PROPAGANDA DETECTION

YASHWANTH AJJILAPURAM
277241

May 13, 2024

1 Introduction

Propaganda is a form of communication with the primary goal of influencing or persuading an audience to support a particular agenda. This type of communication may not be objective and may present facts in a way that favors a certain synthesis or perception, or it may use strong language to generate an emotional reaction from the audience instead of a reasoned one. Politicians, media, big businesses, and state and non-state entities can all employ propaganda to promote false narratives and sow division in society (Da San Martino et al., EMNLP-IJCNLP 2019).
Therefore, it is necessary to identify instances of propaganda in any media with a particular level of precision in order to prevent the success of campaigns to sway and convince people.This report will cover two distinct approaches to recognizing propaganda inside a text. It will also go through how to use these techniques to categorize a span of text that is known to include propaganda into one of the various categories of propaganda (Salman et al., 2023) .

2 Dataset

A specific kind of text classification that requires advanced natural language processing is propaganda detection. The tab-delimited text files "*propaganda_train.csv*" and "*propaganda_val.tsv*," which include the training and validation dataset created especially for this experiment, will be used to train the text classification models. Two columns compose each file: "*label*" and "*tagged_in_context*." One of nine values in the "*label*" field indicates the type of propaganda in the text or indicates it is not an instance of propaganda. The "*tagged_in_context*" field contains a span-identified string of various length that may contain an instance of propaganda. The nine distinct possibilities are as follows:

1.**flag waving:** Patriotic display to evoke support or nationalism.
2. **appeal to fear prejudice:** Using fear or bias to sway opinion.
3. **causal simplification:** Oversimplified cause-and-effect explanation.
4. **doubt:** Casting uncertainty on opposing viewpoints or facts.
5. **exaggeration,minimisation:** Amplifying or downplaying for persuasion.
6. **loaded language:** Emotive words to influence perception or response.
7. **name calling,labeling:** Using derogatory terms to discredit or categorize.
8. **repetition:** Reiterating messages for reinforcement and memorability.
9. **not_propaganda:** Communication free from bias or persuasion tactics.

The first 8 of these possibilities are a subset of the 14 different propaganda types (Giovanni Da San Martino, 2020). There are 2414 entries altogether in the training dataset, split among the nine possible labels. Although the datasets appear uneven at first, they are dispersed evenly in relation to the work we will be performing, which will be discussed in the following section. There are 580 records in all in the validation dataset, spread among the nine potential levels.

3 Tasks for Propaganda Detection

Text classification is by default the process of identifying the type of propaganda included in a text and giving it a name. This propaganda detection assignment consists of two distinct tasks.

Task 1: Build and evaluate at least 2 approaches to classify whether a sentence contains propaganda or not.

The task 1 can be simplified to a binary text classification problem. It must anticipate that the labels "propaganda" and "not_propaganda" will be present.To perform this task, a new label will be created in the dataset 'binary_label' which will have 1 of these 2 values. All the strings known to contain propaganda will have the value – 'propaganda'. This also makes the dataset very balanced for the binary classification task.
We will build classifiers using two entirely different approaches in order to complete task 1 binary text classification. They are as follows:
1) Bi-LSTMs(Bidirectional Long Short-Term Memory) are a type of recurrent neural network architecture that can effectively model sequential data like text by capturing long-range dependencies.They have shown strong performance on many natural language processing tasks like text classification, tagging, and text summarization.
2) A pretrained large language model BERT (Bidirectional Encoder Representations from Transformers) will be used to generate a contextualised embedding, which will be used downstream to perform text classification tasks by a neural network.

Task 2: Given a snippet or span of text which is known to contain propaganda, build and evaluate at least 2 different approaches to classifying the propaganda technique which has been used.

The task 2 can be considered to be a multiclass text classification task. As the model which will be built needs to predict one of 8 labels.This is because the model will only employ material that has been proven to include propaganda; any strings that do not contain any instances of propaganda will need to be screened out. This achieves balance in the dataset, which is therefore suitable for training classification algorithms.
We will build classifiers using two entirely different approaches in order to complete task 2 multiclass text classification. They are as follows:
1) A pretrained GloVe (Global Vectors for Word Representation) word embedding model will be used to generate a vector representation of text, which will be then passed into a neural network classifier.
2) A pretrained large language model BERT (Bidirectional Encoder Representations from Transformers) will be used to generate a contextualised embedding, which will be used downstream to perform text classification tasks by a neural network.

These are entirely different approaches, and we will discuss how Task 1 and Task 2 will be implemented by these 2 methods in detail in the next section. However, we will have a brief overview of the Bi-LSTM,GloVe and BERT models before this.

3.1 Bi-LSTMs(Bidirectional Long Short-Term Memory)

Bidirectional LSTMs (Bi-LSTMs) are a variant of traditional LSTMs (Long Short-Term Memory networks) that can process data in both forward and backward directions.

A standard LSTM processes the input sequence in the forward direction, capturing patterns from the past context. However, in many sequence modeling tasks like natural language processing, the future context is also important for accurate predictions (ScienceDirect, 2024).

Bi-LSTMs address this by having two separate hidden layers that process the input sequence in opposite directions - one from start to end, and another from end to start. This allows the model to capture patterns from both past and future contexts. The outputs from the two hidden layers are then combined at each time step, typically by concatenation or summation, to produce the final output. This combined output encodes information from the entire sequence for that time step(Baeldung, 2024).
Our neural language model is constructed using Keras Sequential API. It consists of an Embedding layer, which converts the integer-encoded words into dense vectors of 100 dimensions. These word embeddings capture the semantic meanings of words based on their context within the text.

3.2 BERT (Bidirectional Encoder Representations from Transformers)

Google's BERT is a significant advancement in natural language processing, incorporating bidirectional contextual information from text data. It undergoes pretraining on unlabeled text to predict omitted words and determine consecutive sentence pairss (Jacob Devlin, 2018). BERT leverages bidirectionality, allowing it to jointly condition on leftward and rightward context around a given word. Its transformer architecture offers advantages in modeling long-range dependencies compared to recurrent neural networks. BERT's transfer learning approach has led to performance gains across multiple NLP benchmarks.

We are interested in using the pre-trained BERT model, also known as the "bert-baseuncased" model, which has an embedding dimension of 768, 12 transformer blocks, and 110 million parameters, to accomplish the propaganda detection tasks. Since all of the text was trained on uncased data, it will all be transformed to lowercase. The model will not take into account the word case variations. The Hugging Face Transformers library licences this model for use.

3.3 GloVe (Global Vectors for Word Representation)

GloVe is an unsupervised learning technique that creates dense vector representations of words based on their semantic and syntactic properties. It uses co-occurrence statistics from a large corpus(Jeffrey Pennington, 2014). of text to derive these representations. GloVe constructs a co-occurrence matrix, capturing the number of times a word appears within a specific context window. This global statistical information allows it to capture nuanced semantic relationships (Abad, et al., 2016) and produce more meaningful word representations. GloVe improves performance on natural language processing tasks.

We will use the gensim package's "glove-wiki-giga-300" pre-trained model, which was trained on a sizable corpus of Wikipedia text, to obtain the word embeddings for the text in datasets. With the text as input, this will produce a vector representation with 300-dimensional dense vectors for each word.

4 Binary Classification - Task 1:

4.1 Bi-LSTM based Binary Classification of propaganda text

The binary classification dataset that we have acquired has two labels, namely "propaganda" and "not_propaganda," as previously mentioned in this study. Prior to training, we used the test_train_split function from the sklearn package to divide the training dataset into separate train (80%) and test (20%) sets. The prepro-

cessing steps are involved as removing HTML tags, punctuation, converting text to lowercase, tokenizing sentences, and removing stop words using NLTK.

Tokenization and Padding

We used Keras' Tokenizer to convert the preprocessed text into sequences of integers, where each integer represents a word in the vocabulary. These sequences are then padded to ensure a consistent length for input to the Bi-LSTM model.

4.1.1 Model Architecture

The Bi-LSTM model consists of the following layers:

1.**Embedding Layer:** An embedding layer maps the integer-encoded words to dense vector representations. The embedding vectors are learned during training, rather than using pre-trained GloVe embeddings.
2. **Bidirectional LSTM Layer:** A Bidirectional LSTM layer processes the input sequences in both forward and backward directions, capturing context from past and future states. We use 128 units in this layer, with 'return_sequences=True' to return the full sequence of outputs.
3. **Dropout Layer:** A dropout layer with a rate of 0.2 is added after the Bi-LSTM layer to prevent overfitting.
4. **Another Bidirectional LSTM Layer:** A second Bi-LSTM layer with 128 units is added, without 'return_sequences', to further process the output of the previous Bi-LSTM layer.
5. **Dropout Layer:** Another dropout layer with a rate of 0.2 is added.
6. **Dense Layer:** A dense layer with 64 units and ReLU activation is added for non-linear transformations.
7. **Output Layer:** The final output layer is a dense layer with a single unit and sigmoid activation for binary classification.

Hyperparameters

The hyperparameters used in this Bi-LSTM model are:
1. Batch Size: The size of the batch of training data (32 in this case).
2. Epochs: The number of iterations used to train the model (5 epochs).
3. Validation Split: 20% of the training data is used for validation during training.
4. Early Stopping: The training will stop if the validation loss does not improve for 3 consecutive epochs.

Training and Evaluation

The model is trained using binary cross-entropy loss and the Adam optimizer. After training, the model is evaluated on a validation set, and the validation loss and accuracy are reported. Additionally, a classification report is generated, providing the precision, recall, and F1-score for each class (propaganda and not_propaganda). After training for 5 epochs, the model achieves a validation accuracy of 64% on the validation set, as shown in the classification report:

Table 1: Bi LSTM binary classification

	precision	recall	f1-score	support
not_propaganda	0.63	0.73	0.68	301
propaganda	0.65	0.53	0.58	279
accuracy	–	–	0.64	580

This demonstrates a decent performance for the binary classification task using the Bi-LSTM model

4.2 BERT-based Binary Classification of propaganda text

As stated earlier, we create a contextualised word embedding using the pre-trained BERT language model. This embedding is then fed into a basic neural network to carry out the binary classification.

Prior to defining the model's architecture, we must create a Dataset class that will convert the dataframe object, together with its vectors and labels, into a tensor object. The inputs are transformed into tensors using the PyTorch library, which can subsequently be loaded onto the CPU or GPU for model validation, testing, and prediction. This process trains the models. The use of CUDA cores, which are exclusive to NVIDIA GPUs, can expedite the process if the platform hosting these operations is equipped with a GPU. We can get a dataset that can be indexed like a Python list or array by building a new Dataset class and inheriting from the "torch.utils.data.Dataset" class. This makes it simple to import and prepare data for training or evaluation. Since this process is the same for all models, it won't be covered in more detail in the next sections.

4.2.1 Model Architecture

The "BertModel" layer, which is the initial layer of this network, creates word embeddings using the pre-trained model. The following layer is a dropout layer, which prevents overfitting by using the pooled output that the BERT model provides as input. Since the BERT model has an embedding dimension of 768 and an output size of 8 (number of data labels), the following layer is a linear layer with an input size of 768. Since this layer comprises contextualised word embeddings that capture the contextual information of each word in the sentence, the linear layer uses the pooled output of the BERT model as its input. The linear output is then subjected to a ReLU activation function to obtain the final output, which is the anticipated label. The network gains additional non-linearity from the ReLU function.
Hyperparameters

The Hyperparameters used in this BERT model are:
1) Dropout: The probability of a neuron being turned off in the dropout layer.
2) Learning Rate: This determines how much the model parameters are updated with respect to the gradient of the loss function.
3) Epochs: The number of iterations used to train the model
4) Batch Size: The size of the batch of training data.
After evaluating the BERT model on the test set, it achieves a test accuracy of 50%, with the following classification report:

Table 2: BERT binary classification

	precision	recall	f1-score	support
not_propaganda	0.33	0.02	0.03	237
propaganda	0.51	0.97	0.66	246
accuracy	–	–	0.50	483

While the model shows high precision for the 'propaganda' class, it struggles with recall, resulting in an overall moderate test accuracy."

5 Multiclass Classification - Task 2:

5.1 BERT based Multiclass Classification of propaganda text

It was attempted to pass in the 2 features – the entire sentence and the propaganda span of text as input features to the BERT model but eventually failed to materialise. The pre-trained BERT model did not accept the concatenated tensors containing the entire sentence and span as inputs. Therefore, the BERT model was trained using the 'tagged_in_context' field alone.

5.1.1 Model Architecture

Contextualized embeddings are extracted from the input text by the neural network using the BERT model. A linear layer with an input size of 768 receives the pooled output of the BERT model as its input after that. The network consists of three dense, fully connected layers, the sizes.
A leaky ReLU activation layer receives the output from every fully connected layer. The network now has non-linearity as a result. A dropout layer comes after the activation layer to prevent the model from being overfit to the training set and to allow it to generalize effectively to new data. In order to predict the label, the Softmax layer, the network's last layer, generates a probability distribution across the number of classes and outputs in the network.

Hyperparameters

The Hyperparameters used in this BERT model are:
1)Batch Size: The size of the batch of training data.
2) Dropout: The probability of a neuron being turned off in the dropout layer.
3) LeakyReLU: The LeakyReLU function parameter can be varied to increase performance
4) Learning Rate: This determines how much the model parameters are updated with respect to the gradient of the loss function.
5) Epochs: The number of iterations used to train the model
6)Size of the hidden layer
After training for 3 epochs with a learning rate of 1e-6 and batch size of 4, the BERT model achieves a validation accuracy of 22% on the multiclass propaganda classification task

5.2 GloVe based Multiclass Classification of propaganda text

A new derived column named "prop_span" was established in order to do multiclass classification; it only contains the textual span that consists of propaganda. The model received this column as input along with the original column that included the complete statement. It was assumed that the propaganda-containing material would provide the classifier additional semantic meaning and possibly improve its ability to predict the various propaganda types.

5.2.1 Model Architecture

The inputs to the classifier are 'inten' and 'outen'. The pre-trained GloVe word embeddings for each token in the input sentence are contained in tensor 'inten', and the propaganda-identified text span is contained in tensor 'outen'. This four completely connected layers make up this multiclass neural network. Tensors 'inten' and 'outen' are concatenated in the first fully connected layer. Consequently, because both sentence vectors have a size of 300, the input size is 600.

A batch normalisation layer comes after the first three connected layers in this network. This layer will facilitate faster model convergence during training and helps in stabilising the training process. Additional advantages encompass decreased overfitting and enhanced model generalisation. The issue of vanishing gradients is resolved by batch normalisation. A LeakyReLU activation function layer is introduced after each batch normalisation layer to reduce the probability of "dead" neurons, which occur when the ReLU function turns off too many neurons by setting their outputs to 0. LeakyReLU solves this issue and might improve performance as well. In order to reduce the risk of overfitting to the training data, deeper understand the features in the training data, and facilitate the model's easy generalisation to new data, a dropout layer is also added after each ReLU layer (PyTorch, n.d.). We apply no Dropout Layer or LeakyReLU Activation Layer after the final fully connected layer, fc4. A softmax function takes the output from the final linear layer and produces a probability distribution over the output labels. Given that this is a multiclass classification problem, it is helpful to choose the highest probability label to be predicted label by generating a probability for each output label.

Hyperparameters

The Hyperparameters used in this GloVe model are:
1) Batch Size: The size of the batch of training data.
2) Dropout: The probability of a neuron being turned off in the dropout layer.
3) LeakyReLU: The LeakyReLU function parameter can be varied to increase performance
4) Learning Rate: This determines how much the model parameters are updated with respect to the gradient of the loss function.
5) Epochs: The number of iterations used to train the model
6) Size of the hidden layer

The GloVe model, trained for 20 epochs with a learning rate of 1e-03, achieves a training accuracy of 53% and a test accuracy of 72% on the multiclass propaganda classification task, with the following classification report on the test set:

6 Analysis of Errors of the Methods

Task 1

The Bi-LSTM model has slightly lower precision for the 'propaganda' class compared to the 'not_propaganda'

class, suggesting it may misclassify some instances of propaganda as non-propaganda. It may struggle with subtle or implicit forms of propaganda and longer or more complex sentences. The BERT model has a high recall for the 'not_propaganda' class but low recall for the 'propaganda' class, suggesting it struggles with propaganda instances that don't align with pre-training data or techniques requiring specific cultural, political, or domain-specific knowledge.

Task 2

The BERT model has a low validation accuracy of 22%, suggesting it struggles to differentiate between propaganda techniques. It may struggle with nuanced or context-dependent techniques and techniques that overlap or share similarities in linguistic patterns. The GloVe model performs well overall with a test accuracy of 72%, but exhibits lower precision and recall for certain classes. The model might struggle with subtle or implicit language patterns or techniques requiring deeper contextual understanding, making it harder to capture using GloVe's word-level representations.

7 Further Work

The BERT model can be fine-tuned by analyzing larger propaganda data or domain-specific datasets to understand specific propaganda techniques. Ensemble methods can be used to combine predictions from different models, leveraging their strengths. Attention mechanisms can be incorporated to focus on relevant text parts. Contextual word embeddings like ELMo or XLNet can capture nuanced information. Multi-task learning can be used to train models on related tasks, such as sentiment analysis, fact-checking, and fake news detection, to learn more robust representations and transfer knowledge across tasks.

8 Conclusion

This study investigates the detection of propaganda in text using two main tasks: binary classification and multi-class classification. The Bi-LSTM model achieved a good validation accuracy of 64% for binary classification, while the BERT-based model had a lower test accuracy of 50%, possibly due to limitations in capturing propaganda techniques that deviate from pre-training data. The multi-class classification task was more challenging, with the BERT-based model achieving only 22% validation accuracy. The GloVe-based model showed promising results with a 72% test accuracy, although it still exhibited lower performance on certain classes. The analysis of errors revealed that both tasks posed challenges in handling subtle, implicit, or context-dependent instances of propaganda. The study emphasizes the need for more advanced models to capture the nuances and variations in propaganda techniques. Future work should focus on larger datasets, attention mechanisms, contextual word embeddings, multi-task learning approaches, and ensemble methods that combine the strengths of different models. Propaganda detection remains a critical task in combating misinformation and safeguarding information ecosystems. The findings contribute to ongoing efforts in this domain and provide insights for further advancements in developing robust and effective propaganda detection systems.

9 Code Appendix

Refer the file named 'adv277241.ipynb'

10 References

Abad, A. et al., 2016. Advances in Speech and Language Technologies for Iberian Languages. Lisbon, Portugal, s.n.

Baeldung (2024) Differences Between Bidirectional and Unidirectional LSTM.

Available at: https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm (Accessed: 12 May 2024)

Da San Martino, G., Yu, S., Barrón-Cedeño, A., Petrov, R., Nakov, P. (2019). Fine-Grained Analysis of Propaganda in News Article. In K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 5636-5646). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1565

Giovanni Da San Martino, A. B.-C. H. W. R. P. P. N., 2020. SemEval-2020 Task 11: Detection of Propaganda Techniques in News Articles. p. 1379.

Jacob Devlin, M.-W. C. K. L. K. T., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Jeffrey Pennington, R. S. a. C. D. M., 2014. GloVe: Global Vectors for Word Representationm Conference on Empirical Methods in Natural Language Processing (EMNLP),. s.l., s.n., pp. 1532- 1543.

PyTorch, n.d. LeakyRELU. [Online]

Available at: https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html [Accessed 2024].

Salman, M., Hanif, A., Shehata, S., Nakov, P. (2023). Detecting Propaganda Techniques in Code-Switched Social Media Text. In H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (pp. 16794-16812). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.1044

ScienceDirect (2024) Bidirectional Long Short-Term Memory Network.

Available at: https://www.sciencedirect.com/topics/computer-science/bidirectional-long-short-term-memory-network (Accessed: 12 May 2024)

Smith, B. L., 2016. Propaganda. In: Encyclopædia Britannica, Inc. s.l.:Encyclopædia Britannica, In

Wikipedia, n.d. Propaganda. [Online] Available at: https://en.wikipedia.org/wiki/Propaganda [Ac-

cessed May 2024]