Propaganda Detection

Introduction

This paper is focused on the Propaganda Techniques corpus (Da San Martino *et al.*, 2020) for the propaganda classification. The process has been divided into two subtasks: Span Identification (SI) and Technique Classification (TC). SI is a binary sequence tagging task to identify propaganda. TC is the task of classifying the type of propaganda technique used within the snippet. Bag of words (BoW) classifier with various experimentation is conducted. Pre-trained large language models, BERT is later used to compare the model performance. Dynamic padding technique is applied in the BERT training process. Each method incorporates various hyper-parameter tuning strategies and experimentation. The models' performance is evaluated by assessing metrics such as F1 score, precision, recall, accuracy, and macro average. Our findings demonstrate that the BERT with dynamic padding technique achieves comparable results.

Data

Propaganda Techniques Corpus (Da San Martino *et al.*, 2020) is used for this paper. Two columns are presented in the dataset. The label column indicates if the content is classified as propaganda and the type of propaganda techniques. The second column tagged_in_context, contains the text content, with specific tokens marked by <BOS> (beginning of span) and <EOS> (end of span) tags to indicate the given propaganda technique of the text. There are 9 labels in the label column. The first eight labels are propaganda techniques including loaded_language, flag_waving, doubt, name_calling,labeling, appeal_to_fear_prejudice, repetition, causal_oversimplification, and exaggeration,minimisation. The last label, "Not propaganda," indicates that propaganda is absent in the text.

2.1 Challenges

1. There is an extremely categorical imbalance of frequency in the target variable, with the highest concentration on 'not_propaganda' taking place 1,191 times. In such a scenario, a biased model is set toward the largest occurring class; hence, it becomes hard to recognize the minority classes correctly. Under binary classification, all the propaganda techniques are taken as one category, 'propaganda,' and in the case of multi-class classification, the label 'not_propaganda' are removed.

2. Some tags in the text, such as <BOS> and <EOS>, are to be taken care of during the preprocessing. In a basic model like a bag of words (BoW), sometimes it tends to wrongly relate these tags with some of the labels.

3. There are 2,402 unique text entries out of 2,414 total entries, meaning some of the data might have been duplicated within the dataset. Also, there is a possibility of the sparse problem in capturing textual relations within the text representation for Bag of Words.

Methodology

3.1 Related Work - BPGC at SemEval-2020 Task 11

Detection of the harmful task of propaganda in news required both traditional machine learning and deep learning methods. The authors (Patil, Singh and Agarwal, 2020) leveraged a rich feature set derived from the text, including lexical and pragmatic elements, while using XGBoost and Logistic Regression among other machine learning algorithms. The features have undergone an effective preprocessing stage with comprehensive preprocessing applied, such as converting to UTF-8, removing non-ASCII characters, lowercasing, lemmatizing, and stemming. For example, the authors developed the stacked long short-term memory networks and the CNN-LSTM hybrids, which are known for capturing the proper representation of the sequential nature of textual data. Further fine-tuning, the model was trained with the following parameters: two hidden layers with a width of 256 units, balancing the model complexity and performance with avoiding overfitting in the minority classes. The use of ensembling further gave more strength and accuracy, since it was done by both machine learning and deep learning models.

3.2 Bag of Word (BoW)

3.2.1 Cleaning and Preprocessing

To address the special tags problem <BOS> and <EOS> in both BoW and BERT models, the context from tags in between will be retrieved and the tags will be removed for the multiclass classification whereas, for the binary classification, the input will remain the same as the original. In the multiclass classification, since the goal is to concentrate on certain words and phrases, the models will focus more on these important contexts if the context between <BOS> and <EOS> is extracted (Dai and Iwaihara, 2023). As for binary classification, these tags can serve as distinct features in the models, helping the model differentiate non-propaganda or propaganda text. For example, if there are more propaganda sentences in the training data that begin or end with specific patterns or phrases, the tags might help the model recognize similar structures in unseen data.

3.2.2 Tokenization

Tokenization is the process of the logical division of the text into smaller elements, called tokens. These tokens can be words, numbers, or punctuation marks. The nltk function word_tokenize was used mainly for the experimentation on the pre-processing method to perform Stemming and Lemmatization before the Bag of Words (BoW) processing steps.

3.2.3 Normalization

3.2.3.1 Stop Words Removal

Since propaganda often uses rhetorical structures (Abbas, 2022) to persuade or influence and sentences that contain stop words might well be part of those constructs, stop words have been decided to remain in the text. For example, the repeated groups of words, "we are the people" and "it is time to," may be of considerable significance. Removing these stop words ("are," "is," "to") would take away parts of a phrase that lend to the overall rhetorical, emotional weight of the statement.

3.2.3.2 Punctuation Removal

After conducting several experiments and analyses, the result shows the importance of having punctuation in the propaganda dataset. Maintaining the punctuation in the content will form a solid basis for the text classification models. Punctuation has been found to play a significant role in stylometric analysis (Sabol, 2022). Hence, punctuation won’t be dealt with.

3.2.3.3 Stemming / Lemmatization

WordNetLemmatizer and PorterStemmer from nltk.stem had both been employed in the experimentation. Between the two, stemming was able to outperform lemmatization for the propaganda dataset. Stemming refers to the reduction of words to the base or root by chopping off the ends of words. It uses simple heuristics and does not consider the part of speech of a word or its context in a sentence. Lemmatization does the same of reducing words to the same basic form but uses complex linguistic rules based on morphological analysis. Lemmatization depends much on proper POS tagging; it must understand the grammatical role of a word so it can infer the right lemma (Gesmundo and Samardžić, 2012).

3.2.3.4 Named Entity Recognition (NER)

Based on the above, I sought to explore Lemmatization further with NER (imported from spacy, en_core_web_sm). Results indicate that lemmatization can be applied to NER. However, comparing it with simply applying stemming without NER, the result still shows the latter performs better. Stemming without NER enables it to perform at least one percentage point better in terms of the macro F1 score. Whereas these NER and lemmatization processes add a lot of linguistic richness within the case, they also add complexity that does not provide a better classification in this particular task. That could mean that some of the key aspects of propaganda are not very closely related to named entities or the actual morphological forms of words.

3.2.4 Bag of Words Vectorization

The Bag of Words (BoW) model is a technique to convert textual data into numerical form data. BoW models create a vocabulary of all unique words (tokens) in the entire corpus. These unique words are hence a feature in the final dataset. Every single word from this vocabulary, when represented, exists as a vector within the vector space. The size of the vector is made of every single word from the vocabulary.

3.2.5 Machine Learning Model

Logistic Regression is used in the training process. Its input is from vectorizations generated by the Bag of Words method. It works well with high-dimensional spaces (Widodo Wijayanto and Sarno, 2018) because the decision boundary in logistic regression is a hyperplane, which can effectively separate points in a high-dimensional space. The function is noted below:

$$\hat{p} = \sigma(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b)$$

Since CountVectorizer Ngram_range (1,7) is applied to our vectorization process, the feature vectors are sparse (1 to 7 grams for each combination of words). This attribute of Logistic Regression works well with this high-dimensional data in Bag of Words. Also, Logistic Regression can incorporate regularization (L1, L2, or both) to help mitigate overfitting, which makes it the ideal candidate for the model construction.

3.3 BERT

3.3.1 Preprocessing

BERT is a transformer-based model with a good understanding of how one piece of text relates to another. Noise typically influences the accuracy when performing the multiclass classification. Contexts are extracted from <BOS> and <EOS>, for BERT tactically focuses the attention on the most important segments of the text, thus enabling the model to make distinctions for the multiclass classification among the categories. The original context is used for binary classification, such that the model can appreciate the details of propaganda in a more complex manner than simple uncontextual embedding. In both classifications, the process of label transformation is used to make the text labels present in the data into numerical forms.

3.3.2 Tokenization

The tokenizer = BertTokenizer.from_pretrained('bert-base-uncased') is used for tokenization in BERT. Firstly, character normalization is converting the characters to lowercase, removing accents and other diacritical marks from characters, segmenting the text into tokens, and then using a subword tokenization algorithm known as WordPiece that deals better with unknown words with an effectively reduced vocabulary size. Additional tag representations such as CLS, SEP, PAD, and UNK would be added to enrich the input feature. After the text is broken into subwords, the input tokens would be transformed into numeric IDs, and positional embeddings for the model input (Nandanwar and Choudhary, 2023).

3.3.3 BertForSequenceClassification and DataLoader

The BertForSequenceClassification is a fine-tuning adaption of the BERT model for the sequence classification task (Xu, 2024), which is suitable for our propaganda dataset. TensorDataset contains IDs of tokens, attention masks, and labels—everything generated by the function of the tokenizer. DataLoader, on the other hand, batches data.

3.3.4 Training Loop

Model training is repeated through some epochs with the consistent reduction of the prediction error and weight updates by using backpropagation and optimization, which further finetunes the pre-trained BERT model on the task of propaganda detection by adapting its understanding of language to the nuances of the content.

3.3.5 Dynamic Padding

I also experimented with dynamic padding to see how the model can be improved. While working with natural language processing, the lengths of different texts or documents usually vary. Sending them to a model without adjustments would be impractical, as neural networks require equal-sized inputs (Collobert *et al.*, 2011). Dynamic padding is done so that sequences are not padded to the maximum length of the dataset, but instead to the maximum length within a batch.

Hyper-Parameter Settings

4.1 Bag of Words

The sklearn.feature_extraction.text.CountVectorizer is imported from the library. To begin with, Experimentation is emphasized in identifying key variations individually to determine suitable ranges of values for the hyperparameters. After that, extensive hyper-parameter tuning is applied to find the optimal parameters for the analyzer and n-grams. ‘word’, ‘char’, ‘char_wb’ three parameters were tried for the analyzer. From (1,2) to (1,7) the n-grams parameters were experimented. The best result is derived from the Analyzer set to char, ngram_range set to (1,7), and lowercase set to True. Parameters include:

- **Analyzer**: Is the feature to be made of word n-gram or character n-grams.
- **Ngram_range**: Defines the lower and upper boundary of the range of n-values to be extracted.
- **Lowercase**: Converts all characters to lowercase before tokenizing.

4.2 Logistic Regression

The same implementation is applied in finding the best parameters for Logistic Regression. Individual parameters will be tried in sets of combinations. The library from sklearn.linear_model.LogisticRegression will be imported. Parameters include:

- **Penalty -** The penalty parameter specifies the norm used in the penalization l1 (Lasso), l2 (Ridge), which helps prevent overfitting by discouraging overly complex models in logistic regression.
- **Solver -** Common solvers include lbfgs, saga, and others. They help improve computational efficiency.
- **max_iter -** This parameter sets the maximum number of iterations taken for the solvers to converge.
- **multi_class -** Logistic regression can be configured to support “ovr” (one-vs-rest) or “multinomial” which handles multinomial loss in a single formulation.

- **C -** It is the parameter that controls the strength of this penalty. A smaller value of C specifies stronger regularization, leading to smaller model coefficients, which can help prevent overfitting by penalizing large values.

Setting the penalty to l2 helps reduce model complexity and enhance generalization in the propaganda dataset. lbfgs solver is suitable for l2 penalization for medium datasets (Olatunji *et al.*, 2022). It approximates the second-order partial derivatives of the log-likelihood. 10000 is set to ensure that the model has enough iterations to converge. Multinomial is set for the multiclass classification and ovr for binary classification. 0.1 is found to be most suitable for the model for the propaganda dataset.

4.3 BERT

4.3.1 Tokenize

- **The tokenizer.batch_encode_plus** function enables data to be batch-encoded and tokenized effectively.
- **max_length**: This should be long enough to capture most of the information from a sentence.
- **pad_to_max_length**: Make sure the sequences have the same maximum length, so they can be batched to be processed.
- **Truncation**: the sequence is truncated to 256 tokens.
- **returns_tensors**: Return PyTorch tensors that can be directly used to train a model.

In the first experimentation with BERT, I use BertTokenizer with max_length=256 and pad_to_max_length=True to fix the size of the input and let it, therefore, take a proper size of long text. In considering these parameters, I find a balance between the power of parallelization offered with a larger batch size versus the inefficiency of the parallelization process with too much token space and its effect on the memory limitations of the dataset, leading to the second experimentation, dynamic padding.

4.3.2 Batch Sizes / Epoch / Learning Rate / Optimizer

- **Batch Sizes**: It is the amount of training samples to work through before updating the model in one iteration.
- **Epoch**: An epoch represents one complete pass of the training dataset through the algorithm.
- **Learning Rate**: This is the parameter that controls the step size of the model with the estimated error.
- **Optimizer**: It controls how the network updates to process the input data and minimize the loss function.

It has been found through hyper-parameter tuning that setting a learning rate of 1e-5, an optimizer of AdamW, an epoch of 3, and a batch of sizes 8 performs best in the model with the propaganda dataset. The sizes of the batches are controlled within 8, 16, and 32 during the experimentation. 1e-5, 2e-5, 3e-5 for the learning rate. The number of epochs was tried with 3, 5, and 8 to prevent overfitting. AdamW is better than Adam because it decouples weight decay from the gradient updates and applies it directly to the weights after adaptive rates have been computed (Loshchilov and Hutter, 2018).

4.4 BERT Dynamic Padding

Dynamic padding is an effective way of handling variable-length input sequences (Ge *et al.*, 2021), with less memory used when sequences are padded to the length of the longest sequence, and computing the processing is also less than when sequences are padded to a fixed maximum length. Parameters to collate_batch:

- **batch**: a bunch of samples, where each of them has input_ids, attention_mask, and label.
- **input_ids**: tokens representations of the text.
- **attention_masks**: The model uses these masks to understand which parts of the input_ids are real data or padding.
- **Labels**: These are the targets showing whether the text is propaganda or not.
- **pad_sequence** : This function is called to pad the sequences in a batch to the length of the maximum sequence.
- **batch_first**: When set to True, it indicates that the padding should adjust the sequence length at the first dimension.
- **padding_value**: It is set to tokenizer.pad_token_id which is the token for a position that contains padding. For attention_masks, this is set to 0, which means the padded position is not attended to by the model.

Evaluation Methods

5.1 Precision

High precision indicates that when the model predicts a positive result, it is very likely to be correct. High precision is relevant where the cost of false positives is high. In propaganda detection, for example, it means that non-propaganda has a low chance of being considered propaganda because of the high precision rate.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

5.2 Recall

Recall is the actual positive instances that were predicted as positive by the model. High Recall means the model is good in catching the positive cases where it tries to minimize the false negatives.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

5.3 F1-Score

An F1 score is the mean of precision and recall. It is important in situations where balance must be made between precision and recall. A high F1 score indicates that a good balance is being achieved between precision and recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.4 Accuracy

For binary classification, the accuracy score merely corresponds to how well the model can correctly classify the text into propaganda or not. In multiclass classification, it scores how successful the classifier is in overall categories.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

5.5 Macro Average

It calculates the metrics of a class individually and averages it, giving equal importance to the propaganda class and the non-propaganda class regardless of their frequency. In multiclass classification, the macro average allows all classes to be equally important to reflect the performance of the model in detecting a specific category.

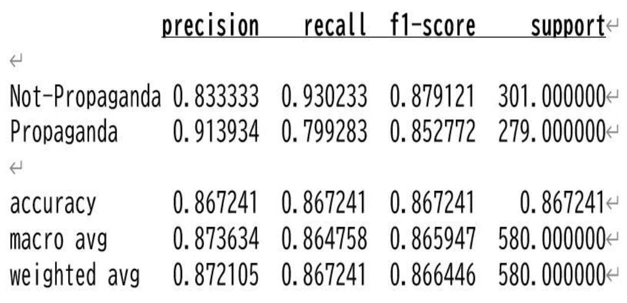
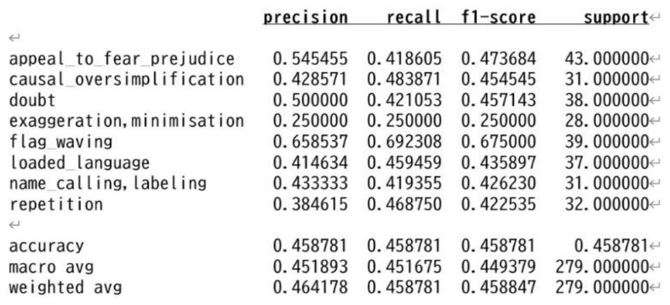
$$M_{\text{macro}} = \frac{1}{N}\sum_{i=1}^{N} M_i$$

Results & Analysis of Errors

6.1 Bag of words

6.1.1 Experimentation1 - Bag of words classifier with optimized CountVectorizer

From the binary classification, Not-Propaganda has a lower precision, but higher recall, indicating the model is conservative at propaganda prediction, meaning it is more in favor of classifying borderline cases as not propaganda. Propaganda gets higher precision but lower recall, meaning missed propaganda and higher correctness of identified cases. The overall accuracy of 86.7% is rather good for the bag of words classifier with optimized CountVectorizer, although the balance between recall and precision for the class of propaganda could be improved.

Binary classification	Multiclass classification
	

```
               precision    recall  f1-score     support

Not-Propaganda  0.833333  0.930233  0.879121  301.000000
Propaganda      0.913934  0.799283  0.852772  279.000000

accuracy        0.867241  0.867241  0.867241    0.867241
macro avg       0.873634  0.864758  0.865947  580.000000
weighted avg    0.872105  0.867241  0.866446  580.000000
```

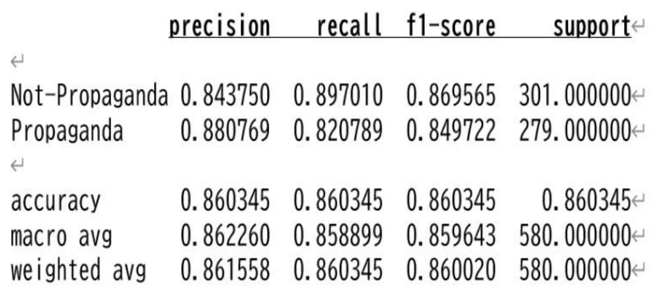
```
                           precision    recall  f1-score     support

appeal_to_fear_prejudice    0.545455  0.418605  0.473684   43.000000
causal_oversimplification   0.428571  0.483871  0.454545   31.000000
doubt                       0.500000  0.421053  0.457143   38.000000
exaggeration,minimisation   0.250000  0.250000  0.250000   28.000000
flag_waving                 0.658537  0.692308  0.675000   39.000000
loaded_language             0.414634  0.459459  0.435897   37.000000
name_calling,labeling       0.433333  0.419355  0.426230   31.000000
repetition                  0.384615  0.468750  0.422535   32.000000

accuracy                    0.458781  0.458781  0.458781    0.458781
macro avg                   0.451893  0.451675  0.449379  279.000000
weighted avg                0.464178  0.458781  0.458847  279.000000
```

For precision and recall in multiclass, "flag-waving" shows a good result, whereas "exaggeration, and minimisation" struggle. F1 scores reflect a similar trend. That the categories "exaggeration, minimisation" perform worse may suggest fewer examples or more complexity in the definition of what denotes exaggeration or minimization with the simple BoW classifier. Other reasons may be that categories like "flag-waving" are defined quite easily with specific words so that BoW can learn from them. General accuracy is 45.8%, while the macro average F1-score is 44.9%, indicating space for improvement in all classes.

6.1.2 Experimentation2 - Bag of words classifier with Stemming

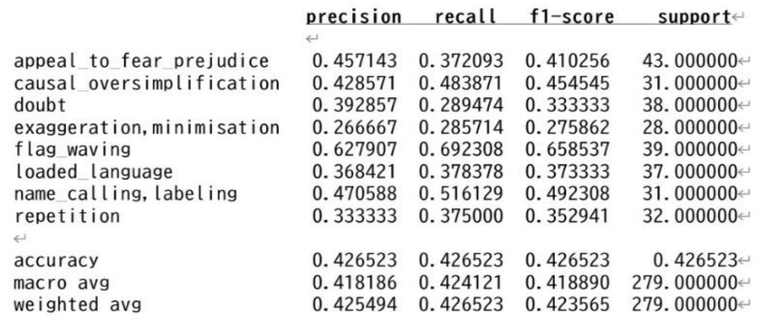
Precision has slightly increased for Not-Propaganda and decreased for Propaganda in the binary classification experimentation 2. This suggests that stemming may have slightly improved the detection of Not-Propaganda but at a small cost in terms of identifying Propaganda. We can see how the degradation of recall for Not-Propaganda and the slight improvement for Propaganda suggest how stemming can reduce overfitting on non-propaganda examples and slightly enhance the sensitivity to propaganda. Both the F1-score and the accuracy of the model were only slightly lower in Experimentation 2 implying that further stemming does not result in the desired benefit for binary classification.

Binary classification



```
               precision    recall  f1-score     support↵
↵
Not-Propaganda  0.843750  0.897010  0.869565  301.000000↵
Propaganda      0.880769  0.820789  0.849722  279.000000↵
↵
accuracy        0.860345  0.860345  0.860345    0.860345↵
macro avg       0.862260  0.858899  0.859643  580.000000↵
weighted avg    0.861558  0.860345  0.860020  580.000000↵
```

Multiclass classification



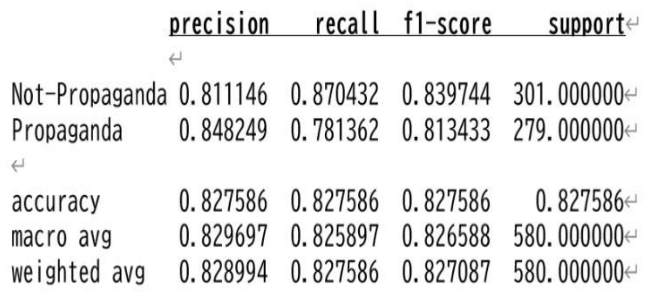
```
                           precision    recall  f1-score     support↵
↵
appeal_to_fear_prejudice    0.457143  0.372093  0.410256   43.000000↵
causal_oversimplification   0.428571  0.483871  0.454545   31.000000↵
doubt                       0.392857  0.289474  0.333333   38.000000↵
exaggeration,minimisation   0.266667  0.285714  0.275862   28.000000↵
flag_waving                 0.627907  0.692308  0.658537   39.000000↵
loaded_language             0.368421  0.378378  0.373333   37.000000↵
name_calling,labeling       0.470588  0.516129  0.492308   31.000000↵
repetition                  0.333333  0.375000  0.352941   32.000000↵
↵
accuracy                    0.426523  0.426523  0.426523    0.426523↵
macro avg                   0.418186  0.424121  0.418890  279.000000↵
weighted avg                0.425494  0.426523  0.423565  279.000000↵
```

In the multiclass classification, the precision, recall, and F1-score in all classes are lower in experiment 2 for the above-compared classes "doubt" and "exaggeration, minimisation." In addition, the accuracy in experiment 2 was also lower, which suggests that stemming is not that optimal for a BoW classifier to be correctly classifying instances for multiple classes. This could be one reason why stemming results in oversimplification of the text, hence the loss of important contextual nuances that help to differentiate the types of propaganda. This could have particular impact while working with datasets like propaganda, where linguistic subtlety is very important for its classification.

6.1.3 Experimentation3 - Bag of words classifier with NER & Lemmatizon

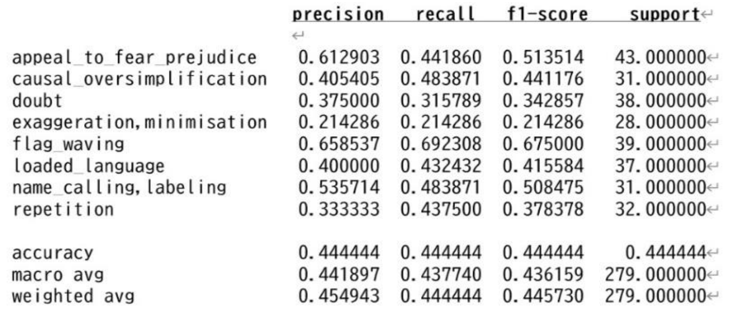
There is a slight decrease for both cases in Experimentation 3. This might imply that, however more sophisticated NER and lemmatization are, they might have brought in more noise or complexity into the model, which did not lead to better recognition of features for this task. The model's general accuracy and F1-scores are also lower in Experimentation 3 with more complex preprocessing.

Binary classification



```
               precision    recall  f1-score     support↵
               ↵
Not-Propaganda  0.811146  0.870432  0.839744  301.000000↵
Propaganda      0.848249  0.781362  0.813433  279.000000↵
↵
accuracy        0.827586  0.827586  0.827586    0.827586↵
macro avg       0.829697  0.825897  0.826588  580.000000↵
weighted avg    0.828994  0.827586  0.827087  580.000000↵
```

Multiclass classification



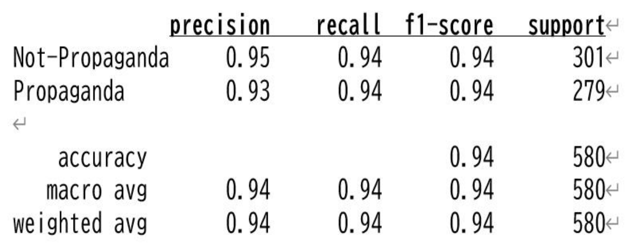
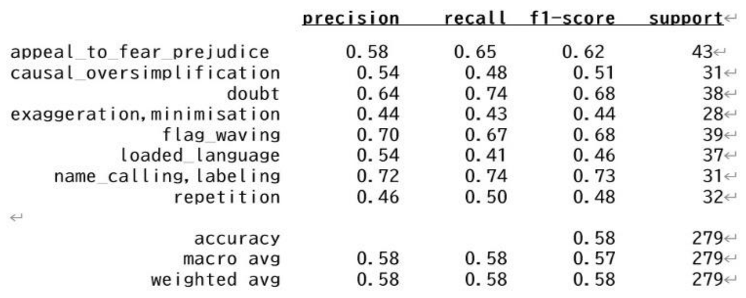
```
                           precision    recall  f1-score     support↵
↵
appeal_to_fear_prejudice    0.612903  0.441860  0.513514   43.000000↵
causal_oversimplification   0.405405  0.483871  0.441176   31.000000↵
doubt                       0.375000  0.315789  0.342857   38.000000↵
exaggeration,minimisation   0.214286  0.214286  0.214286   28.000000↵
flag_waving                 0.658537  0.692308  0.675000   39.000000↵
loaded_language             0.400000  0.432432  0.415584   37.000000↵
name_calling,labeling       0.535714  0.483871  0.508475   31.000000↵
repetition                  0.333333  0.437500  0.378378   32.000000↵

accuracy                    0.444444  0.444444  0.444444    0.444444↵
macro avg                   0.441897  0.437740  0.436159  279.000000↵
weighted avg                0.454943  0.444444  0.445730  279.000000↵
```

The multiclass results exhibit a slight improvement with metrics in Experimentation 3 with the lemmatization and NER method. It helps detect distinctions of the categories of propaganda with some specific entities. An increase in performance is noticed for categories such as "appeal to fear_prejudice" and "flag waving, suggesting that lemmatization and NER may boost the model's ability to identify relevant textual entities and contexts. Although NER and lemmatization are very powerful theoretical tools for text data enrichment, their utility must be balanced against increased complexity. In the task of binary classification, simpler stemming is performed more efficiently, showing that for broad categorization, complex preprocessing does not always lead to better outcomes. From this perspective, preprocessing in Experimentation 3 seems to support better multi-class scenarios, where the impact of specific entity recognition on classification may be significant.

6.2 BERT

6.2.1 Experimentation4 – BERT with Hypermeter Setting

BERT outperforms the Bag of Words model in most metrics, which means it is much more capable in understanding context compared to the classical n-gram-based methods. As seen in precisions and recalls, BERT is quite successful in the identification of most relevant features towards both classes, with much less noise compared to a Bag of Words model. The gain in accuracy and F1 scores tells us about better success not only in defining each class correctly but also in keeping the balance of both precision and recall high.

Binary classification	Multiclass classification
	

```
               precision    recall  f1-score   support

Not-Propaganda      0.95      0.94      0.94       301
    Propaganda      0.93      0.94      0.94       279

      accuracy                          0.94       580
     macro avg      0.94      0.94      0.94       580
  weighted avg      0.94      0.94      0.94       580
```

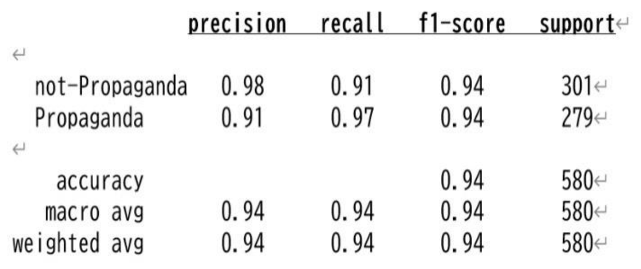
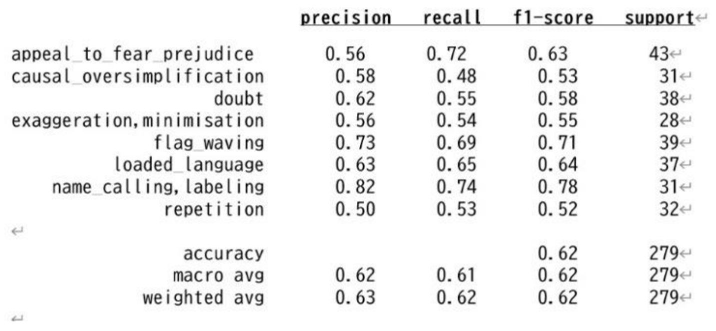
```
                           precision    recall  f1-score   support
appeal_to_fear_prejudice       0.58      0.65      0.62        43
causal_oversimplification      0.54      0.48      0.51        31
                    doubt      0.64      0.74      0.68        38
exaggeration,minimisation      0.44      0.43      0.44        28
              flag_waving      0.70      0.67      0.68        39
          loaded_language      0.54      0.41      0.46        37
    name_calling,labeling      0.72      0.74      0.73        31
               repetition      0.46      0.50      0.48        32

                 accuracy                          0.58       279
                macro avg      0.58      0.58      0.57       279
             weighted avg      0.58      0.58      0.58       279
```

Improvement with BERT is also significant in multiclass classification. While not as dramatic as in binary classification, the rise of precision, recall, and F1-score is obvious. It is an indication that BERT is likely to be successful in being able to differentiate between more nuanced categories by contextual awareness and its embedding capabilities. The increase in accuracy stands for an increase in the general performance of the model for all classes, which is, the more profound the linguistic knowledge from BERT, the more accurate the classification will be.

The results prove BERT to be a much more powerful model in dealing with highly complex classification exercises, achieved by the model's pre-training over a big corpus and its fine-tuning afterward. This is especially true for when trying to detect those subtle differences in text that more classic Bag of Words methods would miss.

6.2.2 Experimentation5 – BERT with Dynamic Padding

I expand Experimentation 4 here and find that dynamic padding gives a slight improvement in accuracy for "Not-Propaganda" while it gives a reduced recall, implying that there is a trade-off between identifying all positive instances and the accuracy of positive predictions. For "Propaganda," the recall increased significantly in Experimentation 5. It would mean that dynamic padding is able to manage contextual nuances in the definition of propaganda, with just a slight decrease in precision. The overall accuracy remains the same, and the changes in precision and recall show that it performs more balanced in handling both classes with dynamic padding.

Binary classification	Multiclass classification
	

```
               precision    recall  f1-score   support

not-Propaganda      0.98      0.91      0.94       301
    Propaganda      0.91      0.97      0.94       279

      accuracy                          0.94       580
     macro avg      0.94      0.94      0.94       580
  weighted avg      0.94      0.94      0.94       580
```

```
                           precision    recall  f1-score   support

appeal_to_fear_prejudice       0.56      0.72      0.63        43
causal_oversimplification      0.58      0.48      0.53        31
                    doubt      0.62      0.55      0.58        38
exaggeration,minimisation      0.56      0.54      0.55        28
              flag_waving      0.73      0.69      0.71        39
          loaded_language      0.63      0.65      0.64        37
    name_calling,labeling      0.82      0.74      0.78        31
               repetition      0.50      0.53      0.52        32

                 accuracy                          0.62       279
                macro avg      0.62      0.61      0.62       279
             weighted avg      0.63      0.62      0.62       279
```

Overall, all metrics improve with dynamic padding: the increase in precision, recall, and F1-score implies that handling variable sequence length more effectively captures more correct and relevant features for classification. There are also individual categories whose performance in precision and recall have impsroved quite a lot such as "appeal to fear_prejudice" and "name calling, labeling." This could have to do with dynamic padding, since it would enable the model to pay more attention to the contents of every sample rather than the noise introduced through padding of shorter texts to a fixed maximum length. Dynamic padding in BERT seems to enable text data processing while minimizing unnecessary computation in padded areas, focusing more on the real content of each text.

Conclusion

These several experimentations aimed at text classification of propaganda to exemplify how different methodologies of complexity level may achieve varying degrees of success. In summary, implementation of some base models can be achieved by BoW coupled with CountVectorizer, but it starts to show its limitations as the text categorization problems grow in complexity. More advanced models, like BERT, further fine-tuned with techniques like dynamic padding, provide a solution that is highly efficient in handling classification problems with great complexity and also experienced in grasping details and the necessary contextual nuances of language in determining accurate propaganda.

Further work

- It could perform even better if the model were tuned and tested with additional preprocessing stages, such as a syntactic parse for the BoW model. The feature selection methods could be applied in a trial to help reduce data dimensionality and focus only on those features that would be most informative in allowing better performance.
- Cross-evaluation tests with a more diversified dataset could also offer further insight into the effectiveness and accuracy.
- BERT can be used to extract features such as a logistic regression model in order to improve its performance and resources.
- Other Transformer family models can also be experimented with for better performance, for instance BERT-Large, RoBERTa, or GPT.

References

Abbas, A.H. (2022) ‘Politicizing the Pandemic: A Schemata Analysis of COVID-19 News in Two Selected Newspapers’, *International Journal for the Semiotics of Law - Revue internationale de Sémiotique juridique*, 35(3), pp. 883–902. Available at: https://doi.org/10.1007/s11196-020-09745-2.

Collobert, R. *et al.* (no date) ‘Natural Language Processing (Almost) from Scratch’, *NATURAL LANGUAGE PROCESSING* [Preprint].

Da San Martino, G. *et al.* (2020) ‘SemEval-2020 Task 11: Detection of Propaganda Techniques in News Articles’, in A. Herbelot et al. (eds) *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Barcelona (online): International Committee for Computational Linguistics, pp. 1377–1414. Available at: https://doi.org/10.18653/v1/2020.semeval-1.186.

Dai, X. and Iwaihara, M. (no date) ‘Utilizing Keyphrase Generation and Semantic Similarity for Extreme Multi- Label Text Classification’.

Ge, Z. *et al.* (2021) ‘Speed up Training with Variable Length Inputs by Efficient Batching Strategies’, in *Interspeech 2021*. *Interspeech 2021*, ISCA, pp. 156–160. Available at: https://doi.org/10.21437/Interspeech.2021-2100.

Gesmundo, A. and Samardžić, T. (2012) ‘Lemmatisation as a Tagging Task’, in H. Li et al. (eds) *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. *ACL 2012*, Jeju Island, Korea: Association for Computational Linguistics, pp. 368–372. Available at: https://aclanthology.org/P12-2072 (Accessed: 11 May 2024).

Loshchilov, I. and Hutter, F. (2018) ‘Fixing Weight Decay Regularization in Adam’. Available at: https://openreview.net/forum?id=rk6qdGgCZ (Accessed: 12 May 2024).

Nandanwar, A.K. and Choudhary, J. (2023) ‘Contextual Embeddings-Based Web Page Categorization Using the Fine-Tune BERT Model’, *Symmetry*, 15(2). Available at: https://doi.org/10.3390/sym15020395.

Olatunji, S.O. *et al.* (2022) ‘A Novel Ensemble-Based Technique for the Preemptive Diagnosis of Rheumatoid Arthritis Disease in the Eastern Province of Saudi Arabia Using Clinical Data’, *Computational and Mathematical Methods in Medicine*, 2022, p. e2339546. Available at: https://doi.org/10.1155/2022/2339546.

Patil, R., Singh, S. and Agarwal, S. (2020) ‘BPGC at SemEval-2020 Task 11: Propaganda Detection in News Articles with Multi-Granularity Knowledge Sharing and Linguistic Features based Ensemble Learning’. arXiv. Available at: http://arxiv.org/abs/2006.00593 (Accessed: 5 May 2024).

Sabol, R. (2022) *Propaganda Detection using Stylometric Text Analysis*. Diplomová práce. Masaryk University, Faculty of Informatics. Available at: https://theses.cz/id/bhcrku/?lang=en#panel_bibtex (Accessed: 8 May 2024).

Widodo Wijayanto, U. and Sarno, R. (2018) ‘An Experimental Study of Supervised Sentiment Analysis Using Gaussian Naïve Bayes’, in *2018 International Seminar on Application for Technology of Information and Communication*. *2018 International Seminar on Application for Technology of Information and Communication (iSemantic)*, Semarang: IEEE, pp. 476–481. Available at: https://doi.org/10.1109/ISEMANTIC.2018.8549788.

Xu, H. (2024) *Applying the Bert Algorithm for Software Requirements Classification*. Available at: https://doi.org/10.13140/RG.2.2.33317.68320.