Sarcasm Detection Evaluation

Ruichen Li rul014@ucsd.edu
Zhichao Liu z5liu@ucsd.edu
Gongxuan Liu goliu@ucsd.edu
Shuyang Cui s3cui@ucsd.edu

1 Introduction

Sarcasm is a complex linguistic phenomenon that uses irony, in which the literal meaning is opposite to what it actually means (JoshiAditya, 2017). It occurs when people communicating in social media like Tweeter, Reddit, Facebook etc. However, sarcasm is difficult to be detected in the social media realm, because it occurs infrequently, needs context to be described, and is difficult for even humans to discern (Wallace et al., 2014). This leads to many explorations on the automatic sarcasm detection by researchers using different techniques.

Sarcasm detection is meaningful by its significant implication to different domains. For example, sarcasm detection can help social media to better understand a user's true intention, then resulting in efficient recommendation for user's preference. On the other hand, sarcasm detection can improve the exploration of complicated contextual interpretation, since sarcasm is always difficult to be analyzed in terms of language structure (Kolchinski and Potts, 2018). The better understanding sarcasm may contribute to better sentiment analysis by identifying irony or sarcastic features.

In our project, we review some past techniques used in sarcasm detection. Given the ideas provided by the previous techniques, we build a self-attention model to perform context based sarcasm detection and an encoder-based classification model to perform non-context based sarcasm detection, code attached below. [1].

[1] https://drive.google.com/drive/u/0/folders/0APkVs-DOwoWKUk9PVA

2 Related work

2.1 Sarcasm Detection using Logistic Regression Model on Rich Features

One approach to tackle sarcasm detection is to extract rich features about contextual information on data, and establish model to train and make prediction, which is similar to the methods used in sentiment analysis. Bamman and smith adopt binary logistic regression with l2 regularization to explore these features to decide whether a tweet is sarcastic or not (Bamman and Smith, 2015). In their works, they collected more than 3,200 tweets online and extracted features into 4 classes: tweet-features, author-features, audience-features, and response-features. For each combination of these 4 features, Bamman and Smith processed cross-validation and parameter tuning to produce various optimal parameters. By trying these optimal parameters in logistic regression model, their approach yield 84.3% accuracy in sarcasm detection on test-set. The similar approach is used by Rajadesingan to classify sarcasm on Reddit and Twitter (Ashwin et al., 2015).

The major disadvantage of their approach is that it required a huge manual works on the features collections. To enable a better prediction on tweets, we have to spend huge amount of time to collect and split the contextual information on each tweet, which may not be very efficient for larger domain.

2.2 Sarcasm Detection using Neural Convolution Model with User Embedding

To avoid the expensive manual collection on features, the researcher proposed a new approach by establishing neural network with pre-trained user embedding (Silvio et al., 2016). The author splits the model into two parts: convolution layer for vector lexical representation, and user embed-

ding for learning user context. The first part will capture the sarcasm features from the content itself, using pre-trained word embedding as input to convolution layer for training. The second part will capture the sarcasm features from the user, in which a user embedding is induced based on relation between users and their contents from previous messages. The method is described in Jiawei Li's work on using conditional probability to encode users' latent information and represent hemophilia (Jiwei et al., 2015). The final prediction on sarcasm is decided by jointed function of both lexical vector and user embedding. This model is named as "Content and User Embedding Convolutional Neural Network (CUE-CNN)"

The advantage is that CUE-CNN model doesn't rely on massive work on features collection, which is more efficient and can apply to larger domain. From authors' experiment, their model prediction increase 2% accuracy from Bamman and Smith work on Logistic Regression, indicating its good performance. The limitation is that User Embedding on context representation may need further exploration to better improve accuracy.

2.3 Sarcasm as Contrast between a Positive Sentiment and Negative Situation

Another notable contribution is the work by Riloff et al. on "Sarcasm as Contrast between a Positive Sentiment and Negative Situation" (Riloff et al., 2013). In their research, Riloff focus on the contrast between positive sentiment and negative situations as a key indicator of sarcasm. They develop a computational algorithm that leverages linguistic features, including *Positive Verb Phrases*, *Positive Predicative Expressions* , and *Negative Situations*, to automatically detect instances of sarcasm. In the dataset collection phase, a total of 3,000 tweets were analyzed, with 693 of them annotated as sarcastic, resulting in a precision rate of 23% if every tweet was classified as sarcastic. The results from the SVM classifiers yielded F scores ranging from 46% to 48%. The significance of their work lies in the recognition of the contrast between positive sentiment and negative situations as a distinguishing characteristic of sarcasm. The algorithm proposed by Riloff et al. offers several advantages. However, they identified that most false hits were due to overly general negative situation phrases. They emphasized the importance of future research in learning longer phrases that represent more specific situations to enhance the accuracy of sarcasm detection.

With the development of CNNs, some work has begun to focus on context modeling using deep neural networks. Amir et al. demonstrate a method that can capture the relationship between context and context representations and combined them with learned user embeddings (Hazarika et al., 2018). proposed CASCADE (ContextuAl SarCasm DEtector) and validate the importance of modeling the relationship between the context and the sarcasm sentences in social media discussions. CASCADE extracts user embeddings by analyzing historical posts, which are concatenated with contextual information and content information extracted by CNN to generate a final representation. The classification of the representation achieves a significant enhancement in the sarcasm classification task on SARC dataset.

2.4 Sarcasm Detection Using Pre-trained Language Models

In recent years, an increasing amount of work in large language models has provided researchers another approach in solving sarcasm detection. In a recent sarcasm classification dataset *iSarcasmEval* (Farha et al., 2022), most authors utilize pre-trained language models such as *BERT-large*, *RoBERTa-large*, *DeBERTa-large* and *XLMRoBERTa-large* for sarcasm detection. A common approach is considering the [CLS] token as the input representation for the sarcasm classification task. Many authors also mix the statistical features and text features for further classification improvements. However, detecting sarcasm remains a challenging task according to the test results, especially considering the massive amount of parameters used in their models. Since very limited work has been done in detecting sarcasm using smaller pre-train language models, a potential future work is to create efficient classification models using smaller pre-trained models while maintaining the same level of classification accuracy.

3 Your dataset

We collect two data-sets: 1) SARC data-set with 1 million Reddit labels, and 2) isSarcamEval dataset with 5735 tweet labels. For SARC data-set, we implement the context-based sarcasm detection; for isSarcamEval data-set, we implement the

non-context based sarcasm methods for detection.

3.1 SARC Dataset

SARC is a large self-annotated text data extracted from Reddit for sarcasm detection (Khodak et al., 2017). SARC contains more than 1.3 million labeled sarcastic text data for training and testing purposes. Given our limited computational resources, this data provides more than enough training labels for our own model. Text data in SARC contains the author, context, parent comments, topic, and labels. This amount of information is enough for the model detect the sarcastic elements in data.

Below is an example of SARC input/output pair. Input: 1) comments: 'NC and NH' 2) parent-comment: 'Yeah, I get that argument. At this point, I'd prefer is she lived in NC as well.' Output label: 0

SARC dataset consists of 50,000 comments, or 25000 for each label. They leverage GloVe word vectors as embeddings to facilitate sentiment analysis. These vectors are dimensioned at 50 and fed into three different networks: a multi-head attention network, a feed-forward network, and a simple attention network. The final classification is conducted by a fully connected layer. The data set includes both comments and parent comments, allowing for different concatenation strategies at various stages.

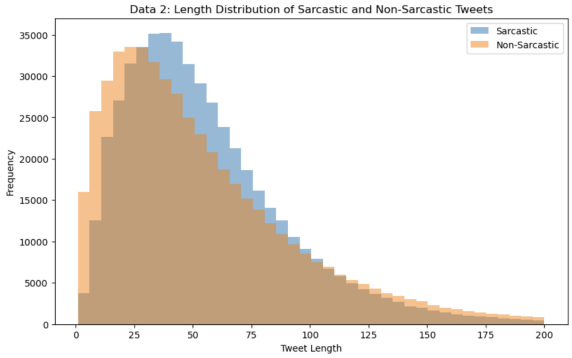

Figure 1: Sarcastic vs Non-sarcastic text length distribution of SARC

The problem of SARC is that all sarcastic text data is labeled by the Reddit users using the $\backslash s$ marker. Despite numerous metrics are applied to reduce the noise in data, we will still suffer from false positive and false negative text data.

3.2 iSarcasmEval Dataset

iSarcasmEval is a collection of tweets in English version for Sarcasm labels (Farha et al., 2022). The author collected them by eliminating labelling proxies. They also asked for linguistic experts to further label each tweets in categories of ironic speech (sarcasm, irony, satire,understatement). In statistic, there are around 5735 tweet comments, each labels as Sarcasm (1) or Non-Sarcasm (2). It is an unbalanced dataset with 1067 sarcastic tweets and 4668 non-sarcastic tweets for training and testing. Our goal is to use non-context based model to perform binary classification task on whether the given text is sarcasm or not.

Below is an example of iSarcasmEval input/output pair: Input: 1) tweet: See Brexit is going well. Output label: 1

The limitation of iSarcasmEval is that it only contains 3500 texts for training, which may be relatively small for training complex machine learning models. There are also limited information and features for each text labels, which makes it unable to implement more sophistic detection method like context-based model. Besides, we will use F1 score to report the performance due to its unbalanced data.

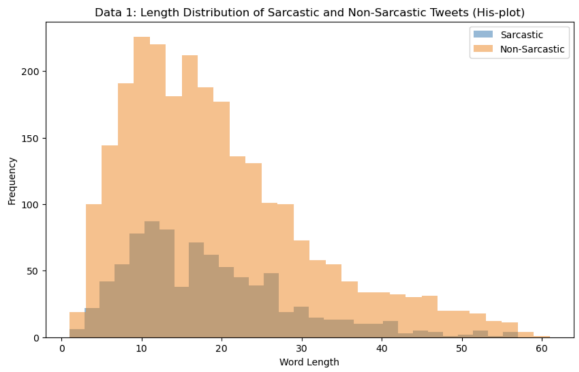

Figure 2: Sarcastic vs Non-sarcastic text length distribution of iSarcasmEval

4 Baselines

4.1 Baseline for Context-Based Sarcasm Detection

In our work, we first wanted to verify whether negative sentiment could be used as a cue to predict potential satirical content.

The two sentiment-based baselines established in our study are designed around the SARC dataset,

The first baseline (Baseline1) concatenates the comment and its parent comment upfront and inputs this into a singular network. The sentiment analysis, performed via Huggingface with model *bert-base-uncased*, is solely based on the comment. If the sentiment is negative, it predicts the content to be sarcastic. The performance of Baseline1 on the test set is 49.994%.

The second baseline (Baseline2), on the other hand, feeds the comment and its parent comment into two identical networks separately. The outputs from the two networks are then concatenated and passed through a fully connected layer for final classification. Here, the sentiment analysis with the same model is performed on both the comment and the parent comment. If the sentiments oppose each other, it predicts the comment to be sarcastic. The performance of Baseline2 on the test set was 50.062%.

Both baselines have been developed using a default set of hyperparameters, with no additional tuning performed on the test set to prevent any form of data leakage. It's noted that if we predict the sarcasm of a comment randomly, the accuracy would be foreseen around 50%. Therefore we believe our baselines are reasonably established. The rationale is that merely determining sarcasm based on sentiment analysis should yield results that are roughly equivalent to random guessing.

4.2 Baseline for Non-context Based Sarcasm Detection

Given the sarcasm classification task without context for additional information, we construct a simple baseline by applying fine-tuned DistilBERT to encode and classify the sentiment of the given input text. If negative sentiment is presented in the text, then we label it as sarcastic; if it has positive sentiment, then we label it as non-sarcastic. The fine-tuned model is *distilbert-base-uncased-finetuned-sst-2-english* provided by Hugging Face and the baseline model is tested on the test set of iSarcasmEval. The F1 score of it is 0.155. Since iSarcasmEval is an unbalanced dataset of sarcastic and non-sarcastic labels, we use the F1 score as the evaluation metric.

5 Your approach

5.1 Tools

We have identified several existing libraries that will be instrumental in training and evaluating our model. These libraries include torch, parameters, transformers, pytorch lightning, sklearn, and tqdm. We will develop our sarcasm detection model, which will leverage pre-trained GloVe word vectors(Pennington et al., 2014), self-attention-based models, and neural networks. Therefore, the aforementioned libraries will be crucial for various tasks such as loading language model checkpoints, processing text, and adding neural network layers. Given the availability of GPUs, we will utilize Google Colab for the training and evaluation of our model.

5.2 Approach for Context-Based Sarcasm Detection

Sarcasm often originates from the semantic inconsistency of the text and its content, and without context, even humans are not always able to tell if a comment is sarcastic or not. Therefore, it is necessary to study context-based sarcasm detection.

In this section, we design a series of networks based on self-attention modules and train and test our proposed methods on SARC dataset.

Considering that the text and its corresponding context are two inputs containing different sarcasm information, we use three different fusion strategies to fuse these two different inputs.

5.2.1 Methods

The network takes the GloVe word vectors as input and outputs the probabilities of sarcasm. The overall architecture including Feature Encoder, Self-Attention Encoder, Attention Pooling Module, and Linear Decoder.

Word Embedding: We first get plain text from the review text and review summary by splitting and removing all symbols and then concrete them. We then pad or cut out the tokens to the fixed length of 50. We experiment on widely used pre-trained 50-dimensional and 100-dimensional GloVe word vectors for each token, which provides a representation of a sentence of size $R^{len\_words \times dim}$, where len_words=50, dim=50 or dim=100.

Feature Encoder: We adapt 1D convolution as a linear encoder in order to extract simple patterns in each word vector and encode them to a high-level feature space. The dimension of the word vectors we use is relatively small, so we keep the output dimension the same as the input. The kernel size is set to 1 in order to keep the temporal information.

Self-Attention Encoder: We first adapt a multi-head self-attention layer with a feed-forward network proposed in [6] and we cancel non-linear layer to model the correlation or representation vectors in the word sequence. The core concept can be described below. Given three vectors of Key(K), Query(Q), and Value(V), which are generated by the linear transformation of the input in Eq. (5-7). Where X is the input, $X \in R^{len\_words \times dim}$, $W_Q, W_K, W_V \in R^{dim \times dim_K}$

$$Q = XW_Q \tag{1}$$
$$K = XW_K \tag{2}$$
$$V = XW_V \tag{3}$$

Then, the attention weights are calculated by Eq. (8), and the weighted sum of values is generated by Eq. (9).

$$\alpha = softmax(\frac{Q \times K^T}{\sqrt{dim_K}}) \tag{4}$$
$$Y = \alpha V \tag{5}$$

The feed-forward network maps the outputs from the multi-head self-attention module to a larger feature space and then maps it to the original space. We also add residual connections to the multi-head self-attention module and the feed-forward network, which enables the network to conduct residual learning.

Attention Pooling Module: The output of the Self-Attention Encoder is still a sequence of vectors, it is necessary to transform it into a single vector. Therefore, we calculated the global weight for each representation vector in the sequence and add them up, the method can be described as follow:

$$A = softmax(XW^T) \tag{6}$$
$$L = A^T X \tag{7}$$

where $A \in R^{len\_words \times 1}$, $L \in R^{1 \times dim_K}$ is the output of the Attention Pooling Module and $W \in R^{1 \times dim_K}$, is the trainable parameter matrix.

Linear Decoder: We use the fully connected neural network with sigmoid function to map the hidden representation from the Attention Pooling Module to the probability value space which is a single value of the probability of sarcasm. And the loss between the predicted valued and the true value is calculated by the Binary Cross Entropy function.

Pre-Fusion: In the pre-fusion strategy, we concatenate Comments and parent comments before inputting them into the network, with the aim of verifying whether the self-attention network could learn the inner semantic inconsistencies given a text.

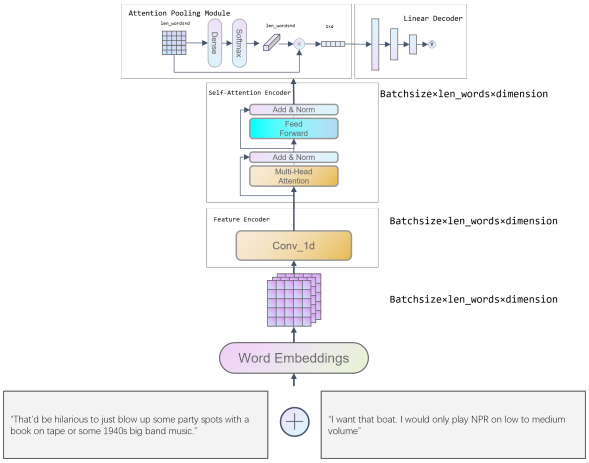

Figure 3: Pre-Fusion

Cross-Attention: In the cross-attention method, we fuse the information of parent comments into the comments by modifying the inputs of the self-attention module. Specifically, the Q and K in Eq. (1-2) are obtained from the word vectors of parent comments, and V in Eq. (3) is obtained from the word vectors of the comments being predicted. Therefore, each vector in Y in Eq. (5) is the weighted sum of vectors in V according to the attention weights calculated by the word vector sequence of the comment based on the word vector sequence of the parent comment.

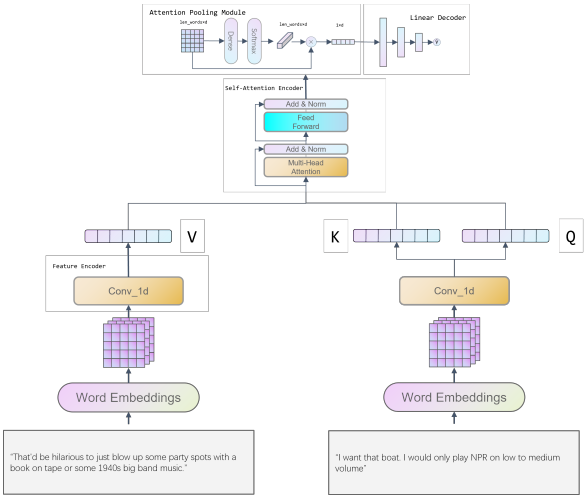

Figure 4: Cross-Attention

Table 1: Hyper-parameter Settings

Hyperparameter	Values
dimension in Linear Decoder	[dim_k,32,16]
dimension in Feed-Forward Network	2×dim_k
dim_k, dim_q, dim_v	50 or 100
learning rate	0.001
batch size	64
optimizer	Adam
dropout	0.1

Post-Fusion: In the post-fusion strategy, we use two independent self-attention networks described in the previous part to get representations of the comment and the parent comment. Then we concatenate these two representations and input them into a Linear Decoder to calculate the label of the input data.

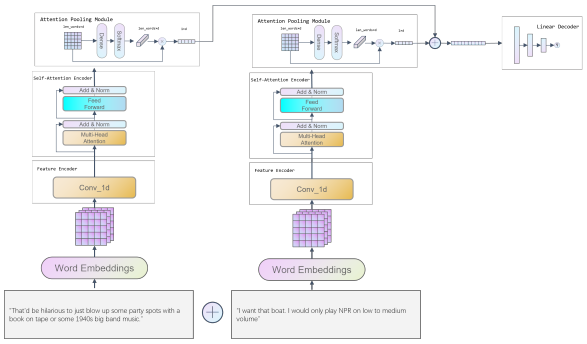

Figure 5: Post-Fusion

5.2.2 Experiments

We conduct experiments on the 50000 samples from the SARC dataset and 80% of the data are used to train the network and the remaining 20% are used to test. For the evaluation metrics, we use accuracy, precision, recall, F1 scores, and AUC ROC (Area under the Curve of ROC). However, the training and testing data are unbalanced, thus the AUC ROC scores are more informative. The hyper-parameters are shown below

5.2.3 Result

The results are shown in 2 and 3. Our approach achieves the AUC score of 0.6824, which means the model learned the information related to sarcasm in the word vector sequences. We observe that the Post-Fusion strategy achieves the best performance on both 50-dimension and 100-dimension word vectors, which indicates using two independent self-attention-based networks will help the network capture different high-level information from comments and corresponding parent comments. We also observe that there is few difference between using 50-dimension word vectors and 100-dimension word vectors. One possible reason is that we designed a shallow network with a very limited number of parameters, so expanding the dimension of the data does not enable the network to extract more meaningful information from text.

5.3 Approach for Non-context Based Sarcasm Detection

The problem of non-context-based classification is given a text *t*, we classify whether it is sarcastic or non-sarcastic. Because less information is provided for the classification model, we need to better encode the textual information from the input text. Encoder-based language models can provide good textual information as they use bi-directional attention. Thus different from the attention-based model described above, we propose an combination of encoder model and classification model to classify sarcasm based on the encoded tokens of *t*.

5.3.1 Methods

Encoder: We select two encoder models DistilBERT and TwHin-BERT for encoding *t*. DistilBERT is small and efficient BERT that provides good baselines of our proposed method; TwHin-BERT is trained on millions of tweets so that it can provide better textual representation of iSarcasmEval data. Both encoder models produce an aggregated representation of input *t* with the [CLS] token. Then, this token is provided to a classification model that outputs a binary label deciding whether *t* is sarcastic or not.

Classification: We select two classification models, logistic regression and multilayer perceptron, for sarcasm detection based on the [CLS] token. Logistic regression as well as a multilayer perceptron as classification models are explicitly tested and the results are presented in Table 4. We construct the MLP with two linear layers of size (768, 64) and (64, 1) with ReLU and sigmoid functions. It is trained with 5 epochs with a learning rate of 0.00001.

5.3.2 Experiments

Our implementation rely on *distilbert-base-uncased* and *Twitter/TwHIN-BERT-base* provided by HuggingFace for encoder models. The MLP is implemented by ourselves and runned on Google Colab. It takes around 15-20 minutes to train this model.

Table 2: Result of 50-dimension word vector

	Accuracy	Precision	Recall	F1 Score	AUC ROC
Pre-Fusion	0.6445	**0.6383**	0.4813	0.5104	0.6644
Cross-Attention	0.6387	0.6308	0.4492	0.4822	0.6451
Post-Fusion	**0.6541**	0.6267	**0.5774**	**0.5491**	**0.6824**

Table 3: Result of 100-dimension word vector

	Accuracy	Precision	Recall	F1 Score	AUC ROC
Pre-Fusion	**0.6499**	**0.6462**	0.5024	0.5165	0.6637
Cross-Attention	0.6478	0.6303	0.5115	0.5127	0.6631
Post-Fusion	0.6466	0.6020	**0.6054**	**0.5566**	**0.6741**

Model	F1 Score
Sentiment Analysis	0.155
DistilBERT + LR	0.203
DistilBERT + MLP	0.250
TwHin-BERT + LR	0.178
TwHin-BERT + MLP	**0.283**

Table 4: F1 scores of the baseline model and our are tested on the test set of iSarcasmEval. LR represents logistic regression and MLP represents multilayer perceptron.

5.3.3 Results

Comparing with the sentiment analysis baseline using fine-tuned DistilBERT, our approach improves the F1 score from 0.155 to the best score of 0.283 as shown in Tabe 4. The experiment shows the encoder models provide meaningful textual encoding that improves the classification performance. Another observation is MLP is better as sarcasm detection than the linear models such as logistic regression. Meanwhile, TwHin performs better than DistilBERT when classifying with MLP. That means TwHin can provide encoding with more textual information for classification. If we have more computational resource, we can switch to larger language models for better classification performance.

6 Error analysis

6.1 Error Analysis on Context Based Approach

For context based detection method, we have baselines that using sentiment analysis via bert-base-uncased on the comments and its parent comments. As the previous results shown, the accuracy are around 50%, which are equivalent to random guessing. For example, we have comments "Great idea, Mom. Just what I needed.", and its parent comments "You always come up with the best ideas.". The baseline incorrectly classify this as non-sarcastic because the sentiment analysis on both the comment and parent comment is likely to be positive. In our developed approach, we build networks with self-attention modules and different fusion strategies on comments and its parent comments. Our model correctly classifies the previous example as sarcastic because it can correctly attend "great idea" to the parent comment and classify its sarcastic tone.

However, the problem of our approach is that due to limitation of computation resource, we only use 50 and 100-dimension word vectors and limited comments' length in network training, which might not sufficient to reach the ideal result. In addition, the missing special character in our vocabulary may also lead to lose of information. For example, in the comment "Nonono ... they're " "jealous" is ." ", "Nonono" is not regarded as a word. Also the word "jealous" is in quotation marks and this situation will also be ignored.

6.2 Error Analysis on Non-context Based Approach

For non-context based detection method, we use the sentiment analysis baseline via fine-tuned DistillBERT. This method has limitation for sarcasm detection, since analysis on tone of sentiment may not enough to identify the linguistic phenomena like sarcasm and irony. For example, our baseline model identifies "Oh, great. Another Monday. Just what I needed." as positive sentiment due to the presence of words like "great" and "needed,"

while missing the sarcastic tone implying the opposite sentiment of actual excitement for Monday. In our developed approach of combining TwHin-BERT encoder and MLP classification, it correctly identifies the sarcasm features from above example, which the baseline model fail to predict.

However, the limitation of this approach is that it has limited contextual understanding and only rely on comment itself. For an instance, "Biden is a great President like none other we have had," our model fail to identify it as sarcastic because it doesn't have the context of this tweet so it fails. Thus, we can observe that having contextual information is crucial in sarcasm detection and having larger language models might alleviate this issue.

7 Conclusion

Our methods outperformed the baseline of sentiment analysis, enhancing the F1 score from 0.155 to 0.283, suggesting that the use of encoder models significantly enhanced our text encoding capabilities, leading to a marked improvement in our detection results. MLP demonstrated greater efficiency than linear models like logistic regression for sarcasm detection, and TwHin proved superior to DistilBERT when paired with MLP, implying the potential of larger language models with sufficient computational resources. Furthermore, our model achieved an AUC score of 0.6824, illustrating the effective learning of sarcasm-related information. Recognizing the auxiliary assistance of the context in sarcasm detection, we made preliminary attempts at context fusion, enhancing model's ability to understand sarcasm better, where special symbols and punctuation marks, often serving as cues for sarcasm, were also taken into account. The Post-Fusion strategy consistently delivered optimal performance with both 50-dimension and 100-dimension word vectors, suggesting the value of utilizing two independent self-attention-based networks to capture diverse high-level information from comments and their parent comments.

Continuing our research on sarcasm detection in textual data, there are numerous intriguing avenues to explore. For non-context based methods, we could explore concatenating multiple language models to improve the text encoding ability. For context-based methods, we need better computational resources to avoid limited input text length as well as a more comprehensive word embedding library to avoid informational loss on special characters.

Regarding the diverse contextual environments, the project so far has only focused on English language data. However, sarcasm is a universal language feature. Therefore, it would be worthwhile to extend the scope of this project to include multiple languages and examine cross-lingual sarcasm detection models. All these efforts collectively aim to advance the frontiers of sarcasm detection and render it more comprehensible in varied linguistic and cultural contexts.

8 What you proposed vs. what you accomplished

1. ~~Research on past papers, complete related work~~: DONE.
2. ~~Prepare Baselines based on data~~: DONE.
3. ~~Build models for experiment, implement each selected technique~~: DONE.
4. ~~Analyze the output of the model, do empirical comparison, error analysis, and complete the report~~: DONE.

9 Contributions of group members

- Ruichen Li: design and implement non-context based sarcasm detection and write report
- Shuyang Cui: design and implement context-based sarcasm detection and write report
- Gongxuan Liu: Templates/ 2.3 Related Works/ Baselines for Context-based Sarcasm Detection / Conclusion
- Zhichao Liu: Introduction/ 2.1-2.2 Related works/ data-set visualization, description / Error Analysis

References

Ashwin, R., Reza, Z., and Liu, H. (2015). Sarcasm detection on twitter: A behavioral modeling approach. In *In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, page 97–106.

Bamman, D. and Smith, N. (2015). Contextualized sarcasm detection on twitter. In *proceedings of the international AAAI conference on web and social media*, volume 9, pages 574–577.

Farha, I. A., Oprea, S. V., Wilson, S., and Magdy, W. (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)*, pages 802–814.

Hazarika, D., Poria, S., Gorantla, S., Cambria, E., Zimmermann, R., and Mihalcea, R. (2018). Cascade: Contextual sarcasm detection in online discussion forums. *arXiv preprint arXiv:1805.06413*.

Jiwei, L., Alan, R., and Dan, J. (2015). Learning multifaceted representations of individuals from heterogeneous evidence using neural networks. In *arXiv preprint arXiv:1510.05198*.

JoshiAditya, B. (2017). J., c.: Automatic sarcasm detection. *ACM Comput. Surv*, 50.

Khodak, M., Saunshi, N., and Vodrahalli, K. (2017). A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Kolchinski, Y. A. and Potts, C. (2018). Representing social media users for sarcasm detection. *arXiv preprint arXiv:1808.08470*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N., and Huang, R. (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, Seattle, Washington, USA. Association for Computational Linguistics.

Silvio, A., Byron C, W., Hao, L., Paula, C., and Mario J., S. (2016). Modelling context with user embeddings for sarcasm detection in social media. In *arXiv:1607.00976v2 [cs.CL]*.

Wallace, B. C., Kertz, L., Charniak, E., 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.