CyberWallE at SemEval-2020 Task 11: An Analysis of Feature Engineering for Ensemble Models for Propaganda Detection

Verena Blaschke **Maxim Korniyenko** **Sam Tureski**
Seminar für Sprachwissenschaft
Eberhard Karls Universität Tübingen
first.last@student.uni-tuebingen.de

Abstract

This paper describes our participation in the SemEval-2020 task Detection of Propaganda Techniques in News Articles. We participate in both subtasks: Span Identification (SI) and Technique Classification (TC). We use a bi-LSTM architecture in the SI subtask and train a complex ensemble model for the TC subtask. Our architectures are built using embeddings from BERT in combination with additional lexical features and extensive label post-processing. Our systems achieve a rank of 8 out of 35 teams in the SI subtask (F1-score: 43.86%) and 8 out of 31 teams in the TC subtask (F1-score: 57.37%).

1 Introduction

Propaganda is defined as "the deliberate and systematic attempt to shape perceptions, manipulate cognitions, and direct behavior to achieve a response that furthers the desired intent of the propagandist" (Jowett and O'Donnell, 2019, p. 6). With the advent of rapid dissemination of news articles through online social media, automatic detection of biased or fake reporting has become more crucial than ever before.

This paper describes our participation in both of the subtasks offered by Da San Martino et al. (2020) in the SemEval 2020 shared task for the Detection of Propaganda Techniques in News Articles. The Span Identification (SI) subtask is a binary classification problem to discover propaganda at the token level, and the Technique Classification (TC) subtask involves a 14-way classification of propagandistic text fragments.

To address the SI task, we combine token-level BERT embeddings and linguistic features with bidirectional LSTMs and data post-processing methods. To address the TC task, we use BERT sequence embeddings and linguistic features to train a feed-forward neural network before post-processing is applied.

While top-scoring teams in a similar shared task (Da San Martino et al., 2019b) focus primarily on leveraging the performance of the pre-trained, context-dependent language model BERT, we find that further encoding of linguistic features offers a meaningful boost over both BERT and GloVe word embeddings alone. Majority voting and span merging in the SI task, as well as pre- and post-processing techniques to account for frequent occurrences of the REPETITION technique in the TC task result in further performance increases. Along with our best-performing model, we provide an extensive exploration into various embedding, feature and classifier combinations.

We release our code at github.com/cicl-iscl/CyberWallE-propaganda-detection.

2 Background

Before the introduction of the 2019 shared task on propaganda detection, approaches to propaganda recognition in news generally focused on classification at the article level. Rashkin et al. (2017) compare the language of "trusted news articles" from a well-known corpus with stories from "unreliable news sites". The publicly-available, feature-based tool Proppy, released by Barrón-Cedeno et al. (2019), ranks articles by their likelihood of containing propagandist content. These previous systems have been prone

to misclassification and lack of explainability, however, due to the assumption that articles from news sources deemed propagandist will always contain propaganda.

The release of the shared task on Fine-Grained Propaganda Detection by Da San Martino et al. (2019a) sparked the creation of numerous new detection systems, resulting in 14 system papers published. In contrast to this year's tasks, the 2019 participants encountered sentences, rather than tokens, in the binary classification round, and were asked to differentiate 18, rather than 14 classes, in the multiclass classification round.

The highest-scoring teams in that task integrate the language representation model BERT (Bidirectional Encoder Representations from Transformers) into their systems. BERT, developed by Devlin et al. (2019), was shown to reach state-of-the-art performance on eleven natural language processing tasks at the time of its release. The `base` version of the model for English contains 110 million parameters over 12 layers, produces embeddings of size 768 and encodes semantic information on a sub-token level.

A number of the previous year's teams use linguistic features in addition to BERT or other word embeddings. Gupta et al. (2019) employ six categories of features, ranging from topic representations to layout-derived features like sentence length. Ferreira Cruz et al. (2019) work with simple linguistic features (e.g. punctuation frequency, sentence length) as well as type-token ratios and TF-IDF scores for uni- and bigrams. Al-Omari et al. (2019), by contrast, employ the output of a twitter-based sentiment analysis tool. Finally, Alhindi et al. (2019) and Li et al. (2019) both use features returned by the LIWC (Linguistic Inquiry and Word Count) text analysis software (Pennebaker et al., 2001).

3 Dataset

The corpus used is an extension of the corpus described in Da San Martino et al. (2019b). The articles, pulled from around 50 news outlets, have been annotated into specific techniques at the fragment level; several classes with few instances are then condensed under single labels, such as in WHATABOUTISM, STRAW MEN, RED HERRING. An example of a fragment containing an instance of the class LOADED LANGUAGE:

$$\text{not } \overbrace{\text{looking as though Trump killed his grandma}}^{\text{loaded language}} .$$

Descriptions of each class can be found in Da San Martino et al. (2020).

Class imbalance exists strongly in the dataset. A minority of sentences are assessed to contain propaganda, and fragments that do contain propaganda are classed as only 3 of the 14 labels around 60% percent of the time. It is important to note that some of the techniques can be identified on the fragment level (e.g. NAME CALLING, FLAG-WAVING), while others often require a broader context of the discourse at hand to be discovered (e.g. REPETITION, RED HERRING).

4 The Span Identification (SI) system

4.1 System overview

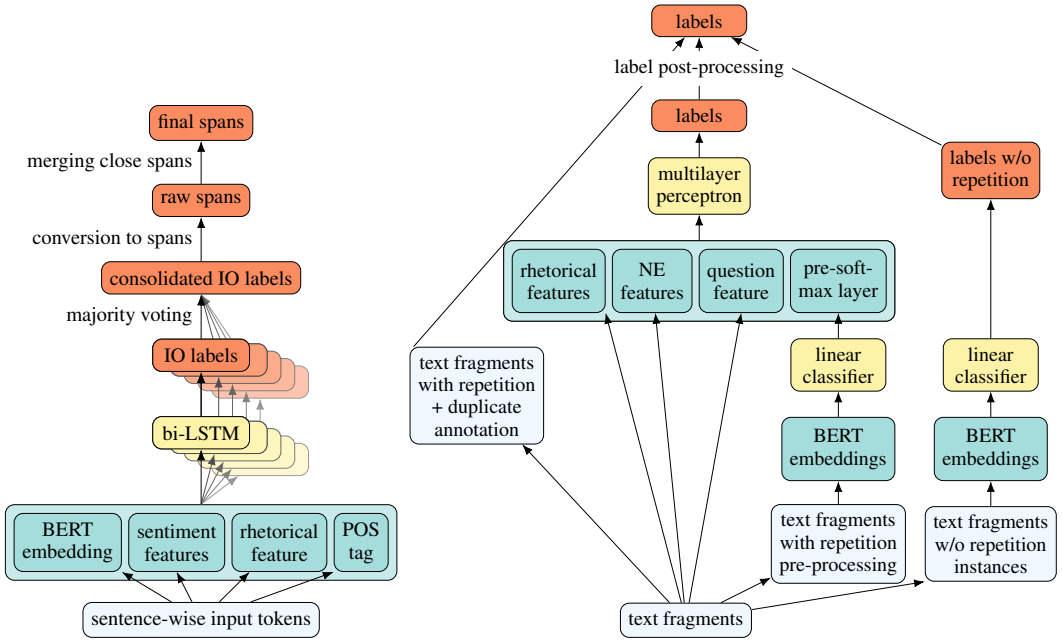
Our architecture for the SI subtask is built around a bidirectional LSTM (Graves and Schmidhuber, 2005). We split the news articles into sentences (or sentence fragments) and tokenize these sentences. We convert the target spans into corresponding token-wise binary labels (*I* - inside a span, *O* - outside a span).

Each token is represented as a vector consisting of a BERT embedding (Devlin et al., 2019) concatenated with two sentiment values (indicating how positive/negative the token is), a binary feature indicating whether the token is part of a rhetorically salient phrase, and a one-hot encoded POS tag representation. This input is fed into a bidirectional LSTM that predicts a label for each token. We carry this out five times in total to abstract over the effect caused by random initializations. The five sets of predicted labels are consolidated via majority voting and then converted into spans. In a final step, we remove short gaps between spans by merging the two surrounding spans into one larger span.

Figure 1a illustrates this architecture.

4.2 Experimental setup

We use the Natural Language Toolkit 3.2.4 (Bird et al., 2009) for sentence splitting and additional heuristics to divide long sentences into shorter fragments by splitting them along quotation marks, semicolons and



(a) The span identification system.

(b) The technique classification system.

Figure 1: Our system architectures.

commas when possible. The maximum sentence length is 35 tokens (long enough to fully include most input fragments, but short enough to reduce vanishing gradient issues). We use spaCy's pretrained model for English (version 2.0.0) as tokenizer.

We use HuggingFace's pretrained BERT embeddings (Wolf et al., 2019), specifically the uncased base version. We initially also experiment with the cased version, as well as small (6B tokens, 100 dimensions) and large (42B tokens, 300 dimensions) GloVe embeddings (Pennington et al., 2014).

The sentiment features are based on SentiWordNet 3.0.0 (Baccianella et al., 2010).[1] Each token is associated with a positive and a negative sentiment score, each bounded between 0 and 1. Tokens that are not in the dataset (even in their lemmatized form as produced using spaCy's lemmatizer) are assigned the value $[0, 0]$. Polysemous tokens that are assigned several different scores by SentiWordNet are assigned the average of these values.

The feature for rhetorically salient tokens is produced using the Arguing Lexicon[2] (Somasundaran et al., 2007), which contains phrase patterns for 17 different rhetorical strategies such as *appeal to authority* or *generalization*. Each token in a phrase contained in this lexicon is assigned the value 1, all others 0.

The last feature is each token's one-hot encoded 15-dimensional POS tag, as determined by spaCy.

We use Keras 2.2.5 with a Tensorflow 1.14 backend to build the LSTM. We work with a batch size of 128, class weights (1.0 for *O*, 6.5 for *I*), 512 hidden units and a dropout rate of 25%. We use the Adam optimizer (learning rate: 0.001) to minimize the cross entropy across ten epochs.

After predicting the labels and transforming them into spans, we merge all spans that are separated by less than 25 characters. We also experiment with using minimum gap lengths of up to fifty characters.

During the development stage, we use all of the training data as training input to tune parameters based on development data performance. For the test set submissions, we extract the development data labels from the input for the TC task and concatenate the training and development data as model input.

The predictions are evaluated using character-level precision, recall and F1-score for the propagandist fragments. More details can be found in the task description paper (Da San Martino et al., 2020).

[1] https://github.com/aesuli/SentiWordNet

[2] http://mpqa.cs.pitt.edu/lexicons/arg_lexicon/

5 The Technique Classification (TC) system

5.1 System overview

The foundation of our strategy for the TC subtask revolves around generating accurate predictions for the large and qualitatively unique REPETITION class. Our key innovation in resolving this issue is to break prediction generation into several sub-models.

For the *base model*, we use lexical information encoded in each span to generate a first set of predictions (see the central branch of Figure 1b). We modify the input fragments slightly to represent repetition information. If a fragment has the REPETITION label (in the case of training data) or is repeated elsewhere in the same news article (development/test data), we append a copy of this sequence to itself. Then, we train a linear classifier on all fragments, which are represented using BERT embeddings. Afterwards, we take the pre-softmax layer from this model and concatenate it with additional features indicating whether a rhetorical technique is contained in the fragment, whether it mentions geopolitical entities or groups of people (two binary features), and whether it contains a question mark. These newly created vectors are used as input to a multilayer perceptron, which generates this model's final predictions.

For our *repetition model*, we isolate two simple features which are then assigned to every instance of the data—how often the segment is repeated in the article and whether it is the first such repetition (the left branch of Figure 1b). This post-processing step classifies instances as REPETITION if the given normalized span has at least one repetition in the article and it does not represent the first occurrence of it in the text. These predictions override the base model's predictions.

If the base model predicts a REPETITION that is not confirmed by this post-processing step, this instance is re-classified by a third model, which we call the *alternative model* (the right branch of Figure 1b). It is a linear classifier which uses BERT embeddings of the data as input and is trained using all training data except the instances of the REPETITION class. As a result, the model always predicts labels of only the remaining thirteen classes.

The last step of this system is handling duplicates. Some of the fragments have several labels and appear as multiple identical instances in the data. We use the prediction from the alternative model to label the duplicate instance. If these predictions are identical or if there are three or more identical spans, we use the first model's runner-up predictions instead.

5.2 Experimental setup

The *base model* and *alternative model* use HuggingFace's pretrained `bert-base-uncased` vectors with a sequence classification head on top, `BertForSequenceClassification`. The maximum sequence length is 200 tokens, as determined by HuggingFace's pretrained BERT tokenizer. Our optimizer is `BertAdam` with a warmup rate of 0.1. The base model is trained for two epochs with a learning rate of 1e-5 and a batch size of 12. The alternative model is trained for four epochs with a learning rate of 2e-5 and a batch size set to 32. This settings are based on the parameters recommended in Devlin et al. (2019).

The rhetorical technique feature is generated using the Arguing Lexicon (Somasundaran et al., 2007), as in the previous task. We create the two named entity features using spaCy's pretrained named entity tagger (version 2.0.0) based on its predictions for 'NORP' (nationalities, religious or political groups) and 'GPE' (countries, cities, states).

In order to build the base model's feed-forward neural network, we use Keras 2.2.5 for Tensorflow 1.14. The best results on the development set are achieved using a hidden layer with 128 units and a dropout rate of 25%. This multilayer perceptron is trained for 15 epochs using a batch size of 128 instances. To minimize the cross entropy, we use the Adam optimizer with a learning rate set to 0.001.

This final base model is chosen after a series of experiments exploring different input representations and machine learning models.

As alternative BERT embeddings, we also test the cased version of HuggingFace's pre-trained base model. Additionally, we experiment with alternatives to the pre-softmax layer of `BertForSequenceClassification`. To embed a sequence with BERT, it needs to be pre- and suffixed with two special embeddings, `[CLS]` and `[SEP]`. BERT's embedding for `[CLS]` in its final layer represents the entire sequence (Devlin et al., 2019). We extract this embedding and concatenate

Configuration	F1-score	Precision	Recall
GloVe-6B-100D (uncased)	0.3148	0.2015	0.7307
GloVe-6B-100D + SentiWordNet (SWN)	0.3171	0.2052	0.7059
GloVe-6B-100D + Arguing Lexicon (AL)	0.3105	0.1989	**0.7359**
GloVe-6B-100D + SWN + AL	0.3223	0.2097	0.7026
GloVe-6B-100D + POS	0.3294	0.2169	0.6897
GloVe-6B-100D + SWN + AL + POS	0.3247	0.2132	0.6875
GloVe-42B-300D (uncased)	0.3453	0.2472	0.5792
BERT-base-cased	0.3759	0.2792	0.5783
BERT-base-uncased	0.3869	0.2884	0.5965
BERT-base-uncased + SWN	0.3893	0.2966	0.5784
BERT-base-uncased + AL	0.3921	0.3066	0.5507
BERT-base-uncased + SWN + AL	0.3912	0.2995	0.5687
BERT-base-uncased + POS	0.3912	0.2939	0.5915
BERT-base-uncased + SWN + AL + POS ("full")	0.3949	0.3025	0.5699
full + majority voting (across 5 initializations)	0.4065	0.3110	0.5901
full + majority voting + span merging	**0.4115**	**0.3136**	0.6026

Table 1: Different embeddings and feature combinations for the development set of the SI task. The results are mean values across five runs. The configuration for our final model is in italics.

it with the final-layer representations of the first ten actual tokens and feed this input to three different machine learning architectures. We use the Keras 2.2.5 implementation of a convolutional neural network (CNN) (3 layers of size 128, max pooling, dense layer) as well as the KimCNN (a CNN architecture developed for NLP tasks by Kim (2014)) and a multilayer-perceptron (same configuration as for our actual base model).

In a further set of experiments, we compare different classifiers on top of the pre-softmax layer from `BertForSequenceClassification`. Firstly, we inspect the actual output of `BertForSequenceClassification`. We also test out the following classifiers: a decision tree with extreme gradient boosting (Chen and Guestrin, 2016), a single-layer perceptron (with the same set-up as the neural net in the base model, only without the hidden layer) and a support vector machine (SVM) with the default configuration of Scikit-learn version 0.22.2 (Pedregosa et al., 2011).

We also explore different input features (and combinations thereof) in an architecture otherwise identical to our actual base model. We use four additional binary named entity features (also created with spaCy) that indicate whether a text fragment contains organizations, names of people, cardinal numbers, and dates. Moreover, we test binary features indicating whether a fragment contains tokens related to the US (*America*) or to the REDUCTIO AD HITLERUM class (*reductio*). Furthermore, we use the Natural Language Understanding Tool by IBM Cloud[3] to get each fragment's anger, disgust, fear, joy and sadness ratings. Each value is between 0.0 and 1.0 and stored as an individual feature. We also examine two numerical features, one encoding the sequence length (in tokens) and the other encoding how often a text fragment is repeated in the given news article.

This subtask is scored based on micro-averaged F1-score.

6 Results

6.1 Span identification

Our best model achieves an F1-score of 42.39% on the development set and 43.86% on the test set (rank 8 of 35). In this subsection, we describe the findings of our feature ablation experiments. The detailed

[3] `https://cloud.ibm.com/catalog/services/natural-language-understanding`

Configuration	F1-score
BERT-base-cased + pre-softmax + multilayer perceptron (MLP)	0.5485
BERT-base-uncased + pre-softmax + MLP	0.5635
BERT-base-uncased embeddings of `[CLS]` & the first 10 tokens + MLP	0.5870
rep + BERT-base-uncased + pre-softmax + MLP	0.6341
rep + BERT-base-uncased + pre-softmax + two named entity classes (NE-2) + MLP	0.6322
rep + BERT-base-uncased + pre-softmax + Arguing Lexicon (AL) + MLP	0.6350
rep + BERT-base-uncased + pre-softmax + question mark feature (Q) + MLP	0.6359
rep + BERT-base-uncased + pre-softmax + NE-2 + AL + Q + MLP ("base model")	0.6359
base model + label post-processing	**0.6640**

Table 2: Different embeddings, feature combinations and models for the development set of the TC task. Where the tokens for the BERT embeddings are not specified, all token embeddings are used as input to the linear classifier, whose pre-softmax layer of the linear classifier is referred to as 'pre-softmax.' The results are mean values across five runs. The configuration for our final model is in italics. The full version of this table can be found in the appendix.

results can be found in Table 1.

Using the simplest token embedding (100-dimensional GloVe) yields a low precision but very high recall score (20.15% compared to 73.07%). Using larger (300-dimensional GloVe) or more sophisticated embeddings (BERT) lowers the system's recall score while increasing precision enough to also increase the F1-score. Adding linguistic features generally has similar effects. It should be noted, however, that while adding features mostly raises the overall score, the strength of this effect and the best composition of feature combinations vary greatly across runs.

We observe a significant increase in performance by adding post-processing steps that make the model both more robust to initialization differences and improve the predictions. Majority voting across 5 model initializations raises the F1-score by more than one percentage point. During almost all runs, we observe that the predictions after majority voting have higher precision and recall scores than any of the individual predictions that went into this voting process. Merging nearby spans also yields better results for both metrics. We achieve good results for a range of minimum gap lengths (between 10 and 40 characters); optimum values within this range do not generalize between model initializations.

6.2 Technique classification

Our final model for the TC task achieves a 66.42% F1-score on the development set and 57.37% on the test set, placing us eighth out of 31 teams. Table 2 presents part of our feature ablation study; the full table can be found in the appendix.

We notice that our model design choices influence the model's performance at all stages of our system architecture. Firstly, when choosing embeddings to represent the input fragments, we observe that, like in the SI task, the uncased BERT embeddings yield better results than the cased versions. We also note that feeding the output of `BertForSequenceClassification`'s pre-softmax layer into another machine learning model leads to a higher F1-score than feeding in BERT embeddings (of `[CLS]` and the first ten tokens) directly: ca. 63% versus 31.89-58.7%. The latter range is so large because the model choice for that set-up matters: the KimCNN yields significantly better results than a standard CNN, but it is still outperformed by the multilayer perceptron.

Furthermore, we observe that the repetition pre-processing step markedly improves final results. In the base model (sans additional features), adding this step boosts the overall F1-score by 7 percentage points to 63.41%, and the REPETITION F1-score alone from 12.36 to 65.5%. We see that the choice of machine learning model has a modest effect for this architecture: Using an SVM or a multilayer perceptron yields marginally better results than using the linear classifier whose layer is fed into the other models, which

Technique	Proportion (dev)	SI	TC		
		Recall (dev)	F1-score (dev)	F1-score (test)	Change
Loaded language	30.6	70.6	76.6	74.7	− 1.9
Name calling, labeling	17.2	63.0	81.0	70.9	− 10.1
Repetition	13.6	63.8	73.3	47.7	− 25.6
Flag-waving	8.2	74.4	73.7	54.4	− 19.3
Exaggeration, minimisation	6.4	57.6	52.7	28.3	− 24.4
Doubt	6.2	46.9	53.8	58.7	+ 4.9
Appeal to fear/prejudice	4.4	62.9	30.6	39.9	+ 9.3
Slogans	3.7	74.6	51.4	39.4	− 12.0
Whataboutism, straw men, red herring	2.7	36.8	0.0	0.0	0.0
Black-and-white fallacy	2.1	46.9	21.4	23.7	+ 2.3
Causal oversimplification	1.7	50.7	21.1	15.4	− 5.7
Thought-terminating clichés	1.6	51.4	17.4	23.8	+ 6.4
Appeal to authority	1.3	49.9	18.2	14.7	− 3.5
Bandwagon, reductio ad hitlerum	0.5	8.4	22.2	12.2	− 10.0
All classes	100.0	63.8	66.4	57.4	− 9.0

Table 3: Technique-level breakdown of model performances for both subtasks. The proportions, recall values and F1-scores are percentages. The change of the F1-score is given in percentage points.

in turn outperforms the decision trees with extreme gradient boosting and the single-layer perceptron slightly.

Adding additional features also has a slight effect on the outcome. Based on mean values across five model initializations, the bag-of-words features (*America*, *reductio*), the emotion feature and NE-2 slightly decrease the overall F1-score, the sequence length feature and NE-6 do not change the result, and the repetition count, the rhetorical phrase lexicon and the question mark feature as well as a combination of NE-2/6, the rhetorical lexicon and the question mark feature improve the score. Interestingly, combining the features does not have an additive effect; some features work better in combination with others than on their own and vice versa. Both the question mark feature on its own as well as together with NE-2 and the rhetorical lexicon score highest (63.59%), but the scores of individual runs with the multi-feature model are more stable. Further combinations of features are omitted from the table, but lead to results in the same range as the presented features and score lower than the base model set-up.

The propaganda technique we paid the most attention to while building the system, both during pre- and post-processing, is REPETITION. This allows our team to achieve the best development phase result among all teams for REPETITION predictions (73.3%). The repetition post-processing steps helps to boost model performance by almost 3 F1 percentage points beyond the base model, allowing our system to reach an overall F1 score of 66.40%.

6.3 Model performance by propaganda technique

Our models do not perform equally well for each propaganda technique. The scoreboard for the TC subtask includes F1-scores for each technique, and with the help of the gold-standard labels for the development data for the TC subtask, we can calculate the recall score for each propaganda technique. Table 3 shows this technique-level breakdown for both subtasks.

In the SI subtask, the largest classes all have high recall scores (at least 63%), likely due to their being well-represented in the data and because they tend to be very short text fragments (making it more probable to retrieve (nearly) complete spans).

The results for the smaller classes are mixed. The recall score for WHATABOUTISM, STRAW MEN, RED HERRING is relatively low, and our technique classification system does not produce any predictions for the this class. This might be due to the fact that such derailment techniques tend to be based on the discourse structure rather than more local syntactic or semantic patterns.

Our technique classification system also demonstrates a clear tendency towards better predictions for the classes representing large proportions of the training data, which is expected because these classes are weighted more heavily in the micro-averaged F1 metric. Each of three most frequent classes (LOADED LANGUAGE, NAME CALLING, LABELING and REPETITION) reaches an F1-score of above 70% during the development phase (Table 3).

Most moderately frequent classes (DOUBT, EXAGGERATION, MINIMISATION, FLAG-WAVING, and SLOGANS) score at least 50% F1 in each category. The only exception is for predictions on APPEAL TO FEAR/PREJUDICE. Despite being a frequent class, predictions here top out at an F1 score of 30.6% (on the development set). As mentioned above, there are no predictions for WHATABOUTISM, STRAW MEN, RED HERRING, but the five least frequent classes reach F1-scores of $20 \pm 3\%$ during the development phase.

The confusion matrix of the development set predictions for the TC subtask can be found in the appendix (Figure 2). Quite often, the model misclassifies instances of other classes as LOADED LANGUAGE, the most frequent class in the data. For example, 36% of APPEAL TO FEAR/PREJUDICE instances were classified as LOADED LANGUAGE.

The model performs worse on test data, presumably due to having been overtuned on the development set or due to different label distributions in the development and test sets. Only four classes manage to achieve better results in the test phase. The most significant improvement is achieved in the APPEAL TO FEAR/PREJUDICE class, with an F1-score increase of more than 9 percentage points. The most frequent class, LOADED LANGUAGE, remains stable in test runs; its F1-score decreases only by 1.9 percentage points. The individual F1-score of 5 classes drop by more than 10 percentage points. All those classes are quite frequent and represent almost half (49.1%) of the instances of the development data. As a result, the overall F1-score on test data drops by 9 percentage points.

The most significant decrease occurs in the REPETITION class with a drop of over 25 percentage points. All teams demonstrate significantly worse test phase results for this technique compared to the development phase. Only two teams out of 31 manage to score at least 50% for this class, while only five more teams manage results above 30%.

7 Conclusion

While fair results on propaganda detection can be achieved with BERT embeddings alone, we further improve the performance on this task through the addition of linguistic features and pre- and post-processing techniques. Our error analysis indicates that majority voting across several runs additionally increases the F1-score and stability of the model.

Future work can proceed in different directions. In our experiments for the SI subtask, we applied majority voting across different runs of the same model, but it could be beneficial to use majority voting to combine predictions from the high-recall GloVe-based model with the high-precision BERT-based model. Another potential source of improvement for either subtask is retraining the BERT model, similarly to Mapes et al. (2019) and Yoosuf and Yang (2019). Finally, training a joint system for both subtasks, as Da San Martino et al. (2019b) have done, may result in more accurate predictions.

Acknowledgments

We thank Dr. Çağrı Çöltekin for useful discussions and his guidance throughout this project.

References

Hani Al-Omari, Malak Abdullah, Ola AlTiti, and Samira Shaikh. 2019. JUSTDeep at NLP4IF 2019 task 1: Propaganda detection using ensemble deep learning models. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 113–118, Hong Kong, China, November. Association for Computational Linguistics.

Tariq Alhindi, Jonas Pfeiffer, and Smaranda Muresan. 2019. Fine-tuned neural models for propaganda detection at the sentence and fragment levels. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 98–102, Hong Kong, China, November. Association for Computational Linguistics.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204.

Alberto Barrón-Cedeno, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: A system to unmask propaganda in online news. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9847–9848.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: Analyzing text with the Natural Language Toolkit*. O'Reilly Media, Inc.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Giovanni Da San Martino, Alberto Barron-Cedeno, and Preslav Nakov. 2019a. Findings of the NLP4IF-2019 shared task on fine-grained propaganda detection. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 162–170.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019b. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*, EMNLP-IJCNLP 2019, Hong Kong, China, November.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. SemEval-2020 task 11: Detection of propaganda techniques in news articles. In *Proceedings of the 14th International Workshop on Semantic Evaluation*, SemEval 2020, Barcelona, Spain, December.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

André Ferreira Cruz, Gil Rocha, and Henrique Lopes Cardoso. 2019. On sentence representations for propaganda detection: From handcrafted features to word embeddings. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 107–112, Hong Kong, China, November. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, 18(5-6):602–610.

Pankaj Gupta, Khushbu Saxena, Usama Yaseen, Thomas Runkler, and Hinrich Schütze. 2019. Neural architectures for fine-grained propaganda detection in news. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 92–97, Hong Kong, China, November. Association for Computational Linguistics.

G. Jowett and V. O'Donnell. 2019. *Propaganda and Persuasion*. Sage, 7th edition.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Jinfen Li, Zhihao Ye, and Lu Xiao. 2019. Detection of propaganda using logistic regression. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 119–124, Hong Kong, China, November. Association for Computational Linguistics.

Norman Mapes, Anna White, Radhika Medury, and Sumeet Dua. 2019. Divisive language and propaganda detection using multi-head attention transformers with deep learning BERT-based language models for binary classification. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 103–106, Hong Kong, China, November. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue*, volume 6.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Shehel Yoosuf and Yin Yang. 2019. Fine-grained propaganda detection with fine-tuned BERT. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 87–91.

Appendix A. Supplemental Material

Configuration	F1-score
BERT-base-cased + pre-softmax + multilayer perceptron (MLP)	0.5485
BERT-base-uncased + pre-softmax + MLP	0.5635
BERT-base-uncased embeddings of `[CLS]` & the first 10 tokens + CNN	0.3189
BERT-base-uncased embeddings of `[CLS]` & the first 10 tokens + KimCNN	0.4835
BERT-base-uncased embeddings of `[CLS]` & the first 10 tokens + MLP	0.5870
repetition pre-processing (rep) + BERT-base-uncased + linear classifier	0.6322
rep + BERT-base-uncased + pre-softmax + XGBoost	0.6219
rep + BERT-base-uncased + pre-softmax + single-layer perceptron	0.6312
rep + BERT-base-uncased + pre-softmax + SVM	0.6341
rep + BERT-base-uncased + pre-softmax + MLP	0.6341
rep + BERT-base-uncased + pre-softmax + America + MLP	0.6312
rep + BERT-base-uncased + pre-softmax + reductio + MLP	0.6322
rep + BERT-base-uncased + pre-softmax + emotion + MLP	0.6331
rep + BERT-base-uncased + pre-softmax + sequence length + MLP	0.6341
rep + BERT-base-uncased + pre-softmax + repetition count + MLP	0.6350
rep + BERT-base-uncased + pre-softmax + two named entity classes (NE-2) + MLP	0.6322
rep + BERT-base-uncased + pre-softmax + six named entity classes (NE-6) + MLP	0.6341
rep + BERT-base-uncased + pre-softmax + Arguing Lexicon (AL) + MLP	0.6350
rep + BERT-base-uncased + pre-softmax + question mark feature (Q) + MLP	0.6359
rep + BERT-base-uncased + pre-softmax + NE-6 + AL + Q + MLP	0.6350
rep + BERT-base-uncased + pre-softmax + NE-2 + AL + Q + MLP (“base model”)	0.6359
base model + label post-processing	**0.6640**

Table 4: Different embeddings, feature combinations and models for the development set of the TC task. Where the tokens for the BERT embeddings are not specified, all token embeddings are used as input to the linear classifier, whose pre-softmax layer of the linear classifier is referred to as ‘pre-softmax.’ The results are mean values across five runs. The configuration for our final model is in italics.

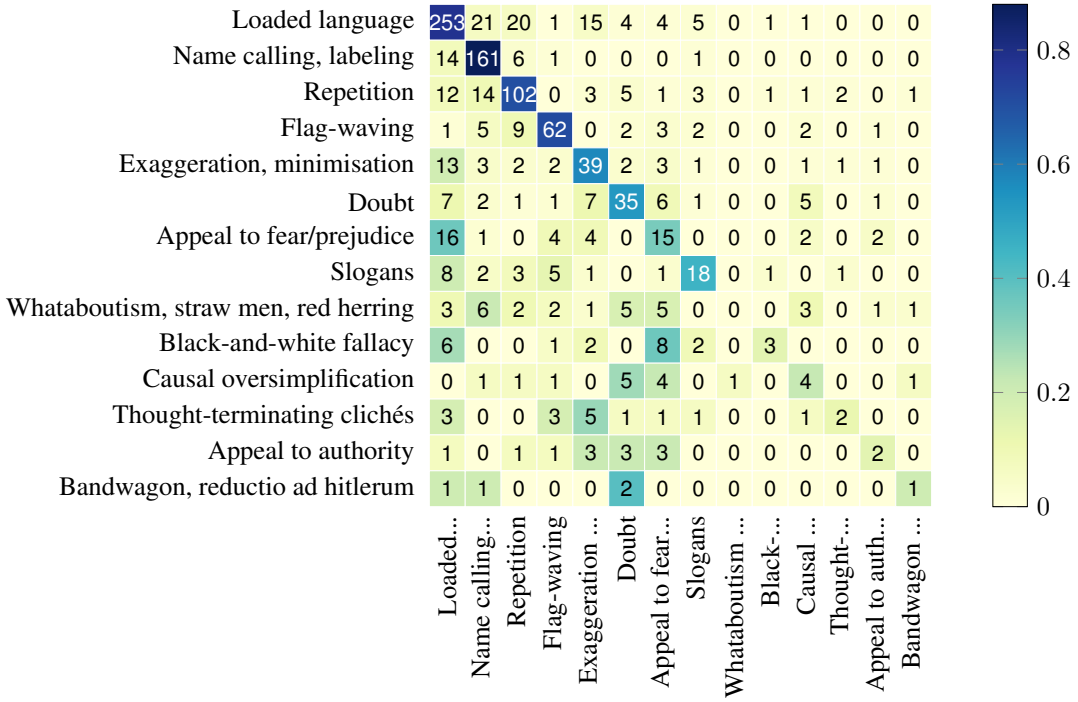


Figure 2: Confusion matrix for technique classification predictions on the development set. Rows represent true labels and columns predicted labels. The numbers in the matrix are absolute instance counts. The colour scale indicates each classification's frequency relative to its row (true label).