Propaganda techniques classification

CS-C3250

Tuan Nguyen[1], My Linh Nguyen[2], Luna Ansari[3], and Anselmi Jokinen[4]

[1]**tuan.t.nguyen@aalto.fi**
[2]**linh.m.nguyen@aalto.fi**
[3]**luna.ansari@aalto.fi**
[4]**anselmi@ajokinen.com**

ABSTRACT

Text Classification has been a well-studied task in Natural Language Processing. In this work we train various classifiers by applying deep learning methods for the purpose of analyzing a data set of propaganda technique samples. We use Long short-term memory neural network as well as transformer-based models in our experiments. The classification is conducted at two levels: sentence level and token level. In addition, this study utilizes scraped data in order to test generalization of the trained models on unseen data sets. We compare the models' performance and conclude that the accuracy of models depends on various components. Though in highly imbalanced and multi-label setting it is a challenge to achieve good performance, especially on rare classes, we find that utilizing context captured by the previous and the next sentences improves performance of models, especially context-based model such as BERT significantly.

Keywords: deep learning, BERT, multilabel text classification, propaganda detection techniques, rationale-based classification

1 INTRODUCTION AND PROBLEM DESCRIPTION

Natural language processing (NLP) technologies are widely applied to solve a range of problems across various domains such as text summarizing, language translation, and sentiment analysis. Whereas earlier attempts to solve these problems relied on rule based methods and probabilistic methods such as hidden Markov model (20) which required much data engineering, more recently NLP methods have relied much more on machine learning and deep learning (6) (23). With the rise of powerful computing systems, it has become possible to train end-to-end systems as machine learning has been shown useful in addressing various problems from fields such as image recognition (12), speech recognition (3), and also NLP (17).

Early approaches to text classification relied on representing documents through a Bag-of-Words representation using machine learning methods where the words were not processed sequentially (23). Nowadays text classification is done by considering the sequential nature of data using convolutional neural networks (CNN) and long short-term memory (LSTM) neural networks, as these models can in certain situations require many fewer training samples due to their use of embeddings. Two of recent popular models are Generative Pretrained Transformer (GPT2) (21) and Bidirectional Encoder Representations from Transformers (BERT) (9), which are both based on transformers, being essentially a neural network block (23). GPT2 is trained to predict the next word from left to right and as a corpus it uses a lot of scraped human generated data to preserve document quality. In addition to word or token embeddings it also uses information about the position of the word, i.e. the position of the words in the sentence, in the form of position embeddings. (24)(.

Sequence to sequence models is another application of deep neural network based models in text classification. Sequence to sequence models, such as sequence labeling and part of speech tagging, are applied for machine translation and text summarization where the input is a larger piece of text and the output is a smaller piece of text (2). Sequence labeling is a task where we assign a label to every sequence of the input.

As methods such as recurrent neural networks, LSTM and attention-based models have transformed speech and natural language processing, we draw from these models to perform analysis and classification of texts by detecting the fragments that contain propaganda techniques. In particular, we study a corpus of

news articles being labeled with eighteen propaganda techniques. Propagandist messages are conveyed through a range of rhetorical and psychological techniques and within text based documents, propaganda techniques have been studied in computational linguistics (10). Research has addressed propaganda detection at various levels of granularity. However the conducted studies have primarily focused on document analysis and in most cases have considered how whole articles of a propagandist new outlet can be labeled (10). Recently research has taken a more fine-grained frame of analysis and has looked into how sequences of words can signify and make up such techniques.

We address this problem at two levels of analysis: first by conducting the classification at the sentence level and secondly at the more granular sequence level. Our primary goal is to experiment with various supervised modeling approaches for text classification and sequence labeling including several BERT architectures and LSTM. Comparing the performance measures between these models will strengthen our understanding of multi-label classification algorithms and model tuning. It will as well shed light on how deep learning pipelines are structured and organized.

This study is structured as follows. First, an overview of the data and exploratory data analysis is provided. Second, the application of data augmentation in this project is described. Third a review of existing research on natural language processing techniques in particular related work utilized in this project on text classification methods is presented. Next the learning methods and the achieved results for propaganda classification are discussed. Finally comparison between these models, concluding remarks, limitation and further research directions are discussed.

2 DATASET DESCRIPTION AND VISUALIZATION

We utilized a dataset of expert annotated news articles provided by Martino et al. (15) through the Propaganda Analysis project. The dataset contains 451 news articles originating from 48 outlets, with sentences annotated with 18 different propaganda techniques. We also retrieved an additional 200 sentences that we labeled as either positive (containing propaganda) or negative ourselves. Both datasets are described in more detail below.

2.1 Expert Annotated Dataset

The expert annotated dataset consists of both propaganistic and non-propaganistic articles sourced from various publications and then annotated through a process developed by Martino et al. (15). The 18 labels used in the annotations were chosen by Martino et al. (15) and are listed fully in Table 2.

The number of sentences in this dataset totals 22487 with average lengths of 22.3 tokens and 110.6 characters (if not hard, add a picture with distribution of those token lengths).

Attribute	Word Count
Maximum number of words in a sentence	129
Minimum number of words in a sentence	0
Average number of words in a sentence	22.2
Maximum number of words in a propaganda sequence	81
Minimum number of words in a propaganda sequence	0
Average number of words in a propaganda sequence	2.6

Table 1. Total word count in full sentences and propaganda spans in training set

The sentences vary significantly in length with the shortest and longest sentences being 1 and 218 tokens long, respectively.

The labels of the data are in the form of text spans annotated with a specific propaganda technique. As an example, the sentence ”Florida gubernatorial candidate and George Soros-backed Andrew Gillum may have just been completely undercut by a member of his own staff.” has annotations for the spans ”Florida gubernatorial candidate and George Soros-backed” and ”completely undercut” with labels *Name calling or labeling* and *Loaded language*, respectively.

For our sentence classification task we simply assigned each sentence the labels of any spans occurring in that sentence. The example sentence above was then given both the labels *Name calling or labeling* and *Loaded language*. Any sentences with no propaganda spans within them were labeled as negative. Due to the nature of this process the sentences could potentially (as seen above) have multiple labels.

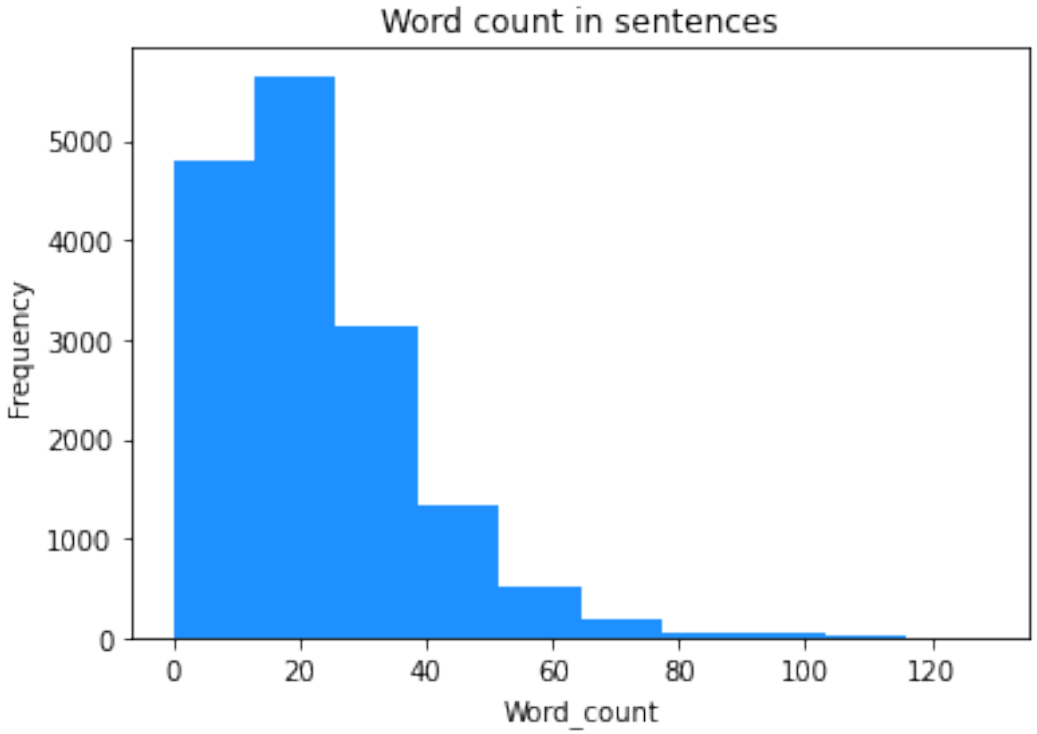

Figure 1. Word count in full sentences in train set

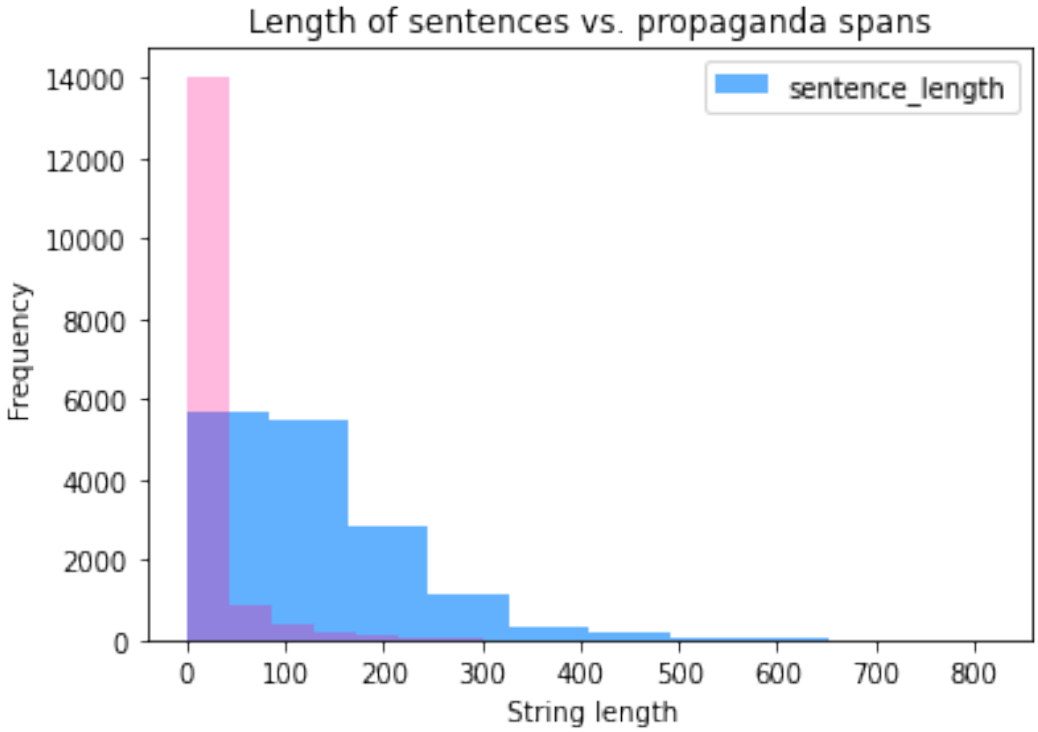

Figure 2. Length of full sentences and propaganda spans in train set

These labeled spans number 7480 in total and show similar variation in length to the sentences themselves, ranging from 1 to 133 tokens in length. These average around 7.9 tokens, which represents around a third of the average sentences length.

A key characteristic of the dataset is a significant imbalance in the number of instances of each class, with the two most common classes accounting for over half of the data. The entire distribution of positive classes is described in detail in Figure 3.

1	Loaded language.	10	Appeal to authority.
2	Name calling or labeling.	11	Black-and-white fallacy, dictatorship.
3	Repetition.	12	Thought-terminating cliche.
4	Exaggeration or minimization.	13	Whataboutism.
5	Doubt.	14	Reductio ad Hitlerum.
6	Appeal to fear/prejudice.	15	Red herring.
7	Flag-waving.	16	Bandwagon.
8	Causal oversimplification.	17	Obfuscation, intentional vagueness, confusion.
9	Slogans.	18	Straw man.

Table 2. Propaganda Labels used in the Expert Annotated Dataset

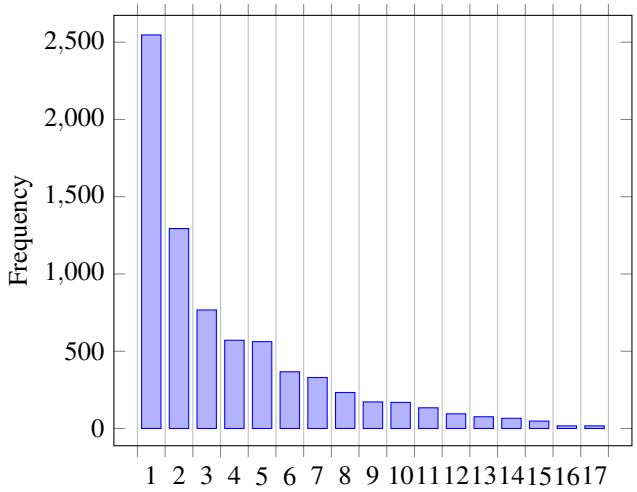

Figure 3. Class distribution of the expert annotated dataset

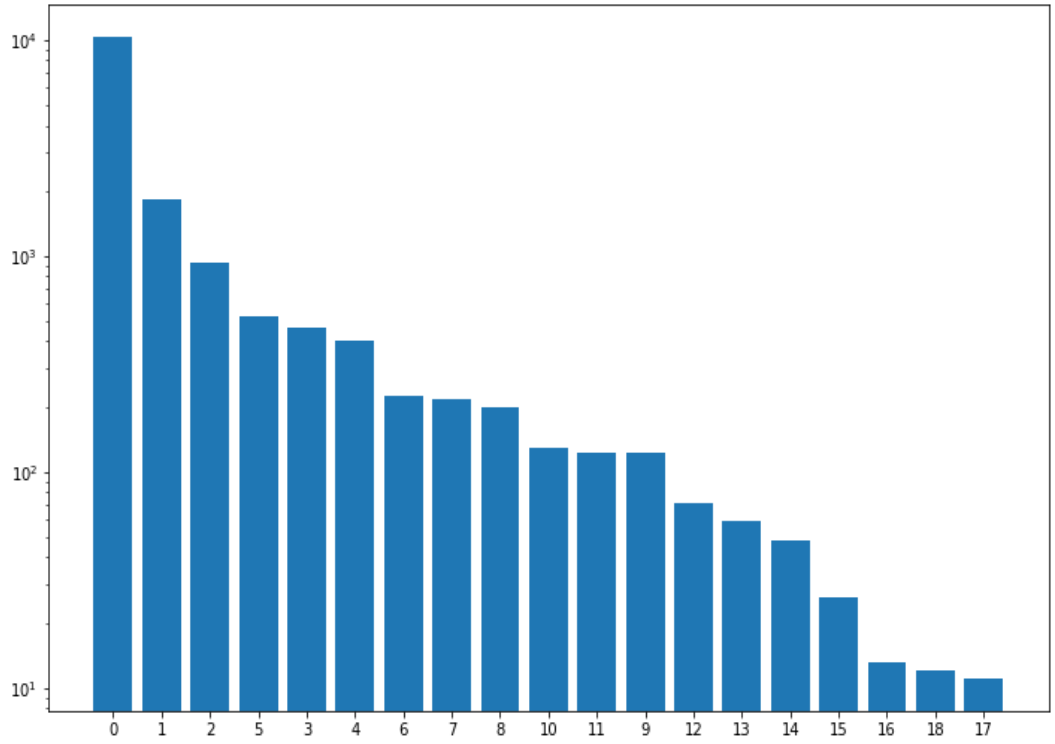

Figure 4. Class Distribution in log scale

2.2 Additional data

In addition to the data from the Propaganda Project, we also collected 200 sentences with and without use of propaganda. Similar to the original dataset, our sentences are splitted from collected articles, or paragraphs, as it helps provide context for individual sentence. Our data was collected from 4 different platforms, including Twitter, Breitbart, HuffPost and Alternet.

The collected data is annotated manually by our team with binary labels (”propaganda”, ”not propaganda”) instead of multi-labels, due to the lack of professional expertise in data annotation.

3 DATA PREPROCESSING AND AUGMENTATION

3.1 Data Preprocessing

As the data and separate labels were in separate file, our first step was to combine them into one place with a more usable format, such as an 2D-array, or a Pandas dataframe.

Once the data is combined, we perform general techniques to clean our data, including removing empty rows, punctuation, numbers, and stop words. Stop words are words often filtered out during text preprocessing, as they do not play an important role in expressing the meaning, and sentiment of the text, such as “the”, “is” or “and”. Although these tends to be the most common words, currently there is not an universal list for these words, in our project, we utilised the stop words list from “Spacy” library [1].

3.1.1 Data Vectorization

As machine learning model can not understand textual data directly, it is necessary that the data is transformed in to vectors for the model to process. This process includes Sentence and Word Tokenization, Text Embedding.

Tokenization Tokenization is the process of dividing the raw text into small meaningful chunks, such as words or sentences, called token (7). In our project, the first task require classifying the propaganda technique on sentence level. Thus, the tokenization process is utilized to break sentences into meaningful words token. As tokenization is a necessary process in developing Natural Language Processing models,

[1] https://github.com/explosion/spaCy/blob/master/spacy/lang/en/stop$_w$*ords.py*

there are several pre-trained models and libraries available for the task. Different libraries might handle the tokenization task differently, instead of just splitting sentence into chunks of text, they might extend into turning textual token into a sequences of integers based on their ids in the pre-trained model dictionary, or an binary vector [2] [3]. In our project, we utilize both self-written tokenizer, and pre-train models, such as BERT tokenizer [4]. The self-written tokenizer creates an index mapping dictionary in such a way that the frequently occurring words are assigned lower indexes.The vocabulary to integer mapping dictionary then is utilized to tokenize each sentence.

3.2 Data Augmentation

Data Augmentation is a popular approach to up-sample data when the dataset is small or balanced. Data augmentation enriches the original dataset by making simple modifications from the training data, such as flipping an image in image data, or replacing words with their synonym.

In the context of Natural Language Processing, Data Augmentation is the technique of adding more samples to the dataset either by modifying existing samples or by adding new data samples from other data sources. There are several Data Augmentation methods in NLP, which can varies both in complexity, and computation resource. These methods includes, but not limited to:

- Thesaurus: generate a new sentence by replacing words with their synonyms.
- Back Translation: generate new sentence by translating a target/ sentence/ paragraph to another language, then translate it back to the original language. For example, we translate a sentence from English to German, and then translate it back to English.
- Contextualized word embeddings: generate new sentence by replacing certain words with other words using contextualized word embeddings. The difference between this method and simple variation by Thesaurus is that the contextualized word embeddings do not only capture the static meaning of the target words but also their meaning in the given context. For example, the word ”apple” in ”I want to buy an apple”, and ”I want to buy an Apple MacBook Pro” are used to indicate two completely different objects.

Moreover, adding irrelevant material into the training data is another simple method presented to be useful, as it highlights the relevant features (22, 25). Based on the same principle, in their research, (14) experimented the “Majority class sentence addition (ADD)” method on their document level classification task. They add random sentences from a majority class document into a random position in a copy of the minority class document. When combine with BERT model it raise up to 21 percent of the result.

3.2.1 Application in our project

The dataset used in our project is highly imbalanced, for example, class ”O” - No propaganda techniques contribute to 65.5 percent of the training data, while rare classes, such as “Straw men”, ‘Bandwagon‘ and ‘Obfuscation, intentional vagueness, confusion‘, only represent less than 0.1%. These cause severe skewness in the data making the model ineffective in predicting rare class. In order to minimize that, Data Augmentation has been utilized in our project to up-sample rare classes.

Even though these methods could be useful, implementing them from scratch requires immerse effort and technical expertise. Thus in our project, we utilize Contextual Word Embeddings Augmentation (CWEA) from NLPAug [5] library, along with our own code for the ADD method. The CWEA from NLPaug, which uses “bert-base-uncased” under the hood, would take a word from a copy of the sentence and replaces with a similar word that captures the same context. While for our ADD implementation, we create a new sentence by concatenating a minority class sentence with a sentence from a majority class.

3.3 Using context

Augmenting input with its context is a powerful method to feed more relevant data into the model. Because our data are sentences or phrases in an article, and the input to our model are those sentences, we would

[2] https://www.tensorflow.org/api$_{d}ocs/python/tf/keras/preprocessing/text/Tokenizer$

[3] https://huggingface.co/transformers/model$_{d}oc/bert.html$

[4] https://huggingface.co/transformers/model$_{d}oc/bert.html$

[5] https://nlpaug.readthedocs.io/en/latest/index.html

like to extract some context relating to the input. Our model will then both analyze the context and the sequence, thus it will learn better and give good results.

For our model, we extract the preceding and succeeding sequence as the context. For getting the surrounding sentence context, we limit the maximum number of words to 130. The limit is set to avoid a longer context form having a greater influence on the final representation than the sentence itself. We select words from both sides in our context until the end of the sentence is reached or the maximum word limit is reached. The context and sequence will then be passed to BERT-based models.

4 SENTENCE CLASSIFICATION APPROACHES

For the task of sentence classification we experimented with several different transformer architectures, as well as one LSTM-based model. We go over some of the characteristics of the models employed.

4.1 BERT

BERT is a transformer-based model that is trained to predict masked words using data from available formal corpora such as Wikipedia and Book corpora (9). In short, the transformer-based model is a model where the transformer is a block of self attention and feed forward networks, consisting of an encoder responsible for encoding the input text and passing it to a decoder that then produces the result (24). The model is self attention based in that when it predicts the masked word, it takes into account the whole sentence not only from left to right, but also considering the words that come after the masked word. BERT also has a token segment, position embedding and segment embedding that allow the model to take as input several sentences and the classes to which the sentences belong (9). The self attention mechanism takes the input and looks for the connection of one word to other words in the same set (hence self attention). Whereas GPT2 is suitable for text generation, BERT is more suitable for working with different parts of text such as machine translation and text classification (9).

4.2 Contextual Transformers-based Model.

Contextual Transformers-based model is a model where we use two pre-trained Transformers architecture, which was developed by (18). We pass the sequence and its context to each of the Transformers. The outputs are sequence vector **S** and context vector **C**.There is an additional hidden layer on top of the context Transformer which reduces the dimension of context vector **C**. This resultant vector is then concatenated with **S** to get contextualized vector representation **V**.The additional hidden layer allows the classifier to give more attention to the propaganda sequence. The contextualized vector **V** is then passed to the classifier layer on top, which performs the final classification.

Sequence and Context Vectors We use pre-trained BERT to obtain the representation for the sequence as well as the context. The context is described in the Data Preprocessing section above. We limit the length of sentence contexts to 130 words. This is because, for sufficiently long propaganda spans, the exact meaning can be inferred directly from the text without any surrounding context.

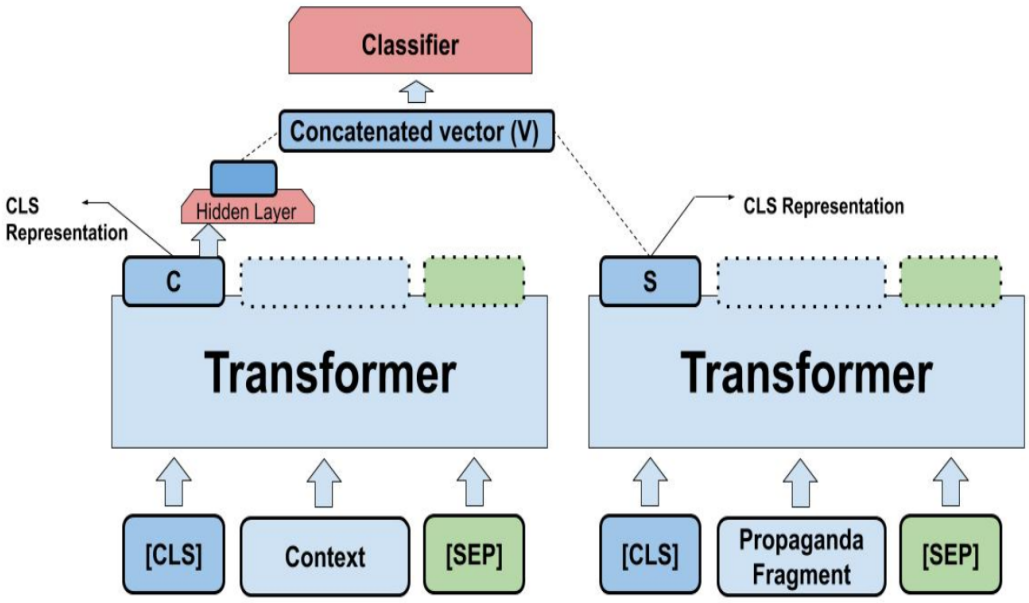

Figure 5. Contextual Transformers-based model.

4.2.1 Application in our project

Our project is strongly related to Natural Language Processing, and context play a crucial role. Applying that logic, we extract the context and feed it into the model together with the propaganda fragment. The result will be based on the contextualized vector representation, which gives better classification than the propaganda fragment alone.

4.3 Rationale-biased BERT

Rationale-biased BERT is an extension of the BERT architecture developed by Melamud et al. (16) for performing classification tasks in which there is little labeled training data available, but sub-sequences serving as explanations for label choices (rationales) can be acquired. We experimented with this architecture with the hypothesis that the incorporation of the rationales might improve accuracy on the imbalanced dataset. We also implemented class weights for the loss function to use with the model to introduce further bias towards rare labels. For the application of the model and any modifications to it we used the implementation provided by original authors to reduce needed development time.

4.3.1 Overview

The model developed by Melamud et al. (16) introduces a few key modifications to the original BERT architecture. Firstly, to improve runtime the model applies BERT to each sentence in the input text separately and represents the whole input as a weighted average of the individual sentence representations. Melamud et al. (16) experimented both with using uniform weights as well as learning the weights as attention.

Secondly, the model incorporates annotator rationales into the learning process by training the model to perform two tasks in parallel and calculating total loss as a sum of the losses of each task. These tasks are the original task of predicting the label of the input as well as predicting for each token in the input if it is part of a rationale or not.

Finally, the model uses the probabilities of the tokens being part of a rationale span calculated in the second task to find the previously mentioned sentence weights as the average of the probabilities for the tokens in that sentence.

4.3.2 Application in our project

In applying the model to our project we made certain modifications to the input and architecture of the original. Due to our input data consisting of single sentences we did not utilize the second extension of learning sentence weights as attention.

A point of consideration introduced by the difference in datasets between our project and the original experiments was the lack of rationales for all negative samples in our dataset, which introduced problems

as the model expected rationales for all inputs. For our implementation we opted for using empty spans for all the negative samples. We justify this by the original hypothesis by Melamud et al. (16) that the secondary task of predicting rationale membership of tokens infuses low-level information of the relevance of input words. We expect the empty rationales will help the model to identify when no relevant words are present, leading to a negative classification.

As a second modification we also implemented class weights, which were not part of the original model implementation. We expected this might improve prediction accuracy, as the classes in our dataset are both more numerous and more imbalanced compared to the data used in the original implementation (16)

4.4 Long Short Term Memory

4.4.1 Overview

Long short term memory is another popular model for text classification problem in Natural Language Processing field

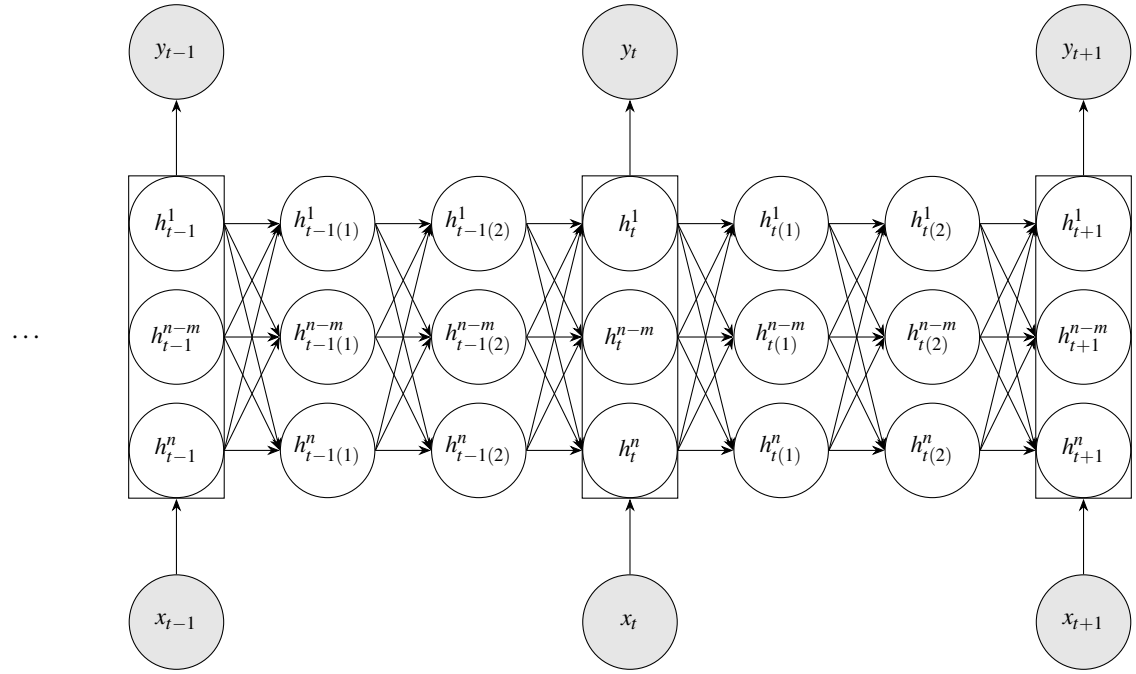

Figure 6. An RNN, with 3 hidden layers, estimating sequence of hidden states y_t from x_t. Typically the final output state, y_{t+1} can be treated as an encoding of the entire sequence.

Recurrent neural networks Recurrent neural networks (RNN) is a class of Neural Network architectures where the output of the network for the current token in a sequence depends on the output of hidden state for the previous tokens (11)..

Long short term memory and gated cells Long short term memory (LSTM) is a special kind of RNN. Despite RNN's theoretical ability to retain information in time, it has difficulty handling "long-term dependencies" in practice (11). LSTM cells have the same chain-like network structure, but each unit contains four different sub-networks - the "cell state", "input gate", "forget gate", and "output gate". The "forget gate" and "input gate" work as a filter to determine which information to remove from the cell state and which information to add to it. The "output gate" determines how the cell state and current token should be combined to produce the output, which is itself the output to the sigmoid function (11).

4.4.2 Application

Bi-directional LSTM and Sentiment features (4) presents that adding additional language features, such as Sentiment features and rhetorical feature along with BERT embedding could improve the classification result. Our Bi-directional LSTM model for the sentence classification was built based on the

same idea. We utilized Bi-directional LSTM model, which concatenate 2 hidden layers of the opposite directions to the same output. However, the length of each sentence in our dataset varies significantly, thus, if we padded all the sentences with 0, or a special character, the vectors would become highly skewed. Moreover, factors such as polarity and subjectivity might be more relevant to propaganda content than positivity and negativity. Therefore, instead of mapping each word token to their own sentiment score as (4), we utilize TextBlob [6] to calculate the positivity and negativity scores for the entire sentence. These score along with Bert embedding is then fed to our Bi-directional LSTM model.

5 SEQUENCE LABELING APPROACHES

Sequence labeling (also referred to as sequence tagging) is a type of pattern recognition task in machine learning which involves algorithmic assignment of categorical labels to each member of a sequence (19). Sequence tagging is a subtask of information extraction in text classification that locates and classifies tagged entities in a text. One example application of sequence labeling is named entity recognition (NER) which identifies and classifies named entities in a text. The named entities could be organizations, persons, locations, and times. Another example of sequence labeling task is part of speech tagging, which seeks to assign a part of speech such as a noun or verb to a word in an input sequence.

Early attempts to recognize patterns in sequences heavily relied on algorithmic methods which were probabilistic in nature and were based on statistical models such as hidden Markov models (HMM) (2). These models were built on the assumption that the choice of label for a particular word is directly dependent on the immediately neighboring words and thus the set of labels followed from a Markov chain (2). Lately however deep learning has been applied to these sets of questions.

In the context of this study the sequence tagging is used for propaganda token level classification. More specifically whereas in the first part we focused on classifying full sentences, in this part we take a granular approach to classify word sequences that characterise each propaganda technique.

5.1 BiLSTM-CRF

A BiLSTM-CRF network combines the previously described BiLSTM architecture with a probabilistic Conditional Random Field (CRF) model in order to utilize both previous input features and surrounding tags for the purposes of sequence tagging (13). The model uses a CRF layer to connect consecutive output layers of the BiLSTM component. This layer adds additional conditional information regarding the neighboring tags, not just the neighboring feature vectors. The final score used in determining the tag is then calculated as a sum of the score outputted by the BiLSTM and the transition score given by the CRF.

We later describe the use of this combined architecture in conjunction with contextual embeddings.

5.2 BERT and BiLSTM-CRF

In our experiments we utilized the previously described BERT model to generate word embeddings which we then used as input to a BiLSTM-CRF network for sequence tagging.

The model has previously been shown to produce state-of-the-art results when used in combination with various word embeddings (8). Due to resemblances to our problem we experimented with the architecture in hopes that it would be sufficient in modeling propaganda use. For easy application of the models we utilized the PyTorch-based NLP library Flair, which provides convenient access to state-of-the-art embeddings and models. (1).

6 EXPERIMENTS

We performed separate experiments for the tasks of sentence- and sequence-level classification. For the former of these we utilized both the expert annotated dataset obtained from the Propaganda Analysis Project, as well as the additional data collected by the team. For the latter task only the expert annotated dataset was used.

Experimentation for both tasks included setups with and without the augmentation techniques described earlier. Results from different approaches were then compared. All results were finally benchmarked against the results obtained by Martino et al. (15) in their propaganda classification.

[6] https://textblob.readthedocs.io/en/dev/index.html

6.1 Experimental Setup and Metrics

Model	Batch size	Sequence length	Epochs	Optimizer
Contextual BERT	32	128	7	Adam with learning rate 3e-5
Bi-LSTM with sentiment features	64	256	10	Adam 1e-3
pretrained BERT	32	256	5	Adam with learning rate 1e-5
unified LSTM	32	200	50	Adam
Rationale-biased BERT	32	256	7	Adam with learning rate 5e-6

Table 3. Model hyper-parameters

We used the PyTorch framework and pre-trained BERT models from huggingface's transformers[7] library. The hyper-parameters for each model can be found in Table 3.

As the data set is imbalanced, with 66 percent non-propaganda class, it is important to choose the right metric for comparing model performance. Accuracy is defined as the sum of the correct predictions divided by the sum of all predictions made on a data set. However for an imbalance data set accuracy as a performance measure, is not appropriately reflective of models' performance. This is due to the fact that the number of data points from the majority class (non propaganda in this data set) will largely outnumber the number of data points in the minority class. As a consequence even for models with low performance a high accuracy can be achieved depending on how large the class imbalance is. (Brownlee)

Precision quantifies the number of correct positive predictions and recall quantifies the number of correct positive predictions made out of all positive predictions that could have been made (Brownlee). Whereas precision only considers the correct positive prediction, recall also provides information about the missed positive predictions. For imbalance data set, recall is specifically helpful in highlighting coverage of the minority class (Brownlee). However neither precision nor recall alone can provide a complete guide to model performance. $F1_{measure}$ combines the two and is the metric variant most often used when learning from imbalance data set, defined as:

F-Measure = (2 * Precision * Recall) / (Precision + Recall)

The best model uses BERT technique along with controlling for class over laps and batch normalization to prevent overfitting. To apply BERT we have used pre-trained word embeddings.

All results are obtained using the validation set, containing 101 articles and 4265 sentences.

6.2 Classification results and discussion

6.2.1 Binary

Our Model	Precision	Recall	F1 Score
Contextual BERT	**75.31**	**62.15**	**68.73**
unified LSTM	34	28	31
Models from the paper	**Precision**	**Recall**	**F1 Score**
All-propaganda	23.92	100	38.61
BERT	63.20	53.16	57.74
BERT-Granu	62.80	55.24	58.76
BERT-Joint	62.84	55.46	58.91
MGN ReLU	60.41	61.58	60.98

Table 4. Sentence-Level Classification Result

Table 4 shows the results for Binary Sentence-Level classification, which asks to predict whether a sentence contains at least one propaganda technique. All-propaganda is a baseline which always output the propaganda class. Contextual BERT is our model and the rest are models from (15). Interestingly, our model yields significant performance improvements comparing to models from the paper. Increases come from all across the board, with 12.11% better precision, 0.57% increase in recall, resulting in a 7.75% improvement of the F1 Score (from the paper's best numbers).

[7] https://huggingface.co/transformers/

6.2.2 Multiclass

Model	F1 Score - Macro	F1 Score - Weighted
Bi-LSTM + Sentiment Features	8.4	
Contextual BERT	**14**	69
Pretrained BERT		64
Unified LSTM		61
Unified LSTM without Glove		62
Rationale-biased BERT	4.4	
Context & ADD Augmentation In Contextual BERT	14	70
Context & ADD Augmentation In Contextual (5 times) BERT	13	67

Table 5. Sentence-Level Classification Result

Table 5 shows the performance for six of our models. This task concerns multiclass classification. It is otherwise similar to binary classification task above, however, the models predict the exact propaganda technique of a sentence. We also include the non-propaganda class, thus there are 19 techniques in total. In this task, there are no results to compare from (15).

Augmentation does not help much with improving F1 Score. However, from our experience, it does help with predicting a bigger variety of propaganda techniques. In detail, Contextual BERT with Augmentation predicts the minority classes better than all other models, with 13 different classes predicted in total. Moreover, the F1 score starts decreasing more when we experiment training the model with a highly augmented dataset including data with 5 augmentation round for both ADD and Context method. One of the potential explanation for this result is the lack of augmented context in the training data. While our Contextual BERT model benefits significantly from the context, the augmentation only augmented the sentence itself, while the context is unchanged from the context of the original sentence. Thus, this might cause over-fitting.

The results does not look relatively good. We think that it is because of the ample amount of propaganda techniques and the evidently imbalanced dataset. Furthurmore, the sample size also needs to be bigger. We suspect that using even more robust architectures (like RoBERTa) can help with the results.

6.3 Sequence labeling results and discussion

Our Model	Precision	Recall	F1 Score
BERT + BiLSTM-CRF	79	7.2	7.3
LSTM	66	90	80
Models from the paper	**Precision**	**Recall**	**F1 Score**
BERT	39.57	36.42	37.90
BERT-Granu	39.26	35.48	37.25
BERT-Joint	43.08	33.98	37.95

Table 6. Sequence Tagging results

Table 6 summarizes the results obtained by our models in the sequence labeling task. Our BiLSTM-CRF based approach to predict all but the majority labels. This is apparent from the high precision, as very few samples are incorrectly labeled with a propaganda tag. The recall suffers as most propaganda spans are labeled as negative. Our pure LSTM approach performs admirably and obtains a very good F1 Score.

The results of the LSTM sequence tagger is measured such that the predicted tags are compared to the true tags, if each tagged prediction is same as each actual tagged label (that is every single item in 1 tagged prediction compared to every single item in 1 tagged label), then that counts as a 1 else it does not. The LSTM sequence tagger appears to be biased towards the 0 (tagged as non words) and this is also reflected in the high recall metric.

We suspect the poor performance of the BiLSTM-CRF model to be caused by the fact that the model is highly complex and not tuned for problems with multiple rare labels. This is explained by the fact that

the original implemenation of the model was tested with a very different dataset (13). Given more tuning we expect the model would perform better.

7 LESSONS LEARNED

We end off by reflecting on the lessons the team learned during the completion of this project. The section begins by briefly going over some more technical details regarding machine learning and model architectures that we think represent useful discoveries to remember. We then go on to discuss more general lessons related to time allocation, teamwork, and project management.

7.1 Technical takeaways

7.1.1 The importance of building a robust architecture

In our experience, using a more robust architecture is better in the context of this project. The problem we are solving is quite complex, with 19 propaganda techniques and an easily noticeable imbalance. Therefore, even professionals can not accurately annotate the techniques in a sentence. With our experiments, we conclude that having a more robust architecture helps in learning more complex features in a propaganda fragment. This results in the model being able to predict more classes. In the end, we managed to predict most of the techniques in the validation dataset, whereas in the beginning we only predicted a few most popular techniques.

7.2 General lessons

7.2.1 Difficulties related to black box models

In this study we analyzed the propaganda data and tried different deep learning methods. Our primary goal was to reach results comparable to the ones achieved by (15), thus we tried a variety of classification methods building the sophistication of the models gradually. This resulted in finding differences in the methods and defining one strongest candidate for solving this problem. The results indicate that BERT architecture is efficient for classifying text for propaganda samples. We conducted the optimization mainly based on trial and error.

This study as well provided understanding on how to approach NLP problems. Preprocessing text data can have an essential impact on performance of text classification models. Future work can consider possibilities for improvement which lie, among others, in utilizing other features such as part-of-speech tags, as well as more complex methods of handling imbalance data sets. Conducting experiments by optimizing the BERT architecture is another dimension that can be further explored.

While it seems that all major recent breakthroughs in machine learning come from the field of deep learning where complex black box models are employed, many old and relatively simply models still seem to enjoy widespread use in industry. From an idealized point of view present in academic courses this may seem silly as presented problems are often widely studied. Working with a real project shed some new light on this issue.

We found on multiple occasions that models which previously had successfully been applied to very similar problems refused to produce satisfactory results. We then challenges of black box models firsthand as we tried to tinker our way to results which more closely aligned with our expectations based on previous research. All of a sudden the reasons a lot of institutions still stubbornly rely on older models became more clear.

In it important to note that the problem the project sought to explore is of a very complex nature and as such the models chosen were a natural choice. The above observations do in no part represent a dislike for these models, simply a realization of the difficulties associated with them.

7.2.2 Interplay of models and data

Another interesting lesson learned by the team was the highly iterative and non-linear nature of the workflow in a project such as this. Much of the data collection and wrangling process was guided by the specific model choices that had been made. This also introduced a need for strong collaboration, as the models explored by different team members each had their own quirks that had to be considered when collecting and processing data.

A concrete example of this is the fact that one of the models explored required wider context information than the others, which meant that more care had to be taken to preserve the original context when collecting new data than was necessary for some of the other models.

8 APPENDIX

GitHub repository GitHub Repository for the project: https://github.com/linhnguyen222/DPS-Silo

REFERENCES

[1] Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

[2] Akbik, A., B. D. and Vollgraf, R., . (2018). Contextual string embeddings for sequence labeling. *In Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

[3] Bahdanau, D., C. J. S. D. B. P. and Bengio, Y. (2016). End-to-end attention-based large vocabulary speech recognition. *In 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4945–4949.

[4] Blaschke, V., Korniyenko, M., and Tureski, S. (2020). Cyberwalle at semeval-2020 task 11: An analysis of feature engineering for ensemble models for propaganda detection.

[Brownlee] Brownlee, J. How to calculate precision, recall and f measure for imbalanced classification.

[6] Cambria, E. and White, B. (2014). Jumping nlp curves: A review of natural language processing research.

[7] Ciaburro, G. and Joshi, P. (2019). *Python Machine Learning Cookbook: Over 100 Recipes to Progress from Smart Data Analytics to Deep Learning Using Real-World Datasets, 2nd Edition*. Packt Publishing.

[8] Dai, Z., Wang, X., Ni, P., Li, Y., Li, G., and Bai, X. (2019). Named entity recognition using bert bilstmcrf for chinese electronic health records.

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

[10] Durrani, N., D. F. S. H. B. Y. and Nakov, P. (2019). One size does not fit all: Comparing nmt representations of different granularities. *n Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*, pages 1504–1516.

[11] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. www.deeplearningbook.org.

[12] He, K., Z. X. R. S. and Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[13] Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint*.

[14] Juuti, M., Gröndahl, T., Flanagan, A., and Asokan, N. (2020). A little goes a long way: Improving toxic language classification despite data scarcity.

[15] Martino, G. D. S., Yu, S., Barrón-Cedeño, A., Petrov, R., and Nakov, P. (2019). Fine-grained analysis of propaganda in news articles. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5636–5646.

[16] Melamud, O., Bornea, M., and Barker, K. (2019). Combining unsupervised pre-training and annotator rationales toimprove low-shot text classification. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3884–3893.

[17] Mikolov, T., S. I. C. K. C. G. and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems*, pages 3111–3119.

[18] Paramansh Singh, Siraj Sandhu, S. K. A. M. (2020). newssweeper at semeval-2020 task 11: Context-aware rich feature representations for propaganda classification.

[19] Peters, M.E., A. W. B. C. and Power, R. (2017). Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

[20] Rabiner, L. and Juang, B. (1986). An introduction to hidden markov models. *ieee assp magazine*.

[21] Radford, A., N. K. S. T. and Sutskever, I. (2018). Improving language understanding by generative pre-training.

[22] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1).

[23] Sun, S., L. C. and Chen, J. (2017). A review of natural language processing techniques for opinion mining system. *Information fusion*, (36):10–25.

[24] Tenney, I., D. D. and Pavlick, E. (2019). Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

[25] Wong, S. C., Gatt, A., Stamatescu, V., and Mcdonnell, M. D. (2016). Understanding data augmentation for classification: When to warp? *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*.