NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CZ4042 Neural Networks and Deep Learning

AY2023-2024 Semester 1

Group Project

Sentiment Analysis on Financial News

Group members:

Name	Matric No.
Chua Yong Xuan	U2020627F
Daniel Yang	U2022630G
Jiang Zixing	U2023720J

1. Introduction

In the fast-paced world of financial markets, it is crucial to understand the general sentiments expressed from a vast array of financial news sources as efficiently as possible. These textual sources of information provide valuable insights into market trends and investor behavior, ultimately affecting financial decisions. Text Sentiment Analysis (TSA) is an essential tool for decoding the tone, into positive, neutral or negative classes, within bodies of texts or documents.

This project aims to explore the potential of deep learning in improving TSA, specifically in the areas of domain adaptation, comparative performances of transformers, and the influence of handling small, imbalanced datasets on model performance.

In domain adaptation, this study investigates how neural networks can leverage their learned knowledge from one domain and apply it to another. This exploration helps us understand the flexibility and applicability of neural networks across different domains.

Furthermore, we conduct a comparative analysis of transformers, evaluating and comparing different transformer architectures, focusing specifically on their performance (test accuracy) in sentiment analysis tasks. In this project, we have chosen two popular transformer-based models—BERT (Bidirectional Encoder Representations from Transformers) and XLNet—for our analysis.

Kaggle's Financial News Sentiment Analysis dataset was intentionally selected for this study. We aim to correct the insufficiency of data points and the imbalance of classes found in this dataset. To enhance the dataset, a variety of augmentation strategies were employed. We used Google Translate to convert the original text into a different language and then back into English, creating variations of the original text that maintain context but augment the structure. We also utilized the nlpaug library's synonym function to replace specific words in the dataset with their synonyms, adding textual diversity. Finally, we adopted GPT-2 to transform the original statements into outputs reflective of GPT-2's processing, thereby adding another layer of variation to our dataset. Throughout our notebook, we rigorously test the impact of these individual augmentation methods in enhancing our dataset and contributing to the field of Text Sentiment Analysis.

2. Literature Review

The evolution of sentiment analysis began with lexicon-based methods, where sentiments were deduced from predefined word lists indicative of positive or negative sentiments. However, these methods faltered in capturing contextual nuances and intricate semantic relationships within text. The field progressed to embracing machine learning models like Naive Bayes, Support Vector Machines, and Random Forests, which showcased the ability to learn from data and decipher more complex patterns, surpassing the capabilities of lexicon-based methods. A significant leap was witnessed with the onset of deep learning, particularly neural networks, which marked a substantial advancement in Text Sentiment Analysis (TSA). These models exhibited prowess in learning hierarchical representations, adeptly capturing intricate relationships and nuanced meanings in textual data. The narrative of sentiment analysis further evolved with the advent of transformer architectures like BERT and GPT, which significantly bolstered the capabilities of sentiment analysis techniques. Transformer-based models with self-attention mechanisms could understand contextual relationships within text, establishing state-of-the-art performance across sentiment analysis tasks. [1]

Domain adaptation in sentiment analysis has been a focal point, aiming to address the challenge of applying models trained on one domain to another. Existing techniques often leverage transfer learning to improve performance on a target domain by utilizing information from rich source data. [2] In the comparative

analysis of transformers, various studies have assessed different transformer architectures like BERT and XLNET for NLP tasks. BERT is known for its bidirectional training, while XLNET, with its permutation-based training, often outperforms BERT in several tasks including sentiment analysis. [3] Also, handling small and imbalanced datasets is a common hurdle. Data augmentation techniques to expand datasets address this. Extensive studies on its effects on various deep learning models' performance are, however, lacking. By creating slightly modified copies or generating synthetic data from existing data, these techniques mitigate data scarcity challenges, which is crucial for the effectiveness of deep learning models in NLP. [4]

The exploration of these areas forms a bedrock for enhancing TSA, especially within the financial domain, by addressing challenges like domain adaptation, evaluating transformer architectures, and data scarcity. Through a meticulous review of the existing techniques on augmented datasets, this project aims to bridge the gaps and contribute towards refining the methodologies employed in sentiment analysis.

3. Data Pre-processing and Augmentation

3.1 Exploratory Data Analysis

We performed data pre-processing and augmentation in the **NN_Project_2_Data_Preprocessing.ipynb** notebook. The dataset (**all-data.csv**) was obtained from Kaggle Sentiment Analysis for Financial News. The dataset contains 4846 rows (4840 unique rows) comprising article titles and their corresponding sentiments (class labels). There are 3 unique classes for financial news sentiment: Positive – bullish news, Neutral – mixture of bullish and bearish news and Negative – bearish news. The class counts of the dataset are imbalanced. Articles having neutral sentiment have a significantly higher representation in the dataset (Figure 1).

Sentiment	Count
Neutral	2873
Positive	1363
Negative	604

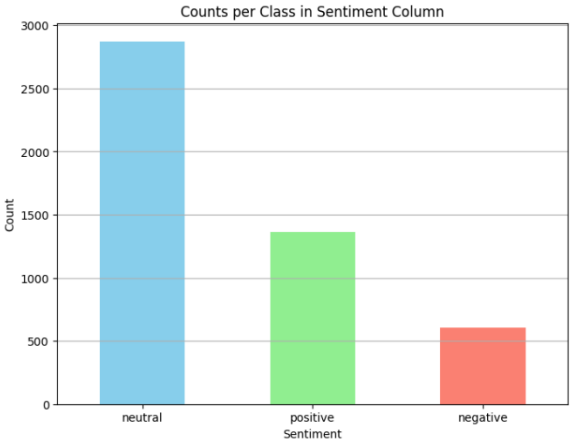

Figure 1. Class counts of article titles

3.2 Data Preparation

Sentiment labels were one-hot encoded from string values to numeric values, *{'positive': 2, 'neutral': 1, 'negative': 0}*.

Three-way data split was performed for hyper parameter tuning in our study. Dataset was partitioned into Train, Validation and Test sets taking 20% of all data points to be the test set, 20% of the remaining data points as the validation set and the remaining were set aside for training (Figure 2). The split datasets were saved as **train.csv**, **validaton.csv**, and **test.csv** respectively.

```
Train set shape: (3097, 2)
Validation set shape: (775, 2)
Test set shape: (968, 2)
```

Figure 2. Row counts for each dataset

Up sampling is performed on the train dataset only to improve model performance. As the negative sentiment label is significantly fewer than other classes, 3 augmentation techniques are adopted: 3.2.1) Google Translate Augmentation 3.2.2) nlpaug Augmentation 3.2.3) GPT-2 Augmentation.

3.2.1 Google Translate Augmentation

Financial new article titles will be back-translated (i.e.. translated from English to French and back into English). This generates additional samples that have the same meaning but with a different text structure. The Google Translate augmented training set was saved as **google-translate-train.csv**.

3.2.2 nlpaug Augmentation

The nlpaug library creates additional samples of the unbalanced classes in the train set by substituting words with their synonyms. This up samples the negative class data points by generating new statements that retain original word meaning. The nlpaug augmented training set was saved as **nlpaug-train.csv**.

3.2.3 GPT-2 Augmentation

The GPT-2 library, developed by OpenAI, functions as a decoder that excels at accurately predicting the next set of tokens in a sequence. This ability enables GPT-2 to generate text that is syntactically coherent when provided with a series of tokens. As a result, we can use GPT-2 to produce more pertinent keywords and text based on an article's title. This enhanced dataset will serve as the new training set of our models. The GPT-2 augmented training set was saved as **gpt-augment-train.csv**.

4. Methodology

4.1 Model Architecture

4.1.1 BERT

We performed sentiment analysis on text data using the BERT (Bidirectional Encoder Representations from Transformers) model in the **NN_Project_2_BERT.ipynb** notebook, a state-of-the-art transformer-based architecture that has shown remarkable performance in various natural language processing tasks.

The process began with importing the training, validation, and test datasets. A custom class, *SentimentDataset*, was established, which requires a tokenizer and the corresponding dataframe as inputs. This class is essential for converting the input text/statements into tokens, which BERT utilizes for its operations, and for generating the attention masks for these tokens. We employed the *'bert-base-uncased'* tokenizer for tokenization as part of the preprocessing of the input text. For modeling, 'prajjwal1/bert-medium' was selected as the pre-trained BERT variant, chosen specifically for its reduced computational demands given limited computing power on Google Colab. [5]

4.1.2 XLNet

We performed sentiment analysis using XLNet in the **NN_Project_2_XLNet.ipynb** notebook. Similarly with XLNet, a custom class, *CustomDataset*, was established, which requires a tokenizer and the corresponding dataframe as inputs.

The *xlnet-base-cased* tokenizer was used to tokenize and preprocess the input dataframe. This is a pretrained tokenizer built on the SentencePiece library.

In this sentiment analysis, we are tackling a classification problem. Therefore, the *XLNetForSequenceClassification* class of pretrained model, which provides a linear layer on top of the pooled output, is used. The *xlnet-base-cased* pre-trained variant is chosen as it has a smaller model size, which also eases the computational demands on Google Colab [5].

4.2 Hyperparameter Tuning

One of the key aspects we want to explore is the impact of hyperparameter tuning on the performance of our model. For this purpose, we will perform hyperparameter tuning based on the range of values recommended by the authors of the original BERT paper, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." [6]

In a similar vein, we adapted the set of XLNet hyperparameters from the XLNet research paper “XLNet: Generalized Autoregressive Pretraining for Language Understanding”. [7] In the choices for XLNet batch sizes, we omitted the recommended value of 128 and replaced with the value of 64 due to the technical limitations of Google Colab T4 GPU. [5]

Hyperparameter	BERT	XLNet
Batch size	16, 32	32, 48, 64
Learning rate (Adam)	5e-5, 3e-5, 2e-5	3e-5, 2e-5, 1e-5
Number of epochs	2, 3, 4	2, 3, 4

Figure 3. Recommended range of hyperparameter values for tuning [6, 7]

The hyperparameters tuned were batch size, learning rate, and the number of epochs (Figure 3). We employed Optuna, a hyperparameter optimization framework, to systematically search for the optimal set of parameters. The function 'objective' (Figure 4) was defined to perform operations sequentially: initializing the datasets, loading the model, and running the training and evaluation processes. Optuna's 'study.optimize' method was then used to conduct ten trials, aiming to identify the combination of hyperparameters that yielded the lowest validation loss.

```python
def objective(trial):
    # Define hyperparameter search space as stated in the original BERT paper
    batch_size = trial.suggest_categorical('batch_size', [16, 32])
    learning_rate = trial.suggest_categorical('learning_rate', [5e-5, 3e-5, 2e-5])
    num_epochs = trial.suggest_categorical('num_epochs', [2, 3, 4])

    # Create dataset and dataloader for both train and val df
    train_dataset = SentimentDataset(train_df, tokenizer)
    train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)

    val_dataset = SentimentDataset(val_df, tokenizer)
    val_dataloader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

    # Load model
    model = BertForSequenceClassification.from_pretrained(model_name, num_labels=3).to(device)

    # Define optimizer
    optimizer = AdamW(model.parameters(), lr=learning_rate)

    # Train and evaluate the model
    for epoch in range(num_epochs):
        train_loss, train_accuracy = train_model(train_dataloader, model, optimizer, device)
        val_loss, val_accuracy = evaluate_model(val_dataloader, model, device)

    return val_loss  # Optimize for minimum validation loss

# Optimize the objective function
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=10, gc_after_trial=True) # We cap the number of trials to 10
```

Figure 4. Creating an Optuna study to find optimal hyperparameters for BERT

5. Results

5.1 Optimal Hyperparameters

From the hyperparameter finetuning, we obtained the following optimal parameters that returned the lowest validation loss.

Hyperparameter	BERT	XLNet
Batch size	32	32
Learning rate (Adam)	3e-5	2e-5
Number of epochs	2	3
Lowest validation loss	0.4178	0.3448

Figure 5. Optimal hyperparameters

5.2 Baseline Data

Having determined the optimal hyperparameters, we proceeded to train the model using the original train dataset.

Model	Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
BERT	1	0.6702	0.7155	0.4787	0.7974
	2	0.3671	0.8621	0.4645	0.8142
XLNet	1	0.3454	0.8654	0.4332	0.8181
	2	0.3454	0.8654	0.3840	0.8387
	3	0.2511	0.9041	0.4129	0.8542

Figure 6. Training results on original train set

Model	Test Loss	Test Accuracy	Test F1 Score
BERT	0.3957	0.8306	0.8305
XLNet	0.4220	0.8554	0.8502

Figure 7. Test results of model trained on original train set

The model was able to generalize well, with similar validation and test accuracy compared to the last epoch's training accuracy. Moreover, the model maintained a robust F1 score, indicative of its ability to effectively balance precision and recall, minimizing both false positives and false negatives.

5.3 Google Translate Augmented Data

We train the model with the Google Translate augmented dataset to evaluate the effects on model performance.

Model	Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
BERT	1	0.7070	0.6972	0.4841	0.8026
	2	0.3523	0.8622	0.4417	0.8116
XLNet	1	0.5744	0.7592	0.3801	0.8245
	2	0.2952	0.8881	0.7526	0.7303
	3	0.2016	0.9242	0.3597	0.8671

Figure 8. Training results on Google Translate augmented dataset

Model	Test Loss	Test Accuracy	Test F1 Score
BERT	0.4147	0.8233	0.8235
XLNet	0.3997	0.8533	0.8537

Figure 9. Test results of model trained on Google Translate augmented dataset

The BERT and XLNet models trained on the Google Translate augmented data performed worse than the original models trained on the original training dataset. This may be due to the introduction of noise into

the data via the translation process. This noise can come in the form of grammatical errors, incorrect word choices, or loss of context, which can confuse the model and degrade performance.

5.4 nlpaug Augmented Data

We train the model with the nlpaug augmented dataset to evaluate the effects on model performance.

Model	Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
BERT	1	0.6772	0.7091	0.4757	0.8039
	2	0.3397	0.8717	0.4089	0.8258
XLNet	1	0.6188	0.7378	0.4809	0.7299
	2	0.3135	0.8798	0.3667	0.8684
	3	0.2161	0.9196	0.5071	0.8206

Figure 10. Training results on nlpaug augmented dataset

Model	Test Loss	Test Accuracy	Test F1 Score
BERT	0.3626	0.8440	0.8435
XLNet	0.4314	0.8368	0.8399

Figure 11. Test results of model trained on nlpaug augmented dataset

The BERT model trained on the nlpaug augmented data performed better than the original model trained on the original training dataset. By augmenting the minority class using synonyms, the model is exposed to a wider range of vocabulary and context and a more balanced dataset, which may have led to better generalization by the model.

However, the XLNet model trained on the nlpaug augmented data performed worse than that of the original training dataset. The augmented dataset may have been less effective in capturing the nuances of the original data, causing the dataset to be dissimilar from the original data and potentially confuse the model. Compared to the original data, the XLNet model performed better in training but not in validation and testing. Additionally, the augmentation process might inadvertently emphasize certain patterns that are not representative of the underlying data distribution. If those patterns differ from what the XLNet model learned on the raw dataset, it could lead to a decrease in performance.

5.5 GPT-2 Augmented Data

We train the model with the GPT-2 augmented dataset to evaluate the effects on model performance.

Model	Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
BERT	1	0.6943	0.7052	0.5204	0.7768
	2	0.3667	0.8595	0.4370	0.8155
XLNet	1	0.6942	0.7029	0.4674	0.8039
	2	0.3407	0.8624	0.4324	0.8516
	3	0.2352	0.9089	0.4106	0.8542

Figure 12. Training results on GPT-2 augmented dataset

Model	Test Loss	Test Accuracy	Test F1 Score
BERT	0.3890	0.8409	0.8416
XLNet	0.3331	0.8647	0.8642

Figure 13. Test results of model trained on GPT-2 augmented dataset

The performance improvement observed in both the BERT and XLNet model trained on the GPT-2 augmented dataset, as compared to the one trained on the original data, suggests that the rephrasing capabilities of GPT-2's decoder architecture may preserve or potentially enrich the semantic content necessary for the classifier to determine sentiment accurately. This may mean that employing more advanced decoder-only language models, such as GPT-3 or LLaMA 2, could offer further enhancements in the quality of text augmentation and subsequent classification performance.

6. Discussion

6.1 Dataset

For the BERT model, we find that the nlpaug augmented data was most effective in improving the test accuracy of the model from 0.8306 to 0.8440.

For the XLNet model, we find that the GPT-2 augmented data was most effective in improving the test accuracy of the model from 0.8554 to 0.8647.

Generally, it is observed that upsampling can help to improve the model performance. The difference between the effectiveness of datasets on different models could be explained by the different models having different architectures and learning objectives — BERT uses masked language modeling, while XLNet uses permutation language modeling. These differences can make certain types of augmented data more suitable for each model. The specific characteristics of the dataset may have played a role. Augmentation techniques like nlpaug and GPT-2 augmentation introduce variations in the data, and the impact of these variations depends on the kind of patterns the model has learned from the original data. It could also be related to the nature of the augmentation itself. For example, GPT-2 augmentation might introduce more diverse and contextually rich variations, which could be beneficial for a model like XLNet that relies on bidirectional context understanding.

6.2 Model

By training on the same datasets, XLNet test accuracy scores are generally higher than that of BERT. This is expected as XLNet is an improvement of BERT with a significantly better performance in 20 NLP

benchmark tasks [7]. XLNet learns more dependency pairs in a corpus given the same target as BERT and contains denser effective training signals. The permutation language modeling objective of XLNet requires the model to consider all permutations of the input sequence during training. This could lead to a more robust understanding of the relationships between words and their positions in the sequence.

7. Future Extensions

7.1 Computing Power

In order to test the full potential of a larger model, we can obtain a GPU/Google Colab Pro to unlock greater computing power, and possibly higher test accuracies.

7.2 Dataset

We can extend our search for a larger corpus to avoid the downside of small and imbalanced dataset, or mitigate the bias introduced by any augmentation techniques. This could be achieved through web-scraping or connecting to third party data providers like GDELT that has consolidated the news in a database and affixed each news with a sentiment score.

7.3 Augmentation Techniques

More robust test generation and augmentation techniques such as the state-of-the-art GPT-4 or Meta's open-source LLaMA 2, can be used to enhance our training sets and therefore model predictive power. Effects of adding these to the test sets (i.e.. Prediction pipeline) has yet to be studied. Ensemble learning could reduce the variance of the predictions and reduce generalization errors that damage test accuracy.

7.4 Exploring Novel Model Architecture

In this project, we used GPT-2 as one of the data augmentation techniques to test the performance of models on a dataset augmented using a decoder-only model. It may be possible to prepend a decoder-only model as part of our model architecture, which may be worth exploring since the models trained with GPT-2 augmented dataset showed a slight improvement over the base training dataset.

8. Conclusion

Our exploration into the realm of deep learning for TSA within the financial domain has unveiled intriguing insights and posed challenges. The importance of understanding sentiments in financial news for market trends and decision-making cannot be overstated, and our project has sought to enhance TSA through the lenses of domain adaptation, transformer architecture comparison, and addressing the hurdles posed by small, imbalanced datasets.

Notably, BERT benefited from nlpaug augmentation, demonstrating improved test accuracy. On the other hand, XLNet showcased better performance when trained on the GPT-2 augmented dataset. The intricacies of each model's architecture and learning objectives, coupled with the characteristics of the dataset, played pivotal roles in determining the effectiveness of augmentation strategies.

Our project serves as a stepping stone in refining methodologies for sentiment analysis within the financial domain. The intricate dance between model architecture, augmentation strategies, and dataset characteristics highlights the need for a nuanced approach in the pursuit of improved TSA.

9. References

[1] A. A. Ansari, "Evolution of Sentiment Analysis: Methodologies and Paradigms," in Trends of Data Science and Applications, S. S. Rautaray, P. Pemmaraju, and H. Mohanty, Eds., vol. 954, Studies in Computational Intelligence, Singapore: Springer, 2021. [Online]. Available: https://doi.org/10.1007/978-981-33-6815-6_8

[2] M. Iman, K. Rasheed, and H. R. Arabnia, "A Review of Deep Transfer Learning and Recent Advancements," Technologies, vol. 11, no. 40, 2023. [Online]. Available: https://doi.org/10.3390/technologies11020040. [Accessed: 31-Oct-2023].

[3] N. Arabadzhieva - Kalcheva and I. Kovachev, "Comparison of BERT and XLNet accuracy with classical methods and algorithms in text classification," 2021 International Conference on Biomedical Innovations and Applications (BIA), Varna, Bulgaria, 2022, pp. 74-76, doi: 10.1109/BIA52594.2022.9831281.

[4] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy, "A Survey of Data Augmentation Approaches for NLP," in ACL 2021 Findings, May 2021, arXiv preprint arXiv:2105.03075. [Online]. Available: https://doi.org/10.48550/arXiv.2105.03075. [Accessed: 31-Oct-2023].

[5] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, "Well-Read Students Learn Better: On the Importance of Pre-training Compact Models," 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1908.08962. [Accessed: 4-Nov-2023].

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805 [cs.CL], Oct. 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1810.04805. [Accessed: 4-Nov-2023].

[7] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1906.08237