Comparing the Performance of Transformer Models with Machine Learning Models for Propaganda Span Identification and Classification

Matthew A. Ford

Department of Informatics, University of Sussex, BN1 9QH, UK

mf472@sussex.ac.uk

Abstract – This paper explores the application of machine-learning (ML), and transformer pipelines for the task of propaganda detection and classification. Here, DistilBERT is compared to simpler models – SVC and CRF – that are applicable to the task with the main goal being to assess if using transformer models has a greater payoff that warrants the computational time and implementation complexity that comes alongside. By analysing the results given, it is shown whether transformer models always perform better, and whether they are more, or less applicable to the task under varying conditions.

1. Introduction

Propaganda has increasing become a pressing issue in society, with much of its rise attributed to the use of the internet and social-media. These spaces enable the fast spread of persuasive content that is commonly designed to manipulate opinions. Propaganda techniques exploit biases, emotion, and linguistic subtleties. This makes them difficult to detect using convolutional methods. (Barron-Cedeno, et al., 2019 ; Rashkin, et al. 2017)

The most prevalent techniques include Appeal to Fear and Prejudice, Causal Oversimplification, Doubt, Exaggeration / Minimization, Flag Waving, Loaded Language, Name Calling/Labelling, and Repetition. (Da San Martino et al., 2019).

Propaganda is mainly used as a persuasive mechanism to try and sway the opinion of a group of people. The urgency to develop methods for detecting propaganda is becoming larger, but there are many challenges posed by subtle manipulation techniques in text. (Sprenkamp, K., 2023). This forms the basis of the research presented in this report.

This research has two main parts, these being:

1. The classification of propaganda techniques within a sentence where the relevant text span has been identified.

2. The sequential identification of both the text span and propaganda technique in any given sentence.

For each section, two models will be created, starting with a simple SVC, compared to DistilBERT for classification before moving into pipeline creation by sequencing these models to make a pure, machine-learning pipeline, and a two-step transformer pipeline.

In this report, first, the methods will be discussed, including the decisions made for each model's implementation and the process of deciding their hyperparameters.

Following this, the results will be shown for each task, and an analysis will be provided to outline the key aspects and comparisons to be made.

Finally, there will be a discussion into the viability of the experiment and results, any limitations that were found, and some thoughts on how this experiment opens opportunities for further research.

2. Methods

This section explains the design choices made in the implementation of the models, alongside explaining the process of hyperparameter optimisation.

2.1 | Data Analysis & Preprocessing

Data preprocessing is essential for both span identification and classification problems. This involves manipulating the data to extract key components, creating word embeddings and tagging the data to show features such as part-of-speech (POS), inside-outside and beginning (IOB), and named-entity recognition (NER). With these techniques, the different models can be trained on the feature data to recognise the semantic content and syntactic structure of the text, improving the ability to identify and classify propaganda span.

However, firstly, the data had to be re-balanced. *Figure 1* shows the original class distributions given by the dataset. Here, there was a disproportionate amount of *'not-propaganda'* labelled data. For both tasks, the data labelled *'not-propaganda'* was removed entirely. This is because it was not seen as useful to the model when identifying a span that is known to have propaganda, and therefore even a guess of a technique woold be more accurate than identifying it as not propaganda.

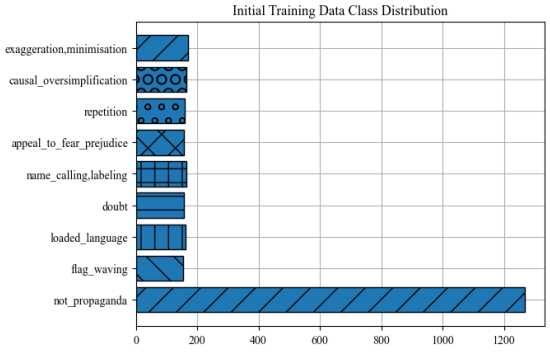

Figure 1: The Original Class Distribution for Each Propaganda Technique

Also, in terms of the span identification, including not-propaganda data is seen as uninformative to the model as its goal is to learn the features that make propaganda present in text. If there was truly no propaganda, then a span should not be identified at all.

Having a random placement of the <*BOS*> and <*EOS*> labels make the data much noisier and therefore reduces its usefulness.

The chosen method of rebalancing the other distributions was to make them all the same length. This was done with respect to the class that contained the least data. *Figure 2* shows the result of this.

Figure 2: The Class Distributions for Each Propaganda Technique after Rebalancing.

To generate the POS and NER representations, the raw text was processed using *spaCy*. This library was chosen over the alternatives such as *nltk* due to its fast-processing speed and accuracy. *SpaCy* can preserve the contextual cues – such as capitalization for proper nouns – as it works on the unaltered text. This was of importance as the data both inside, and surrounding the spans can hold contextual clues and signals that are relevant to propaganda detection. After these representations were generated, the data was lowercased for consistency in the following processing steps.

For all numerical values in the data, a <*NUM*> token was used as a replacement. This is due to numbers not holding any weight in identifying propaganda techniques. This process is more-so based on the linguistic techniques and structure of the spans rather than quantitative data. By replacing these values, the noise in the data was also reduced, but dates, times, etc. were still recognised in the NER tags.

Each propaganda-containing sentence in the training data had an already identified span, using <*BOS*> (beginning-of-span) and <*EOS*> (end-of-span) tokens.

These identifiers were crucial for training span identification models to recognise how propaganda spans begin and end. Although these were not deemed as useful in classification.

To streamline the training of each model and to avoid having too much redundancy, all the preprocessing was performed at once. This generated all relevant representations and eliminated the need to create separate datasets for each model. This resulted in a singular, definitive dataset which allowed for the selective use of feature representations dependent on the task. Therefore, making the workflow more efficient and encouraging modularisation. *Table 1* shows the layout of this dataset.

Dataset
Raw Data
Raw Data (No <BOS> or <EOS> Tokens)
Isolated Span
Part-of-Speech (POS)
Named-Entity Recognition (NER)
Inside-Outside-Beginning (IOB)

Table 1: Full Pre-Processed Dataset Contents

2.2 | Task 1 – Propaganda Classification

For the task of propaganda classification, the spans were first isolated from their sentences. This prevents any unnecessary tokens being used during training, ensuring that the model only learns from tokens that are part of the actual propaganda spans. This helps in narrowing the scope of information provided during training and should help to improve the model's ability to identify propaganda techniques.

Stop words were intentionally retained in the data, although most language models remove them due to the large frequency overbearing truly meaningful words, this task must be based on the entire context, rather than individual tokens.

Therefore, stop words can carry an informative value for certain propaganda techniques – such as loaded language, repetition, and doubt – where subtlety in phrasing and stop words can contribute to the effect.

2.2.1 | SVC Span Classifier

The support vector classifier (SVC) draws boundaries between data points (tokens in this case) to give a hyperplane to best separate the classes. Then, when an unseen span is inputted, the tokens are represented as new points, and the area of which the centroid of those points is located decides the appropriate label.

Figure 3 shows the classification pipeline. While the task of classification is straightforward, the pipeline itself has multiple stages to extract features from the data before the final prediction is made. Semantic and structural features including word embeddings, POS, and NER tags are combined into a single vector to be passed into the model. This helps to capture both the meaning and structure of the span that helps the model make well-informed decisions.

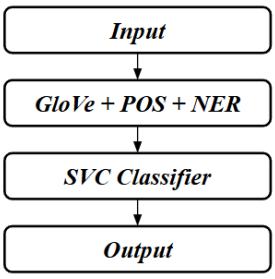

Figure 3: Pipeline Diagram for an SVC Propaganda Classifier.

To represent each token in the span numerically, GloVe embeddings were used. These embeddings come from a pre-trained set and allow for the semantic similarity of words to be captured. This allows similar words to be positioned in nearby vector space proximity to each other. The choice to use GloVe was made due to its ability to provide meaningful representations of words. Unlike other methods – such as one-hot encoding – GloVe embeddings capture the contextual relationships which, as discussed earlier, is valuable for identifying propaganda.

By using these embeddings to feed into the SVC classifier, the ability for the model to differentiate between propaganda techniques based on language and tone is improved.

Using these GloVe embeddings alongside NER and POS representations of the span, a feature vector was constructed as the final representation to be fed into the model.

To maximise the performance of the SVC model, several hyperparameters were chosen for optimisation. These included:

- The regularisation parameter (*C*)
- The kernel type
- The gamma
- The degree (for polynomial kernels)
- "coef0" (for polynomial and sigmoid kernels)

These parameters allow the flexibility of the decision boundaries to be altered and can help the model generalise to unseen data. Each of these hyperparameters have multiple options for their values, and thousands of combinations. Instead of manually selecting these values or running a parameter sweep, a genetic algorithm (GA) was created to optimise them.

The process of a GA simulates a system of natural selection by evolving a population of hyperparameter combinations. This is done over several generations, selecting and crossing over the best performing sets based off their performance on the validation set. This is also known as the fitness of the set, and in this case:

$$fitness \propto macroF1$$

The macro-f1 score was chosen as the main performance metric as it assigns an equal weight to all classes, making sure unbalanced data is accounted for equally. Also, this metric penalises models that perform well on only dominant classes and therefore encourages models that are better generalised across all propaganda techniques, instead of being exclusively effective at identifying just one.

This approach was chosen for its effectiveness in searching high-dimensional and complex search spaces. This was used for every model discussed in this report to ensure fairness and to give a more accurate comparison

Figure 4 shows the best-found model per-generation, and overall, as the algorithm was run over a course of 100 generations with a population size of 20.

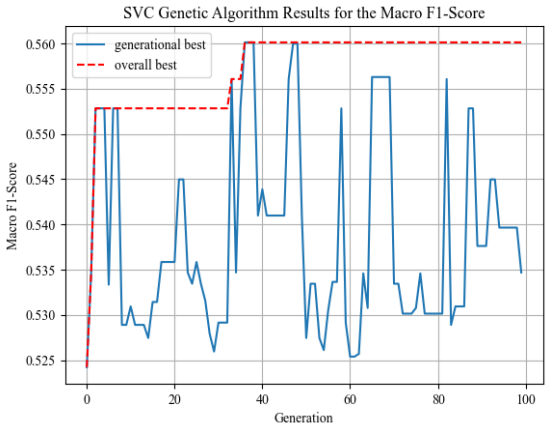

Figure 4: SVC Genetic Algorithm Generational and Overall Highest Fitness Plot.

The resulting hyperparameter values are shown in *Table 2*.

Hyperparameter	Value
'C'	0.01
Kernel	'poly'
Gamma	UNUSED
Degree	3
'Coef0'	0.5

Table 2: Best-Found Hyperparameter Values at the End of the SVC Genetic Algorithm.

The use of a polynomial kernel with degree 3 suggest s that a cubic polynomial kernel is being used. This allows for non-linear, and moderately complex decision boundaries to be drawn between the classes. As there is a lot of crossovers between some techniques, this helps to better distinguish between them.

The low '*C*' value suggests that the model is being heavily regularised, allowing more misclassified points and smoothing the decision boundary, this is also helped by the 'coef0' value.

Therefore, it has been shown that for the SVC classifier, smooth but complex decision boundaries led to the best performance.

The performance of this model, and all subsequent models will be later discussed in *Section 3*. Here, the performance will be analysed and comparisons made.

2.2.2 | DistilBERT Span Classifier

DistilBERT is a transformer-based encoder model that can process input sequences to generate contextual embeddings for each token. It's use of self-attention mechanisms allow for word relationships to be captured at all distances. To prepare the sequence for classification, a [CLS] token is used. This summarises the sequence as the input to the classification layer – similar to how a feature vector was handed to the SVC classifier.

DistilBERT was chosen over other forms of BERT as it is a much smaller (~40%), and faster (~60%) version. This is better for the scope of this experiment as a full transformer model can be expensive both computationally, and regarding time. DistilBERT manages to keep ~97% of BERT's understanding of language and therefore the drop in performance was in this case deemed to have a better payoff as it had a strong balance between performance and computational cost.

Figure 5 shows the pipeline for the DistilBERT classifier. This model has its own encoder and tokenizer, therefore there was no need to use GloVe due to the embeddings being generated within the model from the raw input sequence.

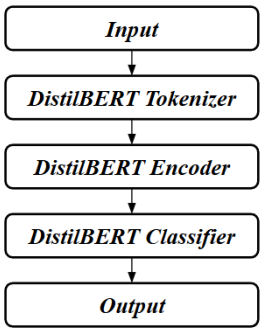

Figure 5: DistilBERT Pipeline for Propaganda Span Classification.

From training and evaluating the DistilBERT model over 50 generations with a population size of 10, a suitable set of hyperparameter values were identified. The best-found solution for each generation and overall can be seen in *Figure 6.*

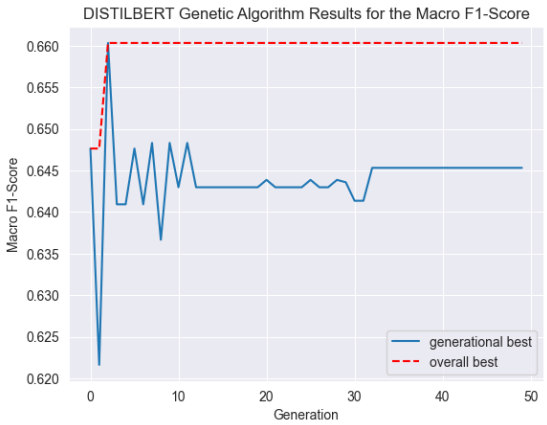

Figure 6: DistilBERT Genetic Algorithm Generational and Overall Highest Fitness Plot.

The resulting hyperparameter values are shown in *Table 3*.

Hyperparameter	Value
Training Epochs	8
Train Batch Size	8
Validation Batch Size	8
Warmup Steps	750
Weight Decay	0.001

Table 3: Best-Found Hyperparameter Values at the End of the DistilBERT Genetic Algorithm.

Having 8 training epochs allows the model to pass over the training data 8 times, adjusting itself every time to become more accurate. Having a high number of training epochs can increase the chance of overfitting and therefore a lower number is better for data as complex as propaganda.

The training and validation batch sizes both ensure the memory usage is smaller and can help with generalisation. Having consistent batch sizes ensures that the metrics given are consistent throughout.

The number of warmup steps determined the period at the start of training where the learning rate increases from zero. This helps to stabilize the training which is essential to transformer models.

Finally, having a small weight decay allows the model to be regularized to discourage large weights in the network. Having a value of 0.001 penalises these weights but is not too restrictive.

2.3 | Task 2 - Propaganda Span Identification

Span identification tasks are much more complex than classification tasks due to the model needing to determine the beginning and end of a sequence in addition to the technique.

Identification models must understand context, semantics, and syntax to outline boundaries of propaganda for token-level predictions to be made.

For this task, the IOB encoded data was modifies to simplify and separate the tasks of identification and classification. Rather than using technique-specific labels such as B-Doubt or I-Repetition, all spans were generically tagged as B-span and I-span. This allowed for each task to then be handled by a dedicated model.

This meant that the complexity of the span identification task was reduced and allowed models to focus entirely on detecting propaganda boundaries. These new pipelines had greater flexibility, and models could be separated for performance analysis, and hyperparameter optimisation with respect to their own subtasks.

2.3.1 | CRF + SVC Pipeline

A conditional random field (CRF) model was chosen for the task of span identification due to its strengths in sequence-labelling tasks (IOB tagging). This is due to their ability to model dependencies between labels, this is crucial for the task of propaganda detection as in most cases, the classification of a token is dependent on its surroundings.

When labelling the sequences of tokens with IOB labels, it is important that the model recognises the dependencies of these tags, where a starting token must be labelled as the beginning, and any relevant tokens proceeding must be labelled as inside the span.

Regular token-level classifiers treat each word individually, whereas CRF models consider the entire sequence. This improves its span identification abilities. The raw sequence, along with its corresponding POS and NER representations are inputted into a feature extractor. Theis extracts token-level features that help the CRF to identify any dependencies. These features are shown in *Table 4*.

Features
Token Suffix Features [-3:] (e.g. “-ing”)
Token Suffix Features [-2:] (e.g. “-es”)
Token POS Label
Token NER Label
Does the Token Begin the Sentence
Does the Token End the Sequence
Token POS Label [±2:] (if applicable)
Token POS Label [±1:] (if applicable)
Token NER Label [±2:] (if applicable)
Token NER Label [±1:] (if applicable)

Table 4: Token-Level Features for the CRF Model

By creating these features for each token, the CRF can learn token-level and contextual patterns for propaganda detection.

The pipeline for this model is shown in *Figure 7*.

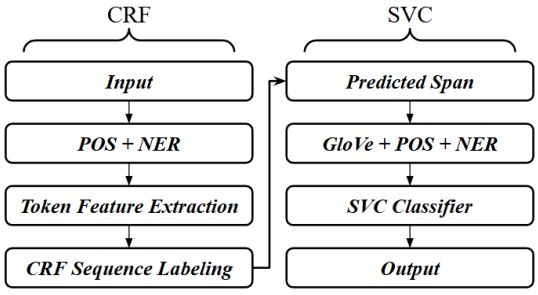

Figure 7: The Pipeline for CRF + SVC Propaganda Identification and Classification.

The use of CRF and SVC models in sequence allows for the benefits of both models to be directed on each subtask. The CRF allows for accurate detection and contextual analysis, whereas the SVC allows for semantic analysis with embeddings for accuracy in classification.

As the SVC model had already been created and optimised in *Section 2.2.1*, it was only necessary to create and optimise the CRF. For this, the ‘sklearn’ library was employed.

The genetic algorithm results shown in *Figure 8* visualise how the model was run over 50 generations of population size 10 and shows the best values found each time.

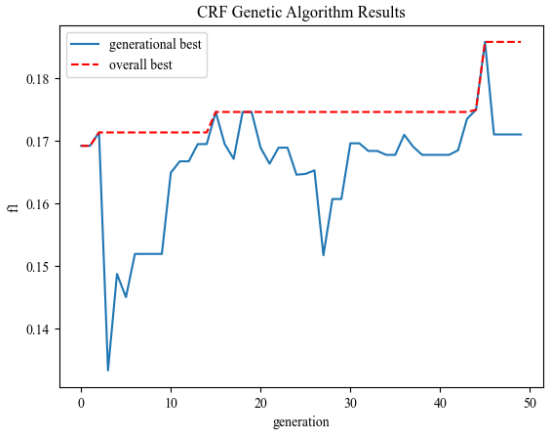

Figure 8: CRF Genetic Algorithm Generational and Overall Highest Fitness Plot.

The resulting parameters are shown in *Table 5*.

Hyperparameter	Value
'c1'	2
'c2'	1
Max Iterations	138
All Possible Transitions	False

Table 5: Best-Found Hyperparameter Values at the End of the CRF Genetic Algorithm.

The values for 'c1' and 'c2' are for regularisation. These help to maintain the model's complexity, whilst avoiding overfitting and encouraging generalization. Having the 'c2' value lower than the 'c1' suggests that here was a stronger sparsity regularisation rather than smoothness.

The max iterations in the model determine how long they should be run for. If convergence is identified before the max iterations are met, the model stops training early. Here, a relatively low number of iterations were chosen. This could be due to the scale of the dataset, meaning more iterations can increase the risk of overfitting.

Having all possible transitions set to false means that the model is constrained to the observed transitions. As patterns are required for IOB labelling, this was important. Having this possibly made the data less noisy and limited the transition rules to keep a defined and robust format.

2.3.2 | DistilBERT Pipeline

For span identification and classification, a two-stage DistilBERT pipeline approach was used. This is illustrated in *Figure 9*.

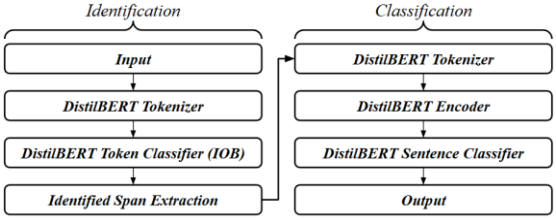

Figure 9: The Two-Stage Pipeline for DistilBERT Propaganda Identification and Classification.

This approach creates a fully transformer-based model that leverages the pre-trained models for both token-level and sequence-level classification. Similar to the CRF model, the token-level DistilBERT classifier is tuned to use the IOB tagging scheme in order to precisely identify propaganda spans within the given sentences.

Having two separate models for the different subtasks enables each to specialize in its given task. This creates a final model that is both accurate in identification and fine-tuned in classification.

Figure 10 shows the best-found parameter value sets over each of the 50 generations, out of a population of 10.

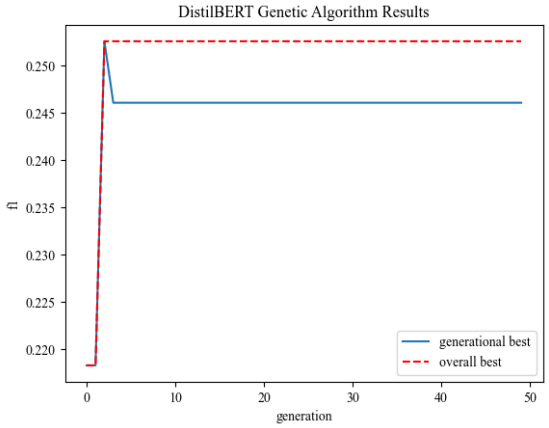

Figure 10: DistilBERT Span Identification Genetic Algorithm Generational and Overall Highest Fitness Plot.

The results of the genetic algorithm game the hyperparameter values, shown in *Table 6.*

Hyperparameter	Value
Training Epochs	8
Train Batch Size	8
Validation Batch Size	8
Warmup Steps	750
Weight Decay	0.001

Table 6: Hyperparameter Values for the best-Found Performing Model in DistilBERT Span Identification.

These values happened to be the exact same as found in *Section 2.2.2*. This could indicate that the models are adapting to system limitations for batch sizes. But the other metrics reveal the same information as discussed previously.

3. Results & Analysis

This section presents and discusses the results given by each of the tasks' models, drawing comparisons between the two approaches used and outlining the benefits and drawbacks of each. Table 7 shows the encoding values for each propaganda technique.

Index	Propaganda Technique
0	appeal_to_fear_prejudice
1	causal_oversimplification
2	doubt
3	exaggeration, minimisation
4	flag_waving
5	loaded_language
6	name_calling,labeling
7	repetition

Table 7: Numerical Encodings for Each of the Propaganda Techniques.

When the results are displayed in a classification report table, these indexes will be used to represent the classes. As each of the models were optimised with respect to the macro-f1 score, this result was highlighted in each table.

3.1 | Task 1 – Span Classification Results

This section outlines the results given by the SVC and DistilBERT models for the task of span classification.

3.1.1 | SVC

The results from testing the model, using the optimised parameters are shown in *Table 8*.

	Precision	Recall	F1-Score
0	0.51	0.56	0.53
1	0.34	0.43	0.38
2	0.55	0.51	0.53
3	0.52	0.57	0.54
4	0.84	0.73	0.78
5	0.54	0.51	0.53
6	0.59	0.65	0.62
7	0.65	0.51	0.57
Accuracy			0.56
Macro Avg.	0.57	0.56	**0.56**
Weighted Avg.	0.57	0.56	0.56

Table 8: The Classification Report for the SVC Classifier Model.

This model achieved an accuracy of 56%, and a macro-f1 score of 0.56. This model performed notably well on class 4 (flag waving), with an f1 score of 0.78. Although it seems the model struggles on other classes such as class 1 (casual oversimplification), where the f1 scorer was 0.38.

These varied results are likely due to the limitations of the feature sets as even though they capture word similarity, part-of-speech, and named-entity labels, this lacks the information needed to identify complex contextual indicators that further separate propaganda techniques as this can be difficult when dealing with those that are semantically similar.

The strengths of this model are with respect to its computational efficiency. This makes it much simpler to interpret and train.

3.1.2 | DistilBERT

The DistilBERT sequence classification model produced the results shown in Table 9.

	Precision	Recall	F1-Score
0	0.60	0.65	0.62
1	0.57	0.69	0.62
2	0.67	0.65	0.66
3	0.59	0.63	0.61
4	0.73	0.75	0.74
5	0.61	0.59	0.60
6	0.80	0.59	0.68
7	0.61	0.56	0.59
Accuracy			0.64
Macro Avg.	0.65	0.64	**0.64**
Weighted Avg.	0.65	0.64	0.64

Table 9: The Classification Report for the DistilBERT Sequence Classifier Model.

Here, with the optimised hyperparameter values, the model had shown a 64% accuracy, and a 0.64 f1 score. Here, the performance was shown to be relatively consistent across all classes, showing the models ability to generalise to high-dimensional data.

The main advantage of this transformer model is the ability to understand the contextual relationships, which allows for a clearer understanding of the classes, and how they differentiate from each other.

However, this model is computationally expensive. Although the DistilBERT model was chosen over BERT for its faster and cheaper computations, the difference between the transformer and machine learning models is still very noticeable.

3.1.3 | Comparison & Conclusion

To conclude the first task, while the SVC model is much faster and simpler regarding implementation, the DistilBERT model offers a superior accuracy for classification, and much better generalisation across every class.

Both models do perform better than a random guess, as this would achieve an accuracy of ~0.125. Therefore, showing that both methods learn meaningful information from the data and representations given and make distinctions between class types.

The choice of whether to use a transformer, or machine-learning model is dependent on the specific needs. If there is a situation where speed and interpretability is most important, then the SVC model would a reasonable choice. But overall, for real-world applications where propaganda classification accuracy is needed, the DistilBERT model is much preferred due to its better contextual understanding and therefore increased performance.

3.2 | Task 2 – Span Identification Results

This section presents the results given from the pipelines discussed in Section 2.3. These results are calculated after both the span identification and classification were completed. For the outcome of the model to be deemed correct, a span must have been identified from the sentence, and the technique used must have been classified correctly.

For this, the location / offset of the span was not deemed important as having exact positions correct would be very difficult to achieve. Instead, approximations for the presence and location of propaganda are considered to be acceptable as real-life applications would simply give warning to users that there may be propaganda present in some text.

3.2.1 | CRF + SVC Pipeline

The performance of the pipeline consisting of a CRF model for span identification with an SVC classifier for span classification is shown in *Table 10*.

	Precision	Recall	F1-Score
0	0.39	0.16	0.23
1	0.25	0.26	0.25
2	0.46	0.28	0.35
3	0.50	0.10	0.17
4	0.53	0.20	0.30
5	0.75	0.08	0.14
6	0.13	0.76	0.23
7	0.67	0.05	0.10
Accuracy			0.23
Macro Avg.	0.46	0.24	**0.22**
Weighted Avg.	0.47	0.23	0.22

Table 10: The Classification Report for the CRF + SVC Propaganda Identification and Classification Pipeline.

Overall, this pipeline achieved an accuracy of 23%, with a macro-average f1 score of 0.22. These low scores indicate the significant challenges faced by the model in both accurately identifying a propaganda span, and then correctly classifying them.

Per class, the pipeline performance varied greatly. A notable point being the high precision (0.75), and extremely low recall (0.08) of class 5 (loaded language). This shows that there were not many true-positive values captured. A similar pattern is again seen in class 6 (name calling).

These inconsistent results suggest the lack of generalisation across the classes, with most f1 scores falling below 0.3. These results show the limitations of the CRF + SVC pipeline used.

Whilst the pipeline is efficient, it relies on hand-crafted features rather than learned weights and deeper context. This results in the underperformance of the pipeline. As the SVC classifier was assessed with a mid-range performance, this means that the pipeline struggles greatly with the span identification.

3.2.2 | DistilBERT Pipeline

Table 11 shows the final set of results given by the DistilBERT two-step pipeline.

	Precision	Recall	F1-Score
0	0.21	0.14	0.17
1	0.24	0.23	0.23
2	0.11	0.02	0.04
3	0.11	0.03	0.05
4	0.00	0.00	0.00
5	0.18	0.10	0.13
6	0.12	0.56	0.19
7	0.17	0.18	0.17
Accuracy			0.15
Macro Avg.	0.14	0.15	**0.12**
Weighted Avg.	0.14	0.15	0.12

Table 11: The Classification Report for the DistilBERT Propaganda Identification and Classification Pipeline.

Despite showing a promisingly strong performance in Section 3.1.2, the two-stage DistilBERT classifier pipeline resulted in a significantly lower performance. This resulted in an overall accuracy of 15% and a macro-average f1 score of 0.12. Although there was a slight drop in performance expected a decrease of this significance signifies the challenges from the complex task of span identification, and how the pipeline failed to effectively overcome them.

The nature of two-stage models means that errors are propagated through the system. This meant that the performance in classification was first reliant on the accuracy of the span identification. If the identification returned incorrect results, it is more likely that the incorrect technique will be assigned in classification.

Given by the variation in class predictions, accuracy, and recall, it can be assessed that the model did not manage to generalise well to the data. This could have been a result of the sparsity of the data. As each class had limited training examples, this could have made it difficult for the transformer model to learn distinct patterns.

3.2.3 | Comparison & Conclusion

To conclude the results given by the span identification and classification pipelines proposed, it is shown that the machine-learning-based pipeline – utilizing a CRF and SVC – outperformed the two-stage DistilBERT pipeline.

Although both pipelines were subject to error propagation, it is likely that the transformer model underperformed due to the data sparsity and their need for informative sequences to be outputted for the context and semantics to be properly analysed.

Therefore, for this task, the recommended pipeline would be dependent on the data available. For sparse, or imbalanced, or noisy datasets, machine-learning pipelines are recommended due to their robustness and interpretability. These models are less prone to overfitting when data is sparse due to their reliance on engineered features and lower data requirements.

Although, in other cases where data may be more abundant, cascading transformer models are a powerful way to create pipeline that create deep, contextual analysis' and therefore may perform better in these scenarios. However, these models are much more sensitive to noise and therefore spans must be identified with care in order to avoid error propagation.

4. Discussion

This report outlines valid uses for both machine-learning, and transformer pipelines for complex language tasks – in this case, propaganda detection.

Transformer models showed better performance and generalisation for simpler classification tasks due to their ability to learn deeper contextual meaning within sentences, and to identify subtle language structures and meanings that distinguish the propaganda techniques to a much more accurate level.

Machine-learning based pipelines showed a robustness in more complex, cascading tasks compared to transformer pipelines. This is due to their ability to train on smaller datasets, and their lessened sensitivity to noise due to features requiring manual creation.

To reflect, SVC and CRF models both worked relatively well given their overall simplicity. DistilBERT outperformed SVC for classification but fell short when identifying spans. This does not inherently mean that they are unusable, but the conditions in this experiment regarding data sparsity may have prohibited the model from training to its full potential.

Further research may want to use DistilBERT for span identification and classification in a single model, rather than sequencing the tasks as separate elements. This could eliminate the error propagation from identified span offsets and yield a better performance.

5. References

Barron-Cedeno, A., Jaradat, I., Da San Martino, G., Nakov, P. (2019) *Proppy: Organizing the News Based on their Propagandistic Content. Information Processing & Management.*

https://doi.org/10.1016/j.ipm.2019.03.005

Da San Maryino, G., Yu, S., Barron-Cedeno, A., Petrov, R., Nakov, P., (2019) *Fine-Grained Analysis of Propaganda in News Articles*

https://doi.org/10.18653/v1/D19-1565

Rashkin, H., Choi, E., Jang, J. Y., Volkova, S. Choi, Y. (2017) *Truth of Varying Shades: Analysing Language in Fake News and Political Fact-Checking*

https://doi.org/10.18653/v1/D17-1317

Sprenkamp, K., Jones, D. G., Zavolokina, L., (2023) *Large Language Models for Propaganda Detection*

https://doi.org/10.48550/arXiv.2310.06422