Propaganda Detection - Advanced Natural Language Processing Assignment

Abstract

This report explores how the NLP models such as Word2Vec and BERT can be used to detect propaganda techniques in text. These pre-trained models are designed to handle two types of tasks: identifying the presence of propaganda and another is classifying the specific propaganda techniques used. This report highlights the strengths and weaknesses of each model in handling the two tasks of propaganda detection. The report found that while Word2Vec is quite good at grasping semantic relationships, it struggles with context sensitivity. On the other hand, BERT performs much better in understanding context, which leads to more accurate classifications.

1 Introduction

Propaganda is a strategic way of spreading information to shape people's attitudes, actions, or even fundamental beliefs and points of view, usually in a way that benefits the person or group sending the message. It usually involves twisting facts or presenting biased information that is circulated across different media platforms, aiming to influence and manipulate a wide audience. Propaganda is used to influence public opinion and advance specific agendas, to manipulate elections and other democratic processes by spreading misinformation and disinformation, spread extremist ideologies and hate speech that can lead to social unrest and division, spread misinformation about public health measures such as vaccinations which can affect public health safety and so on. These issues raise the need to develop means to effectively detect propaganda, ensuring that public opinion always remains based on factual and unbiased information.

The challenge of detecting propaganda is that it requires handling large amounts of data, especially text, to identify if it has a propagandic tone in the language. The propagandists often use subtle manipulation techniques that are context-based, making it difficult to detect them without understanding the broader context. These techniques can vary from appealing to stereotypes, fear-mongering, or more sophisticated methods like logical fallacies or bandwagon appeals. Here, NLP(Natural Language Processing) models can be a useful tool to help with this challenge. The models can be trained to analyze massive datasets, identify patterns, and classify these texts by recognizing particular features that may indicate propaganda.

This report explores the performance of two NLP models, one, pre-trained Word2Vec embedding with a simple Neural Network, and the second, BERT (Bidirectional Encoder Representations from Transformers) language model. The models are individually trained for two types of tasks. The first task requires them to identify if a text contains propaganda or not. The second task requires each model to classify what type of propaganda technique has been used in a known propaganda text.

Some recent work shows how BERT can be integrated with entity mapping and has enhanced model accuracy in classifying different types of propaganda techniques in news articles (Bairaktaris, Symeonidis, & Arampatzis, 2020). Also, a fine-tuned BERT model for word-level classification of text can be used to detect different propaganda techniques (Yoosuf & Yang, 2019). Malik, Imran, & Mamdouh (2023)investigated different feature models for detecting

propaganda, including Word2Vec. Their study noted Word2Vec's strength in enhancing the model's ability to understand complex semantic relationships in propaganda content.

The comparison of the two models (Word2Vec and BERT) can provide a comprehensive view of how these two NLP technologies perform in the context of propaganda detection. This analysis is beneficial because it offers insights into the strengths and weaknesses of each model.

2 Methods

2.1 Task-1

The first requirement is to identify and classify the presence or absence of propagandic language in a given text. The provided training and Validation datasets contain pre-labeled text data where the propaganda text is labeled with a unique label signifying the propaganda technique used and the texts without propaganda is marked as such. Here, identifying a particular technique is not required, hence, the data is re-labeled in a binary format. The 'not_propaganda' label is replaced with '0' and all other propaganda labels with '1'. The following sections explain the implementation of the approaches used detect propaganda.

2.1.1 Word2Vec Embedding with Neural Network

The pre-trained Word2Vec vectors from Google News dataset containing 300-dimensional vectors ('word2vec-google-news-300') are used here for word embedding. The **gensim** library's API is used to download these vectors, which have been trained on a large corpus and capture a vast amount of semantic information and relationships between words. The binary labeled texts are tokenized using nltk after removing all special characters and other special tokens, removing stop-words, and lemmatization. These tokens are then vectorized using the Word2Vec vectors. 20% of the Training data is reserved for testing.

For the neural network, tensorflow keras is used because the neural network implemented is simple and therefore only needs straightforword implementation. Keras has build in methods for training and evaluation which reduces coding requirement significantly. The input layer of the neural network expects vectors of 300 dimensions, which is according to the dimensions of the Word2Vec embeddings used to vectorize the text. Next two layers are made of standard number of 128 and 64 neurons respectively with reLU activation. ReLU has non-linear properties and has the ability to reduce the vanishing gradient problem. The output is a single unit with sigmoid activation. The Dropout is set to 0.5 which randomly sets 50% of the input units to 0 at each step during training, which will help prevent overfitting. For Loss Function, 'binary_crossentropy' is implemented. This loss function measures the "distance" between the model's predictions and the actual labels, and the goal during training is to minimize this value. The training is done with 10 epochs with a batch size of 32.

2.1.2 BERT Language Model

For this approach, the pre-trained BERT model embedding 'bert-base-uncased' is used and integrated with a PyTorch neural network for customisation. The data is encode with BERT tokenizer and prepared as dataloaders for training. The PropagandaClassifier class defined in the code is a custom PyTorch neural network model. The pre-existing BERT model is initialised and moved to GPU (if available, for faster computation). The dropout layer, a dropout probability of 0.5 is set to avoid overfitting. A final linear layer is added for mapping the high-dimensional output of the BERT model into two output classes as per binary classification.

The forward method takes input IDs and an attention mask as inputs and captures the pooled output of the BERT model which is a summary of the input features. The pooled output is applied with dropout for regularization, then the linear layer turns it into logits which

are the raw scores for the two classes. These logits are used with 'nn.CrossEntropyLoss' loss function to evaluate the model's predictions and update its weights through backpropagation during training. CrossEntropyLoss measures the performance of models whose output is a probability value between 0 and 1. It calculates the loss by comparing the model's predicted probabilities with the true distribution. With a 0.00002 Learning rate, the training step is run for only 3 epochs with a batch size of 4. Due to hardware limitations, increasing epochs and batch size is not possible.

2.2 Task-2

The second requirement is to identify and classify the technique of propaganda language in a given text. The propaganda text is labeled with a unique label of the propaganda technique used. Following is list of the labels given.

1. Flag waving
2. Appeal to fear/prejudice
3. Causal simplification
4. Doubt
5. Exaggeration/minimisation
6. Loaded language
7. Name calling/labeling
8. Repetition
9. Not propaganda

Here, the text is already known to have propaganda, hence, the data labeled as 'not_propaganda' is removed. The rest of the propaganda labels are mapped with a unique number between 0-7. The following sections explain the implementation of the approaches used to detect propaganda technique.

2.2.1 Word2Vec Embedding with Neural Network

This implementation is similar to the implementation in Task-1. The same pre-trained Word2Vec vectors ('word2vec-google-news-300') are used here for word embedding too. In the provided data, `<BOS>` and `<EOS>` tags are used to mark the span of text that contains the propaganda. This span of text is extracted, word tokenized and then is embedded with the vectors. These tags help significantly to locate the exact words used in propaganda. The multi-classes are one-hot encoded before using it in the model.

The neural network implemented here is also similar to the implementation in Task-1, except the output layer has 8 unit and softmax is used for activation which aligns with the one-hot encoding done for the classes. The loss function 'categorical_crossentropy' is used here which is suitable for multi-classes. The training is done with 30 epochs with a batch size of 32. The use of a higher epoch in this case is to improve the training of the model.

2.2.2 BERT Language Model

In the second approach for Task-2, the same pre-trained BERT model embedding 'bert-base-uncased' is used and integrated with a PyTorch neural network. The span of text is extracted before it is encoded with BERT tokenizer. The neural network classifier here is similar to the binary classifier before but had to be customised in some places to adjust with multiple classes.

The classifier includes two fully connected linear layers, the first linear layer transforms the output of the BERT model from its hidden size of 768 to 128 neurons and the next from 128 to 32 neurons. Here, LeakyReLU activation is used instead of ReLU, to allow small gradients when a neuron is not active. The output from the second dropout layer is passed through the last linear layer, which maps these features to the final number of classes. The output of this layer, logits, represents the raw predictions for each class, which is passed through a softmax function. The dropout is 0.5 and CrossEntropyLoss function is used as loss function same as before. The training step is executed for 3 epochs with a batch size of 4 and with a 0.00002 Learning rate.

3 Results

The following tables and figures show the results of the model for each task.

3.1 Task-1

3.1.1 Word2Vec Embedding with Neural Network

Results of Word2Vec Embedding with Neural Network after training and its performance report are shown in Table-1 and Figure-1.

Final Evaluation Results on Validation Set:
Validation Loss: 0.58
Validation Accuracy: 0.72

Table 1: Word2Vec Training and Validation Loss and Accuracy per Epoch (Task-1)

Epoch	Time (ms/step)	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	16	0.6832	0.5448	0.6688	0.6724
2	3	0.5879	0.6970	0.5796	0.7023
3	3	0.5604	0.7204	0.5723	0.6983
4	3	0.5307	0.7421	0.5761	0.7069
5	3	0.5120	0.7561	0.5564	0.7431
6	3	0.5200	0.7628	0.5617	0.7190
7	3	0.4507	0.7939	0.5091	0.7224
8	3	0.4279	0.7851	0.5686	0.7241
9	3	0.4422	0.8853	0.5795	0.7190
10	3	0.5795	0.7190	0.5795	0.7190

3.1.2 BERT Language Model

Results of BERT Language Model with Neural Network after training and its performance report are shown in Table-2 and Figure-2.

```
16/16 [==============================] - 0s 3ms/step

------------------Performance Report-----------------

                precision    recall  f1-score   support

Non propaganda       0.77      0.69      0.73       241
    propaganda       0.72      0.80      0.76       242

      accuracy                           0.74       483
     macro avg       0.75      0.74      0.74       483
  weighted avg       0.75      0.74      0.74       483
```

Figure 1: Word2Vec Model Performance Report for Task-1

Table 2: BERT Training and Validation Loss and Accuracy for Three Epochs (Task-1)

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
1/3	0.5810	0.6986	0.5744	0.6914
2/3	0.3434	0.8622	0.5129	0.7707
3/3	0.1406	0.9482	0.7102	0.7724

```
Test Loss: 0.7035, Test Accuracy: 0.7971

------------------Performance Report-----------------

                precision    recall  f1-score   support

Not Propaganda       0.79      0.81      0.80       241
    Propaganda       0.81      0.78      0.79       242

      accuracy                           0.80       483
     macro avg       0.80      0.80      0.80       483
  weighted avg       0.80      0.80      0.80       483
```

Figure 2: BERT Model Performance Report for Task-1

3.2 Task-2

3.2.1 Word2Vec Embedding with Neural Network

Results of Word2Vec Embedding with Neural Network after training and its performance report are shown in Table-3 and Figure-3.

Final Evaluation Results on Validation Set:
Validation Loss: 1.54
Validation Accuracy: 0.52

3.2.2 BERT Language Model

Results of BERT Language Model with Neural Network after training and its performance report are shown in Table-4 and Figure-4.

Table 3: Word2Vec Training and Validation Loss & Accuracy for Epochs (Task-2)

Epoch	Time (ms/step)	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
17	0.48	1.1666	0.5767	1.0957	0.4839
18	0.45	1.1402	0.5624	1.1498	0.5018
19	0.45	1.8877	0.6155	1.1436	0.4946
20	0.45	1.0469	0.6166	1.1151	0.5054
21	0.45	1.0251	0.6483	1.1995	0.5125
22	0.45	1.0089	0.6380	1.5000	0.5305
23	0.45	1.0083	0.6482	1.1598	0.5188
24	0.45	0.9717	0.6534	1.5234	0.5197
25	0.45	0.9337	0.6554	1.5221	0.5269
26	0.45	0.9108	0.6738	1.5338	0.5197
27	0.45	0.9229	0.6871	1.5431	0.5125
28	0.45	0.8906	0.6902	1.5531	0.5161
29	0.45	0.8728	0.6838	1.5407	0.5395

----------Performance Report----------

	precision	recall	f1-score	support
flag_waving	0.59	0.79	0.68	24
appeal_to_fear_prejudice	0.58	0.52	0.55	27
causal_oversimplification	0.43	0.53	0.48	30
doubt	0.44	0.41	0.43	29
exaggeration,minimisation	0.56	0.62	0.59	37
loaded_language	0.44	0.38	0.41	32
name_calling,labeling	0.52	0.47	0.49	34
repetition	0.42	0.34	0.38	32
accuracy			0.50	245
macro avg	0.50	0.51	0.50	245
weighted avg	0.50	0.50	0.50	245

Figure 3: Word2Vec Model Performance Report for Task-2

4 Analysis

4.1 Task-1

4.1.1 Word2Vec Embedding with Neural Network

The Table-1 show a steady improvement in both training and validation accuracy and loss as the epochs progress. Starting with an accuracy of 54.48% in the first epoch, the model's training accuracy improves significantly to 85.53% by the last epoch. Similarly, the validation accuracy starts at 67.24% and reaches up to 72.44%.

The training loss decreases from 0.6932 to 0.4242, and the validation loss also sees a reduction from 0.6688 to 0.5795. This consistent decrease in loss indicates that the model is learning effectively from the training data without showing signs of overfitting, as the validation loss is decreasing with the training loss.

By the end of the training, both accuracy and loss values begin to stabilize, which means that additional training epochs might not produce any significant improvements without changes in model architecture, data, or training procedure.

Table 4: BERT Training and Validation Loss and Accuracy for Three Epochs (Task-2)

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
1/3	2.0135	0.2045	1.9535	0.2115
2/3	1.9121	0.2638	1.8407	0.2545
3/3	1.7857	0.3119	1.7313	0.2796

```
Test Loss: 1.7762, Test Accuracy: 0.2816

Classification Report:

                           precision    recall  f1-score   support

                flag_waving       0.23      0.75      0.35        24
 appeal_to_fear_prejudice       0.00      0.00      0.00        27
causal_oversimplification       0.38      0.87      0.53        30
                    doubt       0.00      0.00      0.00        29
exaggeration,minimisation       0.00      0.00      0.00        37
          loaded_language       0.21      0.09      0.13        32
    name_calling,labeling       0.00      0.00      0.00        34
               repetition       0.26      0.69      0.38        32

                 accuracy                           0.28       245
                macro avg       0.14      0.30      0.17       245
             weighted avg       0.13      0.28      0.17       245
```

Figure 4: BERT Model Performance Report for Task-2

The final reported validation loss is 0.58, and the accuracy is 0.72. These results are consistent with the last observed epoch metrics, indicating the model has generalized well to unseen data in the validation set.

4.1.2 BERT Language Model

From Table-2, over only 3 epochs, the training loss decreased significantly from 0.581 to 0.105 across the epochs, which indicates that the model is efficiently learning from the training data. Also, the training accuracy increased from approximately 69.86% to 94.82% which is another major improvement. The validation loss initially decreased and then slightly increased from 0.574 in the first epoch to 0.710 in the third epoch. Validation accuracy, however, shows a slower improvement from 69.14% to 77.24% across epochs.

Although, increase in training accuracy alongside a decrease in training loss shows good learning in the model's training phase. But, the model's validation accuracy, while improving, is much lower than the training accuracy, particularly in the last epoch (94.82% vs. 77.24%), which means there is some level of overfitting. The test loss also stands at 0.7035 with an accuracy of 79.71% which are closer to the validation scores than to the training scores, meaning there is some overfitting going on in the training phase.

4.1.3 Word2Vec and BERT Language Model - Comparison

Overall, the BERT model learns significantly faster, achieving high training accuracy and lower training loss in just 3 epochs compared to 10 epochs in the Word2Vec model. Although, both models show signs of overfitting as seen by the gap between training and validation accuracy, the BERT model maintains a higher validation accuracy despite a higher validation loss in the final epoch, which means it generalizes slightly better than the Word2Vec model.

The performance reports in Fig-1[Word2Vec] and Fig-2[BERT Model] include the following metrics:

1. **Precision:** Indicates the accuracy of positive predictions.
2. **Recall:** Indicates the coverage of actual positive cases.
3. **F1-Score:** Harmonic mean of precision and recall, providing a balance between the two.
4. **Support:** Number of true instances for each class.

BERT model, compared to Word2Vec model, show better improvements in both precision and recall across classes due to better handling of context within the input texts which is not handled in Word2Vec. But this better performance comes at the cost of increased computational resources and longer training times. In cases where computational resources are limited and faster training is required, Word2Vec could be a better choice.

4.2 Task-2

4.2.1 Word2Vec Embedding with Neural Network

Here, the model is trained over 30 epochs and the last few epoch results are shown in Table-3 and Fig-3. Initially, the training loss decreases, which is a typical indication that the model is learning from the training data. But the reduction in loss slows down and fluctuates as the training progresses. Similarly, the validation loss decreases initially but then stabilizes around 1.54, with minor fluctuations. The final reported validation loss is 1.54. Starts at 57.67% and shows some improvement, peaking at 71.37%. This means that the model is learning, but probably not enough to capture all the complexities of the data. Meanwhile, The validation accuracy initially increases but level out around 52%, with the final validation accuracy also at 52%. This means that the model is not generalizing well beyond the training data on unseen data.

The higher training accuracy compared to validation accuracy in later epochs suggests that the model may be overfitting the training data. The stagnation of both training and validation accuracy at relatively low levels could also suggest underfitting, where the model is too simple to capture the underlying patterns of the data fully. It might be beneficial to increase the model's complexity by adding more layers and working with different types of layers, or adjusting the hyper-parameters to better train the model.

4.2.2 BERT Language Model

The BERT model for Multi-class classifier has shown the poorest performance out of all others as can be seen from Table-4 and Fig-4. Although, both training and validation losses decrease consistently across epochs and training and validation accuracy increase with each epoch, in overall all values remain extremely low compared to others.

This is not a limitation of BERT model, it is because of lack of proper implementation, hyper-tuning and experimentation. This could be overcome but it may require better computational resources.

5 Discussion

On comparing the Word2Vec and BERT models, both models are good at detecting propaganda, but BERT often takes the lead in terms of accuracy and contextual understanding, thanks to its advanced transformer architecture that considers wider textual contexts. But this advantage does mean BERT requires more computing power. In case of multi-class propaganda detection,

Word2Vec perform much poorer than before, while BERT model performs the poorest. This might be because of the oversimplified architecture of both the models. This study also points out a tendency for both models to overfit, though BERT shows a better ability to generalize when tested on new data, despite sometimes suffering higher losses during training. A major challenge was limited computing resources and the need for deeper exploration into optimal settings for these models.

Looking ahead, further research could focus into adding more complex architecture using better computing resources. Combining other relevant NLP models could also potentially be beneficial in improving performance, especially for multiple classes. Another area of study that could be explored is how to adapt these models with new emerging propaganda techniques rather than fixed technique definitions.

6 Conclusion

In conclusion, both Word2Vec and BERT for detecting propaganda has some strengths and weaknesses while performing the task. Word2Vec performed well in understanding semantic relationships but often falls short when context becomes complex. BERT, with its advanced architecture, generally offers more accuracy and a better grasp of context, though it requires more computational power. Both models show signs of being prone to overfitting.

Despite the challenges, BERT still shows a stronger ability to perform well on new, unseen data. In order for the models to perform better, there is a need for better implementation design, computing power and a thorough experimentation of fine-tuning these models to maximize their potential.

Appendix

See File Attached:

- **TASK-1**: NLP_TASK_1.ipynb
- **TASK-2**: NLP_TASK_2.ipynb

References

[1] Bairaktaris, A., Symeonidis, S., & Arampatzis, A. (2020). DUTH at SemEval-2020 Task 11: BERT with Entity Mapping for Propaganda Classification. `https://doi.org/10.18653/v1/2020.semeval-1.227`.

[2] Yoosuf, S., & Yang, Y. (2019). Fine-Grained Propaganda Detection with Fine-Tuned BERT. *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda.* `https://doi.org/10.18653/v1/D19-5011`.

[3] Malik, M., Imran, T., & Mamdouh, J. (2023). How to detect propaganda from social media? Exploitation of semantic and fine-tuned language models. *PeerJ Computer Science*, 9. `https://doi.org/10.7717/peerj-cs.1248`.

[4] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding.