Investigating Propaganda Detection and Classification

Candidate 246518
Advanced Natural Language Engineering G5114
May 2024

1 Introduction & Problem Outline

1.1 What is propaganda?

The Institute for Propaganda Analysis defines propaganda as:

> *(The) conscious attempt to influence others irrespective of whether the appeal is made to the intellect or to the emotions; whether the propagandist is sincere or insincere, whether his motives are selfish or altruistic; whether his power as a propagandist depends upon conscious calculation or comes from some unconscious force.[7]*

In other words, propaganda is an attempt to influence others to some predetermined end by appealing to their thought[7], typically arising when there is some form of conflict of opinion in society[3].

In today's digital world, the rapid dissemination of news occurs at an unmanageable pace via online outlets, notably through social media. These platforms allow for the distribution of decentralised content that undergoes little scrutiny or analysis. The pursuit of clicks, views, and followers serves as a reason for these platforms to heighten emotional engagement, whether through genuine articles or through sensationalised content, commonly referred to as "rage bait,"[5].

The decentralised nature of online channels makes it difficult to provide scrutiny to *all* content due to sheer volume. More subtle propaganda methods must be employed by those who are likely to face increased scrutiny from the public, such as traditional news platforms, governments, and political figures[1]. These actors aim to use propaganda in an attempt to influence their audiences through nuanced emotional appeal, *without* signalling to an audience that they are essentially being manipulated through carefully picked language.

This highlights the core difficulty with propaganda detection, as it is most effective when unnoticed. [10]

1.2 Task Setup

In this investigation, I aim to create, evaluate and compare two systems which address two tasks concerned with the detection and classification of Propaganda:

Task 1 - Span Identification: Given a plain-text sentence, identify if the sentence contains a propaganda technique. (A binary classification problem)

Task 2 - Technique Classification: Given a span of propaganda-containing text, identify the propaganda technique used in that span. (A multi-class classification problem)

Participants of Herbelot et al.'s 2020 International Workshop on Semantic Evaluation treat these two tasks as steps of a larger propaganda detection pipeline. However, for this investigation, these problems will be treat as two separate tasks, with slight differences in the provided dataset and task specification. Nevertheless, the overarching problem remains essentially the same: fine-grained analysis of propaganda in news articles.

[1]Although the radicalisation of political figures seems to have effectively thrown this out of the window in preference of "Fake News", illustrated by Donald Trump's Twitter.

1.3 Propaganda Techniques and Dataset used

There are eight propaganda techniques used for this investigation. They are derived from the 18 techniques proposed by Da San Martino et al. in their 2019 paper[1] on the same topic. Table 1 showcases these eight propaganda classes, along with concise definitions. Additionally, a ninth class labelled "Not Propaganda" is included, reserved for sentences without any propaganda elements.

Techniques	**Definition**
Flag Waving	Appealing to patriotism.
AtFP	Planting fear against other alternatives.
CO	Assuming a simple cause for an outcome.
Doubt	Questioning credibility.
EM	Exaggerating or minimising something.
Loaded Language	Appealing to emotions or stereotypes.
Name Calling	Giving an object an insulting label.
Repetition	Injecting the same message over and over.
Not Propaganda	Regular text with no propaganda

Table 1: List of propaganda techniques and brief definitions. EM: Exaggeration/Minimisation, LL: Loaded Language, CO: Casual Oversimplification, AtFP: Appeal to fear & Prejudice. Adapted from [12]

1.4 Dataset and Dataset Properties

The Above propaganda classes were directly sourced from the dataset provided alongside the assignment brief.

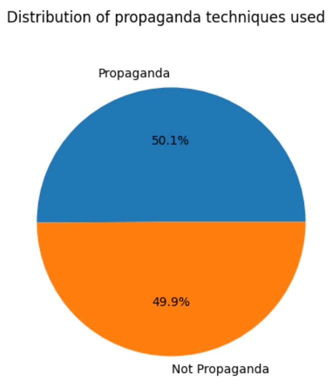
This dataset consists of 2994[2] spans of text, with some surrounding context, split into an 80:20 train/test split. Each span is delineated by a `<BOS>` (Beginning of Span) and `<EOS>` (End of Span) token to separate it from its context. These spans are accompanied by one of nine propaganda technique labels from Table 1, describing the type (or absence) of propaganda in the span.

The dataset has been duplicated and modified to prepare the data for the two different tasks:

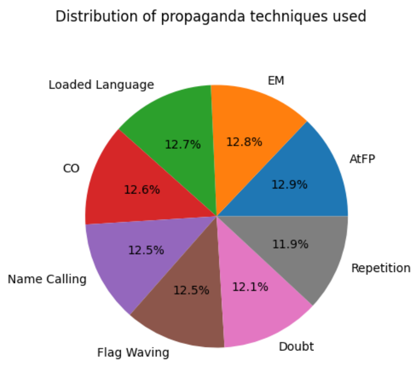
1. Task 1's dataset has had all of the `<EOS>` and `<BOS>` tags removed, and its unique propaganda labels replaced by a single `propaganda` class, creating a dataset with 2 classes.

2. Task 2's dataset has had all of the `not_propaganda` instances removed, and each span's surrounding context, `<BOS>` and `<EOS>` tags removed, creating a snippet-only dataset with 8 propaganda classes.

The class distributions of these two task-specific datasets can be found in Figure 1's pie charts. The balanced distribution of these two datasets minimises concerns related to class imbalance, simplifying the training and evaluation process by eliminating the need for any downsampling or upweighting techniques.



(a) Distribution of task 1 dataset's classes.



(b) Distribution of task 2 dataset's different propaganda classes.

Figure 1: Pie charts illustrating the distribution of the two task datasets' classes.

[2]It is worth noting that the dataset should ideally contain 3200 spans, but due to a formatting error, the actual count is 2994. The incorrectly formatted data points have been removed from the dataset.

While the class distributions themselves may be uniform, the properties of the sentences and spans within the classes are not. There is variation between the word counts[3] of the classes within each task's datasets. Table 2 and 3 illustrate these differences.

Label	**Min**	**25%**	**Mean**	**75%**	**Max**
Not Propaganda (NP)	8.0	19.0	30.07	38.0	118.0
Propaganda (PP)	8.0	26.0	40.30	49.0	167.0

Table 2: Summary statistics of task 1's sentence dataset.

Label	**Min**	**25%**	**Mean**	**75%**	**Max**
Flag Waving	1.0	2.0	12.16	16.5	79.0
Loaded Language	1.0	1.0	3.72	5.0	30.0
Doubt	1.0	9.0	22.71	29.0	160.0
Name Calling	1.0	2.0	4.82	6.0	24.0
AtFP	1.0	8.0	19.01	28.0	70.0
Repetition	1.0	1.0	3.15	4.0	39.0
CO	3.0	14.0	23.80	30.0	90.0
EM	1.0	4.0	8.03	11.0	47.0

Table 3: Summary statistics of task 2's span dataset.

All of these observations may prove helpful in explaining any trends or observations in later experiment findings.

2 Method

As stated previously, two systems are being built, evaluated and compared to complete both tasks.

2.1 System 1

System 1's architecture uses multiple N-gram models to model the language used in the different classes within the datasets. For task 1, this consists of the '`propaganda`' (PP) and '`not_propaganda`' (NP) classes. For task 2, this consists of the eight propaganda classes found in Table 1.

For each task, multiple classifiers are trained with preprocessed training data, and classifications are made by calculating the log probability[4] of a sentence/span through each sub-model; the model with the highest log probability is then chosen, classifying the sentence/span.

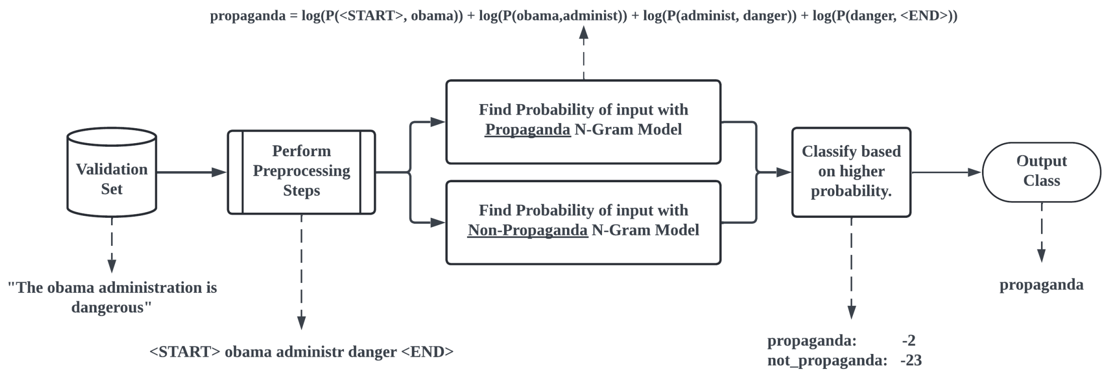

propaganda = log(P(<START>, obama)) + log(P(obama,administ)) + log(P(administ, danger)) + log(P(danger, <END>))
Validation Set
Perform Preprocessing Steps
Find Probability of input with Propaganda N-Gram Model
Find Probability of input with Non-Propaganda N-Gram Model
Classify based on higher probability.
Output Class
"The obama administration is dangerous"
<START> obama administr danger <END>
propaganda: -2
not_propaganda: -23
propaganda

Figure 2: The steps the Multi-N-gram model takes to classify an inputted sentence in Task 1, with examples of the inner representation at each step. This method is the same for task 2, but with the eight propaganda classes.

As this system models language directly through the probability of words (rather than through matrices or tensors, like in more advanced word representations) it is important that a sequence of

[3]Using NLTK's WordTokenizer.

[4]Log probability was chosen for this to minimise any floating-point errors, as the probabilities for this model start to get very small, very quickly.

preprocessing steps are undertaken to normalise the training data, decrease the model's sparsity, and remove semantically meaningless words to ultimately increase the training data's 'semantic density'. I use the following preprocessing steps in the multi-n-gram system:

- Lowercase all of the text for normalisation.
- Tokenise the words into a list using NLTK's WordTokenizer.
- Remove any non-alphanumeric characters as they're semantically insignificant.
- Remove any stop-words as they're also semantically insignificant.
- Stem all words to normalise tense, plurality, and other grammatical variations.

Because of the dataset's large vocabulary, there will likely be many unseen n-grams within the dataset. To counteract this, I've employed a fairly elemental smoothing technique: If an n-gram is not found, a 'default value' will instead be returned. I aim to test different values of this default value for both a unigram and bigram model to optimise my system.

Despite my efforts to compress the model's vocabulary and introduce a pseudo-smoothing technique, I anticipate that this system's performance will be mediocre; It will most likely perform better on task 1 than task 2 due to the longer input length of the sentences when compared to the propaganda spans, with spartsity on these sentences and spans being a concern.

2.2 System 2

System 2 uses a fine-tuned Large Language Model to classify inputted sentences/spans. A pre-trained GPT2Model from HuggingFace is used with two fine tuned classification heads for the two separate tasks. Code for this system was adapted and from George Mihaila's online guide "GPT2 For Text Classification using Hugging Face Transformers"[4].

This system originally aimed to use HuggingFace's BERTForSequenceClassification, but a desire to understand the inner workings of the popular GPT architecture used in OpenAI's impressive ChatGPT led me to instead pick GPT2 as the base transformer model for this task.

GPT2 is an auto-regressive Large Language Model for general purpose, unidirectional text generation. It is built to essentially 'predict the next word' in a sentence[6].

However, by exploiting GPT2's inner representation of text (used to generate text from previous sentence content), one can essentially 'hijack' the final hidden-layer representation (logits) of an inputted sentence to gather a compressed representation of the whole sentence. This representation is then fed into a separately trained head which classifies the logits using a trainable neural network layer, turning GPT2 into a classifier.

I originally aimed to attach a custom built classification head on to a 'headless' GPT2 Model provided by HuggingFace. Research was started on this but little progress was made, as I found HuggingFace's documentation for its headless GPT2 model rather vague, requiring time-consuming exploration of its source code to understand how it worked.

While this original, custom head aim unfortunately didn't result in any models being built, it did give me a better understanding of how GPT2's tokeniser and GPT2 itself worked. I opted to instead use HuggingFace's GPT2ForSequenceClassification, which adds a pooling and linear layer to the top of GPT2, allowing it to use the model's final logits to classify an input sequence.

HuggingFace's pre-trained GPT2Tokenizer is used to handle the conversion of raw input sentences/spans to a GPT2-friendly input ID Tensor + attention mask format, with left padding and truncation being used to ensure all input sequences are the same tensor size. No other preprocessing steps are taken, as GPT2's Byte Pair Encoding system allows for the model to operate without any further preprocessing steps[8].

Figure 3 illustrates the steps this system uses to classify a task, from raw text to a classification.

The training process for the GPT2 based models involves training the classification head over a set of 4 epochs with an AdamW Optimiser. This number and optimiser are taken directly from Mihalia's guide, and are recommended by HuggingFace for model fine-tuning[2]. Training batch

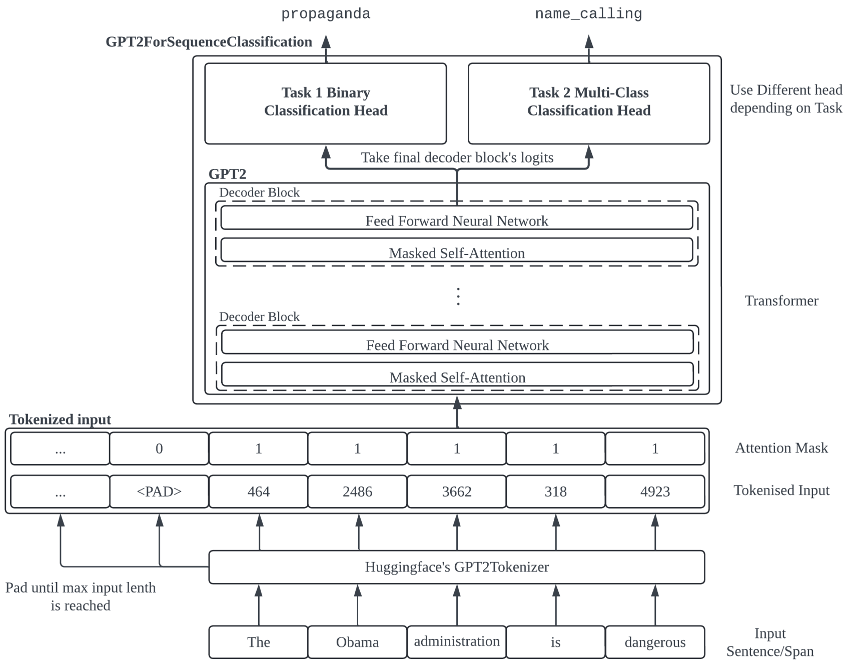

propaganda
name_calling
GPT2ForSequenceClassification
Task 1 Binary Classification Head
Task 2 Multi-Class Classification Head
Use Different head depending on Task
Take final decoder block's logits
GPT2
Decoder Block
Feed Forward Neural Network
Masked Self-Attention
Transformer
Decoder Block
Feed Forward Neural Network
Masked Self-Attention
Tokenized input
... 0 1 1 1 1 1
Attention Mask
... <PAD> 464 2486 3662 318 4923
Tokenised Input
Huggingface's GPT2Tokenizer
Pad until max input lenth is reached
The Obama administration is dangerous
Input Sentence/Span

Figure 3: The steps used to make classifications using GPT2 and a classification head. Adapted from [11].

sizes were limited to 32, due to memory and training time restrictions.

Similarly to System 1's n-gram models, a set hyper-parameter tests will be performed to find an optimal learning rate and maximum input length, with all other parameters being fixed.

3 Experiment Results & Analysis

3.1 Evaluation Metrics

For both tasks, the systems are evaluated using standard evaluation metrics, including *accuracy*, *precision*, *recall* and $F_1 score$.

Normalised confusion matrices, displaying recall, are used to further show any differences between true labels and predictions, possibly showing any unusual model behaviour.

3.2 Analysis of Multi-N-Gram Models

3.2.1 Task 1

Model 1's mini-experiments were carried out in a grid-search fashion, calculating F1-scores of a set of 11 equally spaced 'default probability' values ranging from 0.0000000001[5] to 0.001 over a unigram and bigram model.

[5]This would be 0, but using a value of 0 breaks the log function.

Default Value	Unigram	Bigram
0.0000000001	**0.672**	0.638
0.0001	0.636	0.636
0.0002	0.601	0.639
0.0003	0.585	0.638
0.0004	0.562	0.643
0.0005	0.540	0.639
0.0006	0.526	0.638
0.0007	0.521	0.639
0.0008	0.519	0.634
0.0009	0.513	0.631
0.001	0.504	0.631

Table 4: F1-scores for different default probability values for both a unigram and bigram model.

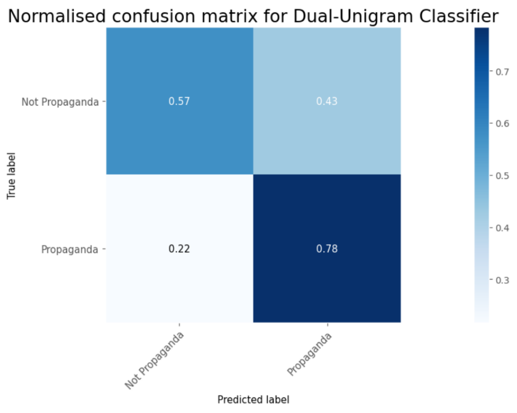


Figure 4: A Normalised Confusion Matrix of the best performing multi-n-gram classifier for task 1.

In this grid search, a unigram model with default value 0.0000000001 performed best, yielding an F1-score of 0.672.

Figure 4 illustrates this model's confusion matrix. We can see that it accurately classifies sentences containing propaganda, but struggles to correctly classify sentences without propaganda.

In the mini-experiment we can observe that as the default value increased, the unigram's performance declined, partly due to the heightened probability of encountering unseen words, which skewed the classifier towards classifying more sentences as NP.

The shorter average sentence length of NP sentences, coupled with a higher percentage of unseen words in the NP test set compared to the PP set, led the NP n-gram model to more frequently output a higher sentence probability, resulting in an increased number of NP classifications for the higher 'default value' unigram models.

Interestingly, the bigram models did not exhibit this trend, showing little variation despite changes in default values. This could be attributed to the large number of unseen bigrams encountered by both the NP and PP models in the validation data, at 75% and 83% respectively.

I believe that the cumulative probability of the sheer number of unseen bigrams may have made the contribution of previously observed bigrams negligible, effectively creating a prediction system based on the number of words in a sentence rather than any observed properties of the words themselves.

3.2.2 Task 2

Task two served to only further illustrate the problems caused by the sparsity found in task 1. Identical parameters to task 1 were explored in a grid search, the results of which, alongside the confusion matrix of the best performing model, are found below.

Default Value	Unigram	Bigram
0.0000000001	0.352	0.163
0.0001	**0.368**	0.163
0.0002	0.364	0.163
0.0003	0.358	0.163
0.0004	0.346	0.163
0.0005	0.345	0.163
0.0006	0.349	0.163
0.0007	0.335	0.163
0.0008	0.332	0.163
0.0009	0.313	0.163
0.001	0.313	0.163

Table 5: F1-scores for different default probability values for both a unigram and bigram model, for task 2.

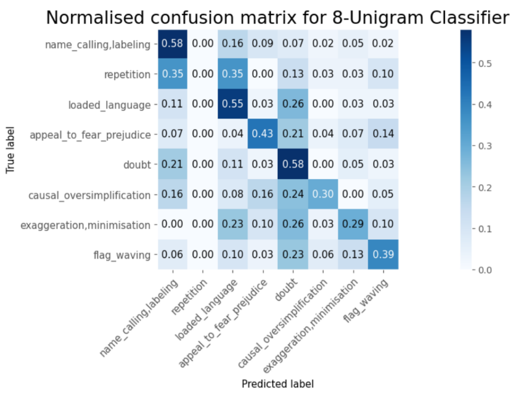


Figure 5: A Normalised Confusion Matrix of the best performing multi-n-gram classifier for task 2.

Task 2's results unfortunately follow a similar trend to task 1's disappointing results, but on a more severe scale over the 8 classes. The best performing model is able to semi-successfully classify some classes with its unigram classifiers, achieving an F-1 score of 0.368.

The classifier performed well when classifying the Name calling, doubt and loaded language, but struggled with others, and completely neglecting the repetition class.

Attempts were made to find the cause for this behaviour, but no full explanations could be deducted. My initial theory while investigating this trend was that, similar to task 1, classes with shorter span lengths and higher rates of unseen n-grams were being predicted more, but comparing these figures with the system's class predictions shown no clear correlation for any of the classes, leaving me without a full explanation as to why the model performed the way it did.

Fortunately, the cause of the bigram model's performance was much clearer. The confusion matrix for the bigram model shows a clear prediction imbalance towards doubt, being predicted for 79% of all spans, and an avoidance of repetition with zero predictions. A combination of average span length and the unseen bigram rate of the doubt class meant that the computed probability would nearly always be largest for the doubt class. I believe the inverse of this happened with the repetition class with its short span length, leading to extremely low probabilities.

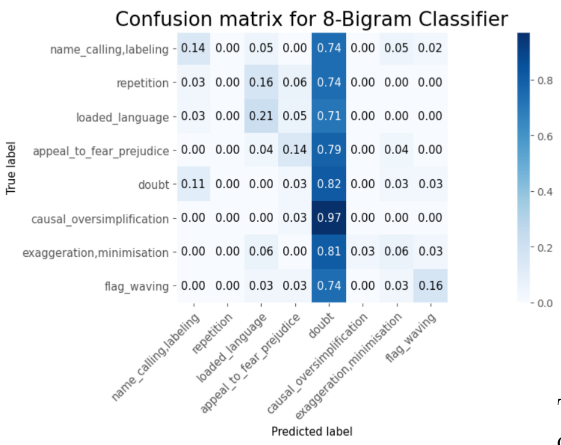


Figure 6: Confusion matrix of the multi-bigram classifier

Class	**% of unseen bigrams**
NC	97.5%
Repetition	97.1%
LL	98.7%
AtFP	94.4%
Doubt	95.5%
CO	100%
EM	96.0%
FW	95.6%

Table 6: Propaganda classes alongside the percentage of unseen bigrams.

This observation helps to explain the lack of variation between the bigram models when changing the default value, as the class probabilities were effectively based off of a span's length and its number of unseen bigrams, which would remain constant throughout the various tests.

Overall, the multi-n-gram model's performance on both tasks was severely limited by the data sparsity and the substandard smoothing technique used.

3.3 Analysis of Fine-Tuned GPT2

3.3.1 Task 1

Initial training of a baseline model a maximum sequence length 160 and learning rate $5e-5$ led to an impressive F1-score of 0.88 for the GPT2 based classifier. Performing a search among a set of 15 hyper-parameter combinations yielded only a slightly higher F1-score of 0.89, giving an optimal learning rate of $1e-4$ and a max sequence length of 150.

With this optimisation, the model performed very well. The confusion matrix in Figure 7 illustrates the model's good performance in the binary classification task, with a very slight preference to classify sentences as containing propaganda, rather than not containing propaganda.

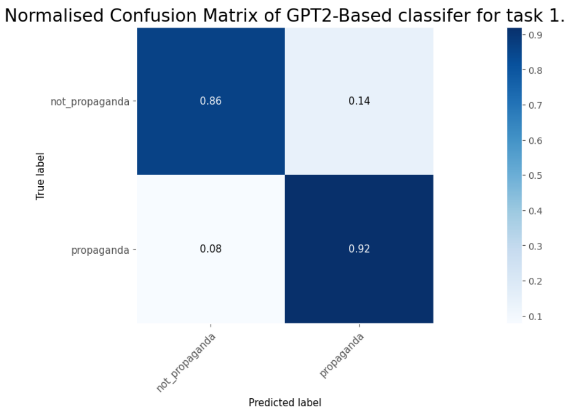


Figure 7: The normalised confusion matrix of the GPT2-Based classifier on task 1.

Among the incorrectly classified sentences, there were a large amount which were fully wrapped in the `<EOS>` and `<BOS>` tokens. This may show a pitfall of the otherwise accurate GPT system, illustrating the importance of a span's surrounding context during classification for the model.

3.3.2 Task 2

Unfortunately, due to GPU usage and time restrictions, I was unable to perform hyper-parameter optimisation for the second task's classification head. The optimal learning rate from task 1 was carried over to task 2, and a maximum length of 160 was chosen to accommodate to the longest span of propaganda text.

Despite this lack of hypertuning, the model performed well. Classifying sentences over the eight classes with a macro F1 Score of 0.59, with strong performance on the flag_waving class (0.80 F1), and weak performance on the repetition and EM classes (0.50 and 0.43 F1 respectively).

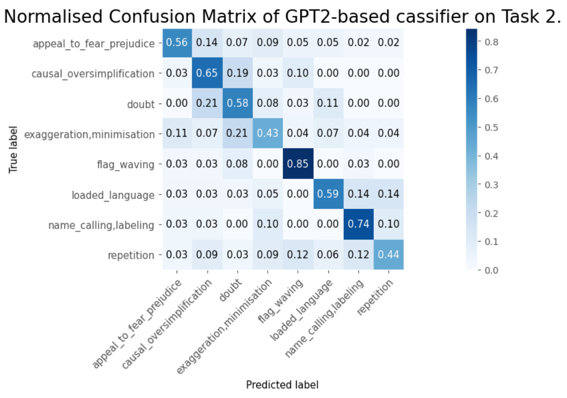


Figure 8: The normalised confusion matrix of the GPT2-Based classifier on task 2.

I believe the high performance of the flag_waving class may be due to the clear nationalistic vocabulary found in its spans. When removing stopwords, the most common words in the class were "America/American/Americans", "people", "country" and "nation", making most occurrences of the class rather overt.

For the repetition class, I believe the observed low performance can be attributed to the fact that instances repetition would typically be spread out over a large amount of text, rather than in a short average 3.15 word span. Participants of Semeval 2020 also found difficulty with this class, with Raj et al. proposing an interesting architecture using a separate repetition classifier alongside a RoBERTa LLM [9].

Unfortunately, I am not sure of root the cause of the low performance of the EM class. Initially, I thought that it could be due to its short average span length. However, the LL and NC classes had a much shorter span length and performed better, which leads me to believe this isn't the cause.

Model 2's better overall performance in task 1 compared to task 2 can likely be attributed to the richer contextual information available in task 1's longer sentences. Given GPT2's reliance on previous context to predict subsequent words, the absence of such contextual information in the isolated propaganda plans in task 2 logically explains its poorer performance.

3.4 General Observations

The observed difference between System 1 and System 2's performance comes with little surprise. The comparison between a simple N-Gram model and an industry leading Large Language Model feels somewhat unfair, given GPT2's usage of self attention, context, and its huge model training dataset of 8 million webpage content[6], compared to the minuscule 2700 sentences used for the n-gram model.

Model	F1-Score
Task 1 Unigram Model	0.672
Task 1 Bigram Model	0.0.639
Task 1 GPT Model	0.890
Task 2 Unigram Model	0.368
Task 2 Bigram Model	0.163
Task 2 GPT Model	0.59

Table 7: F1-Scores of System 1 and System 2 for each task

Despite this contrast between the two, both models encountered similar challenges with the repetition class, likely due to its short length. While the N-Gram model's struggle can be party blamed by smoothing and sparsity issues, the complexity of the GPT2 model makes it difficult to provide a full explanation for its performance beyond saying that classification of repetition may require more than 3 words of context.

4 Further Work

Model 1's disappointing performance means it has several avenues for improvement. The implementation of a more robust smoothing technique would help to create more accurate probabilities for any inputted sentences.

Implementing additional preprocessing steps an attempt to reduce model sparsity could also be beneficial, given the number of unseen n-grams in the validation dataset. One approach could involve normalising words by using synonyms, rather than through stemming. Words could be substituted into their most general form, creating 'clusters' of words that could reduce sparsity, swapping words like "huge" and "giant" with a generalised "big".

For model 2, completing the work started on a more complex classification head (rather than the default linear huggingface head) could lead to improved results. Such a network could have potential improvements through more training epochs, with additional dropout layers to reduce the chance of over-fitting.

Focusing on the pain-points of model 2, introducing an additional system to give the length of a span more weight in the classification task could help to alleviate the issues found with repetition's shorter span lengths, which could be implemented using an additional input into a custom head's input layer.

5 Conclusion

This paper has conducted an assessment on a novel multi-n-gram probability model and GPT2 based classification model over a propaganda detection and technique classificaiton task. The GPT2 based classification model massively outperformed the multi-n-gram due to its increased complexity and better contextual representation, with the multi-n-gram model performing disappointingly due to oversights while implementing smoothing, and large dataset sparsity. High level examples of further improvements have been discussed, with the potential to improve both models performance.

References

[1] Giovanni Da San Martino et al. "Fine-Grained Analysis of Propaganda in News Article". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5636–5646. DOI: `10.18653/v1/D19-1565`. URL: `https://aclanthology.org/D19-1565`.

[2] *fine-tune a pretrained model.* URL: `https://huggingface.co/docs/transformers/en/training`.

[3] Frederick E. Lumley. *The Propaganda Menace.* Appleton, 1933, p. 157.

[4] George Mihaila. *GPT2 For Text Classification using Hugging Face Transformers.* Nov. 2020.

[5] Parker Molloy. *How right-wing media embrace social media-generated rage bait to drive website traffic.* Mar. 2019. URL: `https://www.mediamatters.org/national-review/how-right-wing-media-embrace-social-media-generated-rage-bait-drive-website-traffic`.

[6] *OpenAI gpt2 documentation.* URL: `https://huggingface.co/docs/transformers/en/model_doc/gpt2`.

[7] Institute for Propaganda Analysis. *Propaganda, How to Recognise It and Deal with It.* Institute for Propaganda Analysis, 1938, pp. 2–3.

[8] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: 2019. URL: `https://api.semanticscholar.org/CorpusID:160025533`.

[9] Mayank Raj et al. "Solomon at SemEval-2020 Task 11: Ensemble Architecture for Fine-Tuned Propaganda Detection in News Articles". In: *Proceedings of the Fourteenth Workshop on Semantic Evaluation.* Ed. by Aurelie Herbelot et al. Barcelona (online): International Committee for Computational Linguistics, Dec. 2020, pp. 1802–1807. DOI: `10.18653/v1/2020.semeval-1.236`. URL: `https://aclanthology.org/2020.semeval-1.236`.

[10] Mira Sotirovic. *Propaganda and Journalism.* Aug. 2019. DOI: `10.1093/acrefore/9780190228613.013.864`. URL: `http://dx.doi.org/10.1093/acrefore/9780190228613.013.864`.

[11] Cameron R. Wolfe. *Language Models: GPT and GPT-2.* Nov. 2021. URL: `https://cameronrwolfe.substack.com/p/language-models-gpt-and-gpt-2`.

[12] Seunghak Yu et al. "Interpretable Propaganda Detection in News Articles". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. Ed. by Ruslan Mitkov and Galia Angelova. Held Online: INCOMA Ltd., Sept. 2021, pp. 1597–1605. URL: `https://aclanthology.org/2021.ranlp-1.179`.

6 Appendix

Code used this report can be found in the below repository.
`https://GitHub.com/Dents6679/Propaganda-Detector-Notebooks`