Propaganda Detection and Classification with Bag-of-words and BERT

Mert OLCAMAN
University of Sussex
Brighton, United Kingdom BN1 9BJ
May 13, 2024

Introduction

Propaganda is the intentional spreading of information, whether it's true, arguments, rumours, or even lies, to shape public opinion. It often uses mass media and symbols like words, gestures, or images to spread the message. Propaganda aims to influence what people think or how they behave. Propagandists have clear goals and carefully pick and present information to achieve them. (Smith, 2024).

Many propaganda techniques have been effectively used not only by people, but also by some organisations through different sources. It can lead to some serious problems like the spreading of fake news (Anders, 2024). Additionally, people can do some activities such as the time when COVID-19 came up at first. Many people went to supermarkets to buy products and it caused the products to run out. Also, it is frequently preferred by politicians so that they can get more votes in elections. Even if what is told to the public is wrong, people tend to do whatever is told due to not being able to see the reality. Because of that, these kinds of untrustable sharings must be found and stopped as soon as possible before causing something bad.

As technology grows, we're flooded with tons of data, especially on social media. Bots, which are AI algorithms, often spread propaganda there. Spotting propaganda is tough because it's tricky and spreads fast. This makes it hard to tell what's real and what's trying to deceive us. We need smart tools and algorithms to help out, but even those algorithms might struggle to keep up with all the new tricks of propaganda. On the other hand, some technical words from some fields can be used while creating a propaganda sentence. Therefore, domain knowledge is required to comprehend the sentence thoroughly. It’s not something that everyone can do, since it requires time to gain proficiency (DAlessio, 2021).

There are many studies which have been conducted for years. For example, Large Language Models (LLMs) such as GPT-3 and GPT-4 were used by experimenting on the SemEval-2020 task 11 dataset, which has 14 propaganda techniques. In the research, the highest F1 score was gathered through the GPT-4 base at 58.11% followed by the GPT-4 chain of thought at 57.34% (Sprenkamp et al., 2023).

In another research which was conducted on social media data from Twitter, different machine learning algorithms such as logistic regression, and Multinomial Naïve Bayes were used after using bag-of-words as a method of feature extraction. The model worked with 97.23% accuracy in terms of detecting a sentence containing propaganda (Khanday et al., 2020). Even though Large Language Models don’t perform 100% well exactly, they are trained by loads of data to understand and generate human-like content. LLMs use word embeddings to understand what the context means. Word embeddings are the set of number representations of the words as vectors. In that way, the relationships among the relevant vectors can be evaluated in a multi-dimensional space, the similar sentences are

found close to each other (Barnard, 2024).

I used 2 different approaches. The first one is BERT Model as an LLM and the bag-of-words (BoW) model with the Naïve Bayes classifier. I tested my models on 2 different tasks:

1. A model which checks whether a sentence contains propaganda or not.
2. A model which checks the snippet which is known to contain propaganda and classifies which propaganda type is used.

Data

There are totally 9 different labels in the dataset as follows:

1. **Flag waving:** Using national pride to justify an action or idea.
2. **Appeal to fear prejudice:** Creating support by inducing anxiety towards an alternative.
3. **Causal simplification:** Blaming one cause without considering others.
4. **Doubt:** Questioning credibility.
5. **Exaggeration, minimization:** Making things seem larger or smaller than they are.
6. **Loaded language:** Using emotionally charged words.
7. **Name calling, labelling:** Labeling to evoke strong emotions.
8. **Repetition:** Repeating a message for acceptance.
9. **Not propaganda:** Free from any propaganda content.

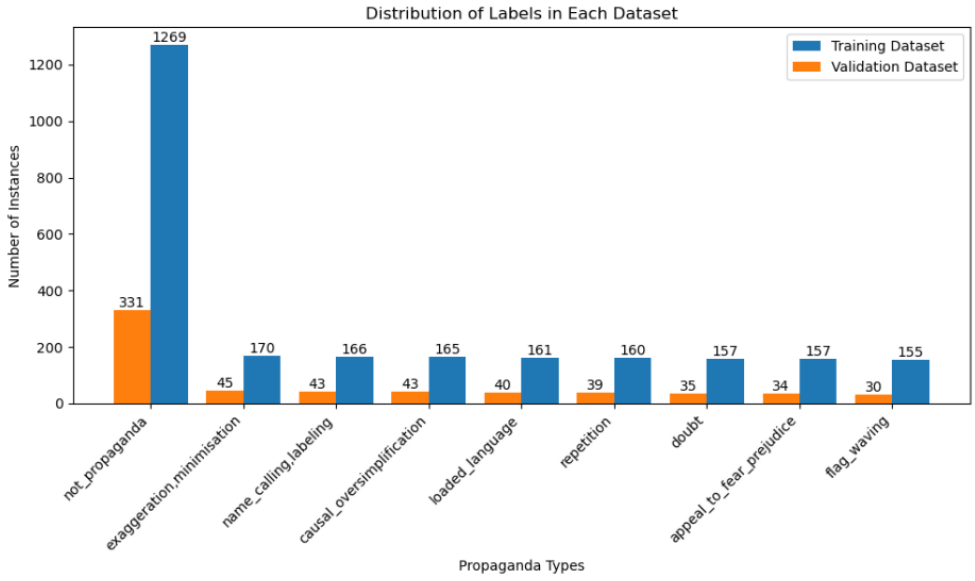

Figure 1: Label distribution for each label in both datasets

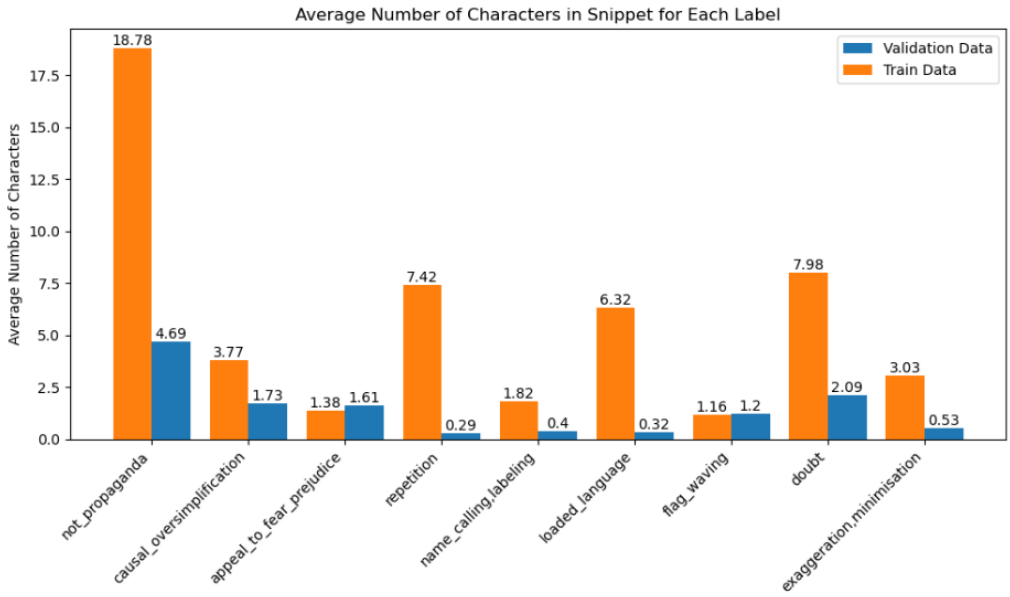

Figure 2: Average number of characters between snippets for each label in both datasets

Methods

I divided the propaganda detection into 2 separate parts. In the first one, I found whether a sentence is a propaganda sentence or not. In the second one, I classified them. Also, I used 2 different approaches in each task.

First Approach: Bag of words with Multinomial Naïve Bayes Classifier

The concept of Bag of Words (BoW) featurization used to process text data for machine learning models. BoW represents text documents by quantifying the frequency of words, ignoring their order and context (Murel PhD & Kavlakoglu, 2024).

Naïve Bayes simplifies the relationship, allowing for efficient classification. Despite their simplicity, naive Bayes classifiers have proven effective in various real-world applications like document classification and spam filtering, requiring minimal training data. They are fast and efficient due to their decoupling of feature distributions, although they may not provide accurate probability estimates (Pedregosa et al., 2011).

As for the first approach, I used the bag of words as a feature extraction. There were some hyperparameters that I took into account for CountVectorizer as can be seen in **Table 1** with their default values and the values that they can take (Pedregosa et al., 2011).

Parameter Name	default value	value range
lowercase	True	True, False
stop_words	None	‘english’, None
ngram_range	(1,1)	(1,1) … (1,n)
analyzer	‘word’	‘word’, ‘char’, ‘char_wb’
binary	False	True, False

Table 1: CountVectorizer parameters

lowercase: converting all characters to lowercase before tokenizing.

stop_words: choosing whether stop words will be considered.

ngram_range: specifying the range of the number of characters or words of n-grams.

analyser: choosing the level of analysis. If the analyser is the word, sets of words are gathered. The difference between char and char_wb is that each character is taken into the set one by one in the char, while space is added at the edges of the words in char_wb.

binary: when it is set to True, the word will take the value 1 or 0 according to presence or absence respectively.

Then, I classified them by using Multinomial Naïve Bayes.

Naive Bayes is a supervised learning algorithm based on Bayes' theorem, with the "naive" assumption of conditional independence between features given the class variable (Pedregosa et al., 2011).

The hyperparameters for this method can be seen in **Table 2**.

Parameter Name	default value	value range
alpha	1.0	0 to n
force_alpha	True	True, False

Table 2: Parameters for Multinomial Naive Bayes

alpha: smoothing parameter to add a value to zero probabilities.

force_alpha: If False and alpha are less than 1e-10, it will set alpha to 1e-10. If it is True, the alpha will remain unchanged. This may cause numerical errors if the alpha is too close to 0.

Finding best parameters

I calculated vectorizer and Naïve Bayes functions separately. Otherwise, it would take a lot of time.

-Vectorizer:

I found the n-gram ranges which give higher accuracy than others. Then, I built for loops for other parameters by taking 5 n-gram ranges to decrease the computation time. After finding the best accuracy that I could gather with those parameters, I built another for-loops algorithm to get the all parameters which have the same accuracy score as the best. After that, I added all into a data frame to see the differences between each other. Some parameters were ineffective. In the end, I found separate model performance metrics such as precision, recall, and f1-score by creating a data frame and confusion matrix.

-MultinomialNB:

As for the classifier function, I created separate lists for both alpha and force_alpha values. Then, I created for loops to get the highest accuracy score. Next, I saved all hyperparameters which give the best accuracy value to create a data frame followed by a confusion matrix.

-Combination:

After finding the best values for each one, I created another for-loops algorithm to find the best accuracy from their combinations. While creating lists for value ranges, I took close values to the ones that I found as the best not to miss any parameters which can give a high accuracy. For example, if I found the best n-gram range as (1, 1), I took from (1, 1) to (1, 5).

Second Approach: BERT

As a second approach, I used BERT (Bidirectional Encoder Representations from Transformers), which considers both left and right context in text. BERT was pre-trained deep bidirectional representations from unlabeled data, which uses left-to-right and right-to-left architecture, to catch the meaning of words better by the masked language model. It was released by Google researchers (Devlin et al., 2018).

As it can be seen in **Table 3**, some of BERT parameters with their default values and the values that they can take.

Parameter Name	Value Range
epochs	1 to n
batch_size	1 to the length of data
learning_rate	0 to n
bert model	base, large, uncased, cased (there are more)
freezing model weights	freezing, unfreezing

Table 3: BERT hyperparameters

epochs: the number of times that the dataset passes through the neural networks.

batch size: the number of examples which take at the same time in each iteration.

learning rate: the step size during the optimization process.

bert model: specifying the fine-tuned model imported.

freezing model weights: it specifies whether the learnt weights in the model are changed.

Each hyperparameter was considered separately at first by specifying default parameters since it takes a lot of time as can be seen in **Table 4**. I took a batch size of 2 to get a higher accuracy by taking every 2 data in each step. Also, I set the number of epochs as 3 to balance the computation time and see the effect of the parameters correctly. The learning rate was specified as small so as not to loss the point for fine-tuning. Otherwise, loss function could increase suddenly just after decreasing a couple of times. As for BERT model, I decided to choose the base model to decrease the computation time compared to the large model at first. Also, the reason why I preferred uncased is that I wanted to take all input as lowercase into account. Moreover, I made the weights in the pre-trained model freezed to decrease the computation time at first.

Parameter Name	Values
epochs	3
batch_size	2
learning_rate	1e-6
bert model	base uncased
freezing model weights	freeze

Table 4: Bert hyperparameter values at the beginning

Sequentially, each hyperparameter was changed one by one. When the validation loss started to increase, the changing of relevant parameters was stopped to prevent overfitting, even though the validation accuracy increased slightly. Each result was printed into a list of dictionaries which consisted of the parameters set. It was done in

separate models for each task. Additionally, Adam was used as an optimizer.

Results

Bag of words with Multinomial Naïve Bayes Classifier

First Task: Propaganda or Not

The accuracy at the beginning was 70%.

After changing the parameters, it was seen that force alpha and stop words parameters didn't affect the model's performance. With all the parameters that I found, I got 81.25% accuracy. The parameters I chose at the end can be seen in **Table 5**. Even though I used the whole sentence in the first task, the best analyzer was found as char, not the word. The F1-score of both can be told as the same. It shows that the model is consistent. The reason why char gave better results in analyzer is because of having less data. There were out-of-vocabulary and therefore it caused sparsity. The analyzer of word couldn't outperform due to having 0 probabilities. All of the zero ones got the same alpha value. However, the character approach worked better, since it checked the sentences on a deeper level.

Function	**Parameter Name**	Values
Vectorizer	**lowercase**	True
	stop_words	None
	ngram_range	(1, 3)
	analyzer	char
	binary	True
Model	**alpha**	0.75
	force alpha	True

Table 5: The hyperparameters for Vectorizer and Naive Bayes for the first task

	Precision	**Recall**	**F1-score**
0 (not_propaganda)	0.843648	0.782477	0.811912
1 (propaganda)	0.783784	0.844660	0.813084
Accuracy	0.812500	0.812500	0.812500
Macro Avg	0.813716	0.813569	0.812498
Weighted Avg	0.814745	0.812500	0.812478

Table 6 Model performance for Vectorizer and Naive Bayes for the first task

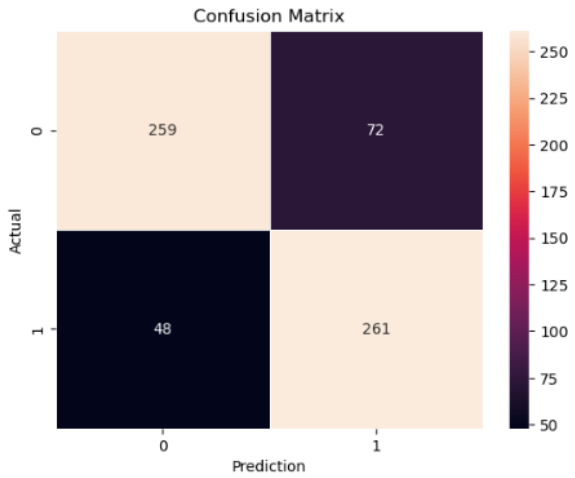

Figure 3: Confusion matrix for Vectorizer and Naive Bayes for the first task

Second Task: Propaganda Classification

The accuracy at the beginning was 39%.

Since only the snippets were taken into account, there were just a few words in each data. The model couldn't catch properly through less amount of data unlike in the first task. The best analyzer was found as char wb with other parameters in **Table 7** by holding an accuracy of 51.46% because the model could catch the context through the characters. However, the accuracy which was gathered with char was 48.87%, which can be seen through **Table 8**. Similarly to the first task, changing stop words and force alpha didn't change anything in the model. Also, flag-waving is the propaganda type that the model was the most successful in classifying them. The minimum F1-score is 0.43 for the 7th label, while the highest one is 0.62 for label 0. There is almost a 20% difference between them. It shows that the model is not stable completely.

Function	Parameter Name	Values
Vectorizer	lowercase	True
	stop_words	None
	ngram_range	(1, 4)
	analyzer	char_wb
	binary	True
Model	alpha	0.175
	force alpha	True

Table 7: The hyperparameters for Vectorizer and Naive Bayes for the second task

	Precision	Recall	F1-score
0 (flag_waving)	0.658537	0.600000	0.627907
1 (loaded_language)	0.516129	0.410256	0.457143
2 (doubt)	0.491803	0.697674	0.576923
3 (name_calling)	0.625000	0.441176	0.517241
4 (appeal_to_fear_prejudice)	0.500000	0.465116	0.481928
5 (repetition)	0.772727	0.425000	0.548387
6 (causal_oversimplification)	0.372549	0.542857	0.441860
7 (exaggeration,minimization)	0.384615	0.500000	0.434783
Accuracy	0.514563	0.514563	0.514563
Macro Avg	0.540170	0.510260	0.510772
Weighted Avg	0.547403	0.514563	0.516651

Table 8: Model performance for Vectorizer and Naive Bayes for the second task

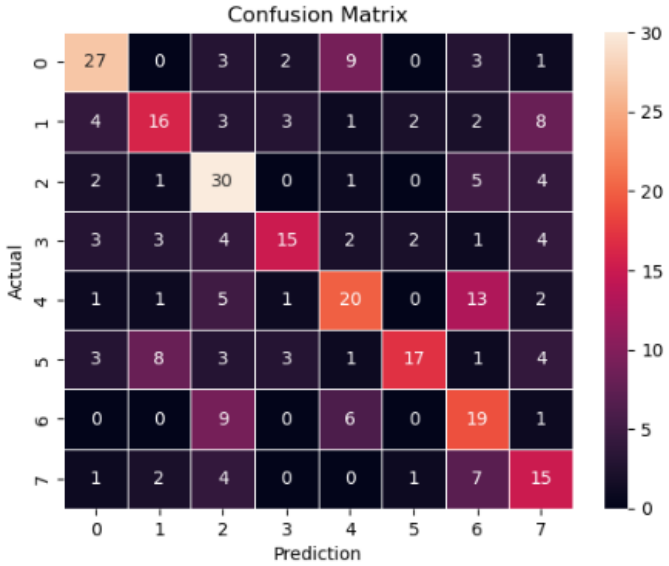

Figure 4: Confusion matrix for Vectorizer and Naive Bayes for the second task

BERT

First Task: Propaganda or Not

The accuracy at the beginning was around 50%.

Even though increasing the epoch was found as a good approach, validation loss started to increase after the 2nd epoch (**Table 9**). Otherwise, overfitting could be observed. Accuracy got 72.34% by increasing nearly a quarter.

epochs	train_accuracy	val_loss	val_accuracy	batch_size	learning rate
1	0.677734	0.256952	0.748437	2	0.0000065
2	0.801172	0.244282	0.775000	2	0.0000065
3	0.883984	0.256098	0.775000	2	0.0000065

Table 9: BERT model performance in training for the first task

Parameter Name	Values
epochs	2
batch_size	2
learning_rate	6.5e-6
bert model	base uncased
freezing model weights	unfreeze

Table 10: BERT optimized parameters for the first task

	Precision	**Recall**	**F1-score**
0 (not_propaganda)	0.669604	0.918429	0.774522
1 (propaganda)	0.854839	0.514563	0.642424
Accuracy	0.723437	0.723437	0.723437
Macro Avg	0.762221	0.716496	0.708473
Weighted Avg	0.759037	0.723437	0.710744

Table 11: Model performance for BERT in the first task

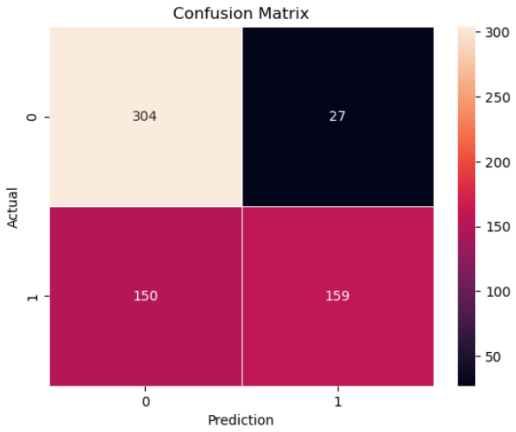

Figure 5: Confusion matrix for BERT in the first task

Second Task: Propaganda Classification

The accuracy at the beginning was 31.07%.

The validation loss started to increase after the 6th epoch with the best parameters at 63.75% accuracy as in **Table 12**. To prevent overfitting, the epoch was accepted as 6. According to that, the parameters were set as in **Table 13**. With these parameters, a model was gathered which has the parameters in **Table 14** and the confusion matrix in **Figure 6**.

epochs	train_accuracy	val_loss	val_accuracy	batch_size	learning rate
1	0.192874	0.942842	0.291262	2	0.0000065
2	0.386522	0.766524	0.482201	2	0.0000065
3	0.580945	0.632216	0.576052	2	0.0000065
4	0.739737	0.557613	0.618123	2	0.0000065
5	0.848954	0.591008	0.627832	2	0.0000065
6	0.908598	0.580146	0.637540	2	0.0000065
7	0.946553	0.623926	0.634304	2	0.0000065
8	0.956623	0.620971	0.650485	2	0.0000065

Table 12: BERT model performance in training for the second task

Parameter Name	Values
epochs	6
batch_size	2
learning_rate	6.5e-6
bert model	base uncased
freezing model weights	unfreeze

Table 13: BERT optimized parameters for the second task

	Precision	Recall	F1-score
0 (flag_waving)	0.777778	0.622222	0.691358
1 (loaded_language)	0.581395	0.641026	0.609756
2 (doubt)	0.694444	0.581395	0.632911
3 (name_calling)	0.785714	0.647059	0.709677
4 (appeal_to_fear_prejudice)	0.627907	0.627907	0.627907
5 (repetition)	0.615385	0.600000	0.607595
6 (causal_oversimplification)	0.526316	0.857143	0.652174
7 (exaggeration,minimization)	0.592593	0.533333	0.561404
Accuracy	0.637540	0.637540	0.637540
Macro Avg	0.650191	0.638761	0.636598
Weighted Avg	0.653929	0.637540	0.638213

Table 14: Model performance for BERT in the second task

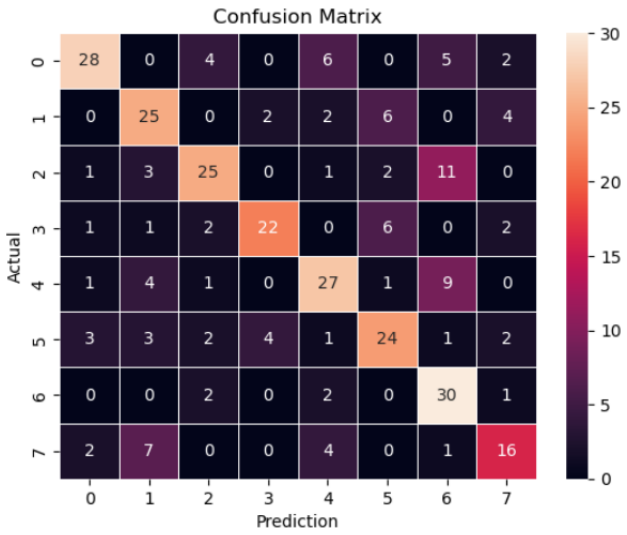

Figure 6: Confusion matrix for BERT in the second task

Conclusion

People's minds have frequently been changed by different propaganda techniques, especially after the development of technology. People expose vast amounts of propaganda content without notice. However, it can lead to serious consequences due to people's actions. Different techniques such as LLMs (Large Language Models) have been used in recent years thanks to the great progress in NLP (Natural Language Processing). I divided the main task into 2 tasks detection of propaganda and classification of it. Also, 2 different approaches were used. The first one was Bag of Words (BoW) with a Multinomial Naïve Bayes Classifier. The second is BERT (Bidirectional Encoder Representations from Transformers). While BoW was more successful in detecting propaganda (Task 1), BERT overcame it better in propaganda classification (Task 2).

Future Work

Bag-of-Words

A different classifier can be used and compared with my findings. Also, different tokenizers and vectorizers can be used. Because I had less data, sparsity was the main problem, despite adding an alpha value. Hence, each sparse data had the same importance as other sparse ones. Instead, distributional probability can be used. However, it won't solve the problem exactly.

BERT

Different epoch-batch-learning rate values can be tried for the base-uncased model as well as the large model with and without cased. As for the same models I used, the epochs could be taken higher by balancing validation loss through a hyperparameter experiment using another external dataset to see the existence of overfitting. Also, different optimizers can be tried.

Overall

The snippets which have less than a specific number of characters or words can be excluded to catch the context in a better way.

REFERENCES

Anders, M. (2024, February 28). Fake News Detection | European Data Protection Supervisor. Www.edps.europa.eu. https://www.edps.europa.eu/press-publications/publications/techsonar/fake-news-detection_en

Barnard, J. (2024, January 23). What are Word Embeddings? | IBM. Www.ibm.com. https://www.ibm.com/topics/word-embeddings

DAlessio, F. (2021, November 3). Computational Propaganda: Challenges and Responses. E-International Relations. https://www.e-ir.info/2021/11/03/computational-propaganda-challenges-and-responses/

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Khanday, A. M. U. D., Khan, Q. R., & Rabani, S. T. (2021). Identifying propaganda from online social networks during COVID-19 using machine learning techniques. *International Journal of Information Technology*, *13*(1), 115-122.

Murel PhD., J., & Kavlakoglu, E. (2024, January 19). What is bag of words? | IBM. Www.ibm.com. https://www.ibm.com/topics/bag-of-words

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(85), 2825–2830. https://scikit-learn.org/stable/modules/naive_bayes.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(85), 2825–2830. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12(85), 2825–2830. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

Smith, B. L. (2024). propaganda. In Encyclopædia Britannica. https://www.britannica.com/topic/propaganda

Sprenkamp, K., Jones, D. G., & Zavolokina, L. (2023). Large Language Models for Propaganda Detection. *arXiv preprint arXiv:2310.06422*.