LT3 at SemEval-2022 Task 6: Fuzzy-Rough Nearest Neighbor Classification for Sarcasm Detection

Olha Kaminska
Computational Web Intelligence
Department of Applied Mathematics, Computer Science and Statistics
Ghent University, Ghent, Belgium

Chris Cornelis
Computational Web Intelligence
Department of Applied Mathematics, Computer Science and Statistics
Ghent University, Ghent, Belgium

Veronique Hoste
LT3 Language and Translation Technology Team
Ghent University, Ghent, Belgium
{Olha.Kaminska,Chris.Cornelis,Veronique.Hoste}@UGent.be

Abstract

This paper describes the approach developed by the LT3 team in the Intended Sarcasm Detection task at SemEval-2022 Task 6. We considered the binary classification subtask A for English data. The presented system is based on the fuzzy-rough nearest neighbor classification method using various text embedding techniques. Our solution reached 9th place in the official leader-board for English subtask A.

1 Introduction

Sarcasm (or irony) can be defined as a trope or figurative language use whose actual meaning is different from what is literally enunciated (Chandler and Munday, 2011). The task of sarcasm detection can be connected with various challenges in the Natural Language Processing (NLP) field, from sentiment analysis to hate speech detection. However, this task is more complicated by its nature. Even for a human, sarcasm detection could be a challenging issue. It can be represented in different shapes, with voice, gestures, mimic, etc. So, text alone may not be sufficient to detect whether a given utterance is sarcastic or not. It makes the labeling of such datasets quite complicated (Ghanem et al., 2020).

The SemEval competition is an annual event that provides a set of challenges for researchers in different aspects of the NLP field. This year we participated in SemEval-2022 Task 6 called "iSarcasmEval" that considers sarcasm detection in two languages: English and Arabic (Abu Farha et al., 2022). We tackled subtask A for English, where for a given text, we should determine whether it is sarcastic. As described by the authors of the dataset, text writers provided the labels by themselves to exclude subjective labeling. The dataset contains text, its binary label for subtask A (sarcastic or not), a non-ironical rephrase of the provided text with an explanation of why is it sarcastic for subtask C (we did not use it in our experiments), and a set of binary classes for six types of sarcasm (irony, satire, rhetorical question, etc.) for subtask B. We note that the size of the non-irony class for subtask A is three times bigger than the size of the irony class (2,601 instances and 867 instances).

A task related to sarcasm detection is emotion detection issue, which we considered in our previous papers. In (Kaminska et al., 2021b) and (Kaminska et al., 2021a), we addressed the intensity task provided by SemEval-2018 Task 1 (Mohammad et al., 2018). Our first paper used the weighted k Nearest Neighbor (wkNN) classification approach with corresponding text cleaning and embedding steps. In the second paper, we tuned the preprocessing steps and, instead of wkNN, we considered the Fuzzy-Rough Nearest Neighbor (FRNN) classification model with Ordered Weighted Average (OWA) operators ((Jensen and Cornelis, 2011), (Vluymans et al., 2019), (Lenz et al., 2019)). The final approach has the shape of an ensemble of FRNN models based on several strong embedding techniques. The solution described in this paper is based on the methodology presented in (Kaminska et al., 2021a), but it is fine-tuned for the iSarcasmEval dataset and task. Our code is provided at the GitHub repository[1].

The remainder of this paper has the following parts: in Section 2 we provide a step-by-step description of our system, including text preprocessing and the used embedding methods, the classification model and its ensemble, and the evaluation metric. In Section 3 we provide results obtained by cross-validation, and identify the best setup ap-

[1] https://github.com/olha-kaminska/frnn_emotion_detection/tree/iSarcasmEval

Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pages 987 - 992

plied on the test data. It also provides an error analysis of the output and post-competition experiments for model improvement. Section 4 presents conclusions and future steps.

2 System Description

Figure 1 gives an overview of our full method, which encompasses the following steps: we embed the raw text with the Twitter-roBERTa-base for Irony Detection model (Barbieri et al., 2020), then we use the output vectors in the OWA-FRNN classification method to obtain a prediction of classes.

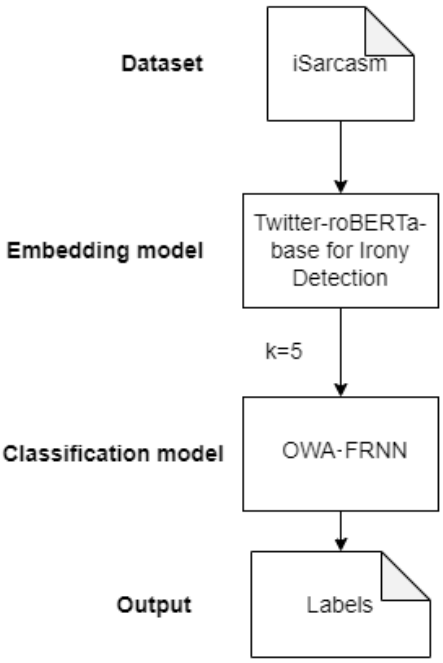

Figure 1: Schematical overview of our architecture.

2.1 Text preprocessing

We manually explored part of the provided dataset and concluded that it has attributes of regular social media posts, like tweets, including user tags, emojis, hashtags, etc. Hence, the first step in our experiments was text preprocessing before we applied text embedding techniques. Initially, we considered three options: no preprocessing, basic cleaning, and extra stop-words removal.

The basic cleaning included deleting the "#" symbol before hashtags and emojis transformation. We did not delete the text of hashtags because it could contain important information about the whole text. Similarly, we kept emojis but replaced them with textual descriptions provided by *"emoji"* package[2].

The third approach of text preprocessing involves the same steps, additionally deleting the stop-words using the *"NLTK"* package[3].

We tried all three text preprocessing setups for each text embedding technique in order to detect the most suitable for each.

2.2 Embedding methods

As a next step of text preparation before classification we investigate different text embedding methods. This technique represents a fragment of text (symbol, word, collocation, sentence, or even paragraph) as a vector (or a set of them). The obtained vector corresponds to the actual text in multi-dimensional vector space with the idea that neighboring vectors represent similar text pieces.

Embedding techniques came a long way from simple bag-of-words and pre-trained dictionaries in a word-vector format to the current state-of-the-art transformer-based solutions and context-based language models. In our method, we explored various types of text embedding techniques: vocabulary Word2Vec from Gensim package[4], sentiment-based DeepMoji[5], Universal Sentence Encoder (USE) by the TensorFlow[6], Bidirectional Encoder Representations from Transformers (BERT), as proposed by (Devlin et al., 2019) and two methods related to BERT - Sentence-BERT (SBERT) by (Reimers and Gurevych, 2019) and the Twitter-roBERTa-based model for irony recognition presented by (Barbieri et al., 2020). As we will see in Section 3.1, the latter performed much better than the others, hence we will describe it in more detail.

The robustly optimized BERT pre-training approach (roBERTa) is comparable to the original BERT model but has a few training technique and architecture differences. In (Barbieri et al., 2020), the authors described several roBERTa-based models for different tasks, for example, hate speech recognition and emotion detection. We considered the one for irony classification[7] that was trained on nearly 58M tweets and fine-tuned on the dataset from Subtask A of the SemEval2018 challenge for Irony Detection presented by (Van Hee et al., 2018).

[2] https://pypi.org/project/emoji/

[3] https://pypi.org/project/nltk/

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

[5] https://deepmoji.mit.edu/

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

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

This model can perform classification directly, but instead we used it to extract tweets' weights from the inner model's layer as their embedding vectors. We will discuss this further in Section 2.3.

2.3 OWA-FRNN classifier

Most state-of-the-art approaches for tasks such as irony detection belong to the deep learning family, and therefore remain black-box solutions. Our idea was to try a more explainable technique based on the Fuzzy-Rough Nearest Neighbor (FRNN) method that already showed promising results in our previous work (Kaminska et al., 2021a).

The FRNN classification model was introduced in (Jensen and Cornelis, 2011). It is an instance-based approach that uses lower (L) and upper (U) fuzzy-rough approximations inside the classification process. In (Vluymans et al., 2019) and (Lenz et al., 2019), FRNN extensions were described based on the Ordered Weighted Average (OWA) operators with the aim of making the method more robust. OWA operators are used to define the membership of a data instance to the lower and upper approximation through an aggregation process.

We will use the following notation: V is the set of OWA aggregation values, where $v_{(i)}$ is the i^{th} largest element of the set V; the weight vector is denoted as $\overrightarrow{W} = \langle w_1, w_2, ..., w_{|V|} \rangle$, where $(\forall i)(w_i \in [0,1])$ and $\sum_{i=1}^{|V|} w_i = 1$. Then, we will have the following formula for the OWA operator:

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

We used additive OWA operators (Vluymans et al., 2019), as they performed the best in our previous paper. They are linearly increasing for lower and linearly decreasing for upper approximations. Additive weights are presented by the Formulas (2) and (3), where p denotes the length of the vector ($p > 1$).

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

$$\overrightarrow{W}_U^{add} = \langle \frac{2}{p+1}, \frac{2(p-1)}{p(p+1)}, ..., \frac{4}{p(p+1)}, \frac{2}{p(p+1)} \rangle \quad (3)$$

OWA-FRNN assigns a test instance y to the class C with the highest sum of $\underline{C}(y)$ and $\overline{C}(y)$:

$$\underline{C}(y) = OWA_{\overrightarrow{W}_L}\{1 - R(x,y) \mid x \in X \setminus C\} \quad (4)$$

$$\overline{C}(y) = OWA_{\overrightarrow{W}_U}\{R(x,y) \mid x \in C\} \quad (5)$$

Here, $R(x,y)$ corresponds to the similarity between vectors x and y. In our experiments, we used cosine similarity:

$$cos\_similarity(A,B) = \frac{1 + cos(A,B)}{2}. \quad (6)$$

Where $cos(A,B)$ is the cosine distance between elements A and B:

$$\cos(A,B) = \frac{A \cdot B}{||A|| \times ||B||} \quad (7)$$

For our setup, A and B denote tweet embedding vectors, $A \cdot B$ is their scalar product, and $||A||$ is the vector norm of A. We considered similarity instead of distance because Formula (6) provides values that fit our classification method: 0 for opposite vectors and 1 for identical ones.

One more parameter that we need for Formulas (4) and (5) is k - the number of nearest neighbors of test instance y. The difference between k in Formulas (4) and (5) is that for the first, it corresponds to the amount of training samples that are y's neighbors outside class C and for the second - those inside class C. The parameter k is used to limit the calculations and, just as for wkNN, there are no general rules on how to choose it. Hence, we will tune this parameter for each model separately.

We used the OWA-FRNN approach as the primary classification technique, with a Python implementation[8] provided by (Lenz et al., 2020).

2.4 Ensembles

To improve our results, we considered the usage of models' ensembles. The ensemble combines several classification models' outputs to provide the final prediction. The idea behind it is to improve the performance of a single model by combining it with other models to fuse their advantages.

We tuned the best setup for each embedding method separately (with parameters as text preprocessing and a number of neighbors k) and then united them in an ensemble. As a combination method for models outputs, or in other words, "a

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

voting function", we used the mean. We tuned the set of the models used in the ensemble to prevent weak models from decreasing the final score. All obtained results are presented in Section 3.

2.5 Evaluation

To compare different approaches on the train data, we used a 5-fold cross-validation technique by splitting the provided dataset into train and test data with an $80/20$ ratio.

As an evaluation metric for predicted labels and actual labels, we used the F1-score (Formula (8)) calculated for the sarcastic class, as was suggested by the competition organizers.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}, \qquad (8)$$

where by $Precision$ we mean the fraction of accurately predicted sarcastic labels from all predicted test labels, and by $Recall$ we refer to the fraction of accurately predicted sarcastic labels from all actual sarcastic test labels. The highest value of the F1-score corresponds to the best model.

3 Results

This section consists of several parts. In Subsection 3.1 we provide an overview of our experiments, using cross-validation on the train data to identify the best model for irony detection and evaluate its result for test data. In Subsection 3.2, we describe error analysis and illustrate the explainability of our approach. We also performed additional experiments after the competition was finished and labels for the test data released to improve our scores and present this process in Subsection 3.3.

3.1 The best setup

Firstly, we evaluated the OWA-FRNN classification model on each embedding method separately by tuning the number of neighbors k and text preprocessing techniques. The best setup for each embedding method with the best parameters and corresponding F1-score for the sarcastic class is shown in Table 1.

Table 1 lists all methods in decreasing order by F1-score. The best result was obtained with the roBERTa-based model with a noticeable gap for the next method - DeepMoji. It can be seen that for almost all embedding techniques, the value of k is equal to 5, whereas for text preprocessing, no particular pattern was observed.

Table 1: Cross-validation best F1-scores for different embedding methods and corresponding setups.

Method	Text	k	F1 sarcastic
roBERTa	raw	5	**0.3722**
DeepMoji	cleaned	5	0.3157
USE	raw	5	0.2808
BERT	cleaned, no stop-words	5	0.2351
Word2Vec	cleaned, no stop-words	5	0.2050
SBERT	raw	7	0.1618

Table 2: Cross-validation F1-scores for ensembles of embedding methods.

Embeddings	F1-score
All six	0.0995
TOP-5	0.1866
TOP-4	0.1317
TOP-3	0.2941
TOP-2	0.1866

Secondly, we combined different embedding methods with their best setups as an ensemble. We calculated the mean for all embedding methods, then for the top-5 (excluding the weakest one - SBERT), top-4 (excluding Word2Vec and SBERT), and so on, until the single top model roBERTa is left. The results are presented in Table 2.

From Table 2 we can see that ensembles provided lower scores than single models. The standalone roBERTa performed better than in ensembles with others, which could already be expected from Table 1, where the gap between the roBERTa method and the rest is remarkable.

Hence, we can conclude that the best setup has the following components: no text preprocessing, roBERTa-based embedding technique for vectors extraction, and an OWA-FRNN classification model with a number of neighbors $k = 5$. This setup was applied to the test dataset.

We calculated labels for the test data using this setup and submitted them to the competition to obtain **F1-score = 0.4242** for the sarcastic class, leading to a **9th place** in the leader-board. Meanwhile, we received higher places for some other metrics: 7th place for averaged F-score = 0.6552 and precision = 0.6422, and 8th place for accuracy = 0.8100.

3.2 Error analysis

For the error analysis, we could rely on the fact that the OWA-FRNN classifier we experimented with has the advantage of being more explainable compared to many other black box approaches. In our case, we mean that we can trace back the test instance and see which five instances from the train data determined its class.

Initially, we traced back several correctly predicted test tweets to see if it was possible to notice any patterns. For example, for the sarcastic test tweet *"So the Scottish Government want people to get their booster shots so badly that the website doesn't even work"*, we got four sarcastic training neighbors out of five. Four neighbors were connected to the health topic and contained collocations such as *"mental health"*, *"health insurance"*, *"covid vaccine"*, and *"healthcare"*. The fifth neighbor was about emails that could be connected to *"website"* word from the test tweet. From this sample, we could conclude that having a common topic is an important feature for neighbors 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: *"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 neighbors about daily routine and lifestyle with mostly non-sarcastic labels, leading to the wrong prediction. For example, the closest training neighbor *"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 labeling and how subjective it could be.

In general, we can see that topic could be a strong feature. However, the same concept could have different meanings in different topics (for example, *"temperature"* in weather or fever). Also, some neighbors have the same emojis as a test instance that can give a hint about emojis importance.

3.3 Model improvements

After the test labels were released, we experimented with more setups to improve our final F1-score.

First, we used other values of the parameter k that showed mediocre results on cross-validation to see how they perform on the test data. For example, we observed that for $k = 17$, we receive an F1-score with cross-validation equal to 0.3408, which is lower than our best setup. However, on the test data, this value of k provided us an F1-score equal to 0.5, which is more than what we got in the leader-board and would lead us to fourth place.

Secondly, we considered the usage of the weighted k Nearest Neighbors (wkNN) algorithm (Dudani, 1976). This approach is close to OWA-FRNN, and we already worked with it in our previous paper (Kaminska et al., 2021b) for the emotion detection task. The wkNN works with k closest neighbors and puts weights based not on the OWA operator but on the neighbors' distances. As a similarity function, we used cosine again.

To test the wkNN method, we applied it inside our best setup - roBERTa-based embedding vectors obtained from the raw tweets and $k = 5$. We got an F1-score for the sarcastic class of 0.3569 with cross-validation and 0.4299 for the test data. We can see a minor improvement for the test data, compared to our final scores from the leader-board.We also checked this setup for $k = 17$ and obtained a sarcastic F1-score for cross-validation equal to 0.2790 and 0.4969 for the test data. It would also top us up to the fourth place, and so in general, we can see that results for the OWA-FRNN and the wkNN methods in our setups are pretty close.

4 Conclusion & Future Work

In this paper, we presented our model for the iSarcasmEval competition. Our solution uses the instance-based classification method OWA-FRNN and a roBERTa-based model for Irony Recognition as an embedding method. We fine-tuned the best setup on the train data with cross-validation and obtained the ninth place on the test data in the competition leader-board.

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. For example, we observe the significance of the tweet topic and even of particular keywords.

In the future, the provided solution may be improved by additional text preprocessing techniques or roBERTa-based model fine-tuning using additional datasets.

Acknowledgment

This work was supported by the Odysseus programme of the Research Foundation-Flanders (FWO).

References

Ibrahim Abu Farha, Silviu Oprea, Steve Wilson, and Walid Magdy. 2022. Semeval 2022: isarcasmeval - intended sarcasm detection in english and arabic. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Daniel Chandler and Rod Munday. 2011. *A dictionary of media and communication*. OUP Oxford.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186.

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

Bilal Ghanem, Jihen Karoui, Farah Benamara, Paolo Rosso, and Véronique Moriceau. 2020. Irony detection in a multilingual context. *Advances in Information Retrieval*, 12036:141.

Richard Jensen and Chris Cornelis. 2011. Fuzzy-rough nearest neighbour classification and prediction. *Theoretical Computer Science*, 412(42):5871–5884.

Olha Kaminska, Chris Cornelis, and Veronique Hoste. 2021a. Fuzzy-rough nearest neighbour approaches for emotion detection in tweets.

Olha Kaminska, Chris Cornelis, and Veronique Hoste. 2021b. Nearest neighbour approaches for emotion detection in tweets. In *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 203–212.

Oliver Urs Lenz, Daniel Peralta, and Chris Cornelis. 2019. Scalable approximate frnn-owa classification. *IEEE Transactions on Fuzzy Systems*, 28(5):929–938.

Oliver Urs Lenz, Daniel Peralta, and Chris Cornelis. 2020. fuzzy-rough-learn 0.1: a Python library for machine learning with fuzzy rough sets. In *IJCRS 2020: Proceedings of the International Joint Conference on Rough Sets*, volume 12179 of *Lecture Notes in Artificial Intelligence*, pages 491–499. Springer.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50.

Sarah Vluymans, Neil Mac Parthaláin, Chris Cornelis, and Yvan Saeys. 2019. Weight selection strategies for ordered weighted average based fuzzy rough sets. *Information Sciences*, 501:155–171.