NULL at SemEval-2022 Task 6: Intended Sarcasm Detection Using Stylistically Fused Contextualized Representation and Deep Learning

Mostafa Rahgouy
Department of Computer Science
Auburn University, Alabama, USA
mzr0108@auburn.edu

Hamed Babaei Giglou
Department of Computer Science
University of Tabriz, Tabriz, Iran
h.babaei98@ms.tabrizu.ac.ir

Taher Rahgooy
Department of Computer Science
University of West Florida, Florida, USA
trahgooy@students.uwf.edu

Cheryl D Seals
Department of Computer Science
Auburn University, Alabama, USA
sealscd@auburn.edu

Abstract

The intended sarcasm cannot be understood until the listener observes that the text's literal meaning violates truthfulness. Consequently, words and meanings play an essential role in specifying sarcasm. Enriched feature extraction techniques were proposed to capture both words and meanings in the contexts. Due to the overlapping features in sarcastic and non-sarcastic texts, a CNN model extracts local features from the combined class-dependent statistical embedding of sarcastic texts with contextualized embedding. Another component BiLSTM extracts long dependencies from combined non-sarcastic statistical and contextualized embeddings. This work combines a classifier that uses the combined high-level features of CNN and BiLSTM for sarcasm detection to produce the final predictions. The experimental analysis presented in this paper shows the effectiveness of the proposed method.

1 Introduction

Sarcasm detection is a specific case of sentiment analysis where the focus is on detecting sarcasm in text. Therefore, the task is to detect if a given text is sarcastic or not. According to (Hacker, 2011), sarcasm refers to the use of words that mean the opposite of what you want to say, especially to insult someone, show irritation, or provide humor. However, (Oprea and Magdy, 2020; Wilson, 2006) define sarcasm as a form of irony marked by a discrepancy between the literal and intended meanings of an utterance, through which the speaker usually manifests a hostile, derogatory, or contemptuous attitude. Generally, sarcasm tweets/texts are the utterances of a positive statement with harmful intent, and since the intent is hard to detect not only for computers but also for humans, it has attracted a considerable body of research in the natural language processing field to study opinions given by the users of online social media platforms such as Twitter, Facebook, Reddit, and Instagram. In the SemEval 2022 at Task 6: iSarcasmEval (Abu Farha et al., 2022), the goal is to identify intended sarcasm in both English and Arabic languages. This task consists of 3 subtasks, and we will focus on SubTask A defined as: *SubTask A: Given a text, determine whether it is sarcastic or non-sarcastic.*

Since sarcasm is an utterance of a statement with negative intent (e.g., *He has the best taste in music.*") and an admiring tone (e.g., "*She always makes dry jokes.*"). We hypothesized that contextualized representation might not be enough to represent intent when considering the sentiment analysis challenge. Therefore, we combined the representation of stylistically statistical and contextualized word embeddings using deep neural networks (DNNs) as a high-level feature extractor and classifier for the task. In this work, we studied the effect of the combination of stylistic and contextualized representation. The rest of the article is organized as follows. We first describe sarcasm studies in Section 2. Section 3 presents the proposed methodology. Sections 4 and 5 describe experimental setups and results. Moreover, in section 6 we made the discussion. Finally, in section 7 we conclude the article.

2 Related Works

Sarcasm detection research has seen a significant surge in interest in the past few years (Potamias et al., 2020; González et al., 2020; Babanejad et al., 2020; Gregory et al., 2020; Kumar et al., 2021; Ahuja and Sharma, 2021). At early stages, most works considered content as the only source for identifying sarcasm (Yaghoobian et al., 2021). To differentiate between sarcasm and non-sarcasm handcrafted rules and features have been applied (Veale and Hao, 2010; Bharti et al., 2015; Riloff et al., 2013). For instance, (Barbieri et al., 2014) used seven sets of lexical features to detect sarcasm by its inner structure. Frequency and Structure (i.e., length, punctuation, emoticons) are two examples of sarcasm features. However, the model based on these stylistic features shows promising results but lacks external knowledge (i.e., common sense). Our proposed method incorporates stylistic features with XLM-RoBERTa model to cope with the limitations of prior methods. Most recently, dominant approaches are Context-based, utilizing background knowledge and contextual dependencies as prior knowledge about the event mentioned in a sarcastic context (Li et al., 2020; Agrawal et al., 2020; Potamias et al., 2020). (Wallace et al., 2014) provided empirical evidence that to make judgments concerning ironic intent annotators require additional contextual information. Hazarika et al. is one of a few works that tried to capture the stylometric and personality features of the users and used user embeddings to encode stylometric and personality features of users combined by a content-based feature extractor (i.e., CNN)(Hazarika et al., 2018). This sarcasm detection model shows promising results on a large Reddit corpus. The improvement in results by this work inspired us to investigate the effectiveness of incorporating content-based features for SemEval-2022 Task 6: iSarcasmEval, but the problem we faced was the lack of data assigned to each user. As a result, instead of a users' unique features, we investigate the stylometric features of each class. In other words, we tried to find general styles of users who tried to write sarcastic texts by finding the distribution of their frequent words and other features provided in section 3. Results provided in section 5 empirically demonstrate the effectiveness of this incorporation. The effectiveness of transfer learning has given rise to a diversity of approaches, methodology, and practice, and they are also gaining attention in Sarcasm detection. BERT, RoBERTa, and XLNet are examples that deliver significant improvements. For example, (Potamias et al., 2020) proposed Recurrent CNN RoBERTa (RCNN-RoBERTa), which obtained %79 on the SARC dataset. Based on its effectiveness and multilingualism, we utilized the RoBERTa model for this research.

3 Model Description

In this section, we describe the details of our proposed deep learning model. The goal is to predict whether the given text is sarcastic or non-sarcastic. Figure 1 depicts an overview of our proposed method.

First, preprocessing (Appendix A.1) is applied to the raw texts. Next, character-based lower dimensionality statistical embedding (Char-LDSE) (Rangel et al., 2017; Giglou et al., 2021) captures the stylistic information about sarcastic and non-sarcastic data in form of class-dependent features based on the term frequency of tokens in sarcastic and non-sarcastic classes. The calculated LDSE features for sarcastic class in combination with contextualized representation are fed into Convolutional Neural Networks (CNNs) (LeCun et al., 1999) to extract local information of the texts. In another setting, the LDSE for non-sarcastic class in combination with contextualized representation is fed into Bidirectional Long Short-Term Memory (BiLSTM) (Schuster and Paliwal, 1997) to learn the long-dependent information in the texts. The features from CNN and BiLSTM layers combined for enhancing the feature extraction ability of the model. Extracted high-level features are fed into the classifier. In the following, we describe each component elaborately.

3.1 Representation

Preprocessed texts are fed into Char-LDSE and contextualized representations. Where it outputs an embedding for models.

Char-LDSE: First, a character n-gram matrix with TFIDF weight is created. Using TFIDF matrix the LDSE weighted probability of terms per class was obtained. As a result, $P(sarcastic)$ and $P(non-sarcastic)$ embeddings are calculated for sarcastic and non-sarcastic, respectively. The union of n-grams with TFIDF weighing on the training set is applied to achieve the input matrix of LDSE. We utilized character-level n-grams features with an n-gram range of $(2, 3)$, and word-level unigrams features for building input matrix to LDSE. For English character-level and word-level settings, and for Arabic, the only character-level setting is applied. As a result, we obtain the following matrix:

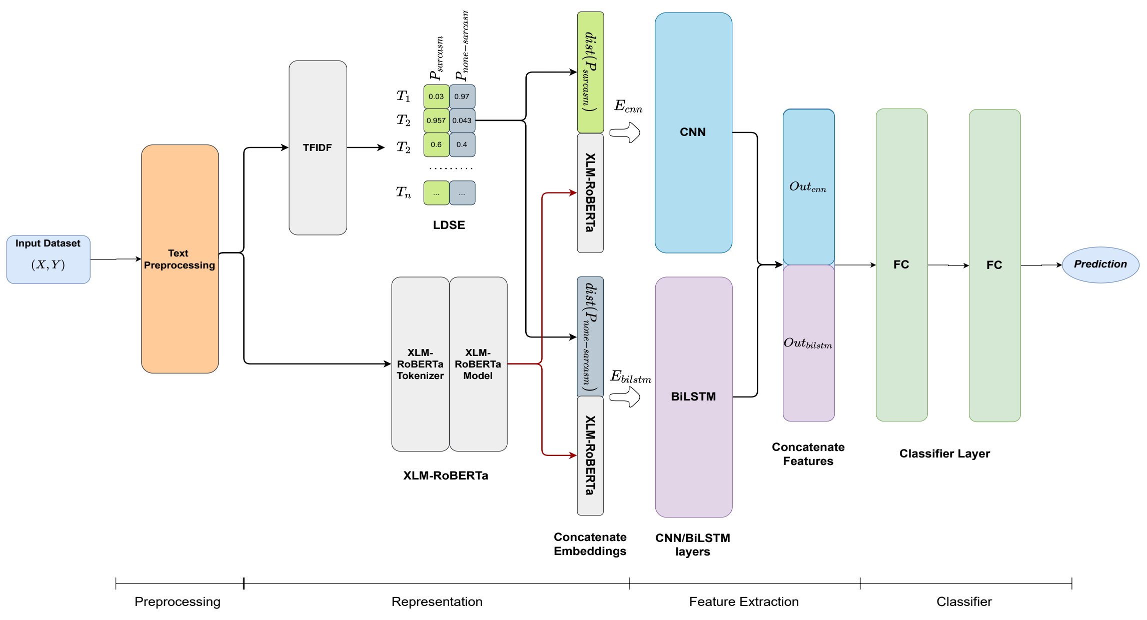

Figure 1: Architecture of Proposed Model

$$M = \left[\begin{array}{cccc|c} W_{11} & \dots & W_{1n} & & \beta(iS_{C_0}) \\ W_{21} & \dots & W_{2n} & & \beta(iS_{C_0}) \\ \dots & \dots & \dots & & \dots \\ W_{(m-2)1} & \dots & W_{(m-2)n} & & \beta(iS_{C_1}) \\ W_{(m-1)1} & \dots & W_{(m-1)n} & & \beta(iS_{C_1}) \\ W_{m1} & \dots. & W_{mn} & & \beta(iS_{C_1}) \end{array}\right]$$

Where each row in the matrix M represents a intended Sarcasm (iS) iS_i, each column represents vocabulary term T and W_{ij} represents its TFIDF weight and β represents the assigned class (C_1 - sarcastic, C_0 - non-sarcastic) of the iS text. Also, m and n represent the number of the training set and vocabulary size, respectively. To obtain the class-dependent term T weight embedding LDSE the following $LDSE(T, c)$ formula has been calculated:

$$LDSE(T, c) = \frac{\sum_{is \in \beta(iS_c)/c=\beta(iS_c)} W_{is,T}}{\sum_{is \in \beta(iS_c)} W_{is,T}}$$

$$\forall is \in iS,\ c \in \{C_0, C_1\}$$

Next, we calculated LDSE for each class for term T:

$$P_{sarcastic} = LDSE(T, c)$$

$$P_{non-sarcastic} = LDSE(T, c)$$

At the end, using $P_{sarcastic}$ and $P_{non-sarcastic}$ we extract the following representations:

$$feat(P_{sarcastic})\ ,\ feat(P_{non-sarcastic})$$

Where $feat(P)$ contains the set of features presented as followings:

$$feat(P) = \{max, min, std, avg, Q_1, ..., Q_{100}\}$$

Where $feat(P) \in \mathbb{R}^{104}$ is the class-dependent LDSE features. Accordingly, max, min, std, and avg are maximum, minimum, standard deviation, and average of weights P, respectively. Moreover, the $\{Q_1, ..., Q_{100}\}$ is the Q-th quantile of P's.

Contextualized Embedding: The XLM-RoBERTa (Conneau et al., 2020) is a pre-trained language model with the Masked Language Modeling (MLM) objective. The XLM-RoBERTa is a multilingual version of RoBERTa (Liu et al., 2019), which is pre-trained on 2.5TB of CommonCrawl data containing 100 languages. The model learns an inner representation of 100 languages that can then be used to extract features useful for downstream tasks. The function $CE \in \mathbb{R}^{768}$ is a base version of contextualized XLM-RoBERTa word embeddings. CE takes iS texts, uses XLM-RoBERTa tokenizer and XLM-RoBERTa model for word embedding generations.

$$CE(iS) := XLMRoBERTa(iS)$$

Concatenate Embeddings: In representations combination, we took the following approach. For CNN layer we combined $feat(P_{sarcastic})$ with $CE(iS)$, and for BiLSTM layer we combined $feat(P_{non-sarcastic})$ with $CE(iS)$. We obtained the following equations.

$$f\colon feat(P_{sarcastic}) \oplus CE(iS) \rightarrow E_{cnn}$$

$$f\colon feat(P_{non-sarcastic}) \oplus CE(iS) \rightarrow E_{bilstm}$$

Where $E_{cnn}, E_{bilstm} \in \mathbb{R}^{872}$, which will be the inputs of CNN and BiLSTM models in feature extraction module.

3.2 Feature Extraction

The CNN is applied on E_{cnn} and BiLSTM is applied on E_{bilstm} to extraction high-level features. Next, outputs of CNN and BiLSTM models are concatenated as an input to the classifier.

CNN Feature Extractor: CNN is a feature extraction technique with strong adaptability and is good at mining data local characteristics. Our CNN's has architecture as follows:

1. Two CNN layer with $Conv2D : conv2d \rightarrow maxpooling1d \rightarrow activation$ scheme.
2. Concatenate output of the two CNN layers: $Out_{cnn} = Conv2D_1 \oplus Conv2D_2$.
3. Dropout layer.

Where $Out_{cnn} \in \mathbb{R}^{200}$ with output channel number of 100 in both CNN layers. The first layer in CNN uses a kernel size of $(4, 872)$, padding of $(2, 0)$, and stride of 3. However, the second layer employs a kernel size of $(3, 872)$, padding of $(1, 0)$, and stride of 2. Next, for an activation function, the Gaussian Error Linear Unit (GELU) (Hendrycks and Gimpel, 2020) that nonlinearity weights inputs by their percentile, rather than gates inputs by their sign as in ReLUs (Agarap, 2019) is applied. It has been used in most of the transformers.

BiLSTM Sequence Learner: For a better representation of contextual information, single BiLSTM layer was employed. The BiLSTM is composed of two LSTM network that is capable of reading input texts in both directions, forward and backward. The forward LSTM processes information from left to right, and backward LSTM processes information vice versa and hidden state can be shown as:

$$\overrightarrow{h}_t = LSTM(E_{bilstm}, \overrightarrow{h}_{t-1})$$

$$\overleftarrow{h}_t = LSTM(E_{bilstm}, \overleftarrow{h}_{t+1})$$

Next, output of BiLSTM can be summarized as following where the max-pooling is applied to output h_t to retrieves a maximum value of each feature in the BiLSTM layer.

$$h_t = \overrightarrow{h}_t \oplus \overleftarrow{h}_t$$

$$Out_{bilstm} = MaxPool1d(h_t)$$

Where $Out_{bilstm} \in \mathbb{R}^{200}$.

Concatenate Features: As a final high-level feature, the Out_{cnn} and Out_{bilstm} outputs are concatenated as follows, Where $Out_{cnn \oplus bilstm} \in \mathbb{R}^{400}$.

$$Out_{cnn \oplus bilstm} = Out_{cnn} \oplus Out_{bilstm}$$

3.3 Classifier

The two fully connected layers use $Out_{cnnbilstm}$ for classification objectives. The first layer takes 400 neurons as input and outputs 200 neurons. Next, GELU activation is applied. Finally, the second layer takes 200 neurons and outputs 2 neurons (the prediction module).

4 Experimental Setup

Dataset: For sarcastic or non-sarcastic detection tasks, the collected dataset where the sarcasm labels for texts are provided by authors themselves, thus eliminating labeling proxies. For hyperparameter tuning and model selection we split the training set into train and dev sets with a 10% split rate for both English and Arabic languages. More information about the datasets (train, dev, and test sets) has been presented in Appendix A.2. The dataset is highly imbalanced.

Training Setup: The preprocessing (Appendix A.1) is applied to text, first. Next, using train and dev sets we made hyperparameters tuning (Appendix A.3). Due to the imbalanced setting of the data we made a data argumentation (Appendix A.4) as one of our experiments. The experimental design shows the positive effect of data augmentation. More information about preprocessing, hyperparameter setting, and data augmentation are described in the appendix section. As an evaluation metric to this task (SubTask A of iSarcasmEval), the main metrics for SubTask A is F1-score for the sarcastic class.

5 Results

Main Quantitative Findings: The table 1 presents the final results on the test set for English and Arabic languages. The main quantitative findings are:

Language	Accuracy	Precision	Recall	F1-Score	F1-Sarcastic	FP	FN	Rank
English	0.6136	0.5290	0.5558	0.4992	**0.2599**	436	105	28
Arabic	0.6671	0.5835	0.6600	0.5667	**0.3581**	396	70	18

Table 1: Evaluation results on test set for SubTask A.

- For English, the proposed model achieved 28th place among 43 teams, and 18th place among 32 teams for the Arabic.
- According to type I error (FP), the model for both languages wrongly predicts the high rate of non-sarcastic samples as sarcastic data. Regarding this phenomenon, the task suffers from type I error mostly.
- For the English language, among 200 sarcastic samples model captures 95 samples correctly, however, for the Arabic language, 130 samples were predicted correctly as sarcastic. About 65% TPR (True Positive Rate) of sarcastic samples in Arabic, but 47.5% of sarcastic samples in English were detected. It shows the task is more sensitive in English rather than Arabic.

Quantitative Analysis: We have used experimentations on the dev set to conclude the CNN-BiLSTM model as a final system for intended sarcasm detection at iSarcasmEval. The experiments are presented in Appendix A.5, and the table 6 shows experimental results. The quantitive analysis is presented as follows:

- The character-based n-gram with an n-gram range of $(2, 3)$ was implemented as a baseline representation with logistic regression, linear SVM, and multi-layer perceptron (MLP) as baseline models. The experimental results on both English and Arabic languages show neural networks (MLP) are performing well on this task.
- The experimentation of *LDSE + MLP* shows that stylistic representations are promising, but they are not enough.
- To find a candidate representation in combination with LDSE, as a performance booster, the XLM-RoBERTa masked language model is been considered. The *XLM-RoBERTa + MLP* shows that contextualized representation is very capable.
- The *Combination + MLP* model is combined LDSE and XLM-RoBERTa representation. The achieved result is promising for Arabic but it is 2% higher for English, this shows stylistic features in combination with contextualized representation perform well. The more complex model may boost the performance.
- The *Proposed Model (1)* uses preprocessing followed by the enriched representation with CNN-BiLSTM model. It is more capable than *Combination + MLP* model.
- Due to the unequal class distribution in the dataset, we employed weights in the cross-entropy loss function. The *Proposed Model (2)* results show its positive effects.
- We experimented with data augmentation techniques (Appendix A.4) and discovered that due to the word importance in achieving high-quality features for LDSE, data augmentation works more considerably well for this task (*Proposed Model (2) + Aug* model).
- The data augmentation probability shifts show the effects of LDSE representations (table 4). And considering *Proposed Model (2)* and *Proposed Model (2) + Aug*, We can observe that data augmentation affects mostly the stylistic features and results in boosting the performance. It represents that high-level feature extraction from a multi-space domain such as LDSE and XLM-RoBERTa play an important role in sarcasm detection.

6 Discussion

Type-1 Error: In either language, non-sarcastic samples are predicted wrongly as sarcastic. Type-1 error is the major source of error. Analysis of English FP samples indicated that most of the samples were in response to a post or reply to another person and indeed, with no information about the text context nor the writer profile, this indicates a lack of primordial clues for sarcasm detection. To make an improvement in reducing the error two approaches can be taken. First, consider more information about the text context such as the tread or post. Second, using a finer feature representation, since the lack of primordial clues leads to low-quality features. Moreover, the analysis revealed the following findings in the data which can be taken into account to improve the results by reducing the error (all of the findings come from the analysis of 436 FP samples for the English test set).

- We have found that samples in the dataset may be labeled wrongly as sarcasm so correcting them may boost the performance (e.g *"10 dec 2021, 4:46 am, All in on $PYR (28,07$)"* or *"pointing-downmedium-skin-tone-emoji this"*)
- A few samples were in single words (e.g *"Rubbish"*) which in a few cases such as *"pointing-downmedium-skin-tone-emoji this"* and *"Followed"* they didn't convey any meaning, tackling them in modelin could be more challenging since they don't have enought information for models.
- In a few cases preprocessing causes a low quality text generation (e.g *"10 dec 2021, 4:46 am, All in on $PYR (28,07$)"* will be converted into *'dec am all in on"* which is less valuable for modeling).
- Sometimes, emojis make the text sarcasm, so using them as features could be useful for this task since most of the comments in FP samples had emojis (almost 60% of them). In the whole preprocessing we converted emojis into texts, but in this way we thread the emojis similar to text, however, threading them in a different way may boost the performance.

Weighted Loss and TFIDF Data Augmentation Improvements: According to table 6, and model *Proposed Model (2)*, adding weighted loss cross-entropy increased results on the dev set by 1% in English, and 2% in Arabic. Similar to weighted loss cross-entropy, the model *Proposed Model (2) +Aug* in table 6 shows TFIDF data augmentation improved results on the dev set by 1% in English, and 4% in Arabic. Overall, weighted loss cross-entropy and TFIDF data augmentation together improved results by 2% in English, and 6% in Arabic. So using these techniques is promising in boosting the sarcasm detection models.

Task Sensitiveness in Languages: We proposed a class-dependent LDSE that considers character n-grams in stylistic information calculations. Character n-gram contains information on the more important tokens and the less important ones as well. So it captures the differences using token counts. However, The stress patterns of most Arabic dialects are broadly similar and appear regularly. Changes happen frequently in English, as word stress can change the lexical variety and meaning of the word (Watson et al., 2011). This affects both LDSE and contextualized representation.

Because LDSE is doing a probability-based calculation for tokens using a character n-gram matrix, the similarity in dialects will lead to a high-quality n-gram matrix in Arabic rather than English. Table 6 experimentation on the TFIDF matrix reveals this fact precisely. Accordingly, as a result of changes in the meaning, neural network feature extractors may have found some variant features which leads to misclassification (Type 1 error). The experimentations with *XLM-RoBERTa + MLP* (table 6) show that neural network models are performing better on Arabic contextualized representation rather than English. Experiments clearly show why this task is more sensitive in the English language than in Arabic.

7 Conclusion

This paper presented our approach for SemEval-2022 Task 6: iSarcasmEval: Intended Sarcasm Detection In English and Arabic. We investigated this problem by employing statistical and contextualized representations with deep learning techniques. We conducted the experimental and statistical analysis and presented the CNN-BiLSTM framework. The proposed studies in this paper show that considering stylistic features with deep learning frameworks is promising for intended sarcasm detection in both English and Arabic languages.

References

Ibrahim Abu Farha, Silviu Oprea, Steven Wilson, and Walid Magdy. 2022. SemEval-2022 Task 6: iSarcasmEval, Intended Sarcasm Detection in English and Arabic. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics.

Abien Fred Agarap. 2019. Deep learning using rectified linear units (relu).

Ameeta Agrawal, Aijun An, and Manos Papagelis. 2020. Leveraging transitions of emotions for sarcasm detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1505–1508.

Ravinder Ahuja and SC Sharma. 2021. Transformer-based word embedding with cnn model to detect sarcasm and irony. *Arabian Journal for Science and Engineering*, pages 1–14.

Nastaran Babanejad, Heidar Davoudi, Aijun An, and Manos Papagelis. 2020. Affective and contextual embedding for sarcasm detection. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 225–243.

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in twitter, a novel approach. In *proceedings of the 5th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 50–58.

Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena. 2015. Parsing-based sarcasm sentiment recognition in twitter data. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1373–1380. IEEE.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Hamed Babaei Giglou, Taher Rahgooy, Jafar Razmara Mostafa Rahgouy, and Zahra Rahgooy. 2021. Profiling Haters on Twitter using Statistical and Contextualized Embeddings—Notebook for PAN at CLEF 2021. In *CLEF 2021 Labs and Workshops, Notebook Papers*. CEUR-WS.org.

José Ángel González, Lluís-F Hurtado, and Ferran Pla. 2020. Transformer based contextualization of pre-trained word embeddings for irony detection in twitter. *Information Processing & Management*, 57(4):102262.

Hunter Gregory, Steven Li, Pouya Mohammadi, Natalie Tarn, Rachel Draelos, and Cynthia Rudin. 2020. A transformer approach to contextual sarcasm detection in twitter. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 270–275.

Hacker. 2011. Merriam-webster.com (8 may 2011).

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. Cascade: Contextual sarcasm detection in online discussion forums. *arXiv preprint arXiv:1805.06413*.

Dan Hendrycks and Kevin Gimpel. 2020. Gaussian error linear units (gelus).

Avinash Kumar, Vishnu Teja Narapareddy, Pranjal Gupta, Veerubhotla Aditya Srikanth, Lalita Bhanu Murthy Neti, and Aruna Malapati. 2021. Adversarial and auxiliary features-aware bert for sarcasm detection. In *8th ACM IKDD CODS and 26th COMAD*, pages 163–170.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, page 319, Berlin, Heidelberg. Springer-Verlag.

Meimei Li, Chen Lang, Min Yu, Yue Lu, Chao Liu, Jianguo Jiang, and Weiqing Huang. 2020. Scx-sd: Semi-supervised method for contextual sarcasm detection. In *International Conference on Knowledge Science, Engineering and Management*, pages 288–299. Springer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Edward Ma. 2019. Nlp augmentation. https://github.com/makcedward/nlpaug.

Silviu Oprea and Walid Magdy. 2020. isarcasm: A dataset of intended sarcasm.

Rolandos Alexandros Potamias, Georgios Siolas, and Andreas-Georgios Stafylopatis. 2020. A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, 32(23):17309–17320.

Francisco Rangel, Marc Franco-Salvador, and Paolo Rosso. 2017. A low dimensionality representation for language variety identification.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 704–714.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Tony Veale and Yanfen Hao. 2010. Detecting ironic intent in creative comparisons. In *ECAI 2010*, pages 765–770. IOS Press.

Byron C Wallace, Laura Kertz, Eugene Charniak, et al. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–516.

Janet CE Watson et al. 2011. Word stress in arabic.

Deirdre Wilson. 2006. The pragmatics of verbal irony: Echo or pretence? *Lingua*, 116(10):1722–1743. Language in Mind: A Tribute to Neil Smith on the Occasion of his Retirement.

Hamed Yaghoobian, Hamid R. Arabnia, and Khaled Rasheed. 2021. Sarcasm detection: A comparative study.

Dataset	Sarcastic	Non-Sarcastic	Overall
English			
Train	772	2349	3121
Dev	95	252	347
Test	200	1200	1400
Arabic			
Train	678	2113	2791
Dev	67	244	311
Test	200	1200	1400

Table 2: English & Arabic dataset stats

Parameter	English	Arabic
Batch Size	8	8
Learning Rate	0.001	0.001
Epoch	10	10
Loss Function	CrossEntropy	CrossEntropy
Weighted Loss	Yes	Yes
Optimizer	Adam	Adam
Weight Decay	1e-9	1e-8
Dropout	0.6	0.4

Table 3: Hyperparameter setting

	English	Arabic
$Avg(P_{sarcastic})$	0.244	0.148
$Avg(P_{sarcastic})$+Aug	**0.327**	**0.213**
$Avg(P_{none-sarcastic})$	0.756	0.852
$Avg(P_{none-sarcastic})$+Aug	**0.673**	**0.787**
LDSE tokens	17691	16941
LDSE tokens + Aug	17961	17073
sarcastic samples	772	678
synthetic samples	760	678
sarcastic samples + Aug	1532	1356
overall training data + Aug	**3881**	**3469**

Table 4: Data augmenter stats

A Appendices

A.1 Preprocessing

Text preprocessing is often the first step in the pipeline of an NLP system. We consider the following preprocessing techniques to be applied to an input text:

- Lowercasing
- Punctuation removal
- Special character removal
- Demojify[1](converting emoji into texts)
- URL, #, @ removal
- Number removal

These preprocessing techniques were applied for both English and Arabic languages.

A.2 Train, Dev, and Test Sets

Each text in the datasets has a label specifying its sarcastic nature (sarcastic or non-sarcastic), provided by its author. The stats of the datasets are presented in table 2.

A.3 Hyperparameter Setting

For training DNNs there are multiple hyperparameters to be fine tuned. Batch size, learning rate, epoch, loss function, weight decay, dropout, wighted loss, and optimizer. The hyperparameters are described in the table 3 for both English and Arabic languages.

A.4 Data Augmentation

Data augmentation is a way to generate synthetic data for improving model performance without manual effort. Since we are dealing with an imbalanced binary classification problem and due to the importance of terms likelihood in LDSE we made a TFIDF data augmenter that learns the word preferences for samples in sarcastic class only, then generates new synthetic samples. For generating new samples we have followed the following steps:

1. Getting sarcastic samples from training data for data augmentation.
2. Preprocessing selected samples.
3. Training TFIDF data augmenter using *nlpaug* (Ma, 2019) a python library.
4. Augment sarcastic samples using data augmenter.
5. Generate new samples for sarcastic samples only.

The overall stats of the data augmentation are presented in the table 4. According to the mean average probability of terms in sarcastic and non-sarcastic, the data augmentation shifts the average probabilities in both classes. Since we were using the same set of data for argumentation so we observe the proper number of changes in tokens. Samples from data augmenter for both languages are presented in the table 5.

A.5 Experiments

The table 6 presents the experimental results for hyperparameter tunings. According to the table

[1]https://pypi.org/project/demoji/

ORG	mentally i m hiding in the walk in
AUG	mentally hiding jewelry the watto in
ORG	i love hour panic attacks
AUG	love limited spoiling attacks
ORG	العيون البصاصة تندب فيها رصاصة
AUG	العيون البصاصة تندب تضرب الأرض
ORG	عامل زى ابو الفراجين معندوش ...
AUG	عامل الحق متعالم الفراجين وغيره ...

Table 5: Data augmentation samples (ORG: original sample, AUG: augmented sample)

Model	English	Arabic
TFIDF + LR	0.16	0.56
TFIDF + LinearSVM	0.31	0.63
TFIDF + MLP	0.32	0.62
LDSE + MLP	0.27	0.58
XLM-RoBERTa + MLP	0.40	0.72
Combination + MLP	0.42	0.72
Proposed Model (1)	0.44	0.74
Proposed Model (2)	0.45	0.76
Proposed Model (2) + Aug	0.46	0.80

Table 6: Experimental results on dev set (F1-sarcastic is reported)

6, *Combination + MLP* refers to LDSE + XLM-RoBERTa Representation, *Proposed Model (1)* is the CNN-BiLSTM model and preprocessing, *Proposed Model (2)* is model with preprocessing, and weighted loss cross-entropy. *Proposed Model (2) + Aug* is a model with preprocessing, weighted loss cross-entropy and TFIDF data augmentation.