MarSan at SemEval-2023 Task 10: Can Adversarial Training with help of a Graph Convolutional Network Detect Explainable Sexism?

Ehsan Tavan[1,*], **Maryam Najafi**[1,*]
[1] NLP Department, Part AI Research Center, Tehran, Iran
{ehsan.tavan, maryam.najafi}@partdp.ai

Abstract

This paper describes SemEval-2022's shared task "Explainable Detection of Online Sexism". The fine-grained classification of sexist content plays a major role in building explainable frameworks for online sexism detection. We hypothesize that by encoding dependency information using Graph Convolutional Networks (GCNs) we may capture more stylistic information about sexist contents. Online sexism has the potential to cause significant harm to women who are the targets of such behavior. It not only creates unwelcoming and inaccessible spaces for women online but also perpetuates social asymmetries and injustices. We believed improving the robustness and generalization ability of neural networks during training will allow models to capture different belief distributions for sexism categories. So we proposed adversarial training with GCNs for explainable detection of online sexism. In the end, our proposed method achieved very competitive results in all subtasks and shows that adversarial training of GCNs is a promising method for the explainable detection of online sexism.

1 Introduction

Online platforms have become a major source of communication, where people express their opinions and views. The online world, however, is not immune to sexism, and increasing social media usage has led to the proliferation of hate speech and offensive language (Alzeer and Amin, 2020). It is critical to automatically identify such content in order to prevent its spread. This study aims to identify and classify sexism in social media texts. The detection of sexism in online conversations requires identifying offensive language and gender-biased text.

Sexism involves the use of abusive or negative language directed towards women based on their gender, or in conjunction with other identity attributes such as race, religion, or gender identity. This form of online hate speech is a significant obstacle to gender equality. Women are disproportionately impacted by hateful and offensive content on social media platforms, including misogynistic and sexist posts, which can lead to real-world violence against them (Deal et al., 2020). The automatic detection and categorization of these types of tweets and posts can be a valuable tool for social scientists and policymakers in their efforts to combat sexism through research.

Detecting and categorizing instances of sexism in social media posts can be a complicated and arduous process, particularly when the content involves sarcasm and various types of sexist language. Recent research has identified multiple factors that make identifying sexist content on social media particularly challenging (Younus and Qureshi, 2022). These reasons include the use of subtle forms of expression with microaggressions woven into the language, the presence of diverse contexts such as ongoing popular events and conversations, and the potential impact of the platform or language used on determining whether a statement is considered sexist or not.

In recent years, graph convolutional networks (GCN) (Kipf and Welling, 2016) and language models have shown significant promise in the field of natural language processing. It is possible to develop robust frameworks for detecting and categorizing online sexism posts using GCN's ability to model complex relationships in graph data. This is coupled with the language model's capability to understand the context and semantics of text.

Adversarial training has been shown to be effective in improving the robustness of neural models by incorporating a small perturbation into the input data during training (Vidgen et al., 2021) (Kirk et al., 2022). This makes it more difficult for the model to make incorrect predictions. Therefore,

[*] Equal contribution. Listing order is random.

Proceedings of the The 17th International Workshop on Semantic Evaluation (SemEval-2023), pages 1011–1020

we propose a novel framework for detecting online sexism posts that combines graph-based models, language models, and adversarial training. Our proposed framework leverages the strengths of each component to detect and categorize subtle forms of sexism in social media content. Additionally, it can enhance the robustness of the model against adversarial attacks.

Our code is available on GitHub for researchers. [1] This paper is organized as follows: A description of the task and its subtasks can be found in Section 2. Section 3 reviews related research. The theoretical background of the neural model is presented in section 4. Section 5 presents implementation details, while 6 describes experiments and results. The conclusions of the paper can be found in section 7.

2 Task Description

Explainable Detection of Online Sexism (EDOS) (Kirk et al., 2023) is a SemEval 2023 task that aims to facilitate the development and assessment of models capable of detecting online sexism in social media posts. Human experts in English have manually annotated social media posts for this task. The following three subtasks are involved in EDOS:

Binary Sexism Detection (Task A): The first subtask of the EDOS task is the detection of binary sexism. In this task, systems need to predict whether a given social media post is sexist.

Category of Sexism (Task B): For posts that are classified as sexist in Task A, the systems have to predict one of four categories: (1) threats, (2) derogation, (3) animosity, or (4) prejudiced discussions.

Fine-grained Vector of Sexism (Task C): Systems have to predict one of 11 fine-grained vectors if posts are classified as sexist in Task A.

3 Related Works

For the IberLEF 2022 dataset, de Paula and da Silva (2022) implemented multilingual versions of BERT, RoBERTa, and the monolingual versions of Electra, and GPT2 and achieved the highest results using back-translation to English with BERT and RoBERTa language models.

With ByT5 as a token-free language model, Younus and Qureshi (2022) proposes to combine byte-level modeling with attention-based neural networks through tabular modeling via TabNet, which can take into account platform and language aspects to detect sexism effectively.

By incorporating recurrent components, Abburi et al. (2021) proposes a neural model for the detection and classification of sexism, combining representations obtained using RoBERTa and linguistic features such as Empath and Hurtlex. In addition to that, Abburi et al. (2021) incorporates external knowledge-specific features into the learning process using emoticons and hashtags.

Jiang et al. (2022) proposes a large Chinese lexicon of abusive and gender-related terms, called SexHateLex, and the first Chinese sexism dataset called Sina Weibo Sexism Review (SWSR). These authors analyse the characteristics of the dataset and how Chinese sexism manifests. SWSR provides labels at different levels of granularity, including sexism or non-sexism, sexism category, and target type. Additionally, the paper uses state-of-the-art machine learning models for the three sexism classification tasks. The research aims to broaden the scope of research in sexism detection and contribute to combating online sexism. A combination of Graph Convolutional Neural Networks (GCN) with different edge creation strategies is applied in Wilkens and Ognibene (2021) as well as an ensemble approach for combining graph embeddings from different GCN models. To capture the linguistic and structural characteristics of sexist posts, the model uses graph-based representations of social media content.

A study from Kalra and Zubiaga (2021) explores the use of deep neural networks, including Long-Short-Term Memory (LSTMs) and Convolutional Neural Networks (CNNs), to identify sexism in a social media text. Using transfer learning through Bidirectional Encoder Representations from Transformers (BERT) and DistilBERT models, as well as data augmentation, the models classify sexism in a dataset of Sexism Identification in Social Networks (EXIST) tasks. With data augmentation, BERT and a multi-filter CNN model produce comparable results.

The authors in Sharifirad et al. (2019) use Convolutional Neural Networks (CNNs) to develop classifiers to detect and annotate different types of sexism. Using CNN models, the study detects semantic categories of n-grams and clusters them to improve the classification task by improving the understanding of sexism. As part of the study, the

[1] https://github.com/MarSanTeam/Explainable_Detection_of_Online_Sexism

authors examine more precise categories of sexism in social media and examine CNN filters, which can help identify the most important n-grams.

To perform document classification, Yao et al. (2019) employ Text Graph Convolutional Networks (Text GCN). A tensor graph convolutional network based on semantic, syntactic, and sequential graphs is introduced in Liu et al. (2020) for document classification. Furthermore, Lin et al. (2021) combine BERT and GCN to improve the performance of text classification.

Gender differences can further compound the challenges of detecting sarcasm and irony. Studies have shown that men and women use sarcasm differently, with men often using it as a form of aggression or dominance, while women may use it to express their frustration or as a coping mechanism in situations where they lack power. Moreover, societal norms and gender stereotypes can influence the way that sarcasm is perceived and interpreted, with women often being judged more harshly for their use of sarcasm than men. These factors must be considered when attempting to detect sarcasm and irony in written or spoken language (van de Kerkhof, 2018; Najafi and Tavan, 2022; Rahgouy et al., 2022; Tavan et al., 2022; Giglou et al., 2022).

4 System Overview

This section presents our proposed model which consists of three modules: the encoder module, the GCN module, and the classifier module. The input text is encoded into a feature representation by the encoder module, and relevant features are extracted by the GCN module based on the graph structure. As a final step, the Classifier module uses these extracted features to classify the input text. After each epoch, the encoder module updates all node embeddings. Figure 1 shows the overview of our proposed model.

4.1 Encoder Module

In the Encoder module, BERTweet (Nguyen et al., 2020) was implemented in order to convert tokens to their corresponding embedding vector. By incorporating the transformer's encoder and a BiLSTM layer at the top of the BERTweet model, we are able to extract more informative features and improve the performance of the model. In this module, the transformer uses a multi-head attention mechanism to identify relationships between tokens, which is particularly useful for detecting sexist contextual

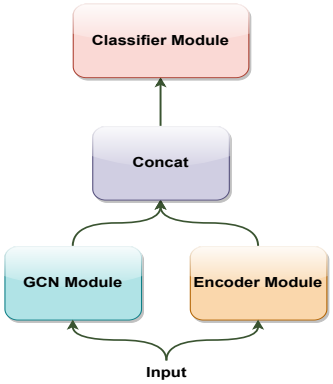

Figure 1: An overview of the proposed model.

features. As a result of our architecture, informative features can be extracted from the input text in an effective manner. Furthermore, it illustrates the importance of leveraging advanced neural network architectures in order to enhance the performance of NLP models.

4.2 Graphs Constructor

We construct multiple heterogeneous text graphs ($G_1, G_2,$ and G_3) with shared nodes to represent text documents that possess various linguistic properties, following the TensorGCN (Liu et al., 2020) approach. These graphs are denoted as $G(V, E, A)$, where $V(|V| = n)$ represents the set of nodes (vertices), E represents the set of edges, and A represents the graph adjacency matrix (edge weights).

These graphs are built using two types of word and document nodes, which incorporate semantic, syntactic, and sequential features. These graphs also have two types of edges: word-document edges and word-word edges. The word-document edges in all three graphs are created based on the frequency of words within documents, and their edge weights are determined by the term frequency-inverse document frequency (TF-IDF) method. The word-word edges in G_1, G_2, and G_3, are constructed based on syntactic dependency, semantic similarity, and local sequential context, respectively.

To initialize the matrix $X \in \mathbb{R}^{n \times m}$, which contains all n nodes along with their embeddings, we utilize BERTweet. For each node, document, or word, the output of the [CLS] token is used as a node embedding.

Semantic-based Graph: A semantic-based graph's adjacency matrix is constructed using BERTweet. The process involves extracting word semantic embeddings from the input text, comput-

ing the cosine similarity between words, and checking if the score exceeds a predetermined threshold. Whenever the threshold is exceeded, it indicates a semantic relationship between the words in the input text. This computation is carried out across the entire corpus, and all semantically related word pairs are counted. The calculation process is illustrated in equation 1.

$$d_{semantic}(w_i, w_j) = \frac{N_{semantic}(w_i, w_j)}{N_{total}(w_i, w_j)} \quad (1)$$

The meaning of the variables in equation 1 is as follows: $d_{semantic}(w_i, w_j)$ represents the edge weight of the semantic relationship between words w_i and w_j. The variables $N_{semantic}(w_i, w_j)$ and $N_{total}(w_i, w_j)$ correspond to the number of times w_i and w_j have a semantic relationship in the corpus and the total number of times they appear together in one sample, respectively.

Syntactic-based Graph: Syntactic adjacency matrices are generated by identifying the syntactic relationships among words in a corpus. We use a spacy parser as the basis of our syntactic parser. The parser output can be visualized as an undirected graph to facilitate analysis. Over the entire corpus, we count the number of times each pair of words has syntactic dependency. This method is used to compute the edge weight of each word pair. The equation below demonstrates how this calculation is performed.

$$d_{syntactic}(w_i, w_j) = \frac{N_{syntactic}(w_i, w_j)}{N_{total}(w_i, w_j)} \quad (2)$$

where $d_{syntactic}(w_i, w_j)$ represents the edge weight for the syntactic relationship between words w_i and w_j in the graph. $N_{syntactic}(w_i, w_j)$ is the number of times w_i and w_j have a syntactic dependency in the corpus, and $N_{total}(w_i, w_j)$ is the total number of times w_i and w_j appear together in the one sample.

Sequential-based Graph: The method of learning text representations commonly involves analyzing sequential context to identify the local co-occurrence of words. This approach is widely used for constructing an adjacency matrix. We employ a sliding window strategy and Point-wise Mutual Information (PMI) to capture sequence context information.

A PMI score was developed because the performance was better than calculating word frequency. This was based on the intuition that the best way to measure the association between two words is to calculate how often they co-occur in the corpus compared to how often they could have appeared by chance (Jurafsky and Martin, 2021).

As a result, we are able to calculate the weight of the edge connecting each pair of words. These words are treated as nodes in a graph based on their sequential relationship. The equation below demonstrates how the calculation is carried out:

$$d_{sequential}(w_i, w_j) =$$

$$\text{PMI}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}$$

$$p(w_i, w_j) = \frac{N(w_i, w_j)}{N(windows)} \quad (3)$$

$$p(w_k) = \frac{N(w_k)}{N(windows)}$$

In equation 3, $N(w_k)$ represents the number of sliding windows that contain the word w_k. Similarly, $N(w_i, w_j)$ refers to the number of sliding windows in the same corpus that contains both the words i and j. Finally, $N(windows)$ represents the total count of sliding windows in the corpus.

4.3 GCN Module

GCNs are a type of neural network specifically designed for processing graph data, where nodes are connected by edges. GCNs aim to generate embedding vectors for each individual node in the graph, using information from its immediate neighborhood. GCN achieves this by applying convolutional layers directly to the graph structure. However, since a single convolutional layer can only capture information from a node's immediate neighborhood, stacking multiple GCN layers enables the integration of information from larger neighborhoods, resulting in richer node embeddings.

The GCN formula, represented by equation 4, is a mathematical function that takes in a graph representation as input and outputs the corresponding node embeddings. The formula involves several key components, including the input node features represented by matrix $H^{(l)}$, the graph adjacency matrix represented by matrix $\tilde{A}$, and the weight matrix for layer l represented by matrix $W^{(l)}$. Additionally, the formula utilizes matrix $\tilde{D}$, which is the degree matrix of the augmented graph, and σ is a nonlinear activation function.

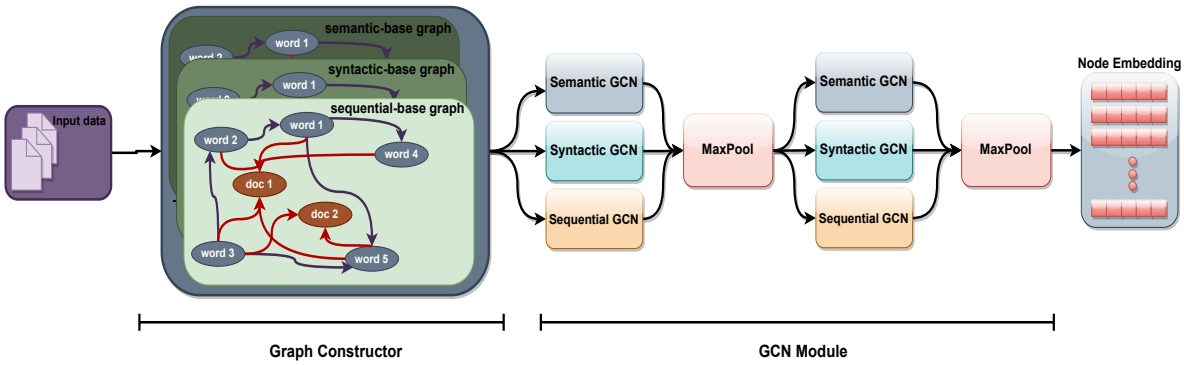

Figure 2: The proposed GCN module

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \quad (4)$$

The GCN module is designed to extract local graph features by utilizing the introduced graphs. The GCN module is shown in Figure 2. Initially, each graph is processed by a single-layer GCN, which generates embedding vectors for each node based on its immediate neighborhood. In order to create a new node embedding vector, the maximum value of the resulting node embeddings is accumulated. To obtain richer node embeddings, we repeat the procedure with another GCN layer to obtain information from larger neighborhoods. To merge the information from the three graphs, we use the maximum value of the three node embedding vectors as the final node embedding. This approach ensures that the final node embedding contains the most informative features from all three graphs.

By leveraging the power of GCNs and combining information from multiple graphs, our method is capable of effectively capturing and utilizing the structural and relational information present in complex graphs.

4.4 Classifier Module

Finally, to determine the probabilities of the labels, the softmax classifier is used. This module is a simple softmax classifier that is used to predict a label $\hat{y}$ from a set of discrete classes. The softmax classifier takes X as input:

$$X = X_{gnn} \oplus X_{encoder}$$

$$P(y \mid X) = \boldsymbol{softmax}(WX + b) \quad (5)$$

$$\widehat{y} = argmax(P(y \mid X))$$

Here, the variable X represents the concatenation of GNN module features and Encoder module features. The weight matrix W has dimensions of R^{h*n}, the bias vector b has dimensions of R^n, and n represents the number of classes.

4.5 Adversarial Training

Adversarial training is a technique used to improve the robustness and generalization ability of neural networks by training them on adversarial examples. The most effective way to do this is by adding small perturbations to the input during the training process. Specifically, we employ the Projected Gradient Descent (PGD) method (Madry et al., 2017), which adds a small perturbation to the original input example in the direction of the gradient of the parameters that maximize the loss. By repeatedly applying this technique, the model learns to handle adversarial examples more effectively, resulting in enhanced robustness and generalization. Our proposed model incorporates perturbations in both the embedding layer of BERTweet and the node embedding of each graph.

Adversarial training has proven to be an effective defense against adversarial attacks and can significantly improve the performance of neural networks on challenging tasks. The resulting models are capable of handling a variety of inputs and can be used in a wide range of applications. The following is a formulation of adversarial training (Madry et al., 2017):

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}\left[\max_{\delta \in S} L(\theta, x+\delta, y)\right] \quad (6)$$

In equation 6, x represents the original input example, y represents the true label for the input example, and L is a loss function that measures the discrepancy between the model's prediction and the true label. The perturbation δ is added to

the input example x to create an adversarial example, and the set S specifies the allowable range of perturbations that can be added to the input. By minimizing the objective function, we can find the optimal value of θ that makes the model more robust to adversarial examples, and thus improves its overall performance.

5 Experimental Setup

This section is structured into two distinct parts. Firstly, we describe our data pre-processing approach in detail. Secondly, we discuss the implementation details.

Pre-processing: The dataset contained URLs and user mentions, which could confuse the model in labeling the data. Therefore, these items were removed during preprocessing. Additionally, the text was converted to lowercase and all numbers were removed. These steps help to reduce complexity, improve uniformity, and increase accuracy, especially in tasks like sexism detection.

Furthermore, the graph constructor utilizes a number of preprocessing techniques, which are only used in the graph constructor, not in the language model. The first step is to remove stop words that are commonly used but do not convey much meaning. Subsequently, all words within the text are lemmatized. The last step involves removing words that appear in the text less than three times.

Implementation Details: PyTorch is employed to implement the model, which was trained on high-performance NVIDIA V100 GPUs. To fine-tune each subtask, hyper-parameters were adjusted using the development set. The model is trained using the back-propagation algorithm with the Adam optimizer, which had a learning rate of 1e-5. An early stopping method is employed by monitoring the validation loss in min mode with patience of five. A batch size of 32 is used for each subtask and cross-entropy loss with class-weights (sklearn is used to calculate class-weights [2]) is used in cases of data imbalance. Additionally, a maximum length of 70 tokens is used in all experiments. (Only 41 samples have more than 70 tokens.)

The transformer consists of eight attention heads, while the Position-wise feed-forward layer has a hidden size of 2048. Additionally, one layer of Bi-LSTM with 512 units is used. The GCN layers have output channels set to 1024, followed by a ReLU function, and a dropout rate of 0.2 is used. The window size in sequential-base graphs is set to ten.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html

Model	F1-score(%)	
	Dev	Test
Subtask A		
BERTweet	**79.04**	**78.26**
BERT-Large	77.05	76.43
XLM-RoBERTa-Large	76.70	75.75
T5-Large	77.46	75.66
Subtask B		
BERTweet	**56.86**	**56.82**
Bert-Large	46.35	47.24
XLM-RoBERTa-Large	55.25	50.34
T5-Large	39.33	37.79
Subtask C		
BERTweet	**42.23**	**36.15**
Bert-Large	37.78	33.31
XLM-RoBERTa-Large	31.61	33.02
T5-Large	10.03	10.75

Table 1: Macro f1-score for three subtasks using different language models.

6 Experiments and Analysis

This section reviews various baselines and compares them with the proposed model. The first challenge is to identify the most appropriate token contexts based on language models. Consequently, we first analyze the performance of different language models for each task, including BERTweet, BERT-Large Devlin et al. (2018), XLM-RoBERTa Conneau et al. (2019), and T5-Large Raffel et al. (2020). According to our analysis, BERTweet outperforms all other models with a higher macro F1-Score. The results of the different language models are presented in table 1.

Table 2 displays the macro F1-score results for three subtasks, A, B, and C, obtain by adding different architecture. The highest macro F1-score for each subtask is shown in bold text, and the best-performing model for each subtask achieves an macro F1-score of 85.78%, 70.82%, and 48.94% for subtasks A, B, and C, respectively. These results suggest that the encoder module with GCN module and adversarial training performs the best across all three subtasks.

Analysis of Encoders and GCN modules: According to table 2, BERTweet's performance on

Model	F1-score(%) Dev	Test
Subtask A		
BERTweet	79.04	78.26
BERTweet + 1 layer GCN	84.75	84.87
BERTweet + GCN module	85.16	85.09
Encoder + 1 layer GCN	85.05	84.48
Encoder + GCN module	85.66	85.07
Encoder + GCN module + Adversarial training (ours)	**85.78**	**85.18**
Subtask B		
BERTweet	56.86	56.82
BERTweet + 1 layer GCN	67.41	61.99
BERTweet + GCN module	67.08	62.51
Encoder + 1 layer GCN	70.45	64.12
Encoder + GCN module	69.95	63.92
Encoder + GCN module + Adversarial training (ours)	**70.82**	**66.09**
Subtask C		
BERTweet	42.23	36.15
BERTweet + 1 layer GCN	50.32	46.4
BERTweet + GCN module	48.29	46.77
Encoder + 1 layer GCN	48.38	47.98
Encoder + GCN module	48.54	48.16
Encoder + GCN module + Adversarial training (ours)	**48.94**	**48.22**

Table 2: Comparison of F1-scores for various architecture on Subtask A, B, and C. We use the term "Encoder" to refer to our proposed encoder module, while "BERTweet" represents the use of BERTweet alone as the encoder module. Similarly, "GCN module" refers to our proposed GCN module, whereas "1 layer GCN" represents the use of only one layer of GCN in our proposed model.

subtasks A, B, and C is improved by the addition of one layer of GCN, however, it is still lower than the proposed Encoder module with GCN module. In all three subtasks, the proposed Encoder module with one layer of GCN performs slightly better than BERTweet with one layer of GCN. Through the addition of the GCN module, the Encoder module has further improved its performance on subtasks A, B, and C, making it the best-performing model on these subtasks.

Analysis of adversarial training: Table 2 clearly demonstrates the effectiveness of adversarial training in enhancing the model's performance across all three subtasks. Specifically, subtask A achieved a 0.09 percentage point increase in F1-score, rising from 85.08% to 85.17%, while subtask B experienced a notable improvement with a 2.17 percentage point increase, from 63.92% to 66.09%. In subtask C, the macro F1-score slightly improved from 48.16% to 48.22%. These results demonstrate the efficacy of incorporating adversarial training to enhance the model's overall performance.

Constructor Setup	F1-score(%) Dev	Test
Subtask A		
submitted	**85.78**	**85.18**
no-lemma	85.08	84.12
no-remove-stopwords	84.82	84.04
no-lemma + no-remove-stopwords	84.41	83.62
Subtask B		
submitted	**79.04**	**78.26**
no-lemma	70.19	65.51
no-remove-stopwords	69.66	65.27
no-lemma + no-remove-stopwords	67.54	64.46
Subtask C		
submitted	**48.94**	**48.22**
no-lemma	47.64	46.45
no-remove-stopwords	47.28	45.96
no-lemma + no-remove-stopwords	46.93	45.94

Table 3: Macro F1-score for different graph constructor setups. The term "submitted" refers to the model we have submitted for the competition that lemmatizes and removes stopwords as part of the preprocessing step. "No-lemma" refers to a preprocessing approach that removes only stopwords from the text before constructing the graph. In the term "no-remove-stopwords", lemmatization is the only preprocessing step. The term "no-lemma-no-remove-stopwords" is applied to a preprocessing approach that does not use any lemmatization or stopword removal.

Analysis of graph constructor: As shown in table 3, the model we submitted to the leaderboard (our model) outperforms the other models in all three subtasks (A, B, and C). In particular, for subtask A, our model achieved the highest macro F1-score on both the development and test sets, with scores of 85.78% and 85.18%, respectively. In subtasks B and C, the "no-lemma" and "no-remove-stopwords" models show a significant decrease in performance compared to our model, indicating that both lemmatization and stopword removal are critical preprocessing steps for constructing the graph. The "no-lemma-no-remove-stopwords"

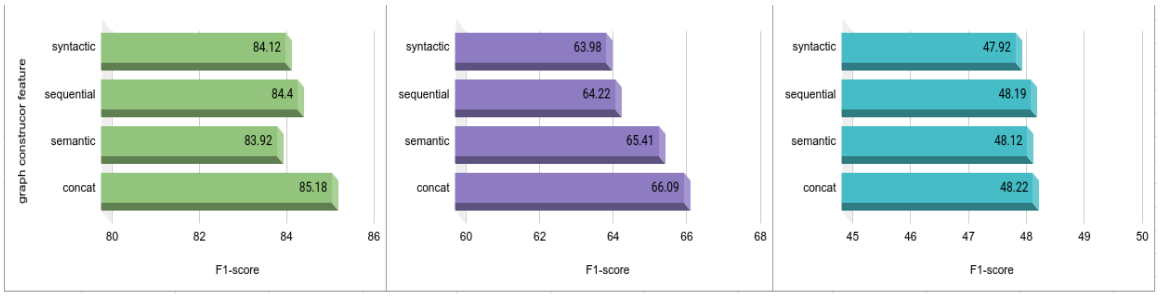

Figure 3: Comparison of the F1 scores of single syntactic, semantic, and sequential graphs with concatenated graphs. The first chart corresponds to Task A, the second to Task B, and the third to Task C.

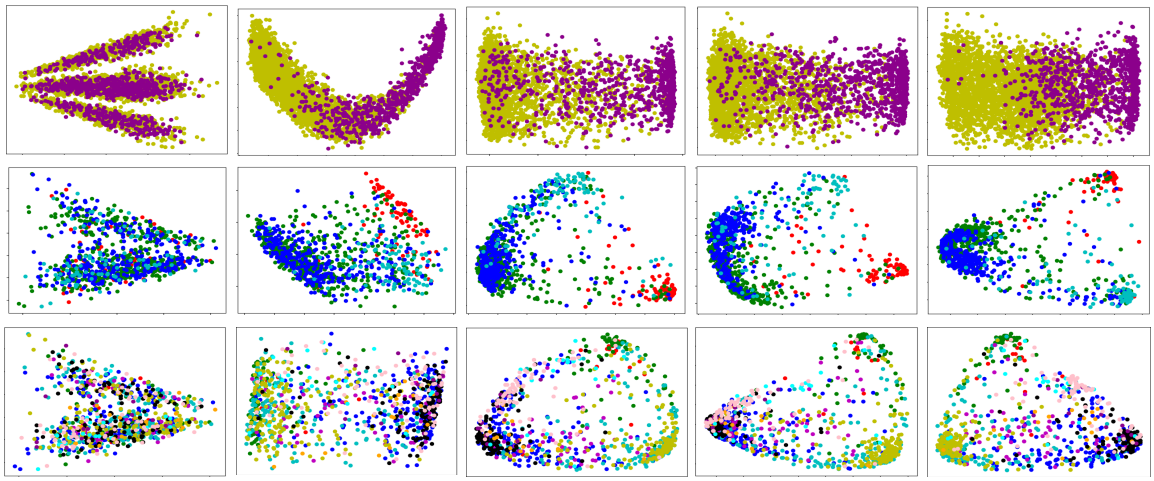

Figure 4: The test set document embeddings are visualized using t-SNE. The first row corresponds to Task A, the second row corresponds to Task B, and the third row corresponds to Task C.

model shows the lowest macro F1-score across all subtasks.

Also, it is important to note that the performance difference between the "no-lemma" and "no-remove-stopwords" models is relatively small compared to our model, suggesting that both preprocessing steps may have a similar impact on the model's overall performance.

Figure 3 compares the F1 scores obtained by single syntactic, semantic, and sequential graphs with concatenated graphs for Sexism detection. The findings of the figure suggest that using concatenated graphs yields better results compared to using a single graph in sexism detection.

6.1 Document Visualization

In order to visually represent the document embeddings learned by our proposed model, we employ t-SNE Van der Maaten and Hinton (2008), a powerful tool for high-dimensional data visualization. Figure 4 shows the results of our document visualization, demonstrating the 2048-dimensional embeddings of test documents learned by our model across all tasks from the initial step to epoch 4. This visualization illustrates the performance of the proposed model in generating discriminative embeddings of documents. As shown in figure 4, it is apparent that the differentiation between classes improves across epochs.

7 Conclusion

Our study presents a novel approach for identifying various types of sexist posts by combining graph and language models. To capture contextual information from the input text, we utilize the encoder module, which is composed of a BERTweet model, a transformer, and a bidirectional LSTM layer. Additionally, we utilize syntactic, semantic, and sequence-based graphs to extract information from the graph structure. By integrating context and graph-based features, our proposed model outperforms existing models in identifying sexist posts and their various types.

In order to evaluate the performance of the proposed model, several experiments are conducted. In the experiments, it is demonstrated that the com-

bination of contextual features extracted from the encoder module and graph-based features provides a reasonable range of results for the detection of online sexism that is explainable due to achieving better performance on fine-grained detection.

References

Harika Abburi, Shradha Sehgal, Himanshu Maheshwari, and Vasudeva Varma. 2021. Knowledge-based neural framework for sexism detection and classification. In *IberLEF@ SEPLN*, pages 402–414.

Gergana Alzeer and Omnia Amin. 2020. Beyond a black sea of sheilas and abayas: do emirati women students have a space of their own? *Journal of International Women's Studies*, 21(2):36–52.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Angel Felipe Magnossão de Paula and Roberto Fray da Silva. 2022. Detection and classification of sexism on social media using multiple languages, transformers, and ensemble models.

Bonnie-Elene Deal, Lourdes Martinez, Brian Spitzberg, and Ming-Hsiang Tsou. 2020. "i definitely did not report it when i was raped . . . webelievechristine metoo": A content analysis of disclosures of sexual assault on twitter. *Social Media + Society*, 6:205630512097461.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Hamed Babaei Giglou, Mostafa Rahgouy, Ali Rahmati, Taher Rahgooy, and Cheryl D Seals. 2022. Profiling irony and stereotype spreaders with encoding dependency information using graph convolutional network.

Aiqi Jiang, Xiaohan Yang, Yang Liu, and Arkaitz Zubiaga. 2022. Swsr: A chinese dataset and lexicon for online sexism detection. *Online Social Networks and Media*, 27:100182.

Daniel Jurafsky and James H Martin. 2021. Speech and language processing,(draft) edition, chapter 4.

Amikul Kalra and Arkaitz Zubiaga. 2021. Sexism identification in tweets and gabs using deep neural networks. *arXiv preprint arXiv:2111.03612*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Hannah Kirk, Bertie Vidgen, Paul Rottger, Tristan Thrush, and Scott Hale. 2022. Hatemoji: A test suite and adversarially-generated dataset for benchmarking and detecting emoji-based hate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1352–1368, Seattle, United States. Association for Computational Linguistics.

Hannah Rose Kirk, Wenjie Yin, Bertie Vidgen, and Paul Röttger. 2023. SemEval-2023 Task 10: Explainable Detection of Online Sexism. In *Proceedings of the 17th International Workshop on Semantic Evaluation*, Toronto, Canada. Association for Computational Linguistics.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bertgcn: Transductive text classification by combining gcn and bert. *arXiv preprint arXiv:2105.05727*.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8409–8416.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Maryam Najafi and Ehsan Tavan. 2022. MarSan at SemEval-2022 task 6: iSarcasm detection via t5 and sequence learners. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 978–986, Seattle, United States. Association for Computational Linguistics.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets. *arXiv preprint arXiv:2005.10200*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Mostafa Rahgouy, Hamed Babaei Giglou, Taher Rahgooy, and Cheryl Seals. 2022. NULL at SemEval-2022 task 6: Intended sarcasm detection using stylistically fused contextualized representation and deep learning. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 862–870, Seattle, United States. Association for Computational Linguistics.

Sima Sharifirad, Alon Jacovi, Israel Bar Ilan Univesity, and Stan Matwin. 2019. Learning and understanding different categories of sexism using convolutional neural network's filters. In *WNLP@ ACL*, pages 21–23.

Ehsan Tavan, Maryam Najafi, and Reza Moradi. 2022. Identifying ironic content spreaders on twitter using psychometrics, contextual and ironic features with gradient boosting classifier.

Ruben Leonard van de Kerkhof. 2018. *Gender information absolutely enhances sarcasm detection (/s)*. Ph.D. thesis, Tilburg University.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1667–1682, Online. Association for Computational Linguistics.

Rodrigo Souza Wilkens and Dimitri Ognibene. 2021. Mb-courage@ exist: Gcn classification for sexism identification in social networks. In *IberLEF@ SEPLN*, pages 420–430.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.

Arjumand Younus and Muhammad Atif Qureshi. 2022. A framework for sexism detection on social media via byt5 and tabnet.