ProsperAMnet at FinCausal 2020, Task 1 & 2: Modeling causality in financial texts using multi-headed transformers

Zsolt Szántó
Institute of Informatics,
University of Szeged
szantozs@inf.u-szeged.hu

Gábor Berend
Institute of Informatics,
University of Szeged
berendg@inf.u-szeged.hu

Abstract

This paper introduces our efforts at the FinCasual shared task for modeling causality in financial utterances. Our approach uses the commonly and successfully applied strategy of fine-tuning a transformer-based language model with a twist, i.e. we modified the training and inference mechanism such that our model produces multiple predictions for the same instance. By designing such a model that returns $k > 1$ predictions at the same time, we not only obtain a more resource efficient training (as opposed to fine-tuning some pre-trained language model k independent times), but our results indicate that we are also capable of obtaining comparable or even better evaluation scores that way. We compare multiple strategies for combining the k predictions of our model. Our submissions got ranked third on both subtasks of the shared task.

1 Introduction

The dominant strategy for solving natural language processing (NLP) problems these days is by taking some large, task-agnostic pre-trained neural language model and fine-tune it on the set of task-specific training data at disposal. This transfer learning setting has been proven to be highly effective in various NLP problems (Devlin et al., 2019; Yang et al., 2019; Raffel et al., 2020).

Despite having demonstrated its efficacy, the above described approach also comes with inconveniences inherent to the stochasticity of the fine-tuning procedure. As the performance of the fine-tuned model is dependant on random factors such as the order in which the training instances are considered during fine-tuning or the choice of hyper-parameters of the fine-tuning procedure. One could possibly account for this variability by performing fine-tuning multiple times with different hyper-parameter choices and performing model selection using the development set performance of the independently fine-tuned models. Performing fine-tuning $k > 1$ many times, however, naturally increases the computational resources required for model selection by a factor of k.

The models that we propose in our work are such that they make the training of $k > 1$ fine-tuned model possible without a k-fold overhead. We achieve this by simultaneously adding k multiple classification heads to the same pre-trained architecture and performing fine-tuning based on the aggregated loss coming from the individual classification heads. Our experimental results indicate that the above fine-tuning strategy not only provides a more resource efficient way of training multiple fine-tuned models, but it also has the potential of providing an increased performance for the models, since fine-tuning classifiers in a simultaneous manner likely acts as a regularizer. Our source code for replicating our results can be accessed at https://github.com/zsozso21/camuh.

2 Data

The shared task consisted of a Sentence Classification (SC) and a Cause and Effect Detection (CED) subtask, that we briefly describe below. A more detailed overview of the shared task can be found in the summary paper (Mariko et al., 2020).

Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation, pages 80–84
Barcelona, Spain (Online), December 12, 2020.

For the SC subtask, the goal was to classify textual utterances whether they convey some form of causality. As such, we treated the SC subtask as a sentence-level binary classification problem for which one can conveniently fine-tune some pre-trained transformer-based model.

In the CED subtask, systems were provided (blocks of) sentences and the goal was to detect the correct spans of the causal elements and their consequences. As a concrete example, the spans formatted in italics and bold in the below sentence are considered as the effect and cause constituents, respectively:

"*Zhao found himself 60 million yuan indebted* after **losing 9,000 BTC in a single day (February 10, 2014)**".

Illustrated by the previous example, the CED subtask can be handled as a token-level sequence classification with three classes, i.e. a token can either belong to a constituent of the cause or the effect or it can be none of the above. We shall refer to these three classes by their initials, i.e. `C` and `E` for the cause and effect classes and `N` for those tokens that are neither the cause or the effect constituent of some text.

3 System

We trained our models using the pre-trained BERT models based on the transformers library (Wolf et al., 2019). During fine-tuning, we relied on the utilization of the cross-entropy loss for both the SC and the CED subtasks. We treated the SC subtask as a binary sequence-level classification task, whereas we framed the CED subtask as an utterance-level classification problem.

3.1 Treatment of subtokens

BERT uses pieces of words (subtokens) as input representations to utilize the information for out-of-vocabulary words. As we mentioned earlier the CED subtask can be handled as token-level sequence classification. To create BERT compatible sequences we split these tokens into subtokens where the subtokens get the label of the original token. We made the classification on this subtoken-level dataset. In the prediction, there will be a label for each subtoken. We merge this subtokens back into tokens and assign the label of the first subtoken to each tokens.

3.2 Strategies for aggregating prediction heads

The main difference in our model – compared to standard single-headed fine-tuning – is that we employ $k \geq 1$ classification heads, resulting in a loss function that needs to be aggregated over k distinct loss terms. A similar method was proposed for LSTM architectures (Kis-Szabó and Berend, 2020), but not in the case of transformers. One of our strategy, that we call *averaging of heads*, calculated k distinct loss terms for each training instance and took their average as the aggregated loss. This strategy also ensures that we are learning k fine-tuned models on top of the same pre-trained architecture simultaneously.

Another strategy for the aggregation that we experimented with was the application of a fully connected layer over the heads. In this case, there are k separated branches in the network too, but these branches are merged in a higher level layer that takes as input the concatenation of the output of the k prediction heads and outputs a single vector of logits. Hence, even though we have k distinct heads, this aggregation strategy ensures that they are merged together into a single prediction and loss term.

3.3 Span-level auxiliary loss

In order to enforce homogeneity of the predicted tags in the case of the CED subtask we added an auxiliary loss to our model which penalizes the changes in the predicted labels within a sentence. We deemed it a useful extension, as our error analysis revealed that each input sequence contained exactly one cause and one effect span (a span being a contiguous sequence of tokens with the same label), whereas our system was not encouraged to behave like that, i.e. it could easily predict spans of label sequences (over the tokens) like `ECE`, `CNCNC`, etc. In order to discourage inhomogeneous predictions, we decided to add an auxiliary loss term to our loss function that was calculated as the squared difference between the number of predicted and gold standard spans over some input sequence.

BERT	k	rs 1	rs 2	rs 3	rs 4	rs 5	avg	std
base	1	96.79	96.69	96.77	96.90	96.68	96.77	0.09
large	1	**96.98**	96.89	96.82	97.13	97.01	96.96	0.12
large	5	96.92	**97.39**	97.02	**97.18**	97.03	**97.11**	0.18
large	20	96.82	97.24	**97.11**	97.10	**97.08**	97.07	0.15

Table 1: The effects of employing different BERT models (base/large) and different number of heads (k) on the Sentence Classification subtask for different random seeds (denoted by rs). Performance is reported as F1 scores.

4 Results and discussion

The parameters of the system were optimized by using the shared task's trial dataset for training and the practice dataset for testing[1]. Based on our evaluation on the practice dataset, we fixed the batch size to on the SC and the CED subtasks to 8 and 1, respectively. We also applied dropout with probability 0.1 in all of our experiments. Those models that we used for making predictions over the evaluation data were trained on the concatenation of the practice and trial datasets and later tested on the evaluation dataset during the post evaluation period. Unless stated otherwise, our models described below use averaging-based aggregation strategy of the heads and no span-level auxiliary loss.

As neural methods are prone to achieving results with high variance when being trained with different choices of random seed, we evaluated each architectures five times with only their random seeds set differently. We report the individual results obtained by the different choices of the random seed as well as the average and standard deviation of the performance metrics.

4.1 Experiments on the number of heads

Sentence Classification subtask Table 1 compares the results of the base-cased and the large-cased BERT models and shows the added value of the extra heads on the SC subtask. There is only a small gap between the performance of the base and the large BERT models, 0.19 percentage point on average, but the base model was outperformed by the large one five out of five times. The usage of more than one heads improved the results in both cases. The five-headed model achieved the best average F1 score if we compare it against the five independent runs of the standard model with a single classification head. Overall, we found that our multi-headed systems were better four out of five times regarding the individual runs.

Cause and Effect Detection subtask Table 2 contains the results of the same experiments on the CED subtask. The BERT large model outperformed the base once again and the application of $k \geq 1$ heads improved the results too. For that task, the higher value of k made the larger improvement, as we obtained the best results with the twenty-headed model.

[1]Our configuration contains an RTX 2080ti GPU, equipped with 11GB RAM.

BERT	k	rs 1	rs 2	rs 3	rs 4	rs 5	avg	std
base	1	81.27	82.96	**82.62**	81.25	81.85	81.99	0.78
large	1	82.54	82.71	82.07	**83.00**	82.04	82.47	0.41
large	5	83.39	83.21	82.25	82.58	82.38	82.76	0.51
large	20	**83.43**	**83.63**	82.00	82.73	**82.68**	**82.90**	0.65

Table 2: The effects of employing different BERT models (base/large) and different number of heads (k) on the Cause and Effect Detection subtask using different random seeds (denoted by rs). Performance is reported as F1 scores.

4.2 Evaluation of the strategies for aggregating prediction heads

Table 3 provides a comparison of the two aggregation strategies, illustrating that backpropagating the loss from the average of the k distinct heads tends to perform better compared to the aggregation of the different heads by applying a fully connected layer on top of their concatenation.

	rs 1	rs 2	rs 3	rs 4	rs 5	avg	std
Using an FC layer	81.91	82.02	**82.64**	**83.27**	82.05	82.38	0.57
Averaging of heads	**83.39**	**83.21**	82.25	82.58	**82.38**	**82.76**	0.51

Table 3: Comparison of the head aggregation strategies on the Cause and Effect Detection subtask using $k = 5$, and the BERT large-cased model. Performance is reported as F1 scores.

4.3 Span number penalization

Table 4 shows that by utilizing the auxiliary loss introduced in Section 3.3, we obtained lower F1 scores on the token level evaluation, but an increased efficiency regarding the fraction of perfectly predicted sequences. By analyzing the results our model predicted 2589 spans on the evaluation dataset by using the original loss function with random seed 1. When using the same architecture with the auxiliary loss that we introduced for encouraging the homogenity of predicted labels, the number of predicted spans decreased to 2018.

In order to see the qualitative effects of the application of our auxiliary loss term, consider the following prediction that we obtained by our original system that did not involve the auxiliary loss term:

> With over $5.2 billion of TRX locked up in staking and an average yield of 13.2%, almost $700 million are made in profits annually from staking on Tron.

Tokens in red were predicted as being part of a *cause*, whereas tokens in blue were predicted as being part of an *effect* by our system. In contrast, the prediction of our model that also incorporated the auxiliary loss produced this more coherent output for the same sequence:

> With over $5.2 billion of TRX locked up in staking and an average yield of 13.2%, almost $700 million are made in profits annually from staking on Tron.

	rs 1	rs 2	rs 3	rs 4	rs 5	avg
Original	83.39 / 61.44	83.21 / 56.58	82.25 / 69.28	82.58 / 14.58	82.38 / 67.08	82.76 / 53.79
+ span	81.62 / 61.91	82.13 / 49.84	83.01 / 61.13	82.50 / 60.50	83.04 / 70.69	82.46 / 60.82

Table 4: The effect of span-level auxiliary loss on Cause and Effect Detection dataset using $k = 5$, and the BERT large-cased model. Performance is reported as F1 / Exact match scores.

4.4 Submission results

We reached the third place in both of the SC and CED subtasks. Our best submission obtained an F1 score of 97.23 in the SC subtask by applying the following parameters: large BERT model, $k = 20$, averaging of heads aggregation, without span penalization. In the CED subtask our top model used BERT large mode, $k = 5$, averaging of heads aggregation, without span penalization, and got 83.71 for F1 score. The reason why the figures that we described in this article slightly differ from our official shared task results is the non-deterministic behavior of GPU computing. We repeated and extended each experiment to get a more detailed image about our system.

5 Conclusions

Our experiments demonstrated that the standard fine-tuning mechanisms work well for modeling causality in financial texts as well. More importantly, we introduced our modified fine-tuning architecture that is capable of training multiple classification model on top of a shared transformer architecture. Our results demonstrate that we did not only obtain multiple classifiers in a less resource intensive manner, but the classifiers also performed better as if we were training them separately of each other. A potential avenue for improving our proposed fine-tuning architecture could focus on more sophisticated aggregation strategies for the predictions of the simultaneously fine-tuned classification models.

Acknowledgements

This research has been conducted in the project "Progressing Service Performance and Export Results of Advanced Manufacturers Networks", no CE1569 ProsperAMnet. The project has been supported by the European Fund for Regional Development in the framework of Interreg CENTRAL EUROPE 2019-2022.

References

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.

Norbert Kis-Szabó and Gábor Berend. 2020. Quasi-multitask learning: an efficient surrogate for obtaining model ensembles. In *Proceedings of the First Workshop on Simple and Efficient Natural Language Processing*, November.

Dominique Mariko, Hanna Abi Akl, Estelle Labidurie, Stephane Durfort, Hugues de Mazancourt, and Mahmoud El-Haj. 2020. The Financial Document Causality Detection Shared Task (FinCausal 2020). In *The 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation (FNP-FNS 2020*, Barcelona, Spain.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.