Text Classification with language models and graph structures

Stanford CS224N {Custom} Project

Fang Shu
Management Science & Engineering
fangshu@stanford.edu

Jian Xu
Civil & Environmental Engineering
xj1998@stanford.edu

Abstract

In this paper, we propose a novel approach to text classification by incorporating graph structure and advanced language model embeddings. We first explore the Mafia player classification task, but due to the low quality of the dataset and marginal model performance improvement, we use the Amazon product dataset to validate our approaches. Our study demonstrates the effectiveness of data augmentation on imbalanced datasets and the superior performance of state-of-the-art NLP models, such as BERT. By integrating textual and structural information, our best-performing hybrid model achieved a 98% accuracy on the Amazon dataset. Furthermore, we compared traditional BOW or GloVe embeddings with advanced BERT embeddings, revealing the potential of using embeddings with richer semantic information to enhance GNN performance. Our findings suggest that combining graph structure, advanced embeddings, and NLP models can lead to more accurate text classification, paving the way for future research in leveraging relational data and advanced embeddings for NLP and GNN tasks.

1 Key Information to include

- Mentor: Rishi Desai; External Collaborators: N/A; Sharing Project: N/A

2 Introduction

Text classification, a fundamental task in natural language processing (NLP), has been widely studied and applied in various domains. Traditional text classification approaches rely on the textual information of documents to predict their labels. However, in many real-world scenarios, the documents are connected or related, forming a graph structure. Ignoring these relationships may lead to suboptimal performance when classifying texts. A compelling example is the Mafia game, which features a small social network of player interactions that can provide valuable insights into the in-game roles of players. In this paper, we aim to combine textual and graph-structured information to push the boundaries of conventional NLP text classification methods.

The Mafia game classification task is inherently challenging due to its deceptive nature and complex logic flow. Detecting deception is an essential skill with applications in intelligence agencies, law enforcement, and online marketplaces. Traditional machine learning (ML) models have shown limitations in handling deceptive text, as they struggle to capture implicit information the author intends to conceal. To tackle this problem, we turn to state-of-the-art language models, which have demonstrated success in various NLP tasks. Moreover, we seek to incorporate textual information and graph structure by combining NLP techniques and graph neural networks (GNNs), a class of ML models that have gained popularity in recent years for their ability to handle graph-structured data. We hypothesize that this hybrid method can capture the complex interactions between texts and yield more accurate classification results.

To validate our approach, we first tested our models on the Mafia game dataset. However, the performance improvement after incorporating textual and graph-structured information was marginal, which we attributed to the low quality of the dataset. Consequently, we selected the Amazon product dataset to further investigate our ideas. Both datasets presented the challenge

of class imbalance, and we addressed this issue through data augmentation techniques, which proved effective on the Amazon dataset but not on the Mafia dataset.

We trained various NLP models, including LSTM, BERT, and BERT+CLS, and observed that the best-performing model achieved an accuracy of about 90% on the balanced Amazon dataset. We then combined textual and structural information, resulting in our best hybrid model reaching a 98% accuracy. This result highlights the potential of integrating graph structure and NLP techniques for text classification tasks.

Another aspect of our study involved comparing the performance of traditional bag-of-words (BOW) or GloVe embeddings with advanced BERT embeddings to determine whether using more semantically rich embeddings would improve the performance of GNN models. Our findings suggest that incorporating advanced embeddings like BERT can indeed enhance the performance of GNN models, particularly in more complex tasks where word order matters.

In summary, this paper presents a novel approach to text classification by leveraging both textual and graph-structured information, along with advanced embeddings. Our results demonstrate the effectiveness of data augmentation for imbalanced datasets, the superior performance of state-of-the-art NLP models, and potential of integrating graph structure and advanced embeddings to improve classification accuracy. This innovative approach provides a more comprehensive understanding of the problem and lays groundwork for future research leveraging relational data and advanced embeddings for NLP and GNN tasks.

3 Background and Related Work

3.1 About Mafia

Mafia, also known as Werewolf, is a social game in which players are randomly assigned an identity and divided into two groups. The Mafia group is a minority, and each member knows the identity of the others. The remaining players are the majority and are referred to as the villagers. They are innocent and uninformed of the Mafia's identities. The game alternates between two phases: day and night. At night, the Mafia select one player to eliminate. During the day, all players have the opportunity to speak and discuss the situation before voting to eliminate a suspected player. The game continues until either all Mafias are eliminated, resulting in a victory for the innocent villagers, or when the number of Mafias left is equal to the number of villagers, resulting in a victory for the Mafia group.

3.2 Related Work

The unique setup of the Mafia game requires players to communicate with each other and deduce each other's identities. As a result, Mafia players often lie in order to hide their true identities. This makes the text generated during the game a suitable dataset for studying and detecting deception. Several studies have been conducted to detect deception using linguistic cues. Different models have been employed on different tasks, and the performance have varied based on methodology and data. For example, Mihalcea et al. (2009) [1] collected three datasets of true and lying text based on written deception task. A support vector machine trained on one task achieved a performance of 70%, and a Naive Bayes classifier achieved 71%. The Support vector machine trained on two tasks correclty classified 58% of the samples in the third task, and the Naive Bayes classifier achieved a classification rate of 60%. In another paper, Hu et al. (2019) [2] used models to identify concealed information in verbal speech and text. A LSTM model was able to achieve a F1-score of 0.71. More recently, Barsever et al. (2020) [3] found that BERT networks outperformed state-of-the-art methods in deception classification accuracy on the Ott Deceptive Opinion Spam corpus, which is a commonly used dataset in deception detection tasks that contains both true and false reviews.

4 Approach

In this section, we detail the various approaches we employed to tackle the problem of predicting player roles in Mafia games and product classification in the Amazon dataset based on linguistic behavior. We first describe the NLP models, followed by our proposed BERT+GNN model, which integrates ideas from graph neural networks (GNNs) into traditional NLP models.

After performing preprocessing on the dataset, we split it into train, validation, and test sets in the ratio of 60%-20%-20%. We implement our models on both the initial imbalanced dataset and the augmented, balanced dataset. Throughtout the process, we experienced a few package version compatibility issues when attempting to use the *torchtext* library to read in data. Therefore,

we used some code located at [4] to load and split the data. We also borrow code from the Mafiascum dataset paper [5] for preprocessing the raw dataset. Finally, we gained insights when building our models from the websites [6] and [7].

4.1 Data Augmentation

Data augmentation is a technique that increases the size and diversity of training datasets by creating new variations of the existing data. The goal of data augmentation is to reduce overfitting and improve the generalization ability of the model. In this project, we use different augmentation methods to overcome the imbalance in the Mafia and Amazon datasets.

4.1.1 Baseline: LSTM

Our baseline is the Long Short-Term Memory (LSTM) model, a type of recurrent neural network model first proposed in 1997 by [8] and further improved by [9] from 2000. The LSTM model consists of components including new memory generation, input gate, forget gate, final memory generation, and output/exposure gate. The LSTM model takes the word embeddings of the input text as input and processes the sequences to generate a final hidden state, which is then fed into a fully connected layer for classification. We use the LSTM model as it is capable of learning long-term dependencies, especially in sequential data. Our baseline LSTM model consists of 2 hidden layers, each with a dimension of 200, followed by a dense layer and a final sigmoid classification layer. We arbitrarily select a learning rate of 0.001 and a batch size of 64. We run the baseline model for 15 epochs, in which each epoch takes 2 minutes to run on the AWS server. We also observed an overfitting issue. We show the mathematical structure of the baseline in the appendix due to page limits.

4.1.2 BERT

The first none-baseline model we use is the Bidirectional Encoder Representations from Transformers (BERT) model, first introduced by Google in 2018 [10]. This model is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. It is a powerful transformer-based architecture that excels in various NLP tasks due to its ability to capture contextual information. We use the pre-trained model as the embedding layer, followed by a bidirectional Gated Recurrent Unit (GRU) layer with 3 layers and a dimension of 256.

4.1.3 BERT+CLS

As for the more complicated BERT model, we use the BERT model for sequence classification. For each sequence, the token is a special classification token (CLS), and the final hidden state that corresponds to this token is used as the aggregate sequence representation for classification tasks.

4.1.4 LSTM

We also use a more complicated LSTM model that includes a LSTM layer followed by a 3D attention layer applied to the LSTM output. The attention mechanism involves a softmax activation on the output of a dense layer applied to the input tensor, which produces a weight vector that is averaged across the time dimension. Finally, the attention vector is multiplied elementwise with the input tensor to obtain the output tensor. The LSTM model then includes a dense layer with sigmoid activation function as output.

4.2 NLP+GNN: Hybrid Models

4.2.1 Graph Construction

We build graphs based on the relationships between text examples. The relationships vary in different tasks, so we design custom algorithms to calculate edge weights, which will be introduced later along with the dataset.

4.2.2 NLP Embeddings

Most GNN datasets use bag-of-words or GloVe embeddings as node features. To find out if using more advanced embedding techniques would improve the model's performance, here we compared the model's performance based on TFIDF and pretrained Bert embeddings.

4.2.3 GNN Models

This section introduces GNN models that we used to incorporate graph-structured information into text classification. There are three different GNN models, which are the baseline GCN, then GAT and finally GATBiGRUModel, a custom hybrid model.

1. Graph Convolutional Network (GCN): A popular GNN model that performs localized first-order approximations of spectral graph convolutions. GCN aggregates information from neighboring nodes by applying a shared linear transformation followed by a non-linear activation function. This model is efficient and simple, but it does not account for different types of edges or the varying importance of neighbors, which may lead to suboptimal performance in certain applications. Furthermore, it assumes a fixed graph structure, which may not be ideal for dynamic graphs.
2. Graph Attention Network (GAT): A GNN model that utilizes attention mechanisms to weigh the importance of neighboring nodes during the aggregation process. GAT dynamically computes the importance of each neighbor by assigning attention coefficients that are learned by the model. This allows GAT to adaptively focus on more informative neighbors and ignore less relevant ones, leading to improved performance in various graph-based tasks. Additionally, GAT can handle graphs with varying edge types and graph structures, making it a more flexible choice for complex applications.
3. GATBiGRUModel: A custom GNN model that combines GAT layers with a bidirectional LSTM (BiGRU) layer on top. This architecture is specifically designed to better fit the sequential nature of the BERT pre-trained embeddings as node features. The GAT layers help capture the local graph structure and relationships among nodes, while the BiGRU layer captures the contextual information embedded in the BERT embeddings. By combining the strengths of both GAT and BiGRU, this model can effectively leverage both the graph structure and rich contextual information from BERT embeddings, potentially leading to better performance in tasks that require a deeper understanding of the relationships between nodes and the semantics of the text.

In our experiments, we compare the performance of these three GNN models on the task of text classification with graph-structured data. We evaluate their ability to incorporate graph structure and textual embeddings, as well as their potential to improve classification accuracy in comparison to traditional NLP models. By analyzing the strengths and weaknesses of each model, we aim to provide valuable insights into the suitability of different GNN and NLP architectures for specific applications, as well as the potential benefits of combining these approaches. Our work contributes to the ongoing development of more effective and efficient GNN and NLP models, while highlighting the advantages of leveraging graph structure and advanced embeddings in a wide range of text classification tasks. This integrated approach not only enhances our understanding of the relationships between textual and structural information but also opens up new avenues for future research in the areas of natural language processing and graph neural networks.

5 Experiments

5.1 Data

We first use the Mafiascum dataset, introduced by Ruiter and Kacergis in the paper [11]. This dataset comprises text generated from over 700 Mafia games played on the Mafiascum online forum. The dataset is labeled for each player in each game as either scum or non-scum, making it ideal for training deception classifiers. Some texts reference other players, as shown in this example from game id 28480: *"I'm less sure that Hoopla is scum and that Cooldog is town than I was but I'm starting to see a scum Gorckat. That would be an excellent lynch for tomorrow."*

The Amazon dataset was obtained from [12]. It comprises about 233 million reviews of electronics products sold on Amazon. In addition to the reviews, the dataset includes information such as co-views, co-buys, product IDs, and electronic labels. We sample a portion of the Amazon data for downstream analysis.

5.2 Preprocessing

The text from Mafia games is available in multiple JSON datasets. We largely follow the data preprocessing steps proposed by the authors, first excluding posts made by eliminated players after the final vote count, as they had no incentive to maintain their facade. Additionally, all posts made by a player in the 24 hours before their last in-game post, during the period known as twilight, were excluded to capture only in-game posts. The remaining in-game posts for each player in each game were merged

into a single text document. Any documents containing less than 50 words were subsequently discarded. Each document was assigned a label, indicating whether the player was a member of the Mafia. This resulted in a total of 9,676 documents, each containing all the messages written by a single player in a single game, with an average length of 3,940 words.

5.3 Graph Construction

5.3.1 Mafia Dataset

- Focus: Our study aims to predict player roles (scum or villager) in Mafia games by analyzing their in-game conversations and leveraging both textual and structural information.
- Structure: The dataset consists of player conversations and associated labels indicating their roles during the game.
- Node & Node Feature: Each node in the graph represents a player, with node feature being his conversation, capturing their linguistic behavior and communication patterns.
- Edge: Edges between nodes are based on reference frequency, which measures the strength of connections between players in the game.
- Edge Weights Method: To calculate reference frequency, we employ a fuzzy string matching algorithm (FuzzyWuzzy) that matches candidate player IDs in conversations. This approach helps identify and quantify player interactions, even when exact player IDs are not explicitly mentioned. These weighted edges provide valuable insights into the underlying social network and help models capture the complex interactions between players in Mafia game.

- Focus: Our study aims to classify products based on their reviews and co-viewed or co-purchased relationships, which provide valuable context for understanding customer preferences and product categorization.
- Structure: The dataset comprises product reviews and lists of co-viewed or co-purchased products for each item, capturing both textual information and relational data.
- Node & Node Feature: Each node in the graph represents a product, with node feature being the text of its product review, encompassing customer opinions and feedback about the product.
- Edge: Edges between nodes are based on product connections. For example, if the two products are purchased or viewed by the same customer, an edge is established between the two nodes(products).
- Edge Weight Method: To create the graph, we connect products based on their co-viewed or co-purchased relationships as provided in the dataset. We use the unique product ID (ASIN) to identify and link related products. The connections between products are binary, representing the presence or absence of a relationship between two items. These binary edges help our models capture the underlying network of product associations and uncover patterns that can improve classification performance.

Figure 1 shows an example of a graph generated based on player communications in game 28480.

5.4 Evaluation method

We use several evaluation metrics as we implement models on both imbalanced and balanced datasets. In the Mafia context, it is equally important for the model to be both specific and sensitive. In other words, the model should be able to capture both Mafias and villagers. Additionally, it is also important for the model to correctly classify both types of electronics on the Amazon dataset. On the imbalanced versions of the two datasets, we pay close attention to the F1-score, which is the harmonic mean of precision and recall, as it provides a more balanced measure of the model's performance by considering both the false positives and the false negatives. On the balanced versions of the datasets, we pay more attention to the accuracy. We also use the Precision and Recall to evaluate our models. In the Mafia game identity classification, high precision means the model is correctly identifying Mafia players with a low false positive rate, which is important to avoid falsely accusing innocent players of being Mafia members. High recall would indicate the model is correctly identifying most of the actual Mafia players in the game, which is important as to not missing out any Mafia members.

5.4.1 Amazon Dataset

5.5 Experimental details

We first implemented both models on the original, imbalanced datasets after basic preprocessing. To overcome the imbalance, we employed three types of data augmentation techniques: synonym replacement, random swapping, and back translation. Synonym replacement involves randomly selecting a non-stop word and replacing it with its synonym. Random swapping refers to randomly selecting two words and swapping their positions. Back translation involves translating the original language to another language and then translating back to the original language.

To be specific, we first randomly sampled $\frac{1}{2}$ of the samples corresponding to scum players. Next, we selected 1 to 5 random words from each text to perform synonym replacement. We then randomly selected another 1 to 5 words and performed random swapping. Finally, we performed back translation by translating the text to Spanish and back to English. After these augmentation steps, we randomly selected a balanced number of scum players and non-scum players for our filtered data, which we use as the input data for the models.

We also compared fitting a multi-layer gated recurrent unit (GRU) layer vs a multi-layer perceptron (MLP) as the output. For all experiments, the GRU layer outperformed. To prevent overfitting, we employed early stopping, batch normalization, and dropout in both models. We used random search for hyperparameter tuning on the learning rate, dropout ratio, batch size, and optimizer. The optimal hyperparameter values for the LSTM model was a learning rate of $1 * 10^{-3}$, a batch size of 64, a dropout ratio of 0.5, along with an Adam optimizer. The optimal hyperparameter values for the BERT+CLS model was a learning rate of $1 * 10^{-5}$, a batch size of 32, a dropout ratio of 0.2, along with an Adam optimizer. With early stopping, we trained the LSTM model for 7 epochs and the BERT model for 15 epochs. Each epoch of the LSTM model takes about 2 minutes on the AWS server, and 6 minutes for the BERT model.

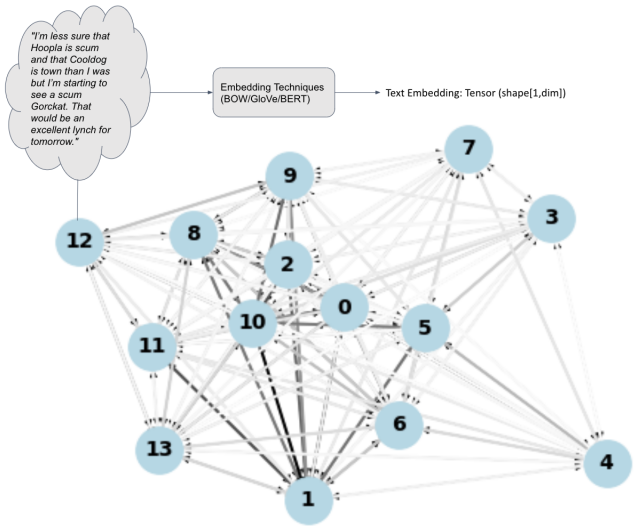

Figure 1: Player Communication Graph for Game #28480

5.6 Results

6 Analysis

6.1 Balance vs Imbalance

The performance of all models on the imbalanced dataset was poor, as they tended to predict all test examples as belonging to the majority class. After data augmentation, we trained our models on the new balanced dataset. The results on the Mafia dataset showed only marginal improvement (LSTM F1-score increased from 0.39 to 0.53), while those on the Amazon dataset

Model	Data	F1-score	Precision	Recall	Accuracy
LSTM	Mafia	0.35	0.72	0.21	0.68
LSTM	Amazon	0.62	0.75	0.53	0.70

Table 1: Baseline Model Performance

Model	Data	F1-score	Precision	Recall	Accuracy
LSTM	Mafia-I	0.39	0.77	0.26	0.72
BERT	Mafia-I	0.35	0.73	0.23	0.70
LSTM	Mafia-B	0.53	0.54	0.53	0.53
BERT	Mafia-B	0.51	0.52	0.51	0.52
LSTM	Amazon-I	0.69	0.96	0.54	0.81
BERT	Amazon-I	0.66	0.91	0.52	0.79
LSTM	Amazon-B	0.87	0.86	0.87	0.88
BERT	Amazon-B	0.89	0.88	0.91	0.90

Table 2: Model Performance (B=balanced,I=imbalanced)

Model	F1-score	Precision	Recall	Accuracy
LSTM	0.87	0.86	0.87	0.88
BERT	0.84	0.86	0.88	0.87
BERT+CLS	0.89	0.88	0.91	0.90

Table 3: Model Performance on Balanced Amazon Dataset

Model	Data	F1-score	Precision	Recall	Accuracy
LSTM	Amazon	0.87	0.86	0.87	0.88
BERT	Amazon	0.89	0.88	0.91	0.90
GCN	Amazon	0.95	0.94	0.96	0.95
GAT	Amazon	0.96	0.97	0.96	0.96
GATBiGRU	Amazon	0.98	0.97	0.98	0.98
GATBiGRU	Mafia	0.05	0.50	0.03	0.57

Table 4: Model Performance on Balanced Datasets

Model	Embedding	F1-score	Precision	Recall	Accuracy
GCN	Bag-of-Words	0.96	0.95	0.96	0.95
GAT	Bag-of-Words	0.97	0.97	0.97	0.97
GATBiGRU	Bag-of-Words	0.98	0.97	0.98	0.96
GCN	PreTrained BERT	0.95	0.94	0.96	0.95
GAT	PreTrained BERT	0.96	0.97	0.96	0.96
GATBiGRU	PreTrained BERT	0.98	0.97	0.98	0.98

Table 5: GNN Model Performance on Balanced Amazon Dataset

improved by 26% (LSTM F1-score increased from 0.69 to 0.87), as shown in Table 2. This indicates that data augmentation through synonym replacement, random swapping, and back translation is effective when dealing with imbalanced datasets in NLP. Furthermore, this demonstrates that imbalance is not the only reason that the Mafia dataset is not machine learnable, and suggests that other factors, such as the inherent complexity of the game or the quality of the data, might also play a role.

6.2 NLP Model Comparison

We compared three NLP deep learning models on the balanced Amazon dataset: LSTM, BERT, and BERT+CLS. Results in Table 3 show that BERT+CLS has the best overall performance, indicating the effectiveness of pre-trained models in NLP tasks. The BERT model achieved better results than LSTM, illustrating the benefits of utilizing more advanced language models for text classification tasks. However, LSTM still performed reasonably well, suggesting that it could be a viable option for less complex tasks or when computational resources are limited.

6.3 Text vs Text+Graph

Instead of only learning from the raw text, we incorporated graph structure data into text classification. By comparing the BERT model (F1-score of 0.89) with the custom GATBiGRU model (F1-score of 0.98) on the balanced Amazon dataset in Table 4, we can see that after learning from both textual and relationship information between examples, F1-score improved (10% increase), as well as accuracy. This demonstrates the value of leveraging graph information along with textual information for certain tasks. Among the three hybrid models, GAT outperformed the simple GCN by adding attention to better weight the message from neighbors. The custom GATBiGRU model had the best performance, which may be due to using LSTM to handle the sequence property of the BERT pre-trained output, while the other two models simply viewed the BERT pre-trained output as a fixed vector. However, the performance on the Mafia dataset was still poor, indicating that the dataset's quality

may not be good. If we want to further research deceptive natural language in the Mafia game, for example, generating good in-game arguments, we should collect higher quality data, such as with stricter rules and higher-level players.

6.4 Embeddings in GNN

Most GNN datasets use bag-of-words or GloVe embeddings as node features. To find out if more advanced embedding techniques would improve the model's performance, we compared TFIDF and BERT pre-trained embeddings. As shown in Table 5, the best model was the custom GATBiGRU model with BERT embeddings. GAT plus BOW embedding also showed pretty good performance. This is probably because the category of products is mainly determined by keywords, so the sequence of words is not that important, and word frequency is enough to predict the category of products. However, it takes much more time to train than BERT embeddings. We believe in more complex tasks where the order of words matters, BERT embeddings will improve performance of GNN models significantly. In summary, the choice of embeddings can have a considerable impact on GNN performance, and more advanced embeddings like BERT can lead to improved results in certain tasks.

7 Conclusion and Future Work

In this paper, we investigated the effectiveness of incorporating graph structure data into text classification tasks using NLP and GNN models. Our experiments were conducted on two datasets, the Mafia game dataset and the Amazon product dataset. We observed significant performance improvements by combining textual and relationship information between examples, particularly on the Amazon dataset.

Our results show that data augmentation through synonym replacement, random swapping, and back translation is effective when dealing with imbalanced datasets in NLP tasks. Moreover, we found that BERT+CLS outperformed other NLP models on the balanced Amazon dataset, highlighting the importance of using pre-trained models for NLP tasks. We also demonstrated the value of leveraging graph information along with textual information for certain tasks, with the custom GATBiGRU model achieving the best performance.

However, the performance on the Mafia dataset remained poor, indicating that the dataset's quality may not be good. Our results suggest that the collection of higher quality data with stricter rules and higher-level players might be necessary for further research on deceptive natural language in the Mafia game.

Based on our findings, there are several directions for future research:

- Use General Adversarial Networks to generate additional examples of text from scum players. We would train the generator to generate fake text examples, and the discriminator to distinguish between fake and real examples.
- Investigate other graph construction techniques, such as dynamically constructing graphs based on context, to better capture the relationships between examples in text classification tasks.
- Study the use of attention mechanisms within GNN models to improve the aggregation of neighborhood information, possibly incorporating transformer architectures for more effective message passing.
- Examine the potential of leveraging other sources of information, such as knowledge graphs or multi-modal data, to further enhance the performance of GNN models in text classification tasks.
- Collect higher quality datasets for the Mafia game to better understand deceptive natural language in the context of the game, potentially leading to the development of models capable of generating good in-game arguments or detecting deceptive players. One higher quality dataset comes from scraping game transcripts generated from PandaKill, a werewolf show, played by experienced players and streamers. Experienced players tend to be more rational in games and incorporate more logical reasoning in their speech. In fact, we scraped data from Pandakill and planned to use it for our project. However, due to long data preprocessing procedures, we will use it for future work.
- Perform grid search for hyperparameter tuning, given more computing resources and work time

Overall, our research demonstrates the potential of combining NLP and GNN techniques for text classification tasks, opening up new avenues for the development of more effective and efficient models in the future.

8 Contributions

Both members contributed to the project. Fang focused on implementing the BERT and LSTM language models, while Jian focused on implementing the BERT+GNN models. Both members contributed to writing up the proposal, milestone, and this final report. We would also like to thank our TA, Rishi, for his generous feedback, guidance and help.

References

[1] Rada Mihalcea and Carlo Strapparava. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 conference short papers*, pages 309–312, 2009.

[2] Shengli Hu. Detecting concealed information in text and speech. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 402–412, 2019.

[3] Dan Barsever, Sameer Singh, and Emre Neftci. Building a better lie detector with bert: The difference between truth and lies. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[4] Pytorch. Migration tutorial. `https://github.com/pytorch/text/tree/master/examples/legacy_tutorial`, 2021.

[5] bopjesvla. The mafiascum dataset. `https://bitbucket.org/bopjesvla/thesis`, 2019.

[6] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[7] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*, 2019.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

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

[11] Bob de Ruiter and George Kachergis. The mafiascum dataset: A large text corpus for deception detection. *arXiv preprint arXiv:1811.07851*, 2018.

[12] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197, 2019.

A Appendix

A.1 Baseline Model Structure

Given the input $\mathbf{X} \in R^{t*n}$, where t is the size of the token embeddings and n is the number of text samples, our model L can be written as follows:

$H_1(X) = LSTM(X) \in R^{n*200}$
$H_2(X) = LSTM(X) \in R^{n*200}$
$D(X) = Dense(H_2) \in R^{200}$
$L(X) = \sigma(D(X))$

A.2 Experimental Platforms

System	GPU	CPU	cuda	pytorch	DGL
Ubuntu 20.04	AMD EPYC 7R32	Nvidia-A10G	11.6	1.12.0	1.0.1

Table 1: Experiment Platforms