NVJPFSI at FinCausal 2021 Span-based Causality Extraction Task

Xianchao Wu
NVIDIA
wuxianchao@gmail.com,xianchaow@nvidia.com

Abstract

We describe our NVJPFSI (NVIDIA Japan Financial Services with AI) system for span-based causality extraction from financial news documents submitted as part of the FinCausal 2021 Workshop. We investigated a list of pretrained language models, such as ALBERT-xxlarge, BERT-large, and RoBERTa-large models fine-tuned under SQuAD2.0. A grid-based ensemble learning algorithm is further introduced to combine n-best predictions from five checkpoints. We show impressive results of F1 (94.77%) and exact match (87.62%) scores through applying these models individually and in grid-based ensemble learning.

1 Introduction

We describe the pipeline architecture and performances of our system that participated the FinCausal2021 span-based causality extraction task[1] (Mariko et al., 2021) which used the same datasets following FinCausal2020 (Mariko et al., 2020a). Pretraining + fine-tuning methodology is adapted. There are three pretrained models, ALBERT (Lan et al., 2020), RoBERTa (Liu et al., 2019) and BERT (Devlin et al., 2019), used in our system. Detailed fine-tuning configurations, loss curves are reported (Section 3). We made 18 submissions (Section 4) in which 3 are combined predictions of 5 checkpoints from individual models and 1 combined prediction achieved the highest F1 (94.77%, ranked 3rd) and exact match (87.62%, ranked 2nd) scores. A simple grid-based ensemble algorithm is described in Section 5. We conclude this paper in Section 6.

2 Dataset for FinCausal2021

We used the full dataset with 2,394 samples for training. There are duplicated usage of sample ids. Thus, in our development set, we only keep the first text for duplicated ids yielding 2,289 samples. Our second development set is a much smaller subset with only 65 samples randomly selected from the full training set. Our experimental setting can be taken as a *closed* test, instead of *open* test. This is to fully utilize the training set with less rich samples.

[1] https://competitions.codalab.org/competitions/33102

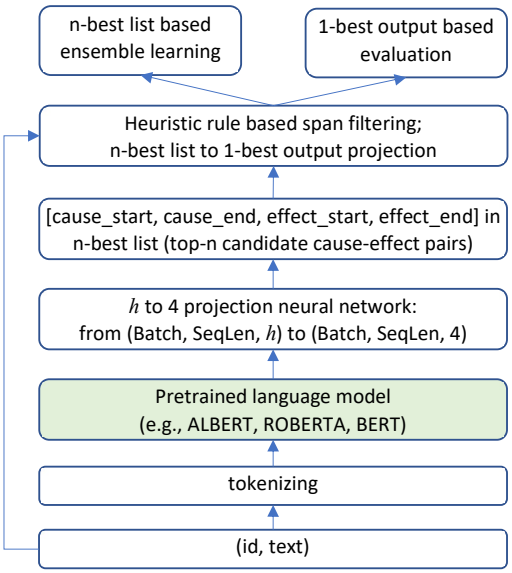

Figure 1: Fine-tuning based system architecture.

3 System Architecture

3.1 Pipeline of the system

Figure 1 depicts our fine-tuning based system architecture. We borrow the major framework from (Becquin, 2020). Starting from a batch of input (id, text), we first use tokenizer of each pretrained language model (PLM) and then use the PLM's forward function to represent the input text into a dense tensor with a shape alike (batch size, maximum sequence length, h) where h stands for the hidden layer dimension (such as 1,024). Then, we use a projection network (one layer linear network) to project from h to 4. That is, for one position in a sequence, we need to compute its probabilities to be start of a cause, end of a cause, start of an effect

and end of an effect. We reuse the two heuristic rules in (Becquin, 2020) for candidate span filtering: (1) one cause or effect should not span over multiple sentences and (2) extend the clause (cause or effect) to the entire sentence when one sentence contains only one clause. These heuristic rules are motivated by the data annotation criteria described in (Mariko et al., 2020b). In addition, one text possibly has more than one pair of cause-effect. At that time, different ids with suffices alike ".1", ".2" will be appended to the original ids, resulting in ids alike "0001.000005.1" and "0001.000005.2". When projecting from n-best list to top-1 results, the id of each input (id, text) is taken into consideration: an id that ends with ".1" picks rank-0 prediction and ".2" picks rank1-prediction. Cross entropy loss is used to compare the top-1 predicted 4 positions and their references.

There are two major differences between our framework and (Becquin, 2020). First, different types of pretrained language models (Section 3.2). Second, grid-based ensemble learning among multiple n-best outputs (Section 5).

3.2 Employed pretrained language models

We use Huggingface's transformer package[2] (Wolf et al., 2019) and employ three pretrained language models which are ALBERT (Lan et al., 2020) fine-tuned by SQuAD2.0 (Rajpurkar et al., 2018) (m1=*elgeish/cs224n-squad2.0-albert-xxlarge-v1*[3]), RoBERTa (m2=*roberta-large*[4]) (Liu et al., 2019) and BERT (Devlin et al., 2019) fine-tuned by SQuAD2.0 (m3=*deepset/bert-large-uncased-whole-word-masking-squad2*[5]). Table 1 lists the major configurations of these 3 PLMs. We run experiments on one DGX-1 machine equipped with 8 NVIDIA V100-16GB GPU cards.

We first selected BERT model fine-tuned by SQuAD2.0 basing on its well-known performance in GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) shared tasks. Also, considering that SQuAD2.0 is another span-based answer extraction tasks of general domain, its domain adaption to financial domain will be meaningful for investigation. Also, we selected RoBERTa since it trained BERT with a list of strategies and yielded significantly better results than BERT. Finally, ALBERT by parameter sharing is with only 2/3 of the parameters of BERT yet still performed better than BERT in GLUE, we thus take it into consideration as well. Each model is attached with an independent vocabulary and out-of-vocabulary words are marked as <UNK> by models' default and independent tokenizers.

	ALBERT	RoBERTa	BERT
dropout prob:	0	0.1	0.1
embedding size:	128	1,024	1,024
finetuning task:	squad2	NA	squad2
hidden act:	gelu	gelu	gelu
hidden dropout prob:	0	0.1	0.1
hidden size:	4,096	1,024	1,024
initializer range:	0.02	0.02	0.02
intermediate size:	16,384	4,096	4,096
layer norm eps:	1.0E-12	1.0E-05	1.0E-12
max position embed:	512	514	512
num attention heads:	64	16	16
num hidden layers:	12	24	24
pad token id:	0	1	0
type vocab size:	2	1	2
vocab size:	30,000	50,265	30,522
parameter size	223M	355M	336M

Table 1: Major configurations of the 3 PLMs.

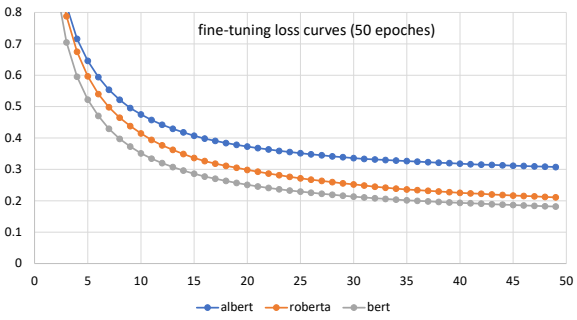

Figure 2: Fine-tuning curves of the 3 PLMs.

Figure 2 depicts the cross-entropy loss curves during fine-tuning. After 50 epoches, BERT's loss is the lowest. However, these losses do not necessary have the same order of the final performances which are given in Table 2 and 3.

4 Our Submission Results

Table 2 lists the F1, Precision, Recall and exact match scores on the official test set of our 18 submitted systems. Submissions from 1 to 12 are selected by sorting the EM scores under a development set with 64 samples.

We first pick the top-1 checkpoints of m1, m2, and m3. These forms the first three submissions. We observe that 2-m2-7563 achieved the best F1 score of 94.82% yet its EM is only 70.53% re-

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

[3] https://huggingface.co/elgeish/cs224n-squad2.0-albert-xxlarge-v1

[4] https://huggingface.co/roberta-large

[5] https://huggingface.co/deepset/bert-large-uncased-whole-word-masking-squad2

	F1	P	R	EM
1-m1-4578	94.10	94.10	94.10	80.56
2-m2-7563	**94.82**	**94.83**	**94.83**	70.53
3-m3-3782	92.28	92.29	92.29	82.29
4-m1-8359	92.44	92.44	92.47	79.31
5-m1-8757	93.99	94.00	94.00	86.36
6-m1-9354	93.83	93.83	93.84	86.52
7-m2-14130	92.01	92.02	92.01	80.88
8-m2-14528	94.39	94.40	94.41	70.53
9-m2-16518	93.85	93.86	93.86	77.90
10-m3-2986	92.01	92.01	92.03	66.30
11-m3-4180	93.41	93.42	93.43	71.00
12-m3-30846	92.86	92.87	92.86	71.47
13-m1-9752	93.81	93.80	93.82	86.83
14-m1-9553	94.23	94.23	94.24	86.36
15-m1-7762	94.27	94.27	94.28	86.68
16-comb.1	94.77	94.78	94.78	**87.62**
17-comb.2	93.68	93.68	93.68	84.95
18-comb.3	93.85	93.86	93.86	85.11
mean	93.59	93.59	93.60	80.07
stdev	0.90	0.90	0.90	7.09

Table 2: F1 (%), P (Precision, %), R (Recall, %) and EM (exact match, %) scores on the official test set of our 18 submissions (m1 = albert-xxlarge-squad2, m2 = roberta-large, and m3 = bert-large-squad2). Numbers after m1/2/3, such as 7762, are iteration number (= checkpoint index) of each individual model.

flecting a significant gap of between F1 and EM. Similar gaps appear frequently in other submitted results. These partially suggest that only using F1/P/R scores for ranking the models is possibly ambiguous. Basing on these observations and considerations, we keep using EM as the major criteria for ranking our models' checkpoints.

Then, rank-2 to rank4 checkpoints from m1, m2, and m3 are respectively selected. These form our 4 to 12 submissions. Even we selected them basing on development set's EM score, their performances at the official test set contain high variances, from 66.30% to 86.52%.

We thus consider change our development set of from 64 samples to the whole training set. There are 5 ids used by 10 texts (i.e., two texts used the same id) and we only keep one text for one unique id. The result training set contains 2,288 samples. We use this development set to rank the checkpoints and submissions from 13 to 18 are related to it. In which, submissions 13 to 15 are the top-3 single models from m1 and both F1 and EM achieved in a relatively high level: 13-m1-9752 achieved the best EM score of 86.83% among all the individual checkpoints.

Finally, we computed means and standard derivations in in Table 2. F1/P/R are averagely 93.60% around and EM is averagely 80.07%. Also, note that the standard derivations are only 0.9% for F1/P/R while EM's standard derivation reached as much as 7.09%. It will be essential to disclose this gap among F1/P/R and EM for better understanding the predicting qualities of the models.

5 Grid-based Ensemble Method

5.1 The algorithm for FinCausal dataset

We utilize grid-based ensemble method to the n-best (n=5) predictions from several types of models. During each ensemble learning, we intentionally pick 5 strong individual models basing on their exact matching scores in the development set and in the whole training set. Each model is assigned a weight of from 0 to 1 with a step length of 0.1 and the sum up of the five weights are ensured to be 1 (line 5 in Algorithm 1).

Algorithm 1 gives our grid-based ensemble algorithm. There are two functions, **main**() and **getEM**(). The first function **main**() reads nbest prediction results from external files, construct a *grid* that ranges over the possible weights of each model during combining. Then, in **getEM**(), under each weight list, we try to compute the weighted probabilities (line 16) of predicted cause-effect pairs for each (id, text) input. One special treatment (lines 20 to 24) is motivated by the fact that one text is allowed to have more than one cause-effect pairs. Given one text with different id, the model outputs the same n-best prediction. Thus, we are forcing a text having several ids to pick different predictions. This treatment brings significant improvements to final scores. Also, this id based *suffix_index* for top-n result determining is not only used for multiple models' ensemble learning, but also for single model's top-1 result selecting from top-n beam search predictions (right-top at Figure 1).

5.2 Experimental results with ensemble

Table 3 lists F1 and EM scores of three ensemble predictions and their source individual models' outputs. In order to select individual models for ensemble learning, we first rank each model's checkpoints by the EM scores on the whole training set. That is, we are using a closed test set which ranges over all the training samples. The top-1 checkpoint of each model, namely m1-7762, m2-31642, and m3-30647, are selected at first. Then, we further append ALBERT model m1-9752 to all the three combinations since it achieved the best EM score (86.83%) among all the individual models' submis-

Algorithm 1: Grid-based ensemble

▷ **main**()
best_w5 = []; *best_em* = 0.0
nbest = [{id:{(text;cause;effect) : probability}}, {}, {}, {}, {}] ▷ read from external nbest files
reference = [(id;text;cause;effect)] ▷ read from external reference file
for *w5 in {0.0, 0.1, ..., 1.0}*5 and sum(w5) = 1.0* ▷ *grid-based searching* **do**
    em = **getEM**(*w5*, *nbest*, *reference*)
    if *em* > *best_em* **then**
        best_em = *em*; *best_w5* = *w5*
return *best_em*, *best_w5*
▷ **getEM**(*w5*, *nbest*, *reference*)
w_nbest = {id:{(text;cause;effect):0.0}}
for *i in [0, 4]* ▷ *range over 5 models* **do**
    for *aid in nbest[i].keys* **do**
        top5_dict = *nbest*[*i*][*aid*]
        for *text_cause_effect in top5_dict.keys* **do**
            w_nbest[*aid*][*text_cause_effect*] += *w5*[*i*] * *top5_dict*[*text_cause_effect*]
output = {} ▷ one (id,text) with the best (cause,effect)
for *aid, entry_map in w_nbest.items()* **do**
    sorted_texts = sort *entry_map* based on descending order of scores
    suffix_index = 0
    if *aid.count('.') == 2* **then**
        suffix_index = int(*aid*.split('.')[-1])
    if *suffix_index > 0* **then**
        suffix_index -= 1
    best_text = *sorted_texts*[*suffix_index*]
    output[*aid*] = *best_text* ▷ *best_text* = (text;cause;effect)
return *em* (= EM(*output*,*reference*))

sions. The fifth checkpoint is then selected to be rank-3 in m1, rank-2 in m2 and rank-2 in m3.

From Table 3, we have the following observations. First, in the development set, through grid-based ensemble learning, the F1 and EM scores are averagely improved 3.39% and 5.14%, respectively. Second, in the test set, only comb.1's EM and F1 are better than all the submitted individual checkpoints, +0.94% of EM and +0.80% of F1,

	test		dev (=train)		
	EM	F1	EM	F1	w
16-comb.1	**87.62**	**94.77**	**95.02**	**97.56**	
15-m1-7762	86.68	94.27	89.89	94.20	0.4
m2-31642	-	-	89.93	94.11	0.2
m3-30647	-	-	89.97	94.10	0.3
13-m1-9752	86.83	93.81	89.80	94.32	0.0
6-m1-9354	86.52	93.83	89.72	94.05	0.1
17-comb.2	84.95	93.68	**94.97**	**97.17**	
15-m1-7762	86.68	**94.27**	89.89	94.20	0.0
m2-31642	-	-	89.93	94.11	0.8
m3-30647	-	-	89.97	94.10	0.1
13-m1-9752	**86.83**	93.81	89.80	94.32	0.1
m2-6369	-	-	89.76	94.07	0.0
18-comb.3	85.11	93.85	**95.10**	**97.23**	
15-m1-7762	86.68	**94.27**	89.89	94.20	0.0
m2-31642	-	-	89.93	94.11	0.6
m3-30647	-	-	89.97	94.10	0.1
13-m1-9752	**86.83**	93.81	89.80	94.32	0.2
m3-3583	-	-	89.93	94.10	0.1

Table 3: F1 and EM scores on the official test set and development set (=full training set) of three combined predictions in ensemble and their source individual models. w $\in [0.0, 1.0]$ is each checkpoint's learned weight.

averagely. The other two combinations, comb.2 and comb.3 performed worse than almost all the submitted individual checkpoints. Due to submission limitations, our current guess is that the usage of the whole training set as the development set for individual checkpoint selecting possibly caused overfitting problem. Third, interestingly, in our best comb.1, the best individual checkpoint m1-9752's weight is assigned to be 0.0, while in comb.2/3, its weight are 0.1 and 0.2 respectively. These partly reflect that using the best checkpoint is optimal in the ensemble output and other checkpoints are strong enough through combination.

6 Conclusion

We have described our system architecture, 3 pretrained language models, 18 submissions in which 3 are system combination results and a simple grid-based ensemble learning algorithm. Our best submission achieved F1 (94.77%, ranked 3rd) and exact match (87.62%, ranked 2nd) scores in the official test set. In addition, we analyzed the performance gap of among F1 and exact match and used exact match for ranking our checkpoints.

In the future, it will be interesting to investigate other types of pretrained language models such as the auto-regressive GPTx (Brown et al., 2020) with prompt-based learning (Liu et al., 2021). Also it will be interesting to employ other deep learning based ensemble methods (El-Geish, 2020).

References

Guillaume Becquin. 2020. GBe at FinCausal 2020, task 2: Span-based causality extraction for financial documents. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 40–44, Barcelona, Spain (Online). COLING.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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. Association for Computational Linguistics.

Mohamed El-Geish. 2020. Gestalt: a stacking ensemble for squad2.0.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing.

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. *CoRR*, abs/1907.11692.

Dominique Mariko, Hanna Abi Akl, Estelle Labidurie, Hugues de Mazancourt, and Mahmoud El-Haj. 2021. The Financial Document Causality Detection Shared Task (FinCausal 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

Dominique Mariko, Hanna Abi Akl, Estelle Labidurie, Stephane Durfort, Hugues de Mazancourt, and Mahmoud El-Haj. 2020a. 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.

Dominique Mariko, Estelle Labidurie, Yagmur Ozturk, Hanna Abi Akl, and Hugues de Mazancourt. 2020b. Data processing and annotation schemes for fincausal shared task.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *CoRR*, abs/1905.00537.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

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

A Illustration of Our 18 Submissions

In addition, we depict the F1 and EM scores of these 18 submissions in Figure 3, where the best F1 (94.82%) for single checkpoint is 2-m2-7563 with EM of 70.53% while the best F1 (94.77%) for ensemble system is 16-comb.1 with EM of 87.62%. The F1 scores are only with a difference of 0.05% yet EM's difference is as much as 17.09%.

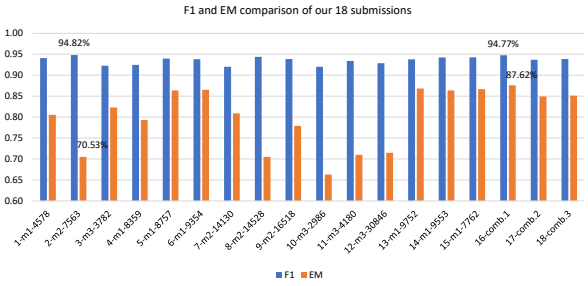

Figure 3: Comparison of stable F1 and astonished EM among our 18 submissions.