Fraunhofer IAIS at FinCausal 2020, Tasks 1 & 2: Using Ensemble Methods and Sequence Tagging to Detect Causality in Financial Documents

Maren Pielka, Anna Ladi, Clayton Chapman, Eduardo Brito, Rajkumar Ramamurthy, Paul Mayer, **Abdul Wahab**, **Rafet Sifa**, and **Christian Bauckhage**
Fraunhofer IAIS, Schloss Birlinghoven, 53757 Sankt Augustin, Germany
Fraunhofer Center for Machine Learning, Germany
maren.pielka@iais.fraunhofer.de

Abstract

The FinCausal 2020 shared task aims to detect causality on financial news and identify those parts of the causal sentences related to the underlying cause and effect. We apply ensemble-based and sequence tagging methods for identifying causality, and extracting causal subsequences. Our models yield promising results on both sub-tasks, with the prospect of further improvement given more time and computing resources. With respect to task 1, we achieved an F1 score of 0.9429 on the evaluation data, and a corresponding ranking of 12/14. For task 2, we were ranked 6/10, with an F1 score of 0.76 and an ExactMatch score of 0.1912.

1 Introduction

Despite the many advances in the field of Natural Language Processing (NLP) in recent years, many challenges dealing with contextual relationships still persist, one of which being causality. Causality, as conveyed in written English, relates a textual description of a cause to a description of its effect(s). While the identification of subordinating conjunctions - such as "because", "since" and "if" - can help in identifying relevant pieces of text forming causal relationships, often richer contextual information is needed. The goal of the FinCausal 2020 shared task (Mariko et al., 2020) is to explore causal relationships from financial and economic context and to determine the likelihood of whether or not a given text contains a causal relationship and then detect the cause and effect parts of the sentence.

2 Data

The data sets are composed of 23808 paragraphs from financial news.[1] The data sets contain paragraphs, with an average size of 214 (+/- 161) characters, each with annotations relevant to the corresponding task. For task 1, the size of the complete data is 22058 paragraphs, each of which is annotated with a "1" (a causal relationship exists in this paragraph - 7.2% of the paragraphs), or "0" (no causal relationship in the paragraph - 92.8% of the paragraphs). The data set for task 2 has a size of 1750 paragraphs, with every paragraph containing exactly one cause and one effect character sequence. These are annotated as character ranges. Since we treat task 2 as a sequence tagging task, we transform these annotations to token level annotations, so that every token in the paragraph gets the label CAUSE, EFFECT or, 0 (for all tokens which belong neither to the cause nor the effect part). This yields a label distribution of 40.8% CAUSE, 40.8% EFFECT, and 18.4% "0". For the experimentation and the development of the models for task 1 and task 2, a 70% training - 30% validation split of the respective data-set (joined practice and trial) was used.

3 System

Tasks 1 and 2 were treated independently. The former was treated as a document classification task (a document being a paragraph in this case) and the latter as a sequence tagging task.

[1]The data set both for task 1 and task 2 is split into a "trial" and "practice" test. To account for any differences in the distribution of the data in the two subsets, we decided to join the two sub-datasets.

Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation, pages 64–68
Barcelona, Spain (Online), December 12, 2020.

3.1 Task 1: Causality Detection

The goal of task 1 is to identify whether a paragraph contains a causal relation or not. The labels are binary, with a 1 indicating the presence of (a) causal statement(s), and a 0 otherwise. For the best performing models, no data pre-processing was used, other than ignoring punctuation and single-character tokens. We explore two approaches: The first is a paragraph embedding paired with shallow Machine Learning (ML) models, while the second is embedding the individual tokens and using a one-dimensional Convolutional Neural Network (CNN) as a classifier. For the first strategy, the text data is transformed into a feature matrix using the scikit-learn (Pedregosa et al., 2011), version 0.23.1 implementation of TF-IDF. This matrix is then used by several classical ML models, including SVMs, Logistic Regression and Random Forests. The best performers were found to be an SVM Classifier [2] and an XGBoost model. XGBoost, or eXtreme Gradient Boosting, is a fairly recent algorithm based on gradient boosting techniques (Chen and Guestrin, 2016). Additionally, a voting classifier model was made from these two models, which predicts the class label based on the argmax of the sums of the prediction confidence values from the SVM classifier and the XGBoost models. We adjust the parameters of our models using the RandomizedSearchCV class from scikit-learn[3]. These models were trained on an Intel i7-8750H CPU. After training, a Voting Ensemble classifier from scikit-learn was built using the trained models as parameters. This ensemble used "soft" voting, that is, the probabilities output by the SGD and XGBoost were averaged and the result was the output of the ensemble.

The second strategy is based on the idea of using CNNs for NLP (Yin et al., 2017). Adjusting the filter size on a one-dimensional CNN also acts similarly to an N-gram, and CNNs are generally very fast at analyzing problems with large feature matrices. This model was implemented using Tensorflow (Abadi et al., 2015) and Keras (Chollet and others, 2015), versions 2.1.0 and 2.3.1 respectively. The data was processed using the Keras tokenizer and then fed into a one-dimensional CNN. The network consists of an embedding layer that uses the 200-dimensional Word2Vec (Le and Mikolov, 2014) embedding from GoogleNews, a convolutional layer, and 2 dense layers. The models were optimized using manual parameter tuning[4]. This model was trained using an Nvidia GTX 1060 Max-Q with 6GB of GPU memory.

3.2 Task 2: Causality Extraction

The goal of task 2 is to identify the parts of a sentence, that correspond to cause and effect, respectively. It is thus a sequence tagging task in which the tokens of a sentence must be assigned either a CAUSE, EFFECT, or 0 tag. One approach to tackle sequence tagging tasks is the use of sequential models such as a Recurrent Neural Network (RNN). We employ the Flair sequence tagger by (Akbik et al., 2018), using ElMo (Peters et al., 2018) and fine-tuned BERT embeddings. For BERT, the transformers library by (Wolf et al., 2019), version 3.0.2 was used.

Using the Flair Sequence Tagger We utilize the Flair Sequence tagger (Akbik et al., 2018) to produce token-level predictions for the causality extraction task. The framework consists of a recurrent neural model with a Conditional Random Field (CRF) and a Long Short Term Memory (LSTM) layer trained for token classification. Learning rate scheduling is applied during training, meaning that the learning rate will be reduced whenever the validation loss does not decrease after 3 epochs. Once the learning rate falls below 0.0001 following this policy, training is stopped.

As a first fine-tuning step, we evaluate different pre-trained word embeddings (ElMo, BERT, Flair, GloVe, and FastText) that are integrated in the Flair framework. We find that ElMo embeddings obtained from the full-sized model (see (Peters et al., 2018) for implementation details) yield the best results on

[2]For the SVM classifier, the scikit-learn implementation was used.

[3]For the TFIDF-SVM the adjusted parameters are: $ngram\_range = (1, 2)$ and $smooth\_idf = false$ for TFIDF, and $alpha = 1e-6$, $loss = log$ for SVM. For TFIDF-XGBoost: $ngram\_range = (1, 1)$ for TFIDF, and $booster = gbtree$, $learning\_rate = 0.3$, $max\_depth = 6$, and $n\_estimators = 100$ for XGBoost.

[4]The best configuration was: a tokenizer that ignored all punctuation and symbols, and set all words to lowercase, and for the CNN: a 1D convolutional layer of 64x3 with relu, a dropout layer with $rate = 0.1$, a global 1D maxpooling layer, a dense layer with an output of 16 with relu, another dropout layer with $rate = 0.1$, and a final dense layer with an output of 1 with sigmoid activation.

Model	F1 score	Recall	Precision
XGBoost	0.942374	0.948957	0.943619
SVM classifier	**0.942909**	0.947604	0.942031
Ensemble	0.937722	**0.949093**	**0.950175**
CNN	0.942066	0.945979	0.940644

Table 1: Task 1 results.

our data, so we use them as word embeddings in the following evaluation steps. The model is being further improved by optimizing the hyperparameters, yielding a batch size of 32, an initial learning rate of 0.1, and a hidden size of 500 neurons. In addition, we adjust the class weights of the model to compensate for the imbalanced label distribution (see section 2). Thus, the weights for the CAUSE and EFFECT classes are decreased to 0.25, and the weight for the 0 class is increased to 0.5. As an alternative embedding method, we also incorporate the fine-tuned BERT model delivered as a baseline for task 1 [5] to the Flair Sequence Tagger. The models were trained on an Nvidia Tesla V100-SXM2 with 32 GB of GPU memory.

Post-processing In addition to the token classification by the Flair sequence tagger, we apply some post-processing to the output to further improve the results. Our first approach is based on the observation, that the classifier sometimes correctly recognizes large parts of a CAUSE or EFFECT sequence, but still predicts 0 for some single tokens in between. Since in most of the cases, every CAUSE or EFFECT is a coherent sequence, it makes sense to account for that using rule-based post-processing. This is done by filling in every ”hole” of up to three tokens in a consequent sequence of CAUSE or EFFECT predictions with the surrounding label. An alternative post-processing approach that was tested, was to smooth the output class probabilities over consecutive tokens, using an average filter with a window size of 3 tokens.

4 Results

The reported results were evaluated on the holdout test set, which was not included in the validation set. The evaluation metrics are reported as defined by the shared task organizers. [6]

4.1 Task 1

For task 1, the best results in terms of F1 score were achieved by the SVM classifier, while the Ensemble of the SVM classifier and the XGBoost classifier performed best in terms of precision and recall (see Table 1). The CNN model performed not significantly different than the less complex ML models.

4.2 Task 2

For task 2, the best scores with respect to the three common metrics are achieved by the Flair sequence tagger model using fine-tuned BERT embeddings, balanced class weights, and applying the first post-processing approach (filling in ”holes” of predicted sequences). Interestingly, our baseline model (ElMo) outperforms the tuned models on ExactMatch. Especially adding balanced class weights seems to cause a drop in the ExactMatch metric. This could potentially indicate some overfitting effect of the simpler model. Table 2 shows two examples that illustrate this behavior. In example 1 (Table 2a), while the simpler model predicts the sequence exactly right, the model with fine-tuning added makes one mistake in between. Example 2 (Table 2b), however, shows that the second model is better at separating cause and effect in a non-straightforward formulated sentence, even though it does not get the labels exactly right. Generally, the fine-tuned model tends to leave larger ”holes” between ”cause” and ”effect” sequences, than the simpler model does.

[5] https://github.com/yseop/YseopLab/tree/develop/FNP_2020_FinCausal/baseline/task1

[6] See website of shared task: https://competitions.codalab.org/competitions/23748

Sentence	They	fell	(...)	to	4p	on	Wednesday	as	analysts	lowered	price	targets	and	cut	forecasts.
Model 1 prediction	E	E	(...)	E	E	E	E	0	C	C	C	C	C	C	C
Model 2 prediction	E	E	(...)	E	E	E	0	0	C	C	C	C	C	C	C
True label	E	E	(...)	E	E	E	E	0	C	C	C	C	C	C	C

(a) Example 1

Sentence	Suppose	(...)	they	could	borrow	at	-0.25%	NLY	would	be	getting	paid	$262	million.
Model 1 prediction	E	(...)	E	E	E	E	E	E	E	E	E	E	E	E
Model 2 prediction	C	(...)	0	C	C	C	E	E	E	E	E	E	E	E
True label	C	(...)	C	C	C	C	0	E	E	E	E	E	E	E

(b) Example 2

Table 2: Example predictions of model 1 (ElMo) and model 2 (fine-tuned BERT, balanced, post-processing), on two sentences from our validation split (part of the practice data set). "C" stands for "cause", "E" for "effect". Due to space constraints, only the relevant part of the sentence is displayed.

Model	F1 score	Recall	Precision	ExactMatch
ElMo	0.740380	0.745353	0.741605	**0.231975**
ElMo, balanced	0.726232	0.71307	0.777991	0.184953
ElMo, balanced, post-processing	0.743371	0.734344	0.769642	0.210031
ElMo, balanced, post-processing (probability smoothing)	0.741773	0.734672	0.759757	0.152038
ElMo, task1 data augmentation	0.731066	0.733866	0.729065	0.188088
ElMo, task1 data augmentation , post-processing	0.737836	0.748546	0.735892	0.191223
fine-tuned BERT, balanced, post-processing	**0.759982**	**0.748933**	**0.799503**	0.191223

Table 3: Task 2 results. The Flair sequence tagger model was used to produce all of those results.

5 Discussion

Our present system relies heavily on the choice of embeddings, as well as on large pre-trained language models. This could possibly be changed if we had a more extensive and consistent data set for training. One of our observations in this regard was, that the provided data included a number of irregularly formatted paragraphs (for example headlines or bullet points). We believe that our algorithms would benefit from further analysis and possible cleaning of such instances. Additionally, the current data does not allow us to explore other approaches that are based on syntactic features.

Regarding task 1, the results of the CNN could be potentially improved by additional hyperparameter tuning. The CNN model showed good results already in early epochs, but was not able to outperform the less complex ML models, which were extensively tuned. With respect to task 2, it would have been interesting to test more combinations of embeddings, data augmentation, and post-processing, which was not possible due to limited time and computing resources.

6 Conclusion and Outlook

Identifying causality in text can be a helpful tool for many empirical use cases. In the context of financial document analysis, it could be used to identify important facts and developments in the annual report of a company. We can potentially extend our previous work on various downstream-tasks on financial reports by incorporating causality, for instance for key performance indicator extraction (Brito et al., 2019b) , contradiction detection (Sifa et al., 2019b) , or content-based text classification and consistency checks (Sifa et al., 2019a) . We can also exploit causality detection in the context of text summarization. Causal sentences may indicate content richness, which is useful not only to extract the most relevant sentences of an original text within an extractive summarization setting (as presented in (Brito et al., 2019a)) but also when we evaluate the quality of a generated summary from a wide range of features (Biesner et al., 2020). In our future work, we are planning to address such applications, building on our experiments and insights from this challenge.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

David Biesner, Eduardo Brito, Lars Patrick Hillebrand, and Rafet Sifa. 2020. Hybrid ensemble predictor as quality metric for German text summarization: Fraunhofer IAIS at GermEval 2020 task 3. In *Proceedings of the 5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*.

Eduardo Brito, Max Lübbering, David Biesner, Lars Patrick Hillebrand, and Christian Bauckhage. 2019a. Towards supervised extractive text summarization via RNN-based sequence classification. *arXiv preprint arXiv:1911.06121*.

Eduardo Brito, Rafet Sifa, Christian Bauckhage, Rüdiger Loitz, Uwe Lohmeier, and Christin Pünt. 2019b. A hybrid AI tool to extract key performance indicators from financial reports for benchmarking. In *Proceedings of the ACM Symposium on Document Engineering 2019*.

Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *CoRR*, abs/1603.02754.

Francois Chollet et al. 2015. Keras.

Q. V. Le and T. Mikolov. 2014. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

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.

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*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Rafet Sifa, Max Lübbering, Ulrich Nütten, Christian Bauckhage, Ulrich Warning, Benedikt Fürst, Tim Khameneh, Daniel Thom, Ilgar Huseynov, Roland Kahlert, Jennifer Schlums, Anna Ladi, Hisham Ismail, Bernd Kliem, Rüdiger Loitz, Maren Pielka, Rajkumar Ramamurthy, Lars Hillebrand, Birgit Kirsch, and Thiago Bell. 2019a. Towards automated auditing with machine learning. In *Proceedings of the ACM Symposium on Document Engineering 2019*. ACM.

Rafet Sifa, Maren Pielka, Rajkumar Ramamurthy, Anna Ladi, Lars Hillebrand, and Christian Bauckhage. 2019b. Towards contradiction detection in German: A translation-driven approach. In *Proc. of IEEE SSCI 2019*.

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.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative Study of CNN and RNN for Natural Language Processing. *CoRR*, abs/1702.01923.