ApplicaAI at SemEval-2020 Task 11: On RoBERTa-CRF, Span CLS and Whether Self-Training Helps Them

Dawid Jurkiewicz* **Łukasz Borchmann*** **Izabela Kosmala** **Filip Graliński**

Applica.ai Zajęcza 15, 00-351 Warsaw, Poland
firstname.lastname@applica.ai

Abstract

This paper presents the winning system for the propaganda Technique Classification (TC) task and the second-placed system for the propaganda Span Identification (SI) task. The purpose of TC task was to identify an applied propaganda technique given propaganda text fragment. The goal of SI task was to find specific text fragments which contain at least one propaganda technique. Both of the developed solutions used semi-supervised learning technique of self-training. Interestingly, although CRF is barely used with transformer-based language models, the SI task was approached with RoBERTa-CRF architecture. An ensemble of RoBERTa-based models was proposed for the TC task, with one of them making use of Span CLS layers we introduce in the present paper. In addition to describing the submitted systems, an impact of architectural decisions and training schemes is investigated along with remarks regarding training models of the same or better quality with lower computational budget. Finally, the results of error analysis are presented.

1 Systems Description

Systems proposed for both SI and TC tasks (Da San Martino et al., 2020) were based on RoBERTa model (Liu et al., 2019) with task-specific modifications and training schemes applied.

Central motif behind our submissions is a commonly used semi-supervised learning technique of self-training (Yarowsky, 1995; Liao and Veeramachaneni, 2009; Liu et al., 2011; Wang et al., 2020), sometimes referred to as incremental semi-supervised training (Rosenberg et al., 2005) or self-learning (Lin et al., 2010). In general, these terms stand for a process of training an initial model on manually annotated dataset first and using it to further extend the train set by means of annotating other dataset automatically. Usually only a selected subset of auto-annotated data is used, however no selection of high-confidence examples nor loss correction for noisy annotations is performed in our case. This is the reason why it can be considered as a simplification of mainstream approaches—the *naïve* self-training.

1.1 Span Identification

The problem of span identification was treated as a sequence labeling task, which in a case of Transformer-based language models is often solved by means of classifying selected sub-tokens (e.g. first BPE of each word considered) with or without applying LSTM before the classification layer (Devlin et al., 2019).

Although pre-Transformer sequence labeling solutions exploited CRF layer in the output (Huang et al., 2015; Lample et al., 2016), this practice was abandoned by the authors of BERT (Devlin et al., 2019) and subsequent researchers developing the idea of bidirectional Transformers, with rare exceptions, such as Souza et al. (2019) who used BERT-CRF for Portuguese NER. Contrary to the above, we approached Span Identification task with RoBERTa-CRF architecture.

Impact of this decision will be discussed in Section 2 along with remarks regarding training models of the same or better quality with lower computational budget in an orderly fashion. In contrast, the following narrative aims at a faithful description of the actual way the model we used was trained.

* Equal contribution. Author ordering determined by a coin flip.

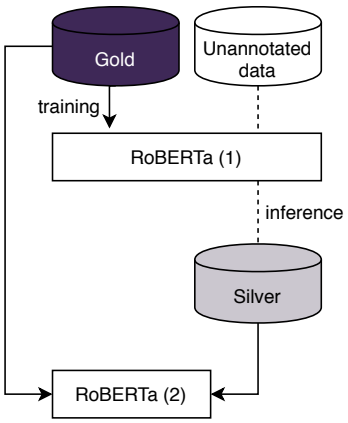

Figure 1: Self-training stands for a process of training an initial model on manually annotated dataset first and using it to further extend train by means of annotating other dataset automatically.

Hparam	SI	TC
Dropout	.1	
Attention dropout	.1	
Max sequence length	256	256
Batch size	8	16
Learning rate	5e-4	2e-5
Number of steps	60k	20k
Learning rate decay	–	
Weight decay	–	.01
Momentum	.9	–
Optimizer	SGD	AdamW
Loss	Viterbi	BCE

Table 1: Optimizers and hyperparameters used for both fine-tuning RoBERTa and training additional parameters.

Hparam	SI	TC
Dropout	.0	
Attention dropout	.0	
Batch size	16	16

Table 2: Hyperparameter overwrites for self-training.

Recipe Take one pretrained RoBERTa $_{\text{LARGE}}$ model, add CRF layer and train until progress is no longer achieved with Viterbi loss, SGD optimizer and hyperparameters defined in Table 1. Use the best-performing model to annotate random 500k OpenWebText[1] sentences automatically. Train the second model on both original (gold) dataset and autotagged one (silver) with hyperparameters defined in Table 1. Repeat the procedure two more times with the best model from previous step, hyperparameters from Table 2 and another OpenWebText sentences.

Note that hyperparameters were indeed not overwritten during the first self-training iteration. Scores achieved by the best-performing models were respectively 50.91 (without self-training) and 50.98, 51.45, 52.24 in consecutive self-training iterations.

A lot of questions may arise regarding this procedure and the role of purely random factors. It is not a problem when rather the best score than its explanation is desired. In a leaderboard-driven exploration one can simply conduct a large set of experiments and choose the best-performing model without reflection whether it is a byproduct of training instability or not. What actually happened here was investigated afterwards and will be discussed in Section 2.

1.2 Technique Classification

Transformer-based language models used in sentence classification setting assume that representations of special tokens (such as `[CLS]` or `[BOS]`) are passed to the classification layer. Since TC task is aimed at classification of spans, it might be beneficial to introduce information about the text fragment to be classified. We experimented with two approaches addressing this requirement.

The first assumes injection of special tokens indicating the beginning and the end of text marked as propaganda, such as sample sentence before BPE applied appears as:

`[BOS]` Democrats acted like `[BOP]` babies `[EOP]` at the SOTU `[EOS]`

In this approach we continue with representation of `[BOS]`, as in usual sentence classification task. The second approach is to stack a small Transformer on the selected tokens only.[2] This one has no own embeddings apart from ones for `[BOS]`, but uses representations provided by the host model instead. This technique is roughly equivalent to adding consecutive layers and masking attention outside the

[1] See: `https://github.com/jcpeterson/openwebtext` OpenWebText is a project aimed at reconstruction of OpenAI's unreleased WebText dataset.

[2] Transformer we used in our experiment had 3 hidden layers, 4 attention heads and intermediate layer of size 512. Note that hidden size depends on host model, since we are using external embeddings.

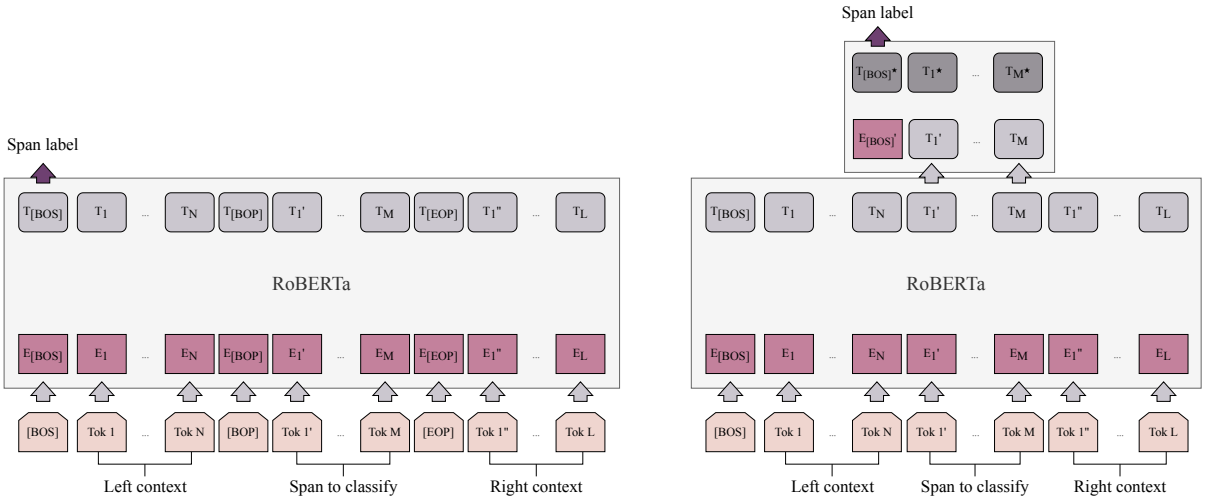

Figure 2: Comparison of span classification by means of special tokens (left) and in Span CLS approach (right). On the left special `[BOP]` and `[EOP]` tokens are introduced and the span is further classified as in usual Transformer-based sentence classification task. On the right an additional, small Transformer is stacked over the selected tokens only. It has no own embeddings apart from one for `[BOS]` token, but uses representations provided by the host model instead.

selected span and will be referred to as Span CLS. Figure 2 summarizes differences between Span CLS and classification by means of special `[BOP]` and `[EOP]` tokens.

The initial experiments have shown that underrepresented classes achieve lower scores. To overcome this problem, we experimented with class-dependent rescaling applied to binary cross entropy. In this setting (further referred to as *re-weighting*) factor for each class was determined as its inverse frequency multiplied by the frequency of the most popular class. The modified loss is equal to:

$$\ell(\mathbf{x}, \mathbf{y}) = -\frac{1}{Nd} \sum_{n=1}^{N} \sum_{k=1}^{d} \left[p^k y_n^k \log x_n^k + (1 - y_n^k) \log(1 - x_n^k) \right]$$

$$p^k = \frac{1}{f^k} \max(\mathbf{f})$$

where N is the batch size, n index denotes nth batch element, d is the number of classes, $\mathbf{f}$ stands for a vector of class absolute frequencies calculated on the train set, $\mathbf{x}$ is the output vector from the last sigmoid layer and $\mathbf{y}$ is a vector of multi-hot encoded ground truth labels. Note that the only difference from the original binary cross entropy for multi-label classification is the addition of the p^k class weights.

In addition to the above, a part of the tested models took the use of the self-training approach. In the case of TC task one had to identify spans first and then predict their classes to generate silver train set (Figure 1). We reused our best-performing model from SI task to identify spans, and the TC model trained on ground truth to automatically annotate these spans.

Regardless of the approach taken, context as broad as possible within the 256 subword units limit was provided on both sides of span to be classified.

The winning TC model (described in recipe below) was an ensemble of three models. Each of them used a different mix of previously described approaches with hyperparameters defined in Table 1 for first and second model, and those from Table 2 in case of the third model.

Recipe Add classification layer (described in Figure 2 on the left) to the pretrained RoBERTa $_{\text{LARGE}}$ model in order to obtain the first model and train until no score gain is observed on development set. Train the second model in the same manner, but this time applying the *re-weighting*. Combine *re-weighting*, Span CLS and self-training approaches to get the third model, and again train until no score improvement on development set is observed. Finally ensemble all three models by averaging class probabilities from their final layers.

As it will be shown later, the approach we took and reported above turned out to be sub-optimal. In-depth analysis of this system and a better one is proposed in Section 2.2.

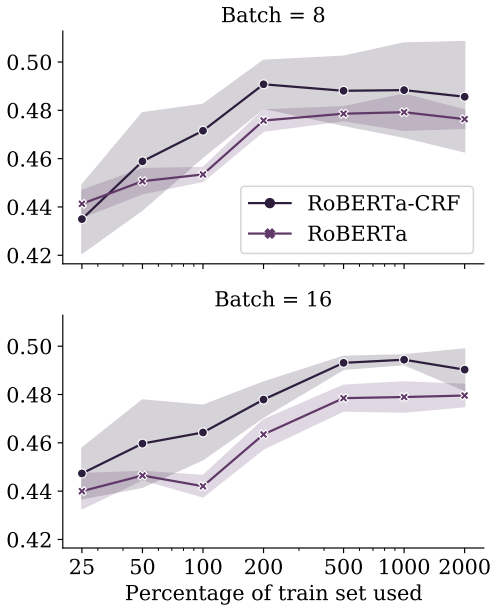

Figure 3: Performance of RoBERTa with and without CRF as a function of percentage of train set available. Values above 100% indicate self-training was performed. Mean FLC-F1 and standard deviation across 5 runs for each percentage.

CRF	**Self-train**	**FLC-F1** (std, max)	
−	−	45.2 ± 0.3	45.6
+	−	47.4 ± 0.8	48.2
−	+	48.9 ± 0.5	50.2
+	+	49.1 ± 3.0	51.7
+	+ (2)	49.7 ± 2.0	51.6
+	+ (3)	50.0 ± 1.8	51.8

Table 3: Best scores on the dev set achieved with RoBERTa large model on SI task. Mean, standard deviation and maximum across 10 runs with different random seeds. Numbers in brackets indicate how many self-training iterations were used.

Batch	**Dropouts**	**Self-train**	**CRF**	**Δ FLC-F1**
$16 \rightarrow 8$	$.0 \rightarrow .1$	+	−	−1.1
			+	−1.6
	$.0$		−	−0.4
			+	−1.1
$8 \rightarrow 16$	$.1 \rightarrow .0$	−	−	−3.9
			+	−7.0
	$.1$		−	−0.7
			+	−1.3

Table 4: Impact of hypothetical lowering batch size during self training or enlarging batch size during initial training, as well as of enabling or disabling both hidden and attention dropouts. Change between means across 10 runs with different random seeds.

2 Ablation Studies

Since different random initialization or data order can result with considerably higher scores,[3] models with different random seeds were trained for the purposes of ablation studies. In the case of SI task, results were evaluated on the original development set, whereas in the case of TC where fewer data points are available, we decided to use cross-validation instead.

2.1 Span Identification

Models with different random seeds were trained for 60K steps with an evaluation performed every 2K steps. This is equivalent to approximately 30 epochs and per-epoch validation in a scenario without data generated during the self-training procedure. Table 3 summarizes the best scores achieved across 10 runs for each configuration.

CRF has a noticeable positive impact on FLC-F1 scores achieved without self-training in the setting we consider. Presence of CRF layer is correlated positively with score ($\rho = 0.27$, $p < 0.001$). Difference is significant according to Kruskal–Wallis test ($p < 0.001$). Unless said otherwise, all further statistical statements within this section were confirmed with statistically significant positive Spearman rank correlation and Kruskal-Wallis test results. Differences in variance were confirmed using Bartlett's test. The 0.05 significance level was assumed.

Statistically significant influence of CRF disappears when the self-training is investigated. In the case of first self-training, whether or not CRF was used, a considerable increase of median score can be observed. Self-trained models with and without CRF layer however are indistinguishable.

Improvement offered by further self-training iterations is not so evident, but is statistically significant. In particular they slightly improve mean scores and decrease variance (see Table 3). As it comes to the latter, CRF-extended models have generally higher variance and scores achieved across the runs.

Table 4 analyzes the importance of using different hyperparameters. Whereas use of smaller batch size and dropout is beneficial for the initial training without noisy data, it impacts self-training phase

[3] See e.g. Junczys-Dowmunt et al. (2018) or recent analysis of Dodge et al. (2020).

#	Re-weight	Span CLS	Self-train	**Micro-F1** (std)
(1)	−	−	−	71.9 ± 1.5
(2)	−	−	+	71.4 ± 1.4
(3)	−	+	−	72.2 ± 1.3
(4)	−	+	+	71.8 ± 1.7
(5)	+	−	−	71.8 ± 1.6
(6)	+	−	+	70.9 ± 1.7
(7)	+	+	−	72.4 ± 1.5
(8)	+	+	+	71.3 ± 1.5

Table 5: Average of 6-fold cross-validation score on TC task with micro-averaged F1 metric.

Ensemble	**Micro-F1** (std)
(1) (6)	72.3 ± 1.7
(1) (2)	72.9 ± 1.8
(3) (5)	73.6 ± 1.5
(1) (5) (8)	74.1 ± 1.7
(2) (4) (7)	74.4 ± 1.5
(1) (4) (7)	74.6 ± 1.4
(1) (4) (7) (8)	74.9 ± 1.2
(1) (2) (4) (5) (7)	75.1 ± 1.5

Table 6: Average scores achieved with ensembles of individual models described in Table 5. Micro-averaged F1 metric.

negatively. Obviously, the largest negative impact is observed when disabling dropout during training on the small amount of manually annotated data.

Figure 3 illustrates scores achieved by models trained for the same number of steps on subsets or supersets of manually annotated data. CRF layer has a positive impact regardless of percentage of train set available. Once again, a large variance in scores of CRF-equipped models can be observed, however it is being substantially reduced with increase of batch size. Interestingly, figures suggest the proportion of automatically annotated data we used might be suboptimal, since it was an equivalent of around 3000% in line with the chart's convention. One may hypothesize better scores would be achieved by model trained with $1:4$ gold to silver proportion.

2.2 Technique Classification

6-fold cross-validation was conducted. The results are presented in Table 5. Folds were created by mixing training and development datasets, then shuffling them and splitting into even folds. Parameters were set according to Table 1 and Table 2, whereas experiments were carried out as follows. Each approach from Table 5 was separately evaluated on each fold using micro-averaged F1 metric. Then, for each approach average score and standard deviation was obtained using 6 scores from every fold.

Moreover, all the 247 possible ensembles[4] were evaluated in the same fashion as in experiments from Table 5. Table 6 shows the performance achieved by selected combinations when simple averaging of the probabilities returned by individual models was used as the final prediction.

Due to a large amount of results available, it is beneficial to conduct a statistical analysis in order to formulate remarks regarding the general trends observed. Each component model of the ensemble was treated as a categorical variable with respect to the ensemble score. Spearman rank correlation between presence of an ensemble component (approaches from Table 5) and achieved scores shows that adding model to the ensemble correlates with a significant increase in score, except for (6) model (see Table 7). Boxplots from Figure 4 lead to the same conclusions.[5]

Re-weighting seems to be beneficial only when ensembled with other models. An interesting finding is that Span CLS offers small but consistent increase of performance both in models from Table 5 and when used in ensembles. Bear in mind we outperformed the second-placed team by ε, so an improvement of point or half is not negligible.

What is most conspicuous however is that self-training based solutions from Table 5 seems to be actually detrimental in the case of TC task. This damaging effect can be potentially attributed to the fact that data automatically generated there accumulate errors from both Span Identification and Classification. Another possible explanation is that much fewer data points are available for span classification task than for span identification attempted as a sequence labeling task. The latter would be somehow consistent with what was found in the field of Neural Machine Translation, where use of back-translation technique in low-resource setting was determined to be harmful (Edunov et al., 2018).

On the other hand, self-training has a positive, statistically significant impact on the score when used in ensembles (see Figure 4 and Table 7). It is not surprising as the beneficial impact of combining individual

[4]It's a number of 8-element set subsets with cardinality greater than one.

[5]Kruskall-Wallis test and Boruta algorithm (Kursa et al., 2010) we used in addition support these findings too.

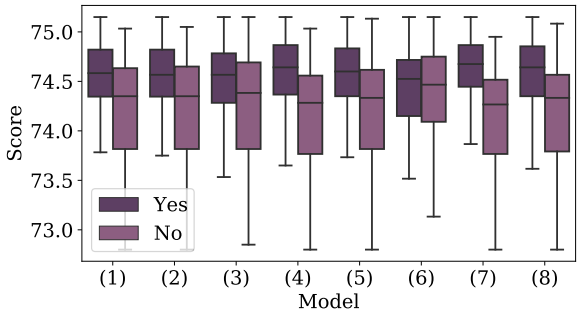

Figure 4: Impact of adding a certain model to the ensemble has on mean scores from different folds. Comparison of results with and without it present in tested combination.

Model	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
ρ	.28	.30	.20	.41	.32	.05*	.50	.36

Table 7: Spearman's ρ between presence of ensemble component (models from Table 5) and score achieved by ensemble. * indicate results were not significant assuming 0.05 significance level.

		Authority	Fear	Bandwagon	B&W	Simplification	Doubt	Minimization	Flag-Waving	Loaded	Labeling	Repetition	Slogans	Clichés	Strawman	Overall
Identified subsequence		57	56	20	36	50	42	48	40	44	45	26	62	41	41	43
Fully identified	%	7	18	0	18	5	6	11	50	25	21	33	7	23	10	23
Not identified		35	25	80	45	44	51	39	9	29	33	40	30	35	48	33
Number of instances		14	44	5	22	18	66	68	87	325	183	145	40	17	29	1063

Table 8: Proportion of partially and fully identified spans (SI task) depending on the propaganda technique used. All the experiments conducted on the original development set.

estimates was observed in many disciplines and is known since the times of Laplace (Clemen, 1989).

3 Error analysis

In addition to providing an overview of problematic classes, the question of which shallow features influence score and worsen the results was addressed. This problem was analyzed in a *no-box* manner, as proposed by Graliński et al. (2019). The main idea is to create two dataset subsets for each feature considered (one for data points with the feature present and one for data points without the feature), rank subsets by per-item scores and use Mann-Whitney rank U to determine whether there is a non-accidental difference between subsets. Low p-value indicates that feature reduces the evaluation score of the model.

3.1 Span Identification

Since FLC-F1 metric used in SI task gives non-zero scores for partial matches, it is interesting to analyze what was the proportion of fully missed (partially identified) spans. Table 8 investigates this question broken down by propaganda technique used.

Our system was unable to identify one third of expected spans, whereas a majority from those identified correctly were the partial matches. The spans easiest to identify in text represented the *Flag-Waving*, *Appeal to fear/prejudice* and *Slogans* techniques, whereas *Bandwagon*, *Doubt* and the group of {*Whataboutism*, *Strawman*, *Red Herring*} turned out to be the hardest. The highest proportion of fully identified spans was achieved for *Flag-Waving*, *Repetition* and *Loaded Language*. Unfortunately, it is not possible to investigate precision in this manner, without training separate models for each label or estimating one-to-one alignments between output and expected spans.

Further investigation of problematic cases in a paradigm of no-box debugging with GEval tool (Graliński et al., 2019) revealed the most worsening features, that is features whose presence impacts span identification evaluation metrics negatively (Table 9). It seems that our system tend to return ranges without adjacent punctuation. This is the case of sentences such as *The new CIA Director Haspel, who 'tortured some folks,' probably can't travel to the EU*, where only the quoted text was returned whereas

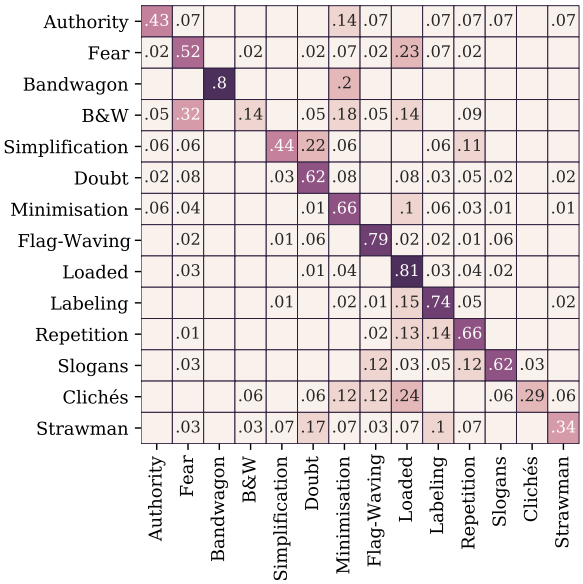

Figure 5: Confusion matrix of the submitted system predictions normalized over the number of true labels. Rows represent the true labels and columns – the predicted ones (TC).

Feature		Count	P-value
question	expected	21	0.036
dot		36	0.037
quotation		58	0.050
exclamation		15	0.064
and	output	14	0.070

Table 9: Selected shallow features one may hypothesize impact evaluation scores negatively (SI).

Feature		Count	P-value
comma	inside	119	< 0.001
we		15	0.002
this		28	0.007
will		40	0.008
not		62	0.013
exclamation		16	0.014
CIA	before	25	< 0.001
according to	after	8	< 0.001
quotation	before	65	0.004

Table 10: Selected shallow features one may hypothesize impact evaluation scores negatively (TC).

annotation assumes it should be returned with apostrophes and comma. This remark can be used to slightly improve overall results with simple post-processing. Returned *and* conjunction refers to the cases where it connects two propaganda spans. The system frequently returns them as single span, contrary to what is expected in the gold standard.

3.2 Technique Classification

Figure 5 presents normalized confusion matrix of the submitted system predictions. Interestingly, there are a few pairs that were commonly confused. *Loaded Language* and *Black-and-white Fallacy* were frequently misclassified as *Appeal to fear/prejudice*. Similarly, *Causal Oversimplification* was often predicted as *Doubt* and *Clichés* as *Loaded Language*.

The most worsening features are presented in Table 10. One of the frequent predictors of low accuracy is a comma character present within the span to be classified. It can be probably attributed to the fact that its presence is a good indicator of span linguistic complexity. Another determinant of inefficiency turned out to be a negation—around a half of the sentences containing word *not* were misclassified by the system. Suggested features of a quotation mark before the span and the digram *according to* after the span are related to reported or indirect speech. Explanation of worsening effect of other features is not as evident as in the case of mentioned above. Moreover, it seems there is no obvious way of improving the final results with our findings and a more detailed analysis might be required.

4 Discussion and Summary

The winning system for the propaganda Technique Classification (TC) task and the second-placed system for the propaganda Span Identification (SI) task has been described. Both of the developed solutions used semi-supervised learning technique of self-training. Although CRF is barely used with Transformer-based language models, the SI task was approached with RoBERTa-CRF architecture. An ensemble of RoBERTa-based models has been proposed for the TC task, with one of them taking use of Span CLS layers we introduce in the present paper.

Analyses conducted afterwards can be applied in rather straightforward manner to further improve the scores for both SI and TC tasks. It is because some of the decisions we have made given lack of or uncertain information, during the post-hoc inquiry turned out to be sub-optimal. These include the

proportion of data from self-training in SI task, as well as the possibility to provide a better ensemble in the case of TC.

The ablation studies conducted however have some limitations. The same subset of OpenWebText was used in experiments conducted within one self-training iteration. This means a random seed did not impact which sentences were used during the first, second and the third self-training phase and in each we were manipulating only the data order. Moreover, an analysis we reported was limited to few hyperparameter combinations and no extensive hyperparameter space search was performed. Finally, only one and rather simple method of cost-sensitive re-weighting was tested and there is a great chance it was sub-optimal. It would be interesting to investigate other schemes, such as the one proposed by Cui et al. (2019).

The error analysis revealed propaganda techniques commonly confused in TC task, as well as techniques we were unable to detect effectively within the SI input articles. In addition to providing an overview of problematic classes, the question of which shallow features influence score and worsen the results was addressed. A few of these were identified and our remarks can be used to slightly improve results on SI task with simple post-processing. This is not the case for TC task, where one is unable to propose how to improve the final results with our findings.

An interesting future research direction seems to be the application of CRF layer and Span CLS to Transformer-based language models when dealing with other tasks, outside the propaganda detection problem. These may include Named Entity Recognition in the case of RoBERTa-CRF, and an aspect-based sentiment analysis that can be viewed through the lens of span classification with Span CLS we proposed.

5 Outro

Developed systems were used to identify and classify spans in the present paper in order to detect fragments one may suspect to represent one or more propaganda techniques. Unfortunately for the entertaining value of this work, none of such were identified by our SI model.

References

Robert T. Clemen. 1989. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5(4):559 – 583.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie. 2019. Class-balanced loss based on effective number of samples. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9260–9269.

Giovanni Da San Martino, Alberto Barr'on-Cede no, 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, September.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*.

Filip Graliński, Anna Wróblewska, Tomasz Stanisławek, Kamil Grabowski, and Tomasz Górecki. 2019. GEval: Tool for debugging NLP datasets and models. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 254–262, Florence, Italy, August. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606, New Orleans, Louisiana, June. Association for Computational Linguistics.

Miron B Kursa, Aleksander Jankowski, and Witold R Rudnicki. 2010. Boruta–a system for feature selection. *Fundamenta Informaticae*, 101(4):271–285.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

Wenhui Liao and Sriharsha Veeramachaneni. 2009. A simple semi-supervised algorithm for named entity recognition. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*, pages 58–65, Boulder, Colorado, June. Association for Computational Linguistics.

Yao Lin, Chengjie Sun, Wang Xiaolong, and Wang Xuan. 2010. Combining self learning and active learning for chinese named entity recognition. *Journal of Software*, 5, 05.

Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 359–367, Portland, Oregon, USA, June. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. In *WACV/MOTION*, pages 29–36.

Fábio Souza, Rodrigo Frassetto Nogueira, and Roberto de Alencar Lotufo. 2019. Portuguese named entity recognition using BERT-CRF. *CoRR*, abs/1909.10649.

Lei Wang, Qing Qian, Qiang Zhang, Jishuai Wang, Wenbo Cheng, and Wei Yan. 2020. Classification Model on Big Data in Medical Diagnosis Based on Semi-Supervised Learning. *The Computer Journal*, 03. bxaa006.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA, June. Association for Computational Linguistics.