding, X. Zhou, X. 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, 2019, pp. 535–539.

[25] C. Van Hee, E. Lefever, V. Hoste, SemEval-2018 task 3: Irony detection in English tweets, in: Proc. 12th International Workshop on Semantic Evaluation, 2018, pp. 39–50.

[26] C. Wu, F. Wu, S. Wu, J. Liu, Z. Yuan, Y. Huang, 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, 2018, pp. 51–56.

[27] C. Baziotis, A. Nikolaos, P. Papalampidi, A. Kolovou, G. Paraskevopoulos, N. Ellinas, A. Potamianos, 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, 2018, pp. 613–621.

[28] O. Rohanian, S. Taslimipoor, R. Evans, R. Mitkov, WLV at SemEval-2018 task 3: Dissecting tweets in search of irony, in: Proc. 12th International Workshop on Semantic Evaluation, 2018, pp. 553–559.

[29] O. Kaminska, C. Cornelis, V. 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.

[30] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, P. Sen, 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, 2020, pp. 447–459.

[31] Z. Wu, Y. Chen, B. Kao, Q. Liu, Perturbed masking: Parameter-free probing for analyzing and interpreting bert, arXiv preprint arXiv:2004.14786 (2020).

[32] M. T. Ribeiro, S. Singh, C. 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.

[33] H. Chen, Y. Ji, Learning variational word masks to improve the interpretability of neural text classifiers, arXiv preprint arXiv:2010.00667 (2020).

[34] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, 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.

[35] R. Akula, I. Garibay, Explainable detection of sarcasm in social media, in: Proceedings of the Eleventh Workshop on Computational Ap-

proaches to Subjectivity, Sentiment and Social Media Analysis, 2021, pp. 34–39.

[36] S. Boy, D. Ruiter, D. 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, 2021, pp. 103–110.

[37] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, CoRR abs/1301.3781 (2013).

[38] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, 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, p. 3111–3119.

[39] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, S. Lehmann, Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm, Proc. 2017 Conference on Empirical Methods in Natural Language Processing (2017).

[40] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope, R. Kurzweil, Universal sentence encoder for English, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2018, pp. 169–174. doi:10.18653/v1/D18-2029.
URL https://www.aclweb.org/anthology/D18-2029

[41] N. Reimers, I. 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), 2019, pp. 3982–3992.

[42] F. Barbieri, J. Camacho-Collados, L. Espinosa Anke, L. Neves, TweetEval: Unified benchmark and comparative evaluation for tweet classification, in: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 1644–1650.

[43] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin, Advances in pre-training distributed word representations, in: Proc. International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[44] M. Honnibal, I. Montani, spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing (2017).

[45] D. Q. Nguyen, T. Vu, A. 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.

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

[47] S. Vluymans, N. Mac Parthaláin, C. Cornelis, Y. Saeys, Weight selection strategies for ordered weighted average based fuzzy rough sets, Information Sciences 501 (2019) 155–171.

[48] O. U. Lenz, D. Peralta, C. Cornelis, Scalable approximate FRNN-OWA classification, IEEE Transactions on Fuzzy Systems 28 (5) (2019) 929–938.

[49] O. U. Lenz, D. Peralta, C. 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 of Lecture Notes in Artificial Intelligence, 2020, pp. 491–499.

[50] T. Chua, W. 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.