The 3rd Financial Narrative Processing Workshop (FNP 2021)

15-16 September 2021. Lancaster, UK

Preface

Welcome to the 3rd Financial Narrative Processing Workshop (FNP 2021). This year we are running a 2-day event sponsored and funded by the Data Science Institute (DSI) at Lancaster University and Yseop. This is an international gathering of researchers and keynote speakers working on Financial Narratives from computing, accounting & finance.

Following the success of the First FNP 2018 at LREC'18 in Japan, the Second FNP 2019 at NoDaLiDa 2019 in Finland and as well as the Multiling 2019 Financial narrative Summarisation task at RANLP in Bulgaria and the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation (FNP-FNS 2020) at COLING 2020 in Barcelona, Spain, we have received a great deal of positive feedback and interest in continuing the development of the financial narrative processing field, especially from our shared task participants. This has resulted in a continuation of the collaboration between FNP and MultiLing workshop series to co-organise the 3rd Financial Narrative Processing Workshop (FNP 2021).

The FNP 2021 workshop achieved our aim of supporting the rapidly growing area of financial text mining. We ran three different shared tasks focusing on text summarisation, structure detection and causal sentence detection, namely FNS, FinToc and FinCausal shared tasks respectively. The shared tasks attracted more than 100 teams from different universities and organisations from around the globe. The shared tasks resulted in the large scale experimental results and state of the art methods applied mainly to financial data. This shows the importance and growth of this field and we want to continue to be associated with top NLP venues.

The workshop focused mainly on the use of Natural Language Processing (NLP), Machine Learning (ML), and Corpus Linguistics (CL) methods related to all aspects of financial text summarisation, text mining and financial narrative processing (FNP). There is a growing interest in the application of automatic and computer-aided approaches for extracting, summarising, and analysing both qualitative and quantitative financial data. In recent years, previous manual small-scale research in the Accounting and Finance literature has been scaled up with the aid of NLP and ML methods, for example to examine approaches to retrieving structured content from financial reports, and to study the causes and consequences of corporate disclosure and financial reporting outcomes.

FNP 2021 continued the excellent collaboration Fortia Financial Solutions (www.fortia.fr) and Yseop (www.yseop.com). Both firms are pioneers in Artificial Intelligence, NLP and Natural Language Generation (NLG). Both firms work on applying those methods to automatically analyse and extract from financial documents and disclosures.

We accepted 20 submissions. Each paper was reviewed by up to three reviewers. The submissions distribution is as follows: 5 main workshop papers and 15 shared task papers. The papers covered a diverse set of topics in financial narratives processing reporting work on financial reports from different stock markets around the globe presenting analysis of financial reports and using state of the art NLP methods such as the use of latest word embeddings.

The quantity and quality of the contributions to the workshop are strong indicators that there is a continued need for this kind of dedicated Financial Narrative Processing workshop. We would like to acknowledge all the hard work of the submitting authors and thank the reviewers for the valuable feedback they provided. We hope these proceedings will serve as a valuable reference for researchers and practitioners in the field of financial narrative processing and NLP in general.

Dr Mahmoud El-Haj, General Chair, on behalf of the organizers of the 3rd FNP workshop, September 2021.

Table of Contents

Data Driven Content Creation using Statistical and Natural Language Processing Techniques for Financial Domain

Ankush Chopra* Fidelity Investments, AI CoE Bengaluru, India ankush01729@gmail.com

Prateek Nagwanshi* Fidelity Investments, AI CoE Bengaluru, India kunal.pn@gmail.com

Sohom Ghosh* Fidelity Investments, AI CoE Bengaluru, India sohom1ghosh@gmail.com

Abstract

Over the years customers' expectation of getting information instantaneously has given rise to the increased usage of channels like virtual assistants. Typically, customers try to get their questions answered by low-touch channels like search and virtual assistant first, before getting in touch with a live chat agent or the phone representative. Higher usage of these low-touch systems is a win-win for both customers and the organization since it enables organizations to attain a low cost of service while customers get served without delay. In this paper, we propose a two-part framework where the first part describes methods to combine the information from different interaction channels like call, search, and chat. We do this by summarizing (using a stacked Bi-LSTM network) the high-touch interaction channel data such as call and chat into short search-query like customer intents and then creating an organically grown intent taxonomy from interaction data (using Hierarchical Agglomerative Clustering). The second part of the framework focuses on extracting customer questions by analyzing interaction data sources. It calculates similarity scores using TF-IDF and BERT (Devlin et al., 2019). It also maps these identified questions to the output of the first part of the framework using syntactic and semantic similarity.

1 Introduction

In the current age information is the key to everything. Faster access to correct information has become an essential need. Customers interact with service providers through various channels looking for information. Information that is being sought, help customers make the right decision or finish a task/transaction that they are intending to do.

Customers use channels including but not limited to on-site search, call, live chat, virtual assistant, and emails. Customers use these channels in an order where they go from simple to more complex channels successively if their information need is not fulfilled by simpler channels. Search is the simplest channel since it is always available, and customers can look for the information very quickly. Next, comes virtual assistant, since even this has high availability, but it may or may not have answers to all the questions. Next, come chat, call, and email. Information provided through a low complexity channel tends to result in not only faster customer query resolution but also proves to be cost-effective for the organization.

Customers interact differently while using these different channels. The nature of the queries from these different channels also differs a lot due to the idiosyncrasies of the channels. For example, the search will have noticeably short inquiries that would not be a well-formed customer question. Virtual assistants get proper context-independent questions. Live chat and call tend to have pleasantries and general chit-chat along with the inquiry that may be broken into multiple sentences.

Often information that is being sought also has a sense of recency attached to it, and a large part of the customer base might also have similar queries. These queries are mostly related to a product or service that an organization offers, a recent event like new tax rules or initial public offering (IPO), specific to customers' current state and so on.

In this paper, we propose a framework to identify the hot topics/intents and questions related to these hot topics that customers are seeking answers to. The framework is divided into 2 parts, first where we identify the hot topics/intents and iteratively cluster those to create an organic intent hierarchy. The second part focuses on identifying and extracting the customer questions from the interaction

*Equal Contribution

data and mapping those to the topics or topic clusters. This also helps content writers to get an idea regarding the topics on which they need to write articles for satisfying the customers' curiosity.

Our contributions: We have developed two NLP based models i) First one performs hierarchical clustering of customer interactions iteratively ii) Second one maps questions to the cluster heads with descending order of frequency.

The next section will consist of a narration of prior works relating to this. After that, we will describe the problem, which will be followed by a section describing the solution methodology. We will shed some light on experimentation and results after the methodology section. Finally, we will conclude the paper by talking about the future enhancements we are about to bring to the framework.

2 Related Works

Tsai et al. in their paper (Tsai and Wang, 2013) described an approach that mined information from financial reports of a set of organizations. Using this information, it ranked these organizations as per their risk levels. They used learning-to-rank algorithms and benchmarked their model with a regression-based one. They also used Term Frequency Inverse Document Frequency (TF-IDF) (Sammut and Webb, 2010) matrix of unigrams as features and trained a Support Vector Machine (SVM) (Vapnik et al., 1995) model. In the paper (Lewis and Young, 2019), Lewis et al. discussed how different Natural Language Processing based approaches (like Word Counts, Latent Dirichlet Allocation etc.) can be used to analyze Financial Texts.

A process of predicting Financial Markets (in terms of volatility, returns and traded volumes) using Google's search queries from four English speaking countries has been narrated by Perlin et al. in their paper (Perlin et al., 2017). They studied how search patterns of some specific finance related terms on Google were related to the behaviour of Financial Markets. They used vector autoregression for modelling. Similar work had been presented by Mao et. al in their paper (Mao et al., 2011). They analyzed how various sentiment scores obtained from Tweets, Google search volume, Negative News Sentiment were related to the market conditions in terms of trading volumes, gold prices and so on.

Chanel	**# Interactions**	**# Distinct Interactions**
Search	29.9 M	2.9 M
Live Chat	12.2 M	96.7 K
Call	31.4 M	31.4 M

Table 1: Data distribution

Li et al. in their paper (Li et al., 2014) proposed a solution for personalizing web search results using semantic and click-based features. They used three types of models namely XCode, Deep Structured Semantic Model and Convolutional Deep Structure Semantic Model. This led to a better ranking of the search results. However, this solution is generic and does not specifically deal with the financial domain.

Litvak et al. in their paper (Litvak et al., 2020) described an approach of creating hierarchical summaries from financial reports. They used CODRA framework (Joty et al., 2015) for discourse parsing.

It is interesting to note that all the papers described above (except the last one) dealt with predicting events/conditions of Financial Markets. None of these works relates to how customer interactions related to finance through different channels like search, calls, live chat, chat with virtual agents can be used to identify what their needs are and help the content writers prioritize their work.

3 Problem Statement

Given a set, $I = \{i_1, i_2 \ldots i_n\}$ consisting of user interactions $(i_1, i_2 \ldots i_n)$ from the financial domain, we develop a system capable of extracting trending topics/intents and questions from it.

4 Data and Preprocessing

The dataset consists of interactions that users had with Fidelity Investments[1] through various channels like search, live chat and call over a period of one year (June 2020 - May 2021). The data distribution is mentioned in Table 1. We list our data collection and pre-processing steps here.

4.1 Calls and Chats data

Customer care representatives (reps) record a summary note for a fraction of customer calls that come in. We call these summaries - 'repnotes'. Due to their idiosyncrasies, reps introduce a lot of variations into the repnotes even if they mean the same

[1] fidelity.com

thing. For example, "*customer contacted to reset the password*" and "*customer asked for the help with account reset*" are worded differently even if the meaning is the same. We used customer call data for training the intent models. We had both repnotes and transcripts for a portion of these calls. This made the call data ideal for modelling, as it gave almost ready to use training data. Call transcripts were input into the model and repnote was used as the summary. We applied certain filter conditions to select the right modelling data:

- We took data only from inbound calls since it'll be applied to inbound interactions later.
- Since we were interested in extracting short intents of the calls, we only took calls where repnotes of length 6 or less were present.

We performed cleaning steps on the call transcripts and repnotes mentioned below:

- Most call transcripts contained system messages (like *"party has left the session"* and *"This conversation is get recorded"*). They carried certain system meta-information only and were removed.
- Call transcript and repnotes were converted to lower case
- Personal information of the customers which were masked were removed from the call transcripts and repnotes. E.g. *[name], [number redacted], [unk]*, etc.
- Non-vocalized noise transcription which had been performed on call transcripts was removed
- Contraction replacement was performed on both call transcript and repnotes to normalize the text like "I've" and "We haven't" were expanded to "I have" and "We have not".
- Non-informative prefixes like "customer contacted" or "customer asked" were removed from repnotes since they did not add any value.

Chat data comprised interaction between customers and representatives through text messages. We perform similar cleaning on the chat data as well.

4.2 Search Data

Search logs needed to be filtered and preprocessed before it is combined with the generated themes from call and chat. Since this data is from financial/ brokerage firm Fidelity Investments, the search log contained a lot of stock tickers, mutual fund tickers, names of the listed entities and so on. As these inquiries were used for getting the quote for respective stocks or funds, they were not relevant for the intent extraction and content creation process.

Analysis of the search logs revealed that a small percentage of search queries were responsible for the majority of the searches. Search logs had an extremely long tail. We selected queries that were searched for at least 5 times. These queries constitute 97% of the search volume. Furthermore, this made the search data manageable.

5 Methodology

This work is divided into two major parts - identification of hot topics by creating an intent tree organically and extracting popular questions from customer interactions. We initiate by describing the first of the two major parts of the framework. The intent tree created represents an organic hierarchy of topics/intents that customers were interested in. We used search queries, call transcripts and live chat from the customers to build this view. By nature, search queries were small and succinctly defined the intent of the customers in the majority of cases. Call transcripts and chat were verbose and had customer intent hidden somewhere in the body of the conversation. To be able to use all these communication channels in tandem, they should have had the same structure. Hence, we first worked towards extracting the customer intent from the calls and chats.

5.1 Extracting Intents from Calls and Chats

We performed abstractive summarization of the calls, given that the nature of the relationship between rep-notes and call transcripts was inherently abstractive. The proposed model is based on 2 stacked Bi-LSTM (Long Short Term Memory) (Hochreiter and Schmidhuber, 1997) Sequence to Sequence with Attention (Vaswani et al., 2017) architecture. We did not use any pre-trained embedding due to the nature of the task. Instead, we are learning the embedding while training for this task.

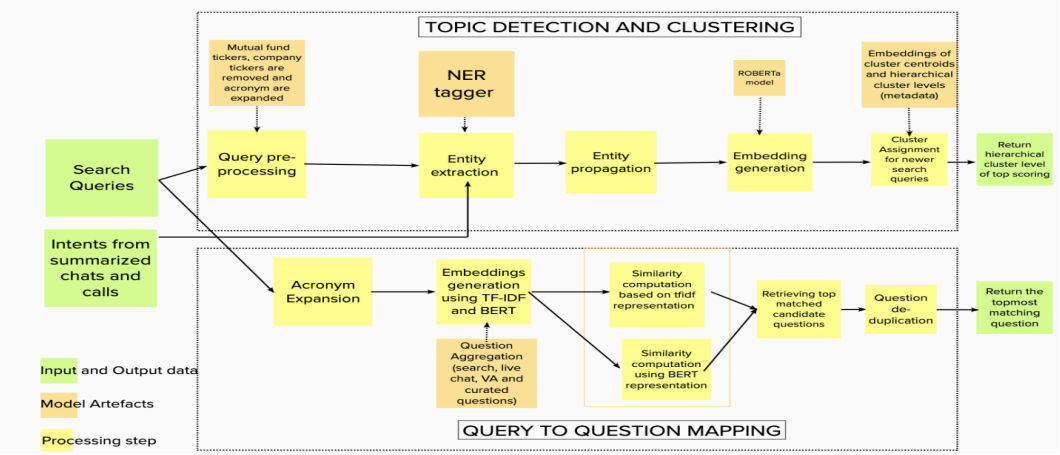


Figure 1: Flowchart representing our proposed approach

This model was trained for 27 epochs and training terminated due to early stopping criteria. The model had categorical cross-entropy accuracy of 79% on the validation set. This model is used to generate the customer intents behind all incoming calls and chat transcripts irrespective of whether they have repnotes or not. These generated repnotes look similar to short search queries and explain the intents behind the calls.

5.2 Combining and clustering search and other interaction data sources

Search queries were mostly specific and short in length. Thus, we used them as it is. Once data from all the channels were in a similar format, we proceeded to the pre-process and normalize them. We replaced acronyms with their definitions, removed repetitive words and any customer specific information. We then converted these intent phrases to embedding using the RoBERTa (Liu et al., 2019) pre-trained model (768 dimensions). We made use of the sentence transformer (Reimers et al., 2019) to come up with sentence embeddings. We performed standardization and reduced the dimensions to 300 using Principal Component Analysis (PCA). This was done to make the data manageable.

We used a Financial Named Entity Recognition (NER) module developed in-house to identify the business entities (product and services) present within customer intents. We divided the intents into smaller groups based on the entities that are present in them. This helped us to achieve cleaner clusters at the product and service level.

We performed iterative clustering within each of the intent cohorts using agglomerative hierarchical clustering (4 levels). We first group the generated intents which had at least x similarity and calculated cluster centroid for all the newly formed clusters. These cluster centroids were again clustered in the groups which had at least $x = x - \Delta x$ similarity. Depending on the business requirement we repeated the above steps multiple times. For example, if we needed 3 level cluster and the similarity threshold for first level clustering was x=0.85 and for subsequent levels, the criteria were relaxed by Δx=0.05. Then we had clusters with 0.85, 0.80 and 0.75 similarity thresholds. We used cosine similarity as the measure of affinity.

5.3 Query to Question Mapping

Here, we propose an algorithm for mapping a search query to its probable questions present in an existing database. The question database contained all the questions that were being asked across different channels (live chat, virtual agent, and search). In addition to it, a set of curated questions were also present. Showing questions for a given search query gave an overall picture of what questions were being asked related to the keyword which was searched.

There were several steps involved in mapping a query to its candidate questions. The following lines explain each of these steps.

Question Detection: We used search queries, live chat, and virtual agent messages for detecting questions using a question detector algorithm developed in-house

Question Aggregation: Due to the different nature of the channels, there were slight variations in detected questions. We needed some bit of preprocessing and normalization of these questions to get the overall volume of certain types of questions.

In addition to simple pre-processing and we also applied acronym expansion (like IPO to Initial Public Offering), to get a unified set of questions along with their aggregated count.

Embedding Creation: For matching a query to its probable questions, we needed to find the similarity between the search query and questions. We further calculated the distance between the query embedding matrix and the embedding matrix created from questions. The embedding techniques used in this work constitute TF-IDF and Sentence BERT. TF-IDF had been used for keyword-based embedding and Sentence-BERT (Bidirectional Encoder Representations from Transformers) (Reimers et al., 2019) had been used for creating semantic embedding.

TF-IDF: We performed different pre-processing steps to clean the data such as acronyms were replaced by their definitions and customer-specific information was removed and so on. All questions had been lemmatized to their base word. Moreover, after removing all stop words these normalized questions were used for creating the TF-IDF (unigrams and bi-grams were used) matrix which was then used for matching the vectors created from questions.

SentenceBERT: (Reimers et al., 2019) has presented a modification of BERT (Devlin et al., 2019) model that uses Siamese network to derive semantic sentence embeddings that can be used to calculate cosine similarity between sentences. For our experiments, we had used Sentence Transformer model based on BERT and RoBERTa. We performed all the pre-processing steps which have been mentioned in the previous section to clean the data excluding the lemmatization step. After removing stop words, the normalized question was used for creating the embedding of 768 dimensions from Sentence-BERT. It was used for finding the similarity with embedding created for the query using Sentence-BERT.

Mapping Query To Questions: For finding probable questions for a given search query, we calculated the cosine similarity between question embeddings (created using TF-IDF and Sentence -BERT) and query embedding. All questions beyond the similarity threshold (0.89 for TF-IDF and 0.86 for BERT) were collected. There was only a slight variation among all such questions. To detect near-duplicate questions, clustering was done and question with high frequencies was assigned as cluster head. For a given search query only the cluster heads were displayed. We present the entire workflow in the Figure 1.

6 Experimentation and Results

We conducted separate experiments for each of the modules as narrated in the following parts.

6.1 Extraction of Intents from Interaction Data

Firstly, we started with the pre-processing and cleaning of the call transcripts and repnotes. We removed system noise, transcription-induced noise, and masked tokens from both sources. We also performed case normalization, contraction replacement on both data sources. We then tokenized both cleaned transcripts and repnotes separately. We divided the data into the train, validation, and out-of-time test set. Close to 35% of the cleaned repnotes were of length 6 and less, close to 40% were of length 7 to 17 and remaining were of more than 17 words. 90% of the call transcripts were of length 450 words or less. For modelling, we considered all the calls with less than or equal to 450 tokes in the transcripts where repnotes of length 6 or less were present.

Since this is an abstractive summarization problem, we chose to use sequence-to-sequence (s2s) Recurrent Neural Networks (RNN) architecture for modelling this. We chose LSTM (Hochreiter and Schmidhuber, 1997) variant of RNNs, since it is proven effective in capturing longer term dependency. We started off with s2s architecture without attention. It did not yield good results. Upon closer examination, it was clear that performance was getting worse with an increase in the input sequence length.

Next, we tried the s2s with attention (Vaswani et al., 2017) since it is known to work better for longer input sequences. This significantly helped in improving the model's performance. We experimented with the common LSTM hyperparameters like the number of layers in LSTM, gradient clipping, dropout, recurrent dropout etc. We also tried pre-trained transformer-based fine-tuning using BERT andT5 but they did not perform as good as LSTM for the validation set. It is probably due to the poorly transcribed data with improper sentence and grammar structure. Inference using transformer-based models was slow as well. We used s2s with attention model for generating the

interaction intent after comparing the performance of the above model using manual validation.

We combined the generated intents with the customer search queries. In our context, interactions belong to two broad categories, customers looking for a quote of a traded entity like a company or mutual fund and seeking information on any product, service, or recent transaction. We had taken the intents belonging to the latter case since the former did not require new content to be written for it to be answered. We also normalized the product name variations, acronyms, contractions, and removed the repetitive words and phrases.

Even after normalization and cleaning, the number of intents was in the range of thousands. We decided to iteratively group these intents into homogeneous clusters. We began with clustering the cleaned intents into groups where constituent members were highly similar to each other. The intent with the highest interaction count was chosen as cluster name. We also calculated the centroid for all the newly formed clusters and clustered them again. This time we reduced the similarity threshold marginally. This gave us a new cluster of clusters. We similarly named these clusters as before. By doing this iteratively a few times, we ended up with few hundred clusters that had links to one of the products or services offered by the organization. We used agglomerative hierarchical clustering since the number of clusters were not known apriori.

We tried few different featurization techniques to get the best clusters. Firstly, we created features using TF-IDF for the cleaned intents and tried clustering these into smaller buckets. We then used PCA to make the embedding size manageable (300 dimensions were retained). As expected, these clusters were failing to capture the semantic similarity. Intents like "account reset" and "password change" were not getting grouped.

Next, we generated the vector representation of the intent phrases using Word2Vec (Mikolov et al., 2013). We tokenized the phrases into words and took the average of the word embeddings present in the intent phrase. It performed better than TF-IDF based approach, but it wasn't doing well where a word had a different meaning in a different context. For example, "option" in "payment option" and "option trade" has a completely different meaning which word2vec based model was not able to disambiguate. Additionally, it was not helping with misspellings as it provides word-level embeddings. Further, we experimented with Transformer based embedding models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). We used Sentence Transformers to get the phrase embedding for the intent phrases while using BERT or RoBERTa in the back-end. We could see that RoBERTa performed better than BERT. RoBERTa (large base model) performed the best when we looked at the silhouette scores to compare the performance of different clustering methods. Manual validation also indicated that transformers based embedding models were able to capture the semantic and syntactic similarity better than the other models. Table 2 shows the model performance. The organically generated hierarchy of intents has been depicted in Figure 2.

cluster level	search query	calls
0	0.09	0.48
1	0.10	0.35
2	0.08	0.23
3	0.07	0.12

Table 2: RoBERTa based Silhoutte Scores

Figure 2: Hierarchical Tree Visualization

6.2 Extraction of Questions from Interaction Data

Query to question mapping is a task to assign 'n' candidate questions to a search query. The value of 'n' differs for each query depending upon the questions being asked around that search query and the similarity between search query and questions. We performed our experiments using different kinds of embeddings such as TF-IDF, BERT and combined

output of TF-IDF and BERT based similarity. We started with normalizing all the questions, which included pre-processing and expansion of acronyms with their definition. Since we did not have tagged data to evaluate this model, we manually evaluated it. We chose the top 10,000 search queries and created the probable questions mapping for them. We used a sample of 1000 records for manual evaluation.

Initially, we started with TF-IDF for creating features of aggregated questions. We performed case normalization and removed noisy characters like '?','.' from the end of questions. Then we replaced the acronyms with their definition and after that, we converted the words with their lemmatized version. We further tokenized the questions and removed the English stop words like 'a', 'the', etc. Using the normalized version of questions, we had created a TF-IDF matrix ($\text{TF-IDF}_{\text{ques}}$). For search queries, we also performed the same preprocessing and created the TF-IDF matrix ($\text{TF-IDF}_{\text{query}}$) for search queries as well. We calculated the cosine similarity between the question $\text{TF-IDF}_{\text{ques}}$ and $\text{TF-IDF}_{\text{query}}$ and evaluated the performance at different levels of threshold. This model was giving descent performance on queries that had some matching words with the question. However, queries like 'withdrawal' did not have any matching questions even if they contained questions related to 'withdrawal'. So, this TF-IDF model was unable to capture the semantic aspect. For covering the semantic aspect as well, we further experimented with sentence transformers to get the phrase level embedding using BERT. We used the normalized version of the question for creating the embedding matrix $\text{SBERT}_{\text{ques}}$ and using the normalized query created $\text{SBERT}_{\text{query}}$. Finally, we calculated the cosine similarity between $\text{SBERT}_{\text{query}}$ and $\text{SBERT}_{\text{ques}}$ to get the questions for a query at the various levels of threshold. We also tried changing the BERT model with RoBERTa for creating $\text{SRoBERTa}_{\text{ques}}$ for questions and $\text{SRoBERTa}_{\text{query}}$ for queries. We calculated the cosine similarity between $\text{SRoBERTa}_{\text{query}}$ and $\text{SRoBERTa}_{\text{ques}}$ to get the set of candidate questions for a query. The model was able to capture most of the variations of questions present in the database. Still, the coverage was very low compared to the TF-IDF method. Finally, we combined the output of TF-IDF and Sentence Transformers with RoBERTa (Liu et al., 2019) at the back-end to get the final candidate set of questions for a given query. It captured the variations from semantic embedding and word-based matching using TF-IDF. Since we did not have the tagged data to evaluate this task. We chose to perform a manual evaluation and calculate the precision score for evaluating the model. We present the individual performance of both the methods at various thresholds in Table 4. We calculated the precision score based on a manual evaluation of 1000 query question pairs. After looking at the precision score we concluded that both the methods were performing well for threshold $>= 0.8$. After doing a failure case analysis, we saw that in plenty of cases where TF-IDF was failing, and Sentence-BERT was performing good and vice-versa. We tried combining TF-IDF that capture the bag-of-words aspect and Sentence-BERT that covers the variations and semantic aspect of the language. We saw if we combine both the models then we are getting much broader coverage with plenty of variations of questions for a query, which is not the case with individual semantic or word-based methods. Some examples of search queries and corresponding questions extracted are presented in Table 4. Finally, we clubbed the output of TF-IDF and Sentence Transformers (with RoBERTa at the back-end) to get the final candidate set of questions for a query.

Th	**Pr TFIDF (%)**	**Pr SBERT(%)**
0.4	84.11	83.80
0.5	87.00	84.20
0.6	87.50	87.70
0.7	87.56	87.32
0.8	90.70	88.93
0.9	92.39	92.30

Table 3: Comparison of Precision(Pr) Scores of Query to Question Model trained using TF-IDF and Sentence BERT based similarity for different thresholds (Th)

SQ	**EQ**
tax form	where do i view my tax form?
tax form	when do tax forms get sent out?
tax form	can i get my tax form?
direct deposit	how to edit direct deposit?
direct deposit	what is a direct deposit?

Table 4: Search Queries (SQ) and Extracted Questions(EQ)

7 Conclusion and Future Works

In this paper, we discussed the process to extract insights from customer interactions by clustering them hierarchically. It led to the creation of an intent hierarchy organically. We further narrated a methodology to mine queries from these interactions and rank them. Various regulations do not permit machine generated answers to be directly given to the customers. This is specifically the case in sectors like finance and healthcare. Hence decoupling of the answer generation from the question mapping was needed to comply with the regulation. Our proposed system would help content writers efficiently identify the topics and questions which are being asked by a large number of customers. Once they write answers to these questions, a system like the one described in (Chopra et al., 2020) could be used to serve these answers directly to the users through channels like search, Virtual Agents (Chatbots) and so on.

In future, we would like to incorporate market events to decide the prioritization of content making process. Furthermore, we would like to remove queries that already have enough content. We want to assign more priority to those topics which the customers searched for but did not lead to any fruitful results. Lastly, we want to do an extensive evaluation using external parameters like measuring the number of searches that are not followed by a call, reduction in call volumes and so on.

References

Ankush Chopra, Shruti Agrawal, and Sohom Ghosh. 2020. Applying transfer learning for improving domain-specific search experience using query to question similarity. In *2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI 2020, New York, NY, USA. Association for Computing Machinery.

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.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2015. CODRA: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

Craig Lewis and Steven Young. 2019. Fad or future? automated analysis of financial text and its implications for corporate reporting. *Accounting and Business Research*, 49(5):587–615.

Xiujun Li, Chenlei Guo, Wei Chu, Ye-Yi Wang, and Jude Shavlik. 2014. Deep learning powered in-session contextual ranking using clickthrough data. In *In Proceedings of NIPS, 2014*.

Marina Litvak, Natalia Vanetik, and Zvi Puchinsky. 2020. Hierarchical summarization of financial reports with RUNNER. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 213–225, Barcelona, Spain (Online). COLING.

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 preprint arXiv:1907.11692*, abs/1907.11692.

Huina Mao, Scott Counts, and Johan Bollen. 2011. Predicting financial markets: Comparing survey, news, twitter and search engine data.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Marcelo S. Perlin, João F. Caldeira, André A. P. Santos, and Martin Pontuschka. 2017. Can we predict the financial markets based on google's search queries? *Journal of Forecasting*, 36(4):454–467.

Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Claude Sammut and Geoffrey I. Webb, editors. 2010. *TF–IDF*, pages 986–987. Springer US, Boston, MA.

Ming-Feng Tsai and Chuan-Ju Wang. 2013. Risk ranking from financial reports. In *Advances in Information Retrieval*, pages 804–807, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vladimir Vapnik, Isabel Guyon, and Trevor Hastie. 1995. Support vector machines. *Mach. Learn*, 20(3):273–297.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

A sequence to sequence transformer data logic experiment

Danxin Cui *
danxin.cui
@sorbonne-nouvelle.fr

Dominique Mariko *
dmariko
@yseop.com

* equal contribution

Estelle Labidurie
elabidurie
@yseop.com

Hugues de Mazancourt
hdemazancourt
@yseop.com

Patrick Paroubek
pap@limsi.fr

Abstract

In this paper we present experiments to evaluate how a T5 model behaves with regard to input data fidelity. The rationale behind these experiments is to evaluate if a sequence to sequence transformer can be constrained into generating the specifics of a financial report, and more generally whether it can trustfully reproduce a semantic logic, and to what extent.

1 Introduction

T5 by Raffel et al. (2020) recently demonstrated strong constrained and data to text generation capabilities. Experiments have been lead on AQG tasks (Grover et al. (2021)) and on the WebNLG dataset as to explore the data to text capabilities of a T5 model. In particular, Kale and Rastogi (2020) demonstrates T5 model shows interesting capacities in generalization to new domains and relations, and Kasner and Dusek (2020) proposes significant text generation experiment even without any in-domain examples.

Text generation in Finance can be very demanding as to the level of constraint a neural model should comply with. The objective of our experiment is to evaluate which data formalism we should build to achieve similar results as the results achieved by T5 models on WebNLG tasks.

In order to produce this evaluation, we chose to create a data set focusing on semantic intentions. Intentions are objects describing Natural Language Generation (NLG) pipelines based on Abstract Categorical Grammars semantic and syntactic items, as defined by Salmon (2017), that can be specialized and combined together. We also implemented a set of metrics for NLG evaluation based on BLEU and BERT-SCORE.

2 Corpus

The initial corpus for this experiment is a set of 4159 public online US and UK Market Reports. We limit the experiments to the financial domain as to prove more accurate results.

2.1 Corpus generation

Raw text is extracted with a home made pdf extractor based on PDFMiner, deleting all tables and titles. A corpus analysis is performed on these raw extractions, leading to the definition of handcrafted grammars describing each intention. These grammars allow to tag each sentence belonging to one of the intentions of interest, and to extract for each sentence a set of relevant chunks which are then transformed into triples. These intentions are currently defined and used by Yseop's generation core engine. For the sake of the experiment, we choose to retain only simple intentions and sentences which can also be produced by this NLG engine.

2.2 Corpus transformation

2.2.1 Data logic

To ensure the precision and accuracy of the generated sentences, we have chosen a data-to-text representation method, which particularizes key elements of sentences in our financial corpus and extract triples, using an automated method close to Li et al. (2020).

This method was applied as to define a corpus of intentions. An intention is a sentence corresponding to a specific expression of a financial indicator's value. Yseop has shown that a handful of such intentions are sufficient to describe a data-driven narrative in a speciality domain, such as Finance. For

instance, an intention *DescribeValue* is a sentence stating the value of a financial indicator at a precise time and an intention *DescribeVariation* is a sentence describing the variation in time of a financial indicator's value. We use the prefix *Merge* to define a sentence composed of two or more intentions. In order to identify and extract these intentions in our corpus, a Ruta grammar (KLUEGL et al., 2016) was created to automate the triples extraction, imitating Gardent et al. (2017) data modeling. This grammar first uses dictionaries as well as POS-tag patterns to identify financial key elements related to these intentions and characterize them into one of the following categories:

- financial indicator
- reference (time and geographical element)
- measure
- predicate

Indicators, measures and dimensions are generic elements that can be found in all intentions. Predicates, on the other hand, vary according to the intention. To create a dictionary that take into account this specificity and can later be used for intention detection, we conducted a manual analysis of the market reports that allowed us to classify the predicates specific to each intention. Synonyms and antonyms have also been included to complete and enrich the dictionary (see Table 2 for some examples)

In a second stage, the grammar looks for syntactic combinations of these key elements. For example, a sentence containing exclusively a financial indicator, a state predicate, one measure and an optional time and/or geographic dimension will be extracted as an intention *DescribeValue*.

Sentences selected from financial corpus are then transformed into a set of triples (hereinafter referred to as **complete triples**), organized into subject-predicate-object structure, *&&* serving as a connector. See Table 11 in Appendix for a detailed overview.

2.2.2 Data construction

There is no theoretical limit to the maximum sequence a T5 can encode, the only constraint being the memory requirements. We did not work on this specific aspect as this is not the purpose of the experiment. We choose to work with a maximum input sequence of 400, trying to keep the experiments into a small memory consumption interval. Owing to the limited capability of our T5 model, triples that are too long cannot be fully processed by the model and therefore the generated sentences will be incomplete. In order to work around this problem, we trimmed the triples by replacing the elements with simpler ones and reducing the length of predicates, then creating simplified triples. In these simplified triples, financial indicators are replaced with their semantic class (predefined in our grammar). For example, *abuse tax* and *absolute organic operating costs* both belong to the class *expenses*, so they were replaced with the generic short form *expenses* in the simplified triple. Measures are replaced a simpler number ($ + two digits), all time dimensions are substituted by a preposition (if there is one in the initial dimension time) plus a year (*in 2019* , for example) and the expression *in America* replaces any term in the geography dimensions. We refer to this trimmed triples as **simple triples**.

We trained a model with simple triples to evaluate if our formalism was rich and accurate enough so the model could infer the data logic, and used the complete triples to train a model for inference.

According to the type and number of key components, target sentences are sorted into different intentions. Our grammar for now is able to recognize and annotate 8 intentions, *DescribeValue* and *DescribeVariation* being the core intentions, based on which we developed 6 others (see Table 1)

Two sets of experiments have been built for each of the data sets created from simple and complete triples. Each experiment is detailed in Section 5.

3 Data sets

Applying the triples generator on a 4159 raw corpus files, we have collected 20615 sentences annotated for both complete and simple triples. The frequency for each intention in each set is presented in Table 1.

We randomly sampled three different training and testing partitions in order to leverage the scarcity of our data. All measures provided below are aggregated means of these three partitions.

4 Models

We used the T5 sequence to sequence transformer from the transformers library by (Wolf et al. (2020)

Intention	Definition	# full data	# test data
DescribeValue	`Measure of an indicator`	4951	990
DescribeVariation	`Variation of an indicator`	7483	1492
DescribeValueWithContributor	`Measure of an indicator with contributing factors`	1294	250
DescribeVariationWithContributor	`Variation of an indicator with contributing factors`	304	61
MergeDescribeValue	`At least 2 DescribeValue`	5744	1149
MergeDescribeValueWithContributor	`At least 2 DescribeValue and one expression contributor`	74	15
MergeDescribeVariation	`At least 2 DescribeVariation`	729	146
MergeDescribeVariationWithContributor	`At least 2 DescribeVariation and one expression contributor`	36	7

Table 1: Complete list of Intentions

Infinitive	Semantics	Semantics +
`{grow}`	*describe object variation*	`{increase}`
`{record}`	*describe object state*	`{null}`
`{record an increase}`	*describe object variation*	`{increase}`
`{be higher than}`	*compare object*	`{above}`

Table 2: Examples of predicate dictionary entries

to run the experiments, using an Nvidia GeForce RTX 2070 GPU with 8 Go RAM.

We trained two models, one from complete triples, and another one from simple triples, for each of our 3 data partitions. A simple [1] and a complete [2] trained models are available for reproducibility on Hugging Face hub.

4.1 Training parameters

Our objective is to evaluate our data formalism and an associated sequence to sequence model capabilities, so we did not experiment much on fine-tuning. We used a standard set of training parameters for all models and trained for one epoch and batches of 6, using the Hugging Face transformers library and the AdaFactor optimization method, keeping all default parameters except for the following:

- learning rate lr=1e-3
- regularization constants eps=(1e-30, 1e-3)
- decay_rate=0.7

4.2 Generation parameters

At inference, we tried to limit hallucinations and omissions while maintaining a good level of richness on the structure and vocabulary of the generated sentences.

We use a mix of top_k _ and top_p sampling for generating. Top_k is a sampling scheme, in which the K most probable next tokens are filtered and the probability mass is redistributed among only those K next tokens. Top_p is also a sampling scheme, managing creativity of the model. It chooses from the smallest possible set of words whose cumulative probability exceeds the probability p. This way, the size of the set of words (a.k.a the number of words in the set) can dynamically increase and decrease according to the next word's probability distribution.

We chose the following process for selecting the most suitable top_p and top_k for our generator:

- select one representative sentence and its corresponding triples for every intention.
- prepare 10 top_p (from 0.12 to 1) and 10 top_k (from 10 to 100) and combine them in a pairwise fashion to get 100 (top_p, top_k) couples.
- generate 10 sentences from a single triple. Then measure the similarity between these 10 sentences with ROUGE [3] and collect the measure under different top_p and top_k couples. We considered this average ROUGE to measure the creativity of our model. The bigger it is (less variation in the 10 generated sentence), the less creative the model is.
- compare the 10 generated sentences with the initial sentence in order to collect the ROUGE measure under different top_p and top_k couples. The bigger it is, the more accurate our model is.

[1] https://huggingface.co/yseop/FNP_T5_D2T_simple

[2] https://huggingface.co/yseop/FNP_T5_D2T_complete

[3] https://github.com/pltrdy/rouge

The top_p and top_k selected for simple triples and complete triples are (0.72, 40) and (0.82, 90), respectively. The model gives the best performance with them, leading to results presented in section 6.

4.3 Evaluation metrics

During the experiment, we have noticed that both the length of elements in the triples and the model's familiarity with them can influence the quality of the generation. We have adopted 3 methods to assess the quality of our models.

- The generated sentences are compared with the initial sentences and the lexical similarity is measured with a BLEU score (Papineni et al. (2002)) [4], adapted so it considers bi-grams.
- The generated sentences are compared with the initial sentences and the semantic similarity is mesured with a BERT-SCORE (Zhang* et al. (2020).
- The generated sentences are reintroduced into the triples generator to obtain regenerated triples. The inspiration for regenerating the triples and evaluate them against the original ones comes from Veksler et al. (2019) 's work on how to assess a key level of information for NLG. The degrees of similarity between the regenerated triples and the original ones offers another point of view on the quality of generated sentences and assesses the credibility of the data logic initially chosen. We used both BLEU and BERT-SCORE to evaluate these similarities.We will refer to these measures as Triple BLEU and Triple BERT.

It is important to notice that the triples comparison results is fully automated and neither human evaluation nor inter-annotator agreement statistics have been performed. Since the triples production process biases the performance measure, and is used both at training and inference, we are in fact evaluating the capability of our model to preserve the "fixed-pointedness" of T5 with respect to our representation rather than the T5 natural language generation power.

5 Experiments

We defined two experiments, one training and evaluating for complete triples (see subsection 5.1), the other for simple triples (see subsection 5.2), and computed the four metrics previously detailed for each experiment. The measures provided are arithmetic means of the scores evaluated for all models created from our three different data partitions.

In the following subsections, we will refer to any element issued from the original corpus sentences as **original**. For each table of results, we present the actual number of triples that could be regenerated in regard to the number of sentences generated at inference available for testing.

5.1 Experiment 1

In this initial experiment, we used complete triples to fine-tune a T5 model. A data sample is available in Table 3, results are provided in Table 8. The BLEU score shows important variations in between intentions, due to the fact that some intentions are more complex and contain more elements than simpler ones like *DescribeValue*, and because they are less represented in the training data. Having around 4000 training examples seems to be a pre requisite to obtain significant improvement on the results.

5.2 Experiment 2

5.2.1 Simple

In this experiment we trained another model to learn and generate from simple triples.

The objective here is to workaround the limitations of our model in low memory consumption mode. The process for training a model for simple triples is the following:

- complete triples are simplified
- we simplify the original sentences by replacing the original elements by the simple ones (indexed by simple triples)

The model is trained with simple triples against these simplified sentences (see example provided in Table 4), then simplified original sentences used for training are compared with the sentences generated at inference (an evaluation sample is provided in Table 5).

5.2.2 Restored

To affect a metric to sentences generated at inference from simple triples models, we retain two additional features:

- in the sentences generated at inference, we restore the original elements using their index

[4] https://www.nltk.org/_modules/nltk/translate/bleu_score.html

in the original sentences, and compare these restored sentences with the original ones. An example of this transformation is provided in Table 6.

- we regenerate triples from these restored sentences, and compare them with the complete triples presented in section 5.1

We will refer to the triples and sentences in this experiment as **restored**. The results are provided in Table 10.

6 Results analysis

BLEU and BERT-SCORE leads to different conclusions and the BLEU score is generally lower than BERT-SCORE. This is because the 2 metrics evaluate the sentence at different levels.

6.1 Evaluating for triples

We expect sentences in financial report to contain all key information provided in the input data. However, BLEU and BERT-SCORE are incapable of examining the completeness of generated sentences. To achieve this goal, we passed the generated sentences to the triples generator and evaluate the regenerated triples with BLEU and BERT-SCORE. The higher the score is, the more complete the generated sentence is.

We were not able to regenerate any triple for a significant amount (22%) of test set sentences generated at inference, neither for complete triples nor for simple triples. The results take into account this information loss, when this happens the Triple BLEU and BERT-SCORE are evaluated to zero. This is partly due to our triples generator, partly to the structure of the sentence generated at inference time. Our triples generator is very dependant from the lexical layout of the sentence. In some cases, generated sentences which would be qualified for triple extractions are not recognized as such and ignored. On the other hand, and for the same reasons, the triple generator will also ignore ill-formed sentences.

The attribution to each case is still a work in progress. We provide examples of ignored generated sentences from which we could not regenerate any triple in Table 7.

This leads to important discrepancy in the Triple BLEU results, between different types of intentions and between complete versus simple triples experiments. Nevertheless we can still directly link the fidelity of the results to input data with the size of the training set.

Simple triples achieve significant better Triple BLEU score than complete triples, due to the fact that sentences generated from simple triples are shorter, and usually mirror the original simple triple sinformation much better than sentences generated from complete triples (often interrupted before a human readable sentence is fully generated at inference time, thus regenerating incomplete triples or none). For complete experiment, we were able to regenerate 85 % of the indicators present in the triples at inference and 95% of the indicators for the simple triples experiment.

6.2 Evaluating for sentences

BLEU evaluates the generated sentences on lexical level. The significant difference between complete and simple results comes mainly from the number of triples we were able to regenerate in each case, the simplest intentions for which a lot of training data was available being once again favored in both cases.

We can witness an improvement on average (from 0.423 average BLEU for sentences generated from complete triples at inference to 0.656 for sentences generated from simple triples), yet the similarity between restored sentences and original sentences (0.429 average BLEU) is only slightly higher than between original sentences and sentences generated from complete triples (0.423 average BLEU), mainly due to the risk of information loss during the process of restoring the original information in simplified sentences. .

Figure 1 shows that, as the complexity of intention increases, average BLEU score for simplified sentences generated from simple triples exceeds BLEU for complete sentences generated from complete triples and also restored generated sentences.

While the sentences are short (intention is less complex), a small lexical change (change of predicate for instance) is reflected in a big drop in BLEU score. For the same intention, the simplified generated sentence is usually the shortest. Therefore, under simpler intention, (e.g. *DescribeValue*), simplified sentences generated from simple triples obtained the lowest score. However, as the intention becomes more complex, the length of simplified sentences increases, which offsets the influence of lexical change in BLEU score. In addition, the BLEU score of simplified generated sentences re-

Triple	Generated Sentence	Regenerated Triple
Operating margin | valIs | 5.8% && 5.8% | comTo | 8.5%	Operating margin was 5.8% (versus 8.5%).	Operating margin | valIs | 5.8% && 5.8% | comTo | 8.5%

Table 3: Complete triples, generated sentence and regenerated complete triples example for original sentence *Operating margin was 5.8% compared to 8.5%.*

Original Triple	Generated Sentence	Regenerated Triple
Results | valIs | 10% && 10% | comTo | 11%	Results was 10% (-0.71) and remained at the same level compared with 11%	Results | valIs | 10% (-0.71) && 10% (-0.71) | comTo | 11%

Table 4: Simple triples, generated sentence and regenerated simple triples example for original sentence *Operating margin was 5.8% compared to 8.5%.*

Original Simplified sentence	Generated Sentence
Results was 10% compared to 11%.	Results was 10% (-0.71) and remained at the same level compared with 11%)

Table 5: Simplified original sentence and sentence generated from simple triples model at inference for *Operating margin was 5.8% compared to 8.5%.*

Restored sentence
Operating margin was 5.8% (-0.71) and remained at the same level compared with 8.5%

Table 6: Sentence generated from simple triples model at inference restored with original elements for *Operating margin was 5.8% compared to 8.5%.*

Original Sentence	Original Triple	Generated Sentence
Non-current liabilities were ¥309.0 billion, an increase of ¥2.4 billion or 0.8%, from the end of the previous fiscal year	Non-current liabilities | valIs | ¥309.0 billion && Non-current liabilities | incBy | ¥2.4 billion or 0.8%	Non-current liabilities were
The right-of-use asset and discounted lease liability related to discontinued operations are €398 million as at 1 January 2019.	Discontinued operations | infBy | the right-of-use asset && lease liability | valIs | €398 million && €398 million | dTime | as at 1 january 2019	The right-of-use asset for the right-of-use asset and the right-
For the nine months ended September 30, 2020, revenues were $421.7 million, up 8.9% or $34.4 million from $387.3 million in the same period in 2019.	Revenues | valIs | $421.7 million && $421.7 million | dTime | for the nine months ended September 30, 2020 && revenues | incBy | 8.9% or $34.4 million	Revenues for the nine months ended September 30, 2020 were $421.7 million, an increase of
Revenue for January-September period amounted to EUR 499.6 (400.5) million, an increase of 24.7%.	Revenue | valIs | eur 499.6 (400.5) && EUR 499.6 (400.5) | dTime | period && revenue | incBy | 24.7%	Revenue during the reporting period amounted to EUR 499.6 (400.5) million
Function costs were €13,266 million in 2018 (2017: € 12,790 million).	Function costs | valis | €13,266 million && €13,266 million | dTime | in 2018 && €13,266 million | comTo | € 12,790 million && € 12,790 million | dTime | 2017	Function costs amounted to €13,266 million in 2018 (2017: € 12,
Long-term Liabilities amount to EUR 5,479k (31 December 2016: EUR 6,866k).	Long-term liabilities | valIs | EUR 5,479k && EUR 5,479k | comTo | EUR 6,866k && EUR 6,866k | dTime | 31 december 2016	Long-term liabilities amount to EUR 5,479k (31 December 2016: EUR 6,866

Table 7: Non regenerated triples sample

Intention	# test	# nan triples	Triple BLEU	Triple BERT	Sentence BLEU	Sentence BERT
DescribeValue	990	163	**0.781**	**0.960**	**0.646**	**0.944**
DescribeVariation	1492	173	0.693	0.951	0.573	0.941
DescribeValueWithContributor	250	146	0.304	0.864	0.364	0.896
DescribeVariationWithContributor	61	14	0.443	0.908	0.325	0.911
MergeDescribeValue	1149	392	0.363	0.888	0.556	0.935
MergeDescribeValueWithContributor	15	6	0.264	0.868	0.346	0.904
MergeDescribeVariation	146	61	0.259	0.865	0.359	0.916
MergeDescribeVariationWithContributor	7	3	0.188	0.854	0.211	0.906
Mean	-	-	*0.412*	*0.895*	*0.423*	*0.919*

Table 8: BLEU and BERT-SCORE (F1) results by intention for complete triples model. *nan triples* stands for non regenerated triples

Intention	# test	# nan triples	Triple BLEU	Triple BERT	Sentence BLEU	Sentence BERT
DescribeValue	990	41	**0.941**	**0.990**	0.469	0.919
DescribeVariation	1492	55	0.914	0.985	0.590	**0.936**
DescribeValueWithContributor	250	93	0.499	0.899	0.407	0.889
DescribeVariationWithContributor	61	14	0.564	0.919	0.453	0.915
MergeDescribeValue	1149	110	0.819	0.964	**0.604**	0.931
MergeDescribeValueWithContributor	15	5	0.397	0.891	0.471	0.899
MergeDescribeVariation	146	27	0.656	0.934	0.491	0.919
MergeDescribeVariationWithContributor	7	2	0.459	0.890	0.405	0.899
Mean	-	-	*0.656*	*0.934*	*0.486*	*0.913*

Table 9: BLEU and BERT-SCORE (F1) results by intention for simple triples model. *nan triples* stands for non regenerated triples

Intention	Triples BLEU	Triple BERT	Sentence BLEU	Sentence BERT
DescribeValue	**0.823**	**0.971**	**0.554**	**0.936**
DescribeVariation	0.684	0.944	0.537	0.930
DescribeValueWithContributor	0.337	0.870	0.298	0.876
DescribeVariationWithContributor	0.351	0.884	0.323	0.897
MergeDescribeValue	0.653	0.936	0.588	0.932
MergeDescribeValueWithContributor	0.274	0.868	0.396	0.891
MergeDescribeVariation	0.494	0.903	0.434	0.911
MergeDescribeVariationWithContributor	0.257	0.859	0.299	0.891
Mean	*0.484*	*0.904*	*0.429*	*0.908*

Table 10: BLEU and BERT-SCORE (F1) results by intention for restored simple triples and sentences (# of test sentences and non regenerated triples is the same as in Table 9)

mains relatively steady compared to the other 2 types of generated sentences. This phenomenon tends to prove that simplification of initial sentences and initial triples does improve the performance. And another proof is that this method generates 1879 sentences with BLEU score in interval 0.98 to 1 for simple triples models, while the number of sentences generated from complete triples and restored triples models scoring within this interval is 1600 and 1783, respectively.

We evaluate with BERT-SCORE on semantic level. Taking BERT as the standard, there is little difference between the aggregated measures for sentences generated at inference from complete triples, simple triples or restored triples models. It's interesting to notice that restored simple sentences for well defined intentions such as *DescribeValue* exhibit a BERT-SCORE close to complete triples generated sentences (0.936 and 0.944 respectively), so this technique might be a way to workaround the memory constraints of the T5.

7 Error analysis

We have observed notable gaps between the BLEU and BERT-SCORE measures. We identified at least 4 reasons why this might occur:

1. Different verbs of same semantic meaning are employed in generated sentences:
    - **Original sentence**: The total gaming margin in online games during the quarter *amounted to* 4.7
    - **Generated sentence**: The total gaming margin in online games during the quarter *was* 4.7
2. The position of dimension time or dimension geography changes (slight influence):
    - **Original sentence**: Revenue *for 2016* amounted to 245 million.
    - **Generated sentence**: Revenue amounted to 245 million *for 2016*.
3. When the indicator in the triples starts with a lowercase, the model adds complement to it, which may be different from the complement in the original sentence. And sometimes, different complements may lead to different conjunctions of verb:
    - **Original sentence**: *The value of* deferred tax assets at 31 December 2016 *was* €190 million.
    - **Generated sentence**: *The total* deferred tax assets at 31 December 2016 *were* €190 million.
4. Predicates used in the triples don't indicate the tense of verbs. Hence, the tense of generated sentence may be different from the original one:
    - **Original sentence**: The annual savings in interest costs from this refinancing *amounts to* approximately US$29 million.
    - **Generated sentence**: The annual savings in interest costs from this refinancing *amounted to* approximately US$29 million.

The first three examples show that a lexical-based measure as BLEU is clearly not suitable to evaluate NLG systems.We tried to leverage this issue by evaluating triples against regenerated triples, this evaluation being less sensitive to semantic variations while retaining enough syntax for comparison.

8 Conclusion and future work

We have evaluated how a T5 sequence to sequence transformer behaves in data to text generation, using a combination of BLEU and BERT-SCORE on triples (with simple triples achieving the best Triple BLEU score of 0.656). The result gap between simple and complete triples experiments demonstrates that transforming initial sentences into simple ones and generating sentences from simple triples contributes to increasing the completeness of the generated sentences and the data logic accuracy.

Future work will focus on leveraging the induced bias of the triple generator as to propose more accurate automation of the triple extraction, and working on the current limitations of the T5 model to extend the length of input sequences keeping memory consumption as low as possible.

Acknowledgements

We thank the FNP 2021 Committee for the opportunity to publish this research experiment.

References

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG

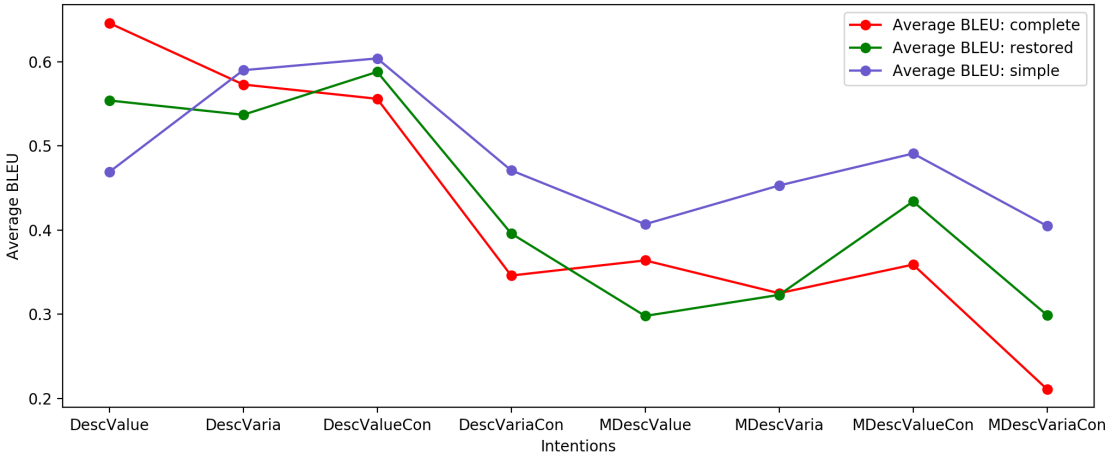


Figure 1: Average BLEU score on every intention for simplified generated sentence, complete generated sentences and restored generated sentences. From *DescribeValue* to *MergeDescribeVariationWithContributor*, the complexity of intention raises.

challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Khushnuma Grover, Katinder Kaur, Karti Tiwari, and Kumar P. Rupali. 2021. Deep learning based question generation using t5 transformer. In *International Advanced Computing Conference (IACC 2020)*, volume 1367 of *Communications in Computer and Information Science*, Singapore. Springer. Https://doi.org/10.1007/978-981-16-0401-0_18.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102, Dublin, Ireland. Association for Computational Linguistics.

Zdenek Kasner and Ondrej Dusek. 2020. Data-to-text generation with iterative text editing. In *Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020, Dublin, Ireland, December 15-18, 2020*, pages 60–67. Association for Computational Linguistics.

PETER KLUEGL, MARTIN TOEPFER, PHILIP-DANIEL BECK, GEORG FETTE, and FRANK PUPPE. 2016. Uima ruta: Rapid development of rule-based information extraction applications. *Natural Language Engineering*, 22(1):1–40.

Ziran Li, Zibo Lin, Ning Ding, Hai-Tao Zheng, and Ying Shen. 2020. Triple-to-text generation with an anchor-to-prototype framework. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3780–3786. International Joint Conferences on Artificial Intelligence Organization. Main track.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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.

Raphael Salmon. 2017. *Natural language generation using abstract categorial grammars*. Ph.D. thesis, Sorbonne Paris Cité.

Yael Veksler, Natalia Vanetik, and Marina Litvak. 2019. EASY-M: Evaluation system for multilingual summarizers. In *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources*, pages 53–62, Varna, Bulgaria. INCOMA Ltd.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Appendix

Predicates	General form	Example
valIs	Indicator | valIs | Measure	Net cash inflow was 9 067 million USD. Net cash inflow | valIs | 9 067 million USD
valIs	Indicator | valIs | Measure	Net cash inflow was 9 067 million USD. Net cash inflow | valIs | 9 067 million USD
chaBy	Indicator | chaBy | Measure	The Company recorded a change of €12. net cash inflow | chaBy | €12
decBy	Indicator | decBy | Measure	Net cash inflow decreased by 12%. Net cash inflow | decBy | 12%
decTo	Indicator | decBy | Measure	Net cash inflow decreased to €10 million. Net cash inflow | decTo | €10 million
dFrom	Indicator | dFrom | Measure	Net cash inflow decreased from €12 million. Net cash inflow | dFrom | €12 million
incBy	Indicator | incBy | Measure	Net cash inflow increased by 12%. Net cash inflow | incBy | 12%
incTo	Indicator | incTo | Measure	Net cash inflow increased to €10 million. Net cash inflow | incTo | €10 million
iFrom	Indicator | iFrom | Measure	Net cash inflow increased from €9 million. Net cash inflow | iFrom | €9 million
Contribute	Indicator | Contribute | value contributed	Cash and cash equivalent was €20 million, with net cash inflow of €12 million. net cash inflow | Contribute | €12 million
CauBy	Indicator in result | CauBy | reason	Due to higher costs in services, costs increased by US$ 4 million. costs | CauBy | higher costs in services
InfBy	Indicator | InfBy | related factor	Full-year capital expenditure amounted to €24.2 million, mainly relating to new finishing capacity Full-year capital expenditure | InfBy | new finishing capacity
ContriBy	Contributed | ContriBy | Contributor	Cash and cash equivalent was €20 million, with net cash inflow of €12 million. Cash and cash equivalent | contriBy | net cash inflow
dTime	Measure | dTime | Date	Cash and cash equivalent was €20 million in 2019. €20 million | dTime | in 2019
startDate	startValue | startDate | Date	The revenue increased from €20 million in 2019 to €23 million in 2020. €20 million | startDate | in 2019
endDate	endValue | endDate | Date	The revenue increased from €20 million in 2019 to €23 million in 2020. €23 million | endDate | in 2020
diGeo	Measure | diGeo | Dimension geography	The revenue increased by €20 million in Europe. €20 million | diGeo | in Europe
cTime	Measure | cTime | Date for comparison	The revenue increased by €20 million compare to the prior year. €20 million | cTime | the prior year
comTo	Measure | comTo | Measure for comparison	The revenue was €20 million (in 2019: €21 million) compare to the prior year. €20 million | comTo | €21 million

Table 11: Usage of predicates

Economic Causal-Chain Search and Economic Indicator Prediction using Textual Data

Kiyoshi Izumi and **Hitomi Sano** and **Hiroki Sakaji**
School of Engineering, the University of Tokyo
Hongo 3-8-1, Bunkyo, Tokyo 113-8656, JAPAN
izumi@sys.t.u-tokyo.ac.jp

Abstract

This paper proposes a method that uses causal information extracted from textual data to predict economic indicators. The method automatically extracts causal information included in each sentence using machine learning and natural language processing methods. The extracted cause-effect expressions are stored in an economic causality database. Then, the method can generate causal chains from the given text using the word similarity between a result expression and a cause expression in the database. The causal chains are used to predict how the numerical values of economic indicators will change in the future due to spillover effects from the given text.

1 Introduction

The causality drives economic phenomena. The people involved in the phenomenon predict the future and decide their actions based on their perception of causality. As a result of the accumulation of these actions, the behavior of the entire economic system is determined. For example, consider a causal series (causal chain) that starts with "Aging society. " The aging society has negative effects on the economy in terms of causing a decline in the labor force. It also has positive effects, causing the increasing demand for products for the elderly. Thus, to predict the future of economic phenomena, it is essential to analyze the perception of cause and effect that people have.

It is, however, difficult to statistically analyze the economic causality from numerical data alone because the economic causality involves human behavior. The key to causality is how humans perceive the causal event and their actions in response to it. The perception of economic causality can change over time. It is almost impossible to extract an objective and universal causal series by statistical analysis of numerical data, like natural science.

Therefore, in this study, we analyze textual data in an economic area that contain human-perceived causal relationships and construct a database of economic causality. We propose an algorithm that constructs causal sequences derived from phrases representing specific events and presents economic indicators related to spillover effects or potential causes. Using this method, we can search for spillover effects and potential causes based on causal information expressed in text and use the relationships between events and economic indicators to predict changes in numerical values.

The main contributions of this research are as follows:

- We developed a new method for integrating causality search using textual data (unstructured data) and economic indicators (structured data) prediction.
- This method enables us to predict the change of economic indicators by tracing a causal sequence from a text representing an event of interest.
- This method can give prediction results in an explainable form that is intuitively understandable by humans.

2 Related Works

In recent years, much research concerning causal information extraction from natural language is based on neural networks.

For example, Dasgupta et al.(Dasgupta et al., 2018) proposed a method for extracting causal information using Long short-term memory (LSTM) architecture.

Furthermore, concerning English causal information extraction, various methods were proposed at the Financial Narrative Processing Workshops (FNP), which is a workshop of Colling 2020 because it is included in the Shared Task FinCausal 2020 of the workshop.

Two types of tasks were set in the shared task, extraction of causal information sentences and extraction of causal-effect expressions from causal information sentences.

Most proposed methods for extracting causal information sentences are based on BERT consisting of Transformer and achieved high performance(Ionescu et al., 2020; Gordeev et al., 2020; Gupta, 2020).

Additionally, BERT based method has also been proposed in the extraction causal expressions task(Imoto and Ito, 2020).

Researches concerning the construction of causal chains, such as Ishii et al.(Ishii et al., 2012), Alashri et al.(Alashri et al., 2018), and Zhao et al.(Zhao et al., 2017) exist.

Ishii et al. proposed a method of constructing causal networks by extracting causal expressions from newspaper articles and combining them with SVO based on the hypernym-hyponym relation dictionary.

Alshri et al. proposed a concept-based causal chain construction method.

Zhao et al. proposed a method for constructing causal chains by a method that considers the cause-to-effect and effect-to-cause paths.

In addition, various applications of causality other than constructing causal chains are expected.

For example, in the world of robotics, Causal World(Ahmed et al., 2020) a new benchmark that considers causality has been proposed.

In the existing research mentioned above, it is limited to causal extraction and causal chain construction, and it is not clear what kind of event or concrete numerical value it is related to.

Therefore, several methods are required to use it for actual economic analysis.

On the other hand, this research can be linked to a numerical index, and the change of the numerical value can be predicted.

It is the novel point of this research.

3 Economic causality detection

First, we analyze textual data containing causal economic information recognized by humans and extract cause-effect expressions from them. In this system, we extracted causal relationships from the text of financial statements, which listed companies regularly publish to disclose their business performance and financial status, using a method that uses cue expressions(Sakaji et al., 2017).

Input: A list of cause–effect expressions CI
CI_i = (Cause Expression c_i, Effect Expression e_i, Company cp_i, Date d_i)
Output: A list of causal chain LCC

$LCC \leftarrow \emptyset$
for each $(c_i, e_i, cp_i, d_i) \in CI$ **do**
  for each $(c_j, e_j, cp_j, d_j) \in CI$ **do**
    $similarity \leftarrow getSimilarity(e_i, c_j)$
    if $similarity \geq threshold$ **then**
      $LCC \leftarrow LCC + (c_i, e_i, cp_i, d_i, c_j, e_j, cp_j, d_j)$
    end if
  end for
end for
return LCC

Figure 1: Economic causal-chain construction

- Textual data: approximately 20,000 financial summary texts issued by approximately 2,300 companies between October 2012 and May 2018.

- Extracted causal-effect expressions: 1,078,542 pairs.

The extracted causal-effect expressions are stored in the database along with the publishing date and the company name of the financial summary containing the causality information.

4 Economic causal-chain construction

To construct causal chains from the economic causality database, we connect an effect expression of a cause-effect expression and a cause expression of another cause-effect expression. We show an algorithm (Izumi and Sakaji, 2020) for constructing causal chains in Figure 1.

In Figure 2, "Company" indicates the company that issues the financial statement summary from which the cause-effect expression has been extracted. Additionally, "Date" is the date the financial statement summary was published. In Figure 1 $getSimilarity(e_i; c_i)$ is a function that calculates the similarity between the effect expression e_i and the cause expression c_j. Our method estimates the similarities based on vectors of word embedding. First, our method obtains the word embedding average of the words included in the expressions. Here, we define the average obtained from the effect expression e_i as $\tilde{W}_{e_i}$ and the average obtained from the cause expression c_j as $\tilde{W}_{c_i}.\tilde{W}_{e_i}, \tilde{W}_{c_i} \in R^m$ and m is the dimension size of word embedding. Then, our method calculates a cosine similarity between $\tilde{W}_{e_i}$ and $\tilde{W}_{c_i}$ and employs the similarity as the similarity between the effect expression ei and

the cause expression c_j. Finally, our approach acquires pairs of cause-effect expressions as a causal chain when the similarities are larger than a threshold.

5 Numerical economic forecasting using causal information

A causal chain is constructed using the algorithm in the previous section from the user's input phrase. Then the following method is used to estimate the numerical values that are expected to change as ripple effects and potential factors (Figure 2).

5.1 Forward search

In the case of forward causal-chain (ripple effect) search (Figure 2a), the procedure is as follows.

1. Construct a causal chain that represents the spillover effects from a text representing a particular phenomenon.

2. The text contained in the cause and effect expressions that appear in the causal chain becomes the expression related to the ripple effect from the first specific phenomenon.

3. When the text related to the spillover effect is given, the system presents the related numerical index from the text using the learning results using the combination of the numerical index and the related text set.

4. For the text data given by the user, the numerical indexes related to the result representation are presented as prediction results as economin indicators that are likely to change as a ripple effect.

5.2 Backward search

In the case of backward causal-chain (latent cause) search (Figure 2b), the procedure is as follows.

1. Construct a causal chain representing the potential causes from the text representing a particular phenomenon.

2. The text contained in the cause and effect expressions that appear in the causal chain becomes the expression related to the potential cause of the first specific phenomenon.

3. When the text related to the potential factor is given, the system presents the related numerical index based on the results of learning using the combination of the numerical index and the set of related texts done in advance.

4. For the text data given by the user, the numerical indexes related to the causal expressions are presented as prediction results as economic indicators that are likely to change as potential factors.

6 Experiments to predict economic indicators

In this section, we show the experimental results using proposed methods. In these experiments, we used the alternative data provided by the company "Deep Data Research" to predict economic indicators. This data consists of monthly corporate reports published on their official homepages. Therefore, this alternative data includes the economic indicators, the numeric values, and some economic texts describing the company's situation. We used almost 150,000 data with non-blank texts in these experiments out of 470,000 data (from Jan.2015 to Dec.2020).

6.1 Experimental Methods

1. Extracting causal information:

   Given arbitrary texts and periods extract causal information to causal-chain search.

2. Extracting the economic indicators:

   Economic indicators that have the causal information in the economic texts of alternative data are extracted, and each frequency is counted.

3. Calculating the relevance:

   The odds ratio between causal information and economic indicators is calculated. Furthermore, the economic indicators which have the highest odds ratio are detected as the results of prediction. The odds ratio (represented as "R") is shown as equation(1)-(3).

$$R = \frac{(P + 0.5) * (1 - Q + 0.5)}{(1 - P + 0.5) * (Q + 0.5)} \tag{1}$$

$$P = \frac{Pa}{Pb} \tag{2}$$

$$Q = \frac{Qa}{Qb} \tag{3}$$

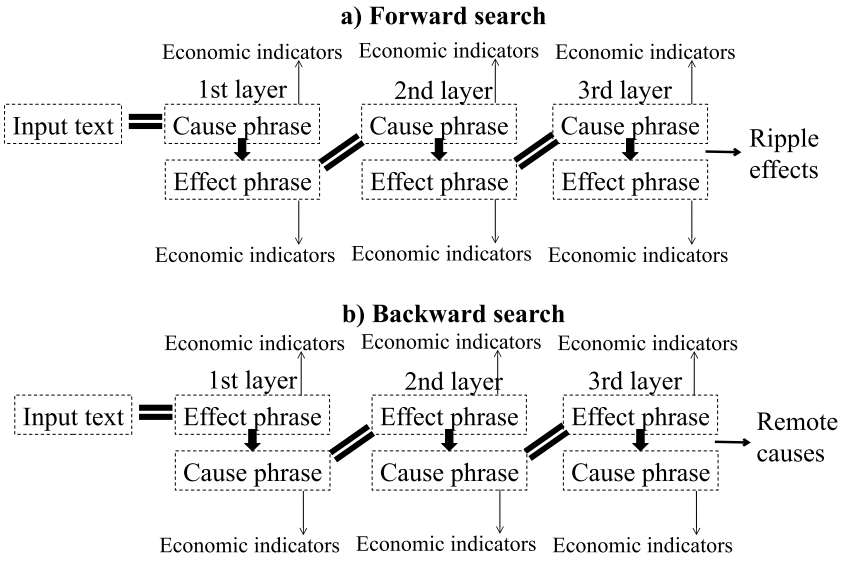


Figure 2: Economic causal-chain and related economic indicators.

Pa: The frequency of target indicator
Pb: The summary of the target indicator
Qa: The frequency of all indicators
Qb: The summary of all indicators

6.2 Experimental Results

We show the experimental results to predict economic indicators related to causal information in the following 4 cases. The First 3 cases give the texts ("Infectious disease," "US presidential election," and "Global warming") to causal-chain search. The following case gives a text "Olympics" and two divided periods to causal-chain search.

Prediction of spillover in the case of "Infectious diseases"

The text "Infectious disease" was given to the causal-chain search, and the causal information was extracted, repeating the spillover effects of multiple layers. Then, the economic indicators related to the causal information were predicted using our proposed methods. The causal information of the second layer in the causal-chain search and the top 5 economic indicators that are predicted to be strongly related to the causal information is shown in Figure 3. Furthermore, among these economic indicators, the numerical change of "room occupancy rate" is shown in Figure 4.

Regarding Figure 4, the transition of "Guest room occupancy rate" decreased sharply around May 2020, when the lockdown was announced in Japan due to the spread of COVID-19. Since the numerical values fluctuate more than usual, it is considered that the predicted related economic indicators are strongly related to the causal information that spillover from "infectious diseases."

The causal results by causal chain search

Cause (1): World affairs, Infectious diseases, Impacts
Result (1): Japanese, Departures

Cause (2) : Japanese, Guests, Decrease
Result (2) : Supply and demand, Guest room, Unit price

Related numerical indicators <Top 5 >

	Indicators	Relevance
1	Average rent per tsubo	8.093
2	NOI	8.093
3	ADR	6.641
4	RevPAR	6.636
5	Guest room occupancy rate	3.994

Figure 3: Prediction of spillover in the case of "Infectious diseases"

Prediction of spillover in the case of "US presidential election"

The text "US presidential election" was given to the causal-chain search, and the extracted causal information of the second layer in the causal-chain search was "Yen depreciation," "High stock prices," "Economy," and "Boom". The top 3 economic indicators that are predicted to be strongly related to the causal information are shown in Figure 5. Furthermore, the numerical change of "Amount of foreign exchange" among these economic indicators is shown in Figure 6.

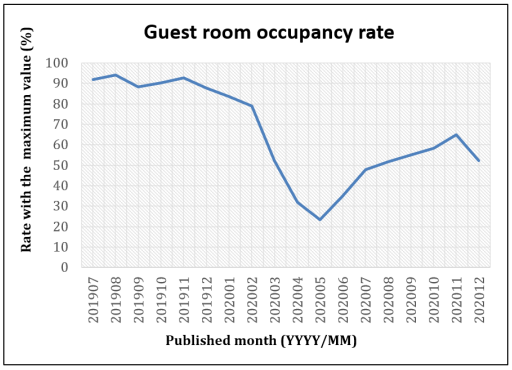


Figure 4: Changes in "Room occupancy rate"

Prediction of spillover in the case of " US presidential election

	Indicators	Relevance
1	Deposited assets	1.200
2	Amount of foreign exchange	1.069
3	Number of accounts	1.052

Figure 5: Prediction of spillover in the case of "US presidential election"

Regarding Figure 6, the amount of foreign exchange rose sharply during both presidential elections (November.2016 and November.2020). COVID-19 caused the rise in February.2020, but the related indicators that spilled over from the "US presidential election" changed at the time of the election.

Prediction of spillover in the case of "Global warming"

The text "Global warming" was given to the causal-chain search, and the extracted causal information of the first layer in the causal-chain search was "Greenhouse gas," "Emission," "Reduction," "Target." The top 3 economic indicators that are predicted to be strongly related to the causal information are shown in Figure 7. Furthermore, among these economic indicators, the numerical change of "CO2 reduction" is shown in Figure 8.

"Reduction of greenhouse gases," which is one of the causal information that spilled over from "Global warming," is in progress as a global goal. And the related numerical index "CO2 reduction" (Figure 8) is on an upward trend over the long term. Therefore, it is considered that the text "Global warming," causal information, and the economic indicators are strongly related.

Prediction of spillover in the case of "Olympics"

The text "Olympics" and two target periods were given to the causal-chain search, and economic indicators were predicted using the proposed method.

In the case where "January.2016-December.2016" held at the Rio de Janeiro Olympics was given as the target period, the extracted causal information was "Golf," "Industry," "Revitalization," and "Expectation." These results show the spillover effect of the new addition of golf as a new Olympic sport. The economic indicators predicted from the causal information were "Number of stores," "Number of customers," "Unit price per customer," and "Number of stores opened."

On the other hand, in the case where "January.2020-December.2020" held at the Tokyo Olympics was given as the target period, the extracted causal information was "Development project," "Construction period," "Review," and "Postponement." These results show the spillover effect that the new COVID-19 expanded worldwide during this period, and the Olympic Games were postponed for one year. The economic indicators predicted from the causal information were "Number of stores closed" and 'Occupancy rate."

In this way, the causal-chain search visualizes the chain of spillover with given texts and periods. Therefore, applying our proposed method makes

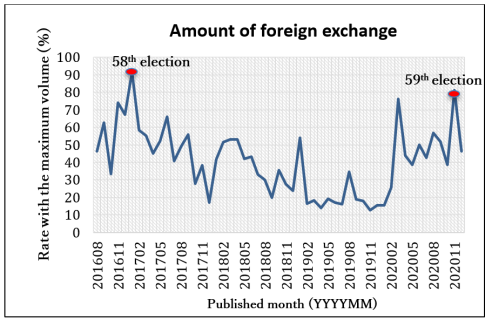


Figure 6: Changes in "Amount of foreign exchange"

Prediction of spillover in the case of "Global warming"

	Indicators	Relevance
1	Lumber purchase	1.200
2	CO2 reduction	1.069
3	Power generation	1.052

Figure 7: Prediction of spillover in the case of "Global warming"

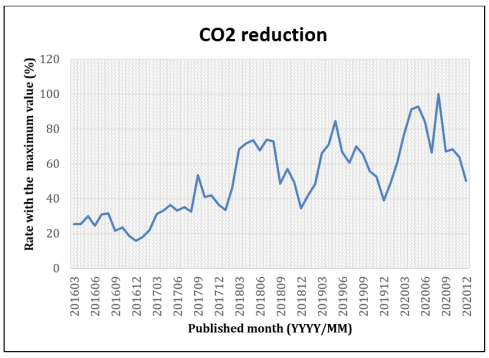


Figure 8: Changes in "CO2 reduction"

it possible to obtain related economic indicators from events that are usually difficult to see the connection and predict trends through causal-chain search.

7 Conclusion

This paper proposes a method that uses causal information extracted from economic texts to predict numerical indicators related to economic and financial fields, such as macroeconomic indicators and stock prices.

Using the proposed method, we identified numerical indicators that are expected to change due to spillover effects from three keywords: "infectious diseases," "U.S. presidential election," and "global warming." The hotel occupancy rate, a numerical indicator related to "infectious diseases," dropped sharply around April 2020, when a state of emergency was declared nationwide in response to the spread of the new coronavirus infection in Japan. The trading volume of foreign exchange markets, which is a numerical indicator related to the "U.S. presidential election," has risen substantially during the presidential election periods of November 2016 and November 2020. CO2 reduction, a numerical indicator related to "global warming," continues to rise. This numerical change indicates that measures to reduce greenhouse gases are continuing.

Furthermore, by giving a target period to the causal chains, more relevant indicators can be extracted. In the case of extracting the relevant indicators of the "Olympic Games," the number of customers and the number of stores opened were extracted related to golf in 2016, and occupancy rate and the number of stores closed were extracted related to COVID-19 in 2020. In this way, the causal chain can extract numerical indicators that are highly relevant to given text data and periods.

In future work, we will add a method for polarity analysis of texts that appears in the middle of a causal series. This method allows us to predict whether the spillover effects and potential factors estimated from the causal series will impact the relevant numerical indicators in the direction of increasing or decreasing changes.

References

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. 2020. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning.

Saud Alashri, Jiun-Yi Tsai, Anvesh Reddy Koppela, and Hasan Davulcu. 2018. Snowball: Extracting causal chains from climate change text corpora. In *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, pages 234–241.

Tirthankar Dasgupta, Rupsa Saha, Lipika Dey, and Abir Naskar. 2018. Automatic extraction of causal relations from text using linguistically informed deep neural networks. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 306–316.

Denis Gordeev, Adis Davletov, Alexey Rey, and Nikolay Arefiev. 2020. Liori at the fincausal 2020 shared task. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 45–49.

Sarthak Gupta. 2020. Finlp at fincausal 2020 task 1: Mixture of berts for causal sentence identification in financial texts. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 74–79.

Toshiya Imoto and Tomoki Ito. 2020. Jdd fincausal 2020, task 2: Financial document causality detection. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 50–54.

Marius Ionescu, Andrei-Marius Avram, George-Andrei Dima, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020. Upb at fincausal-2020, tasks 1 & 2: Causality analysis in financial documents using pretrained language models. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 55–59.

Hiroshi Ishii, Qiang Ma, and Masatoshi Yoshikawa. 2012. Incremental construction of causal network from news articles. *Journal of Information Processing*, 20(1):207–215.

Kiyoshi Izumi and Hiroki Sakaji. 2020. Economic causal-chain search using text mining technology.

In *Artificial Intelligence. IJCAI 2019 International Workshops*, pages 23–35. Springer International Publishing.

H. Sakaji, R. Murono, H. Sakai, J. Bennett, and K Izumi. 2017. Discovery of rare causal knowledge from financial statement summaries. In *The 2017 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)*, pages 602–608.

Sendong Zhao, Quan Wang, Sean Massung, Bing Qin, Ting Liu, Bin Wang, and ChengXiang Zhai. 2017. Constructing and embedding abstract event causality networks from text snippets. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, page 335–344. Association for Computing Machinery.

Preliminary experimentation with combinations and extensions of forward-looking sentence detection wordlists

Jan Štihec
University of Ljubljana
Ljubljana, Slovenia
stihec.jan@gmail.com

Senja Pollak
Jožef Stefan Institute
Ljubljana, Slovenia
senja.pollak@ijs.si

Martin Žnidaršič
Jožef Stefan Institute
Ljubljana, Slovenia
martin.znidarsic@ijs.si

Abstract

Forward-looking sentences are often a subject of studies of financial texts. Detection of such sentences is usually performed with wordlists of inclusive and exclusive keywords that are used as indicators of the forward-looking nature of the sentences at hand. In this paper we describe our assessment of potential improvements of forward-looking sentence detection wordlists by combining them together and by extending them with neighboring words in word-vector representations. Our current results indicate that simple combinations and straightforward extensions of wordlists with vector-space representation neighbors might not be suitable for FLS detection without further methodological improvements.

1 Introduction

Many studies of financial texts focus specifically on the contents of the forward-looking sentences (FLS). Detection of such sentences is then either a part of the methodological approach of a study or even one of the main aims of research.

Approaches to detection of these sentences usually employ lists of keywords, which are used as indicators whether a given sentence tends to be forward-looking or not. Keyword lists usually consist of: words that imply the future (e.g. "future"), future years numbers, conjugations of verbs that imply the future (e.g. "we intend") and combinations of certain adjectives and time indicators (e.g. "next year"). Some approaches also use lists of exclusive words, which are used to exclude a sentence that contains them from the forward-looking sentences identification process. Exclusive keywords are not always correlated with a nature of the sentence not being forward-looking, but might only indicate that a sentence containing them should not be analyzed in a specific study. It might for example contain keywords that are indicative for the parts of text, which aren't relevant for the study at hand.

The aim of our work is to study various wordlists for forward-looking sentence detection that appear in the literature, assess their combined use and experiment with wordlist extensions that are based on vector representation distances (similar to related work in terminology extraction, see e.g. (Pollak et al., 2019; Vintar et al., 2020)).

In this paper we report preliminary results of four wordlists, their combination and one wordlist's vector-space based extension on two manually labeled datasets. The current results indicate that the addition of exclusive wordlists might not always improve the results, which stands also for merging of the wordlists. The extension of the wordlists with word vector neighbors increases the amount of detected forward-looking-sentences, but it also increases the amount of sentences that are wrongly classified as FLS. With the current approach, this issue could not be alleviated with a similar extension of the corresponding exclusive wordlist.

2 Related work

Future-oriented information is recognized as very relevant to investors and is the subject of varius studies (Mio et al., 2020). Some studies rely on manual collection and analysis of FLS, while others employ automatic procedures that are mostly based on a number of widely used FLS wordlists. Each of these two approaches has its benefits and drawbacks, but we are interested in the latter one and the impact of the wordlists that are used for such purposes.

We identified four wordlists which are proposed in works that are commonly cited with regards to FLS identicifation and provide complete wordlists. Chronologically ordered the first one is the work by Li (2010), which is focused on information content and tone of FLS sentences. Next is the one by Athanasakou and Hussainey (2014) which is aimed at the assessment of the frequency of such statements and its relations with financial indicators.

The work of Muslu et al. (2015) studies the relation among FLS quantity and the firms' information environments. It suggests also use of word combination patterns, so the FLS wordlist that corresponds to this approach is relatively extensive. Tao et al. (2018) study the relationships among FLS features and IPO valuation. They use a wordlist based FLS detection approach (similar to the one by Muslu et al. (2015), but with additional consideration of sentence structure) in the stage of data preparation for machine-learning of a neural network based FLS classifier. All the listed studies provide a list of FLS inclusive keywords and all with the exception of Athanasakou and Hussainey (2014) also provide a list of FLS exclusive keywords.

3 Methodology

The approach that we used for the study described in this paper consists of: (I) selection of relevant wordlist-based approaches to FLS detection, (II) preparation of the data for testing and learning, and (III) design and running of the experiments.

We selected wordlists from four works (Li, 2010; Athanasakou and Hussainey, 2014; Muslu et al., 2015; Tao et al., 2018), which are often cited with regards to FLS detection and also provide the wordlists and detailed explanation of their FLS detection processes. We denote these wordlists as wl-Li, wl-At, wl-Mu and wl-Ta respectively. The data that was used for assessments and for learning the vector representations (also referred to as embeddings) in our experiments is described in detail in Section 3.1. For efficient experimentation with the selected wordlists we implemented a general wordlist-based labeling tool in python. Section 3.2 is dedicated to description of the methodological details of experiments.

3.1 Data

For the assessments of FLS detection approaches we used the sentences that were selected at random from recent (since 2017) annual reports of randomly selected FTSE 350 index constituents and were annotated as forward-looking/non-forward-looking by two human annotators. As the data was annotated by two annotators who worked on separate (not overlapping) groups of sentences, we treat this data as two datasets of 467 and 459 annotated sentences respectively and we denote them as D_1 and D_2. There are 260 FLS and 207 non-FLS sentences in D_1, while D_2 contains 122 FLS and 337 non-FLS sentences.

Table 1: Size of the used wordlists in terms of the amount of keywords.

	inclusive	exclusive
wl-Li	17	31
wl-At	45	/
wl-Mu	332	6
wl-Ta	373	6

Data was necessary also in the approach for extending wordlists, where it was used for learning vector space representations of words. Annotations are not needed for this purpose, but the data should be from the same domain as the task in which the vector representations are to be employed. We used a corpus of 604 periodic (10-Q and 10-K) reports. Specifically, it consisted of the 2018 Q4 reports from the Stage One 10-X Parse Data collection (from file `10-X_C_2016-2018.zip`) of the well known Notre Dame Software Repository for Accounting and Finance that was established by Loughran and McDonald (2016).

3.2 Experimental setup

In our experiments we used each individual selected wordlist and a merged wordlist that is denoted as wl-all and contains a set of all the words appearing in any of the wordlists. The wordlists were used for labelling the sentences as FLS or non-FLS. The results were calcualted separately for each of the two datasets.

With the exception of the approach by Athanasakou and Hussainey (2014), all the selected approaches provide an inclusive and an exclusive wordlist. First, we used only inclusive wordlists with a straighforward classification approach: the sentences that contained any word from a given inclusive list were classified as FLS. In the next series of experiments we used also all the corresponding exclusive wordlists in the sense that any sentence classified as FLS was re-classified into non-FLS, if it contained any word from the given list of exclusive words.

Note that our use of the wordlists is not completely comparable with most of the related works, from which the wordlists originate, as they were focused on specific sections of financial reports and some of the FLS detection approaches additionally considered numeric indications of future years or, in case of the approach by Tao et al. (2018), the

Table 2: Accuracy (acc) and recall (rec) of FLS classification with inclusive wordlists only.

	acc D_1	rec D_1	acc D_2	rec D_2
wl-Li	0.67	0.60	0.68	0.70
wl-At	0.68	0.66	0.62	0.74
wl-Mu	0.64	0.45	0.71	0.59
wl-Ta	0.64	0.45	0.71	0.59
wl-all	0.71	0.78	0.60	0.87

use of wordlists represented only a part of the FLS detection approach.

The last series of experiments, assessment of the effect of embeddings-based extensions of wordlists, was done only with one original wordlist - the one proposed by Li (2010). Again, both only the inclusive and the inclusive/exclusive options were experimented with. The word vector representations were learned with the fastText approach (Bojanowski et al., 2016) in the ClowdFlows3[1] prototype online tool for data analysis (parameters for learning the fastText model and neighbors selection: *vector size*=20, *context window size*=5, *minimal word occurences*=5, *distance threshold*=0.9). For each of the words in the original wordlist, the original word and five of the neighboring words from the vector space were included in the extended wordlist. The word neighbors were post-processed as follows: (I) any punctuation character at the start or the end of the word was removed, (II) any words that are considered English stop-words by the NLTK language toolkit[2] were removed.

The exclusive wordlist from Li (2010) includes also some bi-grams that are combinations of words: 'expected','anticipated','forecasted','projected','believed' that are preceded with each of the following auxiliary verbs: 'was' , 'were', 'had' and 'had been'. To obtain the corresponding embedding-based neighbors of these terms, we first calculated the neighbors of the words without the auxiliary verbs and then added all the combinations with auxiliary verbs to all the resulting word neighbors.

The resulting extended wordlists are provided in Appendix A.

4 Results and findings

Results of the assessment for inclusive wordlists are presented in Table 2 in terms of accuracy and recall of the FLS class. The recall might be more of interest if the aim of FLS detection is to analyse FLS contents or pre-filtering. For estimation of the amount of FLS the more relevant measure is accuracy, but it needs to be considered carefully in case of unbalanced datasets such as D_1 and D_2. From Table 2 we can see that on D_1 the best individual wordlist results are obtained with wl-At and that the merged wordlist yields better results as any of the individual approaches in terms of both performance measures. This is not the case on D_2, which has more non-FLS sentences. On D_2 these two approaches are better in terms of recall, but worse than others in terms of accuracy.

Addition of excluding wordlists into consideration slightly reduced all the recalls, with profound effect mostly in case of the merged wordlist. In such a setting, the combined wordlist did not outperform individual ones on any of the two datasets as it for example performs worse than wl-Li with respect to both measures on both datasets.

What we can draw from the first two experiments is that a combination of individual wordlists is not necessarily beneficial, particularly not for the case of considering also exclusive keywords.

Experimental assessment of the embeddings-based extensions of a wordlist are presented in Table 4. The extended inclusive wordlist improves recall, but in D_2 at the expense of accuracy. In comparison with the wordlist extension approach of merging the wordlist with other proposed ones, the

Table 3: Accuracy (acc) and recall (rec) of FLS classification with inclusive and exclusive wordlists.

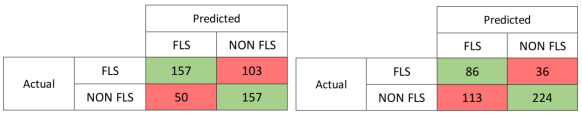
	acc D_1	rec D_1	acc D_2	rec D_2
wl-Li	0.67	0.60	0.68	0.70
wl-At	0.68	0.66	0.62	0.74
wl-Mu	0.63	0.41	0.71	0.51
wl-Ta	0.63	0.40	0.71	0.51
wl-all	0.65	0.58	0.63	0.65

Table 4: Accuracy of FLS detection with embeddings-based extensions of the wordlists by Li (2010). Use of extension is denoted by e(), in stands for the use of the inclusive and ex for the use of the exclusive list.

	acc D_1	rec D_1	acc D_2	rec D_2
in	0.67	0.60	0.68	0.70
e(in)	0.68	0.69	0.59	0.78
e(in) ex	0.68	0.69	0.59	0.78
e(in) e(ex)	0.63	0.59	0.60	0.64

[1]ClowdFlows3 homepage: https://cf3.ijs.si/ The used workflow is available at: https://cf3.ijs.si/workflow/223

[2]https://www.nltk.org/

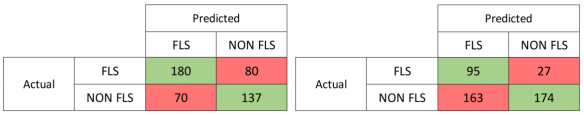


		Predicted	
		FLS	NON FLS
Actual	FLS	157	103
	NON FLS	50	157

		Predicted	
		FLS	NON FLS
Actual	FLS	86	36
	NON FLS	113	224

Figure 1: Contingency matrix for original inclusive wl-Li on D_1 (left) and D_2 (right).

extension with embeddings performs worse than the merged wordlist for both measures on both datasets.



		Predicted	
		FLS	NON FLS
Actual	FLS	180	80
	NON FLS	70	137

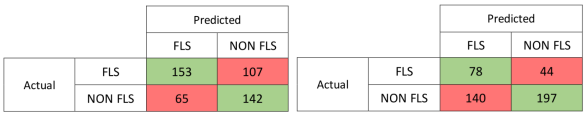
		Predicted	
		FLS	NON FLS
Actual	FLS	95	27
	NON FLS	163	174

Figure 2: Contingency matrix for extended inclusive wl-Li on D_1 (left) and D_2 (right).

Consideration of exclusive keywords was expected to compensate for some of the accuracy lost on D_2 due to potentially too wide reach of the inclusive keyword extensions, but consideration of the original exclusive keywords did not have an effect on results (a single sentence was classified differently in D_1), while use of an embedding-extended exclusive wordlist caused more non-FLS sentences to be correctly classified and vice-versa for the FLS (for details see Figures 1 to 3). This caused slight changes in accuracy in line with the class distributions of the two datasets. Most importantly, overall the approach with both the extended inclusive and extended exclusive wordlist in all aspects performed worse than the approach with original state of these two wordlists (for comparison see Table 3).

Our study is preliminary and we intend to conduct more experiments on larger datasets, but the current results indicate that straightforward extensions of wordlists with vector-space representation neighbors might not be suitable for FLS detection. In most experimental settings this holds also for extensions of wordlists by merging them together, although by a lesser extent.

This does not mean that such approaches cannot improve FLS detection, but it indicates that it might be necessary to go beyond a simple automated word vector neighbor extension and that such methodological improvements would be sensible already before further experimentation.



		Predicted	
		FLS	NON FLS
Actual	FLS	153	107
	NON FLS	65	142

		Predicted	
		FLS	NON FLS
Actual	FLS	78	44
	NON FLS	140	197

Figure 3: Contingency matrix for extended inclusive and extended exclusive wl-Li on D_1 (left) and D_2 (right).

Acknowledgements

This paper is supported by the project Quantitative and qualitative analysis of the unregulated corporate financial reporting (No. J5-2554), which was financially supported by the Slovenian Research Agency. The paper was supported also by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 825153, project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media). The authors acknowledge also the financial support from the Slovenian Research Agency for research core funding for the programme Knowledge Technologies (No. P2-0103). For access to the dataset of labeled forward looking sentences we thank the Faculty of Economics of the University of Ljubljana.

References

Vasiliki Athanasakou and Khaled Hussainey. 2014. The perceived credibility of forward-looking performance disclosures. *Accounting and business research*, 44(3):227–259.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Feng Li. 2010. The information content of forward-looking statements in corporate filings—a naïve bayesian machine learning approach. *Journal of Accounting Research*, 48(5):1049–1102.

Tim Loughran and Bill McDonald. 2016. Textual analysis in accounting and finance: A survey. *Journal of Accounting Research*, 54(4):1187–1230.

Chiara Mio, Pier Luigi Marchini, and Alice Medioli. 2020. Forward-looking information in integrated reports: Insights from "best in class". *Corporate Social Responsibility and Environmental Management*, 27(5):2212–2224.

Volkan Muslu, Suresh Radhakrishnan, KR Subramanyam, and Dongkuk Lim. 2015. Forward-looking md&a disclosures and the information environment. *Management Science*, 61(5):931–948.

Senja Pollak, Andraž Repar, Matej Martinc, and Podpečan Vid. 2019. Karst exploration: extracting terms and definitions from karst domain corpus. In *Proceedings of eLex19*, pages 934–956, Sintra, Portugal.

Jie Tao, Amit V Deokar, and Ashutosh Deshmukh. 2018. Analysing forward-looking statements in initial public offering prospectuses: a text analytics approach. *Journal of Business Analytics*, 1(1):54–70.

Špela Vintar, Larisa Grčić Simeunović, Matej Martinc, Senja Pollak, and Uroš Stepišnik. 2020. Mining semantic relations from comparable corpora through intersections of word embeddings. In *Proceedings of the 13th Workshop on Building and Using Comparable Corpora*, pages 29–34, Marseille, France. European Language Resources Association.

A Embedding-based extensions

Extension of the inclusive part of wl-Li. The words from the original wordlist are in bold, followed by up to five extensions (less, if removed as stop-words or duplicates with extensions of preceeding original words):
will accordingly furthermore **should** relied regarded ultimate context **can** frequently unreliable predicate producibility problem **could** harm harmed adverse **may** even substantial us **might** materialize pursued occur difficult **expect** expand effectively continue expansion **anticipate** profitable **believe** proactively history believes regularly **plan** plans sponsors sponsor **hope** hopes success perspectives identify teamwork **intend** intends **seek** seeking stop decide **project** progress feasibility projects predevelopment **forecast** quarter-to-quarter profitability forecasting forecasts **objective** objectively objectivity maximize **goal** toward targeting driving striving excellence

Extension of the exclusive part of wl-Li. The words from the original wordlist are in bold, followed by up to five extensions (less, if removed as stop-words or duplicates with extensions of preceeding original words):
undersigned, undersigned's, duly, thereunto, countersigned, **herein**, reference, referenced, **hereinafter**, hereinabove, mean, indicated, **hereof**, TAA, **hereon**, henceforth, **hereto**, confirms, **theretofore**, grantor, asserted, party, deemed, obligated, **therein**, documents, **thereof**, therefor, **thereon**, **expected**, differences, future, reversals, different, **was expected**, was differences, was future, was reversals, was different, **were expected**, were differences, were future, were reversals, were different, **had expected**, had differences, had future, had reversals, had different, **had been expected**, had been differences, had been future, had been reversals, had been different, **anticipated**, negative, forecast, unanticipated, results, **was anticipated**, was negative, was forecast, was Unanticipated, was results, **were anticipated**, were negative, were forecast, were Unanticipated, were results, **had anticipated**, had negative, had forecast, had Unanticipated, had results, **had been anticipated**, had been negative, had been forecast, had been Unanticipated, had been results, **forecasted**, magnified, imbalance, movements, variability, fluctuation, **was forecasted**, was magnified, was imbalance, was movements, was variability, was fluctuation, **were forecasted**, were magnified, were imbalance, were movements, were variability, were fluctuation, **had forecasted**, had magnified, had imbalance, had movements, had variability, had fluctuation, **had been forecasted**, had been magnified, had been imbalance, had been movements, had been variability, had been fluctuation, **projected**, projecting, **was projected**, was projecting, **were projected**, were projecting, **had projected**, had projecting, **had been projected**, had been projecting, **believed**, likelihood, verified, mistaken, livelihood, **was believed**, was likelihood, was verified, was mistaken, was livelihood, **were believed**, were likelihood, were verified, were mistaken, were livelihood, **had believed**, had likelihood, had verified, had mistaken, had livelihood, **had been believed**, had been likelihood, had been verified, had been mistaken, had been livelihood, **shall**, hereunder,

Summarization of financial reports with AMUSE

Natalia Vanetik
Shamoon College of
Engineering (SCE)
Beer-Sheva
Israel
natalyav@sce.ac.il

Marina Litvak
Shamoon College of
Engineering (SCE)
Beer-Sheva
Israel
marinal@ac.sce.ac.il

Abstract

This paper reports an approach for summarizing financial texts that combine genetic algorithms and neural document modeling. We treat summarization as the task of binary classification of sentences. Financial reports in the shared data of the FNS workshop are very long, have many sections, and are written in "financial" language using various special terms, numerical data, and tables. Our approach follows two main stages: (1) filtering the most irrelevant information with help of a supervised state-of-the-art summarizer and (2) extracting the most relevant sentences from the selected sentences in stage (1), using a novel deep neural model. As all participants of the Financial Narrative Summarization (FNS 2021) shared task, we used FNS 2021 dataset for training and evaluation.

1 Introduction

There is a growing interest in the application of automatic and computer-aided approaches for extracting, summarizing, and analyzing both qualitative and quantitative financial data, as a series of FNP and related workshops (El-Haj et al., 2018; El-Haj, 2019; El-Haj et al., 2020b) recently demonstrates. However, before these workshops, only a few attempts were made to summarize financial reports (Isonuma et al., 2017), which are different from the news articles in at least four parameters: length, structure, format, and lexicon.

The 1st Joint Workshop on financial Narrative Processing and MultiLing financial Summarisation (FNP-FNS 2020) (El-Haj et al., 2020a) ran the financial narrative summarisation (FNS) task, which resulted in the first large-scale experimental results and state-of-the-art summarization methods applied to financial data. The task focused on annual reports produced by UK firms listed on the London Stock Exchange (LSE). Because companies usually produce glossy brochures with a much looser structure, this makes automatic summarization of such reports a challenging task. A total number of 9 teams participated in the FNS 2020 shared task with a total of 24 system submissions.

The participating systems used a variety of techniques and methods ranging from rule based extraction methods (Litvak et al., 2020; Vhatkar et al., 2020; Arora and Radhakrishnan, 2020; Azzi and Kang, 2020) to traditional machine learning methods (Suarez et al., 2020; Vhatkar et al., 2020; Arora and Radhakrishnan, 2020) and high performing deep learning models (Agarwal et al., 2020; Singh, 2020; La Quatra and Cagliero, 2020; Vhatkar et al., 2020; Arora and Radhakrishnan, 2020; Azzi and Kang, 2020; Zheng et al., 2020). The text representation was also very diverse among the participating systems—very basic morphological and structure features (Li et al., 2020; Suarez et al., 2020), syntactic features (Vhatkar et al., 2020), and semantic vectors using word embeddings (Agarwal et al., 2020; Suarez et al., 2020) were applied. In addition, some teams (Litvak et al., 2020; Zheng et al., 2020) investigated the hierarchical structure of a report. Different ranking techniques, such as Determinantal Point Processes sampling (Li et al., 2020), a combination of Pointer Network and T-5 (Test-to-text transfer Transformer) algorithms (Singh, 2020) were applied for extractive approaches. Deep NN language models (La Quatra and Cagliero, 2020; Zheng et al., 2020), hierarchical summarization under different discourse topics (Litvak et al., 2020), and an ensemble-based models (Arora and Radhakrishnan, 2020) have also been reported.

One of the main challenges and limitations reported by the participants was the average length of annual reports (around 60,000 words), which made the training process extremely inefficient. In addition, participants argued that extracting text and then structure from PDF files with numerous tables, charts, and numerical data resulted in a lot of noise. These limitations open up an interesting

research problem that is worth investigating.

This paper reports an approach for extractive summarization of financial reports. Our approach utilizes MUSE (Litvak et al., 2010) as a filtering tool for the most irrelevant content. Then, we extract the most important sentences, using a novel combination of BERT vectors, neural node embeddings, and LSTM neural network, from MUSE's selections.

2 The method

We treat the task of extractive summarization as a binary sentence classification task. We aim to generate sentence representations, train an LSTM neural model on the training data, and predict sentence labels for every sentence in the test data. The main steps of our method are: (1) to produce large (3,000) summaries with MUSE algorithm (Litvak et al., 2010) to drastically reduce the amount of text to process; (2) to parse the summaries, extract syntactic, sentiment, and embedding data for every sentence (details in Section 2.2); (3) for every type of sentence data to construct a similarity graph and compute node embeddings for nodes representing sentences (see Section 2.3); (4) to concatenate BERT embeddings of sentences with all the node embeddings to obtain final sentence representation; and (5) finally, to train an LSTM neural model for a binary sentence classification task on a training set using the generated sentence representation; a sentence label is set to 1 if the sentence is contained in one of the gold summaries, and is set to 0 otherwise. This pipeline is illustrated in Figure 1.

2.1 MUSE algorithm

MUSE (MUltilingual Sentence Extractor) (Litvak et al., 2010) is an approach to multilingual single-document extractive summarization where summarization is considered as an optimization problem. MUSE uses a genetic algorithm, trained on a collection of document summaries, to find an optimal weighted linear combination of 31 statistical sentence scoring metrics. Because most sentence scoring methods have linear computational complexity, the inference phase of MUSE is very fast.

We used MUSE that was trained on 30 randomly selected gold standard summaries provided with FNS-2020 dataset (El-Haj et al., 2020b) and applied it to the training, validation, and test datasets with a word limit of 3,000 (we have used the MUSEEC tool (Litvak et al., 2016)). The reason for this selection is two-fold: (1) the documents are very long and parsing them as is would prevent us from creating the neural model in a reasonable time, and (2) the MUSE algorithm is very fast and it has demonstrated an excellent capability to find content that appears in gold summaries, as previous reports of its ROUGE scores demonstrate.

2.2 Preprocessing and data generation

Our preprocessing is performed with spacy (Honnibal and Johnson, 2015) and includes, as its first step, sentence splitting and tokenization. We use *en_core_web_sm* Spacy model for the parsing and *en_core_web_trf* for BERT sentence embedding extraction. We eliminate empty sentences and very short (2 words or less) sentences but do not perform any additional data cleaning. The following information is generated for every sentence in a document: (1) a BERT sentence vector; (2) a multi-set of basic part-of-speech (POS) tags in a sentence (*token.pos_*); (3) a multi-set of detailed POS tags in a sentence (*token.tag_*); (4) a multi-set of syntactic dependencies in a sentence; (5) a multi-set of lemmatized tokens in a sentence; (6) a sentiment data for a sentence that includes sentence polarity and subjectivity as numeric values; and (7) a multi-set of named entities tokens (NER) in a sentence.

2.3 Document graph and node embeddings

Once we have the data for all sentences in a document, we generate separate complete edge-weighted undirected document graphs for every data type. In all of these graphs, every sentence is a separate node. The data associated with a pair of nodes X, Y and Y is used to compute the weight of the edge (X, Y) – for BERT sentence vectors and sentiment data, the $weight(X, Y) = cosine\_similarity(X, Y)$; it is set to $jaccard\_similarity(X, Y)$ otherwise. After all the edge weights have been generated, we prune the graphs by deleting all edges with weights less than or equal to the median edge weight in the graph. The pipeline of a graph construction is depicted in Figure 2. Node2Vec algorithm (Grover and Leskovec, 2016) generates a representation of the graph and its nodes as real vectors reflecting the network neighborhoods of nodes.

2.4 Sentence labels and summary generation

For every sentence, we concatenated all of its node embeddings together with the original BERT sentence vector, and broken them into 'chunks' of

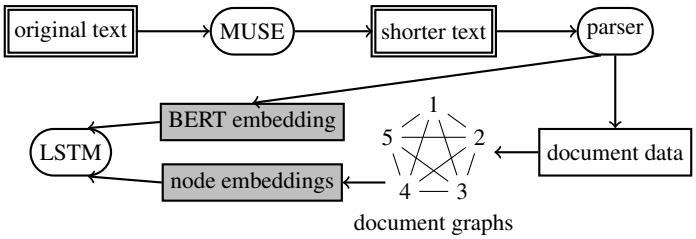


Figure 1: Pipeline of our approach

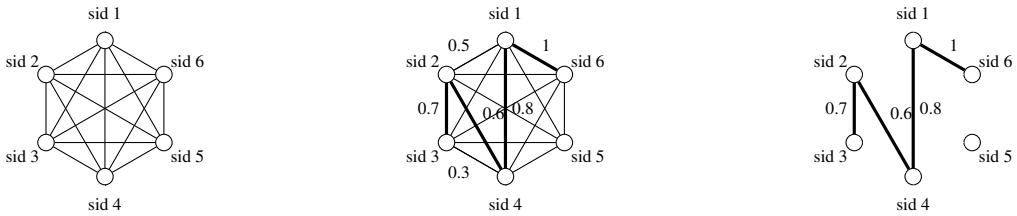


Figure 2: Pruning example for a graph on 6 sentences and median edge weight 0.5

length 128, so that every node embedding will appear separately. Then we trained a Bidirectional LSTM network on this data using the sentence label obtained as described in Section 2.2. To generate a summary, we first added all the sentences for which label 1 was predicted by our neural model in the order of their appearance in the 3,000-word MUSE summary. If their total word count exceeded the limit of 1,000 words, we selected the sentences that appeared first. If, however, the total word count was less than 1,000 we added the sentences with label 0 from the 3,000-word MUSE summary in order of their appearance until the 1,000-word limit was reached.

We also experimented with an enhanced method, called AMUSE$_{en}$, *en* short for *enriched*. The purpose of this model is to address the original financial reports that we were not able to build a neural model for due to their size. To produce an enriched summary, we first found the locations (sentence indexes) of the sentences with label 1 predicted by our model. Then, if the number of sentences between two of these sentences was less than a predefined parameter (in our experiments, it was set to 2 sentences empirically[1]), we also labeled the in-between sentences as 1 regardless of whether they appeared in the MUSE 3,000-word summary or not. Then, we applied the same summary generation procedure as in Section 2.4. This procedure aimed to try and catch the cases where the human experts who generated the gold standard summaries have used the entire paragraphs or sections of the original document.[2]

3 Experiments

The Financial Narrative Summarization (FNS 2021) shared task aims to demonstrate the value and challenges of applying automatic text summarization to financial text written in English, usually referred to as financial narrative disclosures. For the creation of the financial narrative summarization dataset, 3,863 UK annual reports published in PDF file format were used. UK annual reports are lengthy documents with around 80 pages on average, some annual reports could span over more than 250 pages, while the summary length should not exceed 1000 words. The training set includes 3,000 annual reports, with 3-4 human-generated summaries as gold standard. For the evaluation and system development the validation set of 363 files was provided. Table 1 contains the dataset statistics.

3.1 Methods and baselines

We evaluate two variations of our approach (denoted by AMUSE and AMUSE$_{en}$), which are described in Section 2 and compare their results with MUSE. As a reference, we also present the results of a trivial TOP-K baseline that includes the first 1,000 words of a document. Our and baseline models were applied to the validation part of the FNS-2021 shared task dataset, and results are reported

[1] We tested larger distances such as 5 and 10 but they did not produce any improvement in summary quality.

[2] The gold-standard summaries of the FNS data are mainly extracts of entire sections.

dataset	# documents	# gold summaries	avg sentences	avg words	avg characters
Train	3,000	9,873	2,700	58,838	291,014
Validation	363	1,250	3,786	82,906	416,040
Test	500	NA	3,743	82,676	412,974

Table 1: FNS 2021 dataset statistics.

System	R1 R	R1 P	R1 F	R2 R	R2 P	R2 F
TOP-K	0.266	0.241	0.221	0.040	0.038	0.034
MUSE	0.261	0.297	0.243	0.042	0.052	0.040
AMUSE	0.281	0.284	0.248	0.046	0.050	0.042
AMUSE$_{en}$	0.283	0.281	0.247	0.047	0.049	0.042
System	**RL R**	**RL P**	**RL F**	**RSU4 R**	**RSU4 P**	**RSU4 F**
TOP-K	0.264	0.239	0.220	0.081	0.076	0.069
MUSE	0.255	0.292	0.238	0.084	0.100	0.079
AMUSE	0.271	0.275	0.239	0.091	0.096	0.082
AMUSE$_{en}$	0.272	0.271	0.238	0.091	0.094	0.081

Table 2: ROUGE results for FNS-2021 validation set.

System	R1 F	R2 F	RL F	RSU4 F
BASE	0.45	0.24	0.42	0.27
MUSE	0.50	0.38	0.52	0.43
AMUSE	0.50	0.27	0.44	0.30
AMUSE$_{en}$	0.35	0.11	0.26	0.18
LexRank	0.31	0.12	0.27	0.16

Table 3: ROUGE results for FNS-2021 test set.

below.[3]

Experiments were performed on a cloud server with 32GB of RAM, 150 GB of PAGE memory, an Intel Core I7-7500U 2.70 GHz CPU, and two NVIDIA GK210GL GPUs.

3.2 Evaluation results

We applied four ROUGE (Lin, 2004) metrics—ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU4 on the validation set. Table 2 shows the results, with recall, precision, and F-measure for each metric. It can be seen that AMUSE outperforms MUSE in most metrics (8 out of 12) on the validation set, meaning that applying two-stage AMUSE, including the former neural modeling, produces better summaries than the simple-stage MUSE application. According to the Wilcoxon pairwise non-parametric tests, the difference in results for AMUSE and AMUSE$_{en}$ was significant for all twelve ROUGE scores, and the difference between AMUSE and MUSE was significant for ROUGE-2 F, ROUGE-L F, and ROUGE-SU4 F measures. Another interesting observation is that AMUSE outperforms AMUSE$_{en}$, meaning that completing/enriching pure MUSE's selections at the first stage of our method mainly introduces irrelevant sentences. Table 3 shows the results for the same ROUGE metrics, F-measure, obtained on the test set (provided by the FNS organizers). Unfortunately, these results do not demonstrate any superiority of AMUSE over MUSE.

[3]The results on the test set, provided by the FNS organizers, can be seen on the FNS leaderboard https://www.lancaster.ac.uk/staff/elhaj/docs/fns2021_results.pdf and in the Appendix.

4 Conclusions and Future Work

This paper introduces a two-stage method for the summarization of financial reports. The method combines several techniques, such as supervised optimization with GA, unsupervised learning of BERT and node embeddings, and supervised binary classification with LSTM. The evaluation results show that (1) preliminary filtering of irrelevant parts of a text with an efficient summarizer enables the subsequent application of the computationally consuming neural models for producing the final high-quality summaries, and (2) additional stages with help of neural modeling are capable to represent and detect relevant parts of an input text efficiently. The future work may include exploring transformer-based models that are designed to process long sequences such as Longformer (Beltagy et al., 2020), other summarizers as filtering tools at the first stage.

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, Dandelion 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.

Raksha Agarwal, Ishaan Verma, and Niladri Chatterjee. 2020. Langresearchlab_nc at fincausal 2020, task 1: A knowledge induced neural net for causality detection. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 33–39.

Piyush Arora and Priya Radhakrishnan. 2020. Amex ai-labs: An investigative study on extractive summarization of financial documents. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 137–142.

Abderrahim Ait Azzi and Juyeon Kang. 2020. Extractive summarization system for annual reports. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 143–147.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

François Chollet et al. 2015. Keras. https://keras.io.

Mahmoud El-Haj. 2019. Multiling 2019: Financial narrative summarisation. In *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources*, pages 6–10.

Mahmoud El-Haj, Vasiliki Athanasakou, Sira Ferradans, Catherine Salzedo, Ans Elhag, Houda Bouamor, Marina Litvak, Paul Rayson, George Giannakopoulos, and Nikiforos Pittaras. 2020a. Proceedings of the 1st joint workshop on financial narrative processing and multiling financial summarisation. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*.

Mahmoud El-Haj, Marina Litvak, Nikiforos Pittaras, George Giannakopoulos, et al. 2020b. The financial narrative summarisation shared task (fns 2020). In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 1–12.

Mahmoud El-Haj, Paul Rayson, and Andrew Moore. 2018. The first financial narrative processing workshop (fnp 2018). In *Proceedings of the LREC 2018 Workshop*.

Kavita Ganesan. 2018. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. *arXiv preprint arXiv:1803.01937*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo, and Ichiro Sakata. 2017. Extractive summarization using multi-task learning with document classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2101–2110.

Moreno La Quatra and Luca Cagliero. 2020. End-to-end training for financial report summarization. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 118–123.

Lei Li, Yafei Jiang, and Yinan Liu. 2020. Extractive financial narrative summarisation based on dpps. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 100–104.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Marina Litvak, Mark Last, and Menahem Friedman. 2010. A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 927–936.

Marina Litvak, Natalia Vanetik, Mark Last, and Elena Churkin. 2016. Museec: A multilingual text summarization tool. In *Proceedings of ACL-2016 System Demonstrations*, pages 73–78.

Marina Litvak, Natalia Vanetik, and Zvi Puchinsky. 2020. Sce-summary at the fns 2020 shared task. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 124–129.

Abhishek Singh. 2020. Point-5: Pointer network and t-5 based financial narrative summarisation. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 105–111.

Jaime Baldeon Suarez, Paloma Martínez, and Jose Luis Martínez. 2020. Combining financial word embeddings and knowledge-based features for financial text summarization uc3m-mc system at fns-2020. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 112–117.

Amit Vhatkar, Pushpak Bhattacharyya, and Kavi Arya. 2020. Knowledge graph and deep neural network for extractive text summarization by utilizing triples. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 130–136.

Siyan Zheng, Anneliese Lu, and Claire Cardie. 2020. Sumsum@ fns-2020 shared task. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 148–152.

A Appendix

For preprocessing such as sentence splitting, tokenization, obtaining word vectors and BERT sentence vectors we used spacy v3.0 package (Honnibal and Johnson, 2015); for LSTM neural model training and predictions we used Keras (Chollet et al., 2015) with Tensorflow (Abadi et al., 2015) as a back-end. The LSTM network that we used had 50 neurons, and it was trained for 100 epochs (the parameters were chosen empirically). We experimented with different networks such as 2- and 3-dimensional CNNs, convolutional LSTM, and their combinations, but none of them gave an advantage over a simple LSTM network.

Document graphs were constructed using the networkX package (Hagberg et al., 2008). Graph-based sentence embeddings were computed with the node2vec package (Grover and Leskovec, 2016). We have used the MUSEEC tool (Litvak et al., 2016) to compute MUSE summaries to be used as a baseline and as the first stage of our method, with 1000-word and 3000-word limits, respectively. For the Node2Vec algorithm, we use the implementation of `https://github.com/eliorc/node2vec` to generate node embeddings of size 128, setting the number of walks to 10 and the walk length to 80 (the parameters were chosen empirically). We used ROUGE 2.05 java package (Ganesan, 2018). .

NUS-IDS at FinCausal 2021: Dependency Tree in Graph Neural Network for Better Cause-Effect Span Detection

Fiona Anting Tan, See-Kiong Ng
Institute of Data Science
National University of Singapore, Singapore
tan.f@u.nus.edu, seekiong@nus.edu.sg

Abstract

Automatic identification of cause-effect spans in financial documents is important for causality modelling and understanding reasons that lead to financial events. To exploit the observation that words are more connected to other words with the same cause-effect type in a dependency tree, we construct useful graph embeddings by incorporating dependency relation features through a graph neural network. Our model builds on a baseline BERT token classifier with Viterbi decoding, and outperforms this baseline in cross-validation and during the competition. In the official run of FinCausal 2021, we obtained Precision, Recall, and F1 scores of 95.56%, 95.56% and 95.57% that all ranked 1st place, and an Exact Match score of 86.05% which ranked 3rd place.

1 Introduction

We worked on the shared task of FinCausal 2021 (Mariko et al., 2021) that aims to identify cause and effect spans in financial news. This task builds on the previous shared Task 2 (Mariko et al., 2020) by introducing additional annotated data.

Contributions: We propose a solution to include dependency relations in a sentence to improve identification of cause-effect spans. We do so by representing dependency relations in a graph neural network. Our model is an extension of a baseline BERT token classifier with Viterbi decoding (Kao et al., 2020), and outperforms this baseline in cross-validation and test settings. This improvement also holds for two BERT pretrained language models that were experimented with. During the competition, we ranked 1st for Precision, Recall, and F1 scores and 3rd for Exact Match score.

Organization: Section 2 outlines our approach for this task. Section 3 introduces the task dataset and evaluation datasets. Our results are presented and discussed in Section 4 while Section 5 concludes with some future directions.

2 Our Approach

In this section, we outline our approach [1]. Additional architectural and experimental details are provided in the Appendix.

2.1 Framing the Task

Given an example document, which could be one or multiple sentence(s) long, the task is to identify the cause and effect substrings. We converted the span detection task into a token classification task, similar to many state-of-the-art methodologies for span detection (Pavlopoulos et al., 2021) and Named Entity Recognition (Lample et al., 2016; Tan et al., 2020) tasks. Figure 1 demonstrates an example sentence that has its Cause (C) span highlighted in green, and Effect (E) span highlighted in orange, while all other spans are highlighted in grey. The sentence was tokenized and subsequently aligned against the target token labels. We included the BIO format (Begin, Inside, Outside) (Ramshaw and Marcus, 1995) in our labels to better identify the start of spans. Thus, we have five labels: B-C, I-C, B-E, I-E and O.

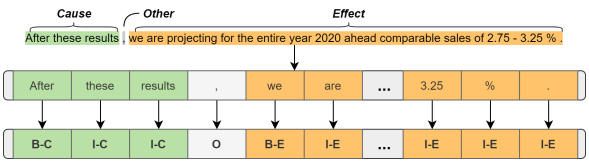


Figure 1: Illustrative training example (ID: 0477.00020) with Cause (C) and Effect (E) spans highlighted in green and orange respectively. We include Begin (B), Inside (I) and Outside (O) prefixes to create 5 labels in our token classification task.

2.2 Baseline

Kao et al. demonstrated that their BIO tagging scheme with a Viterbi decoder (Viterbi, 1967) that

[1] Our source code is available at https://github.com/tanfiona/CauseEffectDetection.

utilised BERT-encoded document representations is useful for this cause-effect span detection task. Their model topped the competition last year across all metrics. Therefore, we adapted their pipeline and proposed distinct additions highlighted later in Section 2.3 for improved performance. In the immediate subsections, we motivate the benefits in retaining the key components of Kao et al.'s model, and highlight any differences in our approach.

2.2.1 BERT Embeddings

We employed the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) for its tokenizer and encoder model, fine-tuned on our task. We used pretrained language models from Huggingface (Wolf et al., 2020). Apart from `bert-base-cased`, we also used `bert-large-cased` for improved performance.

2.2.2 Viterbi Decoder

The Viterbi decoding algorithm is only applied during evaluation. Since the true cause-effect spans are consecutive sequences, but the token classifications from the neural network could produce non-consecutive sequences, the Viterbi algorithm serves as a forward error correction technique and is an important element for the success of this pipeline.

2.2.3 Parts-of-Speech

Kao et al. (2020) showed that Parts-of-Speech (POS) did not improve their model performance. We reconfirm this finding later in Section 4.2. However, we found POS features to be useful inclusions for our proposed model.

2.3 Dependency Tree

Our key contribution is the inclusion of dependency tree relations into the neural network for improved cause-effect token classification. Dependencies in text can be mapped into a directed graph representation, where nodes represent words and edges represent the dependency relation. Figure 2 shows some example sentences, highlighted by their Cause and Effect labels. We notice that there is a tendency for words of the same cause-effect label to be more connected in these graphs and thus wish to incorporate these information into the model as features.

Figure 3 reflects our neural network model, with the addition of our GNN module that produces graph representations, which are then concatenated with the BERT and POS embeddings and fed into a linear layer for token classification. The following subsections describe the GNN module further.

2.3.1 Document to Graph

Each example was represented as a directed graph, where nodes are token features (the concatenation of BERT and POS embeddings), while edges are directed connections pointing head to tail tokens [2] based on dependency tree parsing [3].

2.3.2 Graph Neural Network

Our graph neural network (GNN) comprised of two graph convolutional layers for message passing across dependency relations that are two steps apart. Specifically, we used SAGEConv operator (Hamilton et al., 2017) for its ability to include node features and generate embeddings by aggregating a node's neighbouring information. To capture the long-term contextual dependencies in both forward and backwards order of the original sentence, we added a bi-directional long short-term memory (BiLSTM) layer (Hochreiter and Schmidhuber, 1997) onto the graph embeddings.

3 Data

3.1 Task Dataset

Our main dataset is the FinCausal 2021 dataset (Mariko et al., 2021) comprising of 2393 train and 638 competition test examples.

3.2 Evaluation

We evaluated our proposed models in cross-validation (CV) against the Viterbi BERT model (Kao et al., 2020) that achieved first place in the previous run of this shared task. We refer to this model as the Baseline in subsequent sections. To check if our proposed models has statistically significant improvements from the Baseline, we adapted Dietterich (1998)'s approach using a 3-fold CV setup for 5 iterations, each iteration initialised with a random seed [4]. This gives us $5 * 3 = 15$ sets of evaluation results to perform Paired T-Tests for statistical significance.

To obtain test predictions for submission on Codalab [5], we used a new $seed = 123$ to train on the full train data and applied the model onto the unseen competition test data for online submission.

[2] If head or tail words are split into multiple tokens, each head (sub) piece is connected to each tail (sub) piece.

[3] Stanza (Qi et al., 2020) was used for dependency parsing.

[4] The 5 random seeds used were $916, 703, 443, 229, 585$

[5] `https://competitions.codalab.org/competitions/33102`

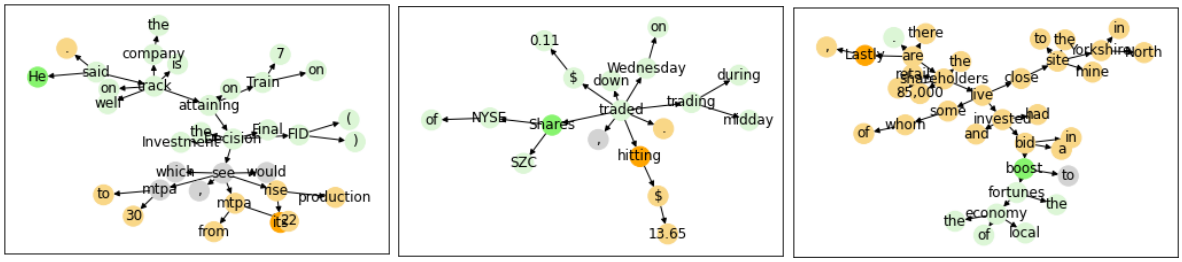


Figure 2: Dependency-tree represented as directed graphs, with Cause, Effect, and Other spans highlighted in green, orange and grey respectively.

	Model	Precision	Recall	F1	ExactMatch
A.	**`bert-base-cased`**				
1	Baseline	95.62	95.90	95.76	88.18
2	+ Node features, BiLSTM	95.42	95.95	95.68	88.07
3	Baseline w/ POS	95.39^	96.02	95.70	88.24
4	+ Node features	95.66	95.81	95.73	88.20
5	+ BiLSTM	95.46	95.99	95.72	88.06
6	+ Node features, BiLSTM (Proposed)	**95.73**	**96.05^**	**95.89**	**88.35**
B.	**`bert-large-cased`**				
7	Baseline	95.64	96.01	95.82	87.65
8	+ Node features, BiLSTM	93.75*	95.56	94.61*	86.44
9	Baseline w/ POS	94.46	96.08	95.22	87.28
10	+ Node features	95.61	95.98	95.79	88.24
11	+ BiLSTM	95.60	95.78^	95.69	87.94
12	+ Node features, BiLSTM (Proposed)	**95.72**	**96.11**	**95.91**	**88.35**

Table 1: Average evaluation results over cross-validation sets from 5 random seeds, each with 3 folds. *Notes.* Scores are reported in percentages (%). Best score per Panel per column is bolded. Baseline models are our replications of the models introduced by Kao et al. (2020). For each Panel A and B, Paired T-test of the models was conducted against Row 1 and 7 respectively, with statistical significance indicated by: ***<0.05, **<0.10, *<0.15, ^<0.20.

Model	Precision	Recall	F1	ExactMatch
Baseline	93.47	93.42	93.65	80.25
Proposed (`bert-base-cased`)	94.24	94.21	94.37	83.23
Proposed (`bert-large-cased`)	**95.56**	**95.56**	**95.57**	86.05
Best Score	**95.56**	**95.56**	**95.57**	**87.77**
Our Ranking	*1st*	*1st*	*1st*	*3rd*

Table 2: Results over test sets submitted to Codalab (As of 01 September 2021). *Notes.* Scores are reported in percentages (%). Best score per column is bolded. The models of the first three rows corresponds to Rows 1, 6, and 12 in Table 1 for CV respectively.

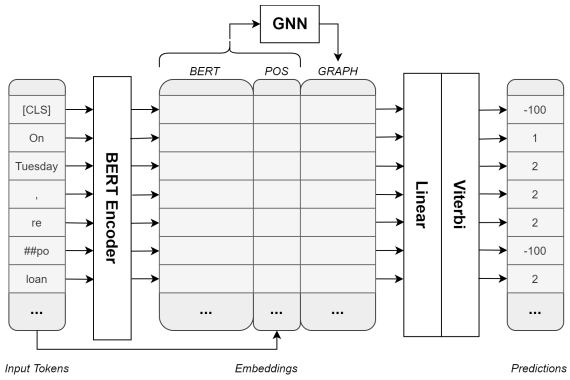


Figure 3: Neural network pipeline. Appendix A.1 outlines this further.

4 Results and Analysis

Table 1 reflects the model performances in CV, while Table 2 reflects the results during the competition. In both cases, we demonstrate that our model (Proposed) surpasses the Baseline.

For the CV setting, Proposed (Row 6) obtained 95.73% Precision (P), 96.05% Recall (R), 95.89% F1 and 88.35% exact match (EM) scores. These exceed the Baseline (Row 1) by 0.11%, 0.15%, 0.13% and 0.18% respectively [6]. For test setting, large performance increments were observed: Proposed achieved P/R/F1/EM scores of 94.24%, 94.21%, 94.37% and 83.23%, which are improvements from Baseline by a magnitude of 0.77%, 0.78%, 0.72% and 2.98% respectively [7].

4.1 Size of Pretrained Models

The two sections of Table 1 shows that the inclusion of dependency-based graph embeddings improved performance against Baseline irregardless of the pretrained BERT model choice.

Between the two investigated BERT models, `bert-large-cased` outperforms `bert-base-cased` by a small amount in CV and by a significant amount in testing across metrics. With `bert-large-cased`, the Proposed model achieved P/R/F1/EM scores of 95.56%, 95.56%, 95.57% and 86.05% during the competition, which are improvements from Baseline by a magnitude of 2.09%, 2.14%, 1.92% and 5.80% respectively [8].

[6] We did not obtain P-values ($< 5\%$) of statistical significance when comparing Proposed against Baseline.

[7] We were unable to run repeated iterations to conduct Paired T-Tests for significance testing in competition test sets as we do not have access to the true labels.

[8] We did not upload a Baseline model using `bert-large-cased`, which would have allowed for a fairer comparison, due to time and upload constraints.

4.2 Features and Layers

Table 1 also includes results from CV experiments where we removed components of our model. In this subsection, we discuss the importance of each component in the context of `bert-base-cased`, but note that similar findings persisted in the `bert-large-cased`.

POS: Inclusion of the POS into the Baseline led to mixed outcomes across the four metrics against the Baseline (Row 3 vs Row 1). However, adding POS features in Proposed improved performance in all metrics (Row 6 vs Row 2).

Node features: We reran a model that takes in nodes with no features (i.e. all nodes are represented by "1"). The results from this model corresponds to Row 5. No obvious improvements from Baseline (Row 1) was found, however, the performance was consistently worse off than Proposed for all metrics (Row 6), suggesting that informative node features are important to include in the GNN.

BiLSTM: Comparing our proposed model with (Row 6) and without (Row 4) the BiLSTM layer suggested the layer was an important addition. Our hypothesis is that the BiLSTM helps to align the graph embeddings into a sequential manner corresponding to the original token order. A simple example is the punctuation full-stop "." in Figure 2. The target label of the full-stop in these cases coincides with the immediate label of the word before it. However, our dependency tree attributed the full-stop as a tail of another word far away from it in the sentence (E.g. *"said"* → ".").

5 Conclusions and Future Work

We have demonstrated the benefits of including dependency tree features as graph embeddings in a neural network model for better cause-effect span detection. A key caveat of our approach, which requires further research, is that dependency parsing occurs within sentences, resulting in disconnected graphs for examples with multiple sentences [9]. Another future work is to apply our cause-effect detection model trained on financial texts onto other domains (E.g. academic journals) to study its generalizability.

[9] Preliminary experiments to tie coreferential entities together to link dependency across sentences did not produce fruitful results.

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. Association for Computational Linguistics.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10(7):1895–1923.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Pei-Wei Kao, Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. NTUNLPL at FinCausal 2020, task 2:improving causality detection using Viterbi decoder. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 69–73, Barcelona, Spain (Online). COLING.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

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 *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 23–32, Barcelona, Spain (Online). COLING.

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.

John Pavlopoulos, Jeffrey Sorensen, Léo Laugier, and Ion Androutsopoulos. 2021. SemEval-2021 task 5: Toxic spans detection. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 59–69, Online. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. 2020. Boundary enhanced neural span classification for nested named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9016–9023.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Appendix

A.1 Architecture of Proposed Model

An example with n tokens can be represented by the vector of tokens $\underline{w} = (w_0, w_1, ..., w_n)^\intercal$, where w refers to a BERT input token. A single token in this vector is represented by w_i, where i denotes the location index of the token within the example. The tokenized example also has a matrix of POS features $V = (\underline{v_0}, \underline{v_1}, ..., \underline{v_n})^\intercal$, where each $\underline{v_i} \in \mathbb{R}^{d_{POS}}$ refers to a one-hot encoding across all POS tags, and so d_{POS} reflects the number of possible POS tags.

To obtain BERT representations (r), we run the tokenized sequence vector through the BERT encoder (T) to obtain r for each token i. Thus, we have that,

$$\underline{r_i} = T_w(\underline{w})_i, \qquad \underline{r_i} \in \mathbb{R}^{d_{BERT}} \tag{1}$$

where d_{BERT} refers to the output dimension for BERT encoder.

A dropout layer is represented by $\delta \sim Bernoulli(\rho)$, where ρ refers to the dropout probability. We apply the dropout layer onto the BERT representations as follows,

$$\underline{\tilde{r_i}} = \delta * \underline{r_i}, \qquad r_i \in \mathbb{R}^{d_{BERT}} \tag{2}$$

We combine the BERT and POS representations of each token together by a simple concatenation along the feature dimension.

$$\underline{r_{2i}} = [\underline{\tilde{r_i}}, \underline{v_i}], \qquad \underline{r_{2i}} \in \mathbb{R}^{d_{BERT}+d_{POS}} \tag{3}$$

Our GNN model generates graph embedding based on these concatenated features, which are then concatenated together to arrive at our final embeddings.

$$\underline{r_{3i}} = GNN(\underline{r_{2i}}), \qquad \underline{r_{3i}} \in \mathbb{R}^{d_{GNN}} \tag{4}$$

$$\underline{r_{4i}} = [\underline{r_{2i}}, \underline{r_{3i}}], \qquad \underline{r_{4i}} \in \mathbb{R}^{d} \tag{5}$$

d_{GNN} refers to the output dimension for the last layer in our GNN module introduced in Section 2.3. That is, if BiLSTM is opted, then d_{GNN} refers to the size of the output dimension of the BiLSTM layer. If not, it refers to the output dimension for the second SAGEConv layer. Our final representations have a feature dimension size of $d = d_{BERT} + d_{POS} + d_{GNN}$.

Next, we run our combined embeddings through a linear layer to obtain predicted probabilities per token. c refers to the number of classes to be predicted.

$$\underline{o_i} = \underline{r_{4i}} * W + \underline{b_i}, \qquad W \in \mathbb{R}^{d \times c}, \quad \underline{o_i}, \underline{b_i} \in \mathbb{R}^{c} \tag{6}$$

Cross entropy loss was used during training to optimize model weights. In evaluation, the logits ran through a Viterbi decoder for adjustment. Finally, all logits ran through an argmax function to obtain the predicted class that had the highest probability.

In our implementation, we set $d_{BERT} = 768$, $d_{POS} = 51$, $d_{GNN} = 512$ and $c = 5$.

A.2 Replication Checklist

- Hyperparameters: Our pretrained BERT models were initialized with the default configuration from Huggingface (Wolf et al., 2020). To train our model, we used the Adam optimizer with $\beta 1 = 0.9$, $\beta 2 = 0.999$. Learning rate was set at $2e{-}05$ with linear decay. GPU train batch size was set as 4. Maximum sequence length was 350 tokens. For GNN, the graph hidden channel dimensions (i.e. output dimension of the first SAGEConv layer) was 1024, the graph output dimension (i.e. output dimension of the second SAGEConv layer) was 512, and the BiLSTM output dimension was also 512. Probability for all dropout layers was 0.1.
- Device: All experiments were ran on the NVIDIA A100-SXM4-40GB GPU.
- Time taken: For 3 folds over 10 epochs each, the Proposed model took us on average (over the 5 random seeds) $1hour : 48minutes : 28seconds$ for `bert-base-cased` and $1hour : 22minutes : 24seconds$ for `bert-large-cased` to train, validate and predict. For a single run over 10 epochs to generate our submission, the code took $28minutes : 17seconds$ and $20minutes : 52seconds$ to train and predict for the base and large models respectively.

A.3 Qualitative Results

In Table 3, we provide examples where the Proposed versus Baseline model predicts correctly when the other predicts wrongly. The predictions are obtained from the CV set of the `bert-large-cased` model with $seed = 916$ and the first fold.

Index	Baseline	Proposed	Right?
0036.00011	<E>Future sales agreements with suppliers increased during the period, and aggregate contracted sales volumes are now 11.7m tonnes per annum</E>, following <C>new European supply agreements.</C>	<C>Future sales agreements with suppliers increased during the period, and</C> <E>aggregate contracted sales volumes are now 11.7m tonnes per annum</E>, following new European supply agreements.	Baseline
0270.00009	<E> It comes with a £250 free overdraft and requires a £1,000 monthly deposit</E> to <C>avoid a £10 monthly fee.</C>	<C>It comes with a £250 free overdraft</C> and requires a £1,000 monthly deposit to <E>avoid a £10 monthly fee.</E>	Baseline
0209.00033	<C>Fiserv believes that this business combination makes sense from the complementary assets between the two companies, projecting higher revenue growth than</C> <E>it would achieve on its own and costs savings of about $900 million over five years.</E>	<C>Fiserv believes that this business combination makes sense from the complementary assets between the two companies</C>, <E>projecting higher revenue growth than it would achieve on its own and costs savings of about $900 million over five years.</E>	Proposed
0003.00019	<E>Additionally, the Congress provided $125 million in the current fiscal year for sustainable landscapes programming</E> to <C>prevent forest loss.</C>	<E>Additionally, the Congress provided $125 million in the current fiscal year</E> for <C>sustainable landscapes programming to prevent forest loss.</C>	Proposed

Table 3: Predicted Cause-Effect spans for CV set from $seed = 916$ on first fold (i.e. $K0$). *Notes.* Cause and Effect spans highlighted in green and orange respectively.

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

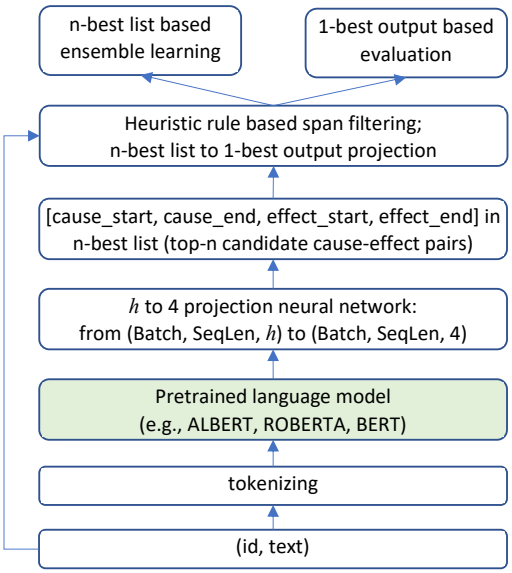


Figure 1: Fine-tuning based system architecture.

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.

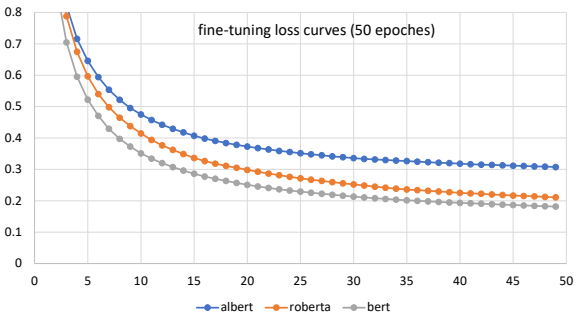


Figure 2: Fine-tuning curves of the 3 PLMs.

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%.

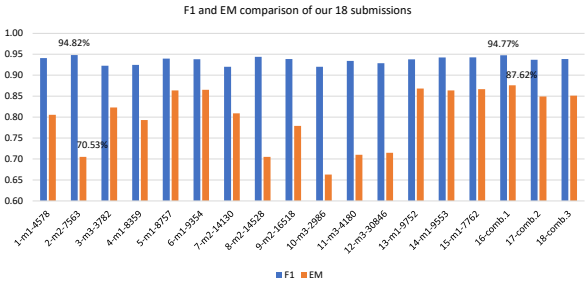


Figure 3: Comparison of stable F1 and astonished EM among our 18 submissions.

DSC-IITISM at FinCausal 2021: Combining POS tagging with Attention-based Contextual Representations for Identifying Causal Relationships in Financial Documents

Gunjan Haldar[1⊕], **Aman Mittal**[1⊕], **Pradyumna Gupta**[2⊕]
[1]Department of Mechanical Engineering
[2]Department of Electronics Engineering
[⊕]Indian Institute of Technology (ISM), Dhanbad 826004, India

Abstract

Causality detection draws plenty of attention in the field of Natural Language Processing and linguistics research. It has essential applications in information retrieval, event prediction, question answering, financial analysis, and market research. In this study, we explore several methods to identify and extract cause-effect pairs in financial documents using transformers. For this purpose, we propose an approach that combines POS tagging with the BIO scheme, which can be integrated with modern transformer models to address this challenge of identifying causality in a given text. Our best methodology achieves an F1-Score of 0.9551, and an Exact Match Score of 0.8777 on the blind test in the FinCausal-2021 Shared Task at the FinCausal 2021 Workshop.

1 Introduction

Integrating causality information as text features can substantially benefit a plethora of applications such as text mining (Girju and Moldovan, 2002), event prediction (Wei and Wang, 2019), question answering (Sharp et al., 2016) and many more. One of the primary motives of this, which has been explored in this challenge, is to extract causality in the financial domain, which can be applied to various tasks such as financial services support (Chen et al., 2020), consumer review (Patil, 2016), stock movement prediction (Chen, 2021) as well as help different institutions to gain insights into the financial sector.

By examining the financial documents carefully, one can observe that single and multiple causal events in a given paragraph may exist. Additionally, there can also be the existence of numerous causal chains in the same. To deal with such cases, we formulate causality detection and extraction task as a sequence labeling and modeling problem and propose an approach using POS tagging (Dhumal Deshmukh and Kiwelekar, 2020) with BIO scheme tagging (Liu et al., 2015) integrated with an ensemble of BERT Large-cased (Devlin et al., 2018), XLNet Base (Yang et al., 2019), BERT Large-Cased Whole Word Masking, GPT-2 (Radford et al., 2019) and RoBERTa Base (Liu et al., 2019), achieving an F1-Score of 0.9551 and Exact Match score of 0.8777 on Blind test dataset provided by the workshop.

2 Dataset

The dataset provided (Mariko et al., 2021) (Mariko et al., 2020) for this challenge[1] has been extracted from 2019 financial documents provided by Qwam[2], consisting of the complete text and the extracted cause and effect pairs along with offset markers. It was also observed that multiple instances comprised of the same text but different causality pairs, due to presence of multiple chains of causal relationships. Total instances present in the database were 2393 which were split into 2101 training and 292 validation instances.

3 Methodology

3.1 Part-Of-Speech (POS) Tagging

We tokenize each sentence and generate rule-based part-of-speech (POS) tags (Dhumal Deshmukh and Kiwelekar, 2020) for each token. Rule-based POS tagging uses contextual information and a set of handwritten rules to assign POS tags to tokens in a sentence.

After tokenizing the data, the tokens are converted into POS tags. The POS tags are enumerated, which are further mapped on the tokenized sentences. These POS tags are represented in the form of a one-hot vector. This vector is concatenated with the model's hidden state output of the last layer, which is then sent to the final linear layer

[1] http://wp.lancs.ac.uk/cfie/fincausal2021/
[2] https://www.qwamci.com/

Token	POS Tag	BIO Tag	Token	POS Tag	BIO Tag
The	DT	B-E	It	PRP	B-C
Sunshine	NNP	I-E	is	VBZ	I-C
State	NNP	I-E	consistently	RB	I-C
drew	VBD	I-E	one	CD	I-C
...	...	I-E	...	...	I-C
...	...	I-E	...	...	I-C
older	JJR	I-E	taxes	NNS	I-C

Table 1: Pre-processed Output stored in text format, The above text represents an example instance from the training set.

of the model. Predictions are performed on the concatenated vector or tensor.

Tag	Description	BIO Label
B-E	At the Beginning of Effect	3
B-C	At the Beginning of Cause	1
I-C	Inside of Cause	2
I-E	Inside of Effect	4
-	Padding	0

Table 2: Tagging Scheme explanation. BIO tag "O" will be converted to padding.

3.2 BIO Scheme Tagging

To extract the causal relations and positional information of the words, considering the semantics of the causal events, we use the BIO tagging (Liu et al., 2015) scheme i.e. Begin-Inside-Outside tagging with Cause and Effect labels (C-E). BIO tagging scheme will represent whether the token is at the beginning (B) of the target phrase, inside (I) of a target phrase and tokens which are not a part of cause or effect are considered as being outside (O) of the target phrase and are labelled as padding (-). Additionally, due to varying sequence length, extra tokens which are not included in cause and effect tuples are converted to padding as shown in Table 2.

3.3 Pre-processing

To begin with, two different modes are given as input for pre-processing. When the mode is "training", the corresponding sentence and cause-effect tuples in the training data are append to a dictionary, otherwise when the mode is "test", sentences in the test dataset are appended to a dictionary. Each sentence in the paragraph is tokenized, subsequently, separate tokens and their positional index are stored in a list.

Further, for the preparation of BIO tags, the index of the tokenized words are identified in each sentence using its respective index and stored in a dictionary. The beginning of cause and effect pairs are found in the sentence, and this pair is tokenized. Tokens at the beginning of the cause and effect are labelled as B-C and B-E respectively. Subsequent tokens in cause and effect sentences are labelled as I-C and I-E respectively. These labels along with the words are stored in a dictionary identified by their index. The tags are extracted from the dictionary. This process is iterated over all the instances in the training set.

To end with, each word is concatenated with its respective POS tags and BIO tags as shown in Table 1. The pre-processed file is stored in a text format which is further passed onto the model as input.

3.4 Transformer Architecture

For the purpose of this challenge, our best approach utilizes an ensemble developed using BERT (Bidirectional Encoder Representations from Transformers) Large-Cased model (Devlin et al., 2018), RoBERTa (Robustly Optimized BERT Pre-training Approach) (Liu et al., 2019), GPT-2 (Generative Pre-trained Transformer) (Radford et al., 2019), BERT Large-Cased Whole Word Masking (Devlin et al., 2018) (BWM), XLNet (Yang et al., 2019) by applying the huggingface[3] (Wolf et al., 2019) package.

3.4.1 Models

BERT Large-cased transformer model has been pre-trained on the English language with a masked language modeling (MLM) objective distributed into Masked Language Modelling and Next Sentence Prediction (NSP), which converges to learn

[3] https://huggingface.co/bert-large-cased

Model	Epochs	MSL*	Validation Score†	Blind Test†
BERT-base	40	256	0.9197	0.9253
RoBERTa-base	50	256	0.9201	0.9372
GPT-2	20	128	0.9251	0.9422
XLNet-base	50	128	0.9368	0.9466
BERT-large	**50**	**256**	**0.9389**	**0.9517**
BWM	50	256	0.9327	0.9476

Table 3: Model Comparison by Experimentation; *Maximum Sequence Length, †F1 Score

Model	F1-Score	Recall	Precision	Exact Match
BERT-large + RoBERTa + XLNet + GPT-2 + BWM	0.9551	0.9580	0.9554	0.8777

Table 4: Best performing method on the official Blind Test of FinCausal-2021

an internal representation that can be utilized to extract features from downstream tasks. This model consists of 24 transformer encoder layers with 1024 hidden dimensions with 16 self-attention heads.

BWM model has been pre-trained on the same language corpus as BERT Large-Cased model but with a whole word masking technique, wherein all of the tokens corresponding to a word are masked at once. The overall masking rate remains the same. The model was pre-trained on 4 cloud TPUs for one million steps with a batch size of 256. The sequence length was limited to 128 tokens for 90% of the steps and 512 for the remaining 10%. The optimizer used is Adam with a learning rate of 1e-4, β_1 = 0.9 and β_2 = 0.999 and a weight decay of 0.01

RoBERTa is pre-trained with the same objective as BERT but on 1024 V100 GPUs for 500K steps with a batch size of 8K and a sequence length of 512. Adam optimizer is used with a learning rate of 6e-4, β_1 = 0.9, β_2 = 0.98 and ϵ = 1e-6, and a weight decay of 0.01 with dynamic masking where the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and has to predict the masked words.

GPT-2 transformer model takes sequences of continuous text as input and uses an internal mask-mechanism to predict the token at any position "i" by the inputs at position 1 to "i".

XLNet is a generalized autoregressive pre-training method enabling learning of bidirectional contexts.

3.4.2 Training

The pre-processed output file was procured, and for every instance, the corresponding POS and BIO tags of each token was extracted and stored in an array. According to the maximum sequence length, these arrays were padded. Depending on the transformer model utilized, `[CLS]` and `[SEP]` tokens were appended to the tokens. For instance, if the transformer model was BERT Large-Cased, `[CLS]` token was appended at the beginning and `[SEP]` token at the end and when the transformer model is XLNet, `[CLS]` is appended at the end. Pseudo POS tag ID and BIO tag ID for `[CLS]` or `[SEP]` token was set as "0" and "-100" respectively. All ID sequences were padded with padding token ID - "0" in POS tag sequence and "-100" in BIO tag sequence.

Each model consumed on an average 3-4 hours for training. The configurations of the best models which are used for the ensemble are reported in Table 3. All these models have been trained with a batch size of 64 with cross-entropy loss (Gordon-Rodriguez et al., 2020) so that only real IDs contribute to the loss function and not the padding IDs.

3.5 Post-processing & Exact Match Optimization

The received predictions are in the format of tuples of tokens and their corresponding predicted BIO tag. The BIO tags are retrieved and stored in a list with the index of each token in the prediction. Further, this process is iterated over all the predicted instances and recorded. We tried to optimize the Exact Match metric by selecting the longest cause-effect pair when multiple causal chains are present in a given data instance. If the number of padding tokens was less than a given threshold between two similar predicted phrases (Cause/Effect), the two pairs were merged.

Text	Cause	Effect
The company also recently announced a quarterly dividend, which was paid on Tuesday, September 3rd. Shareholders of record on Thursday, August 15th were paid a $0.03 dividend. This represents a $0.12 annualized dividend and a yield of 3.42%.	The company also recently announced a quarterly dividend, which was paid on Tuesday, September 3rd.	Shareholders of record on Thursday, August 15th were paid a $0.03 dividend.
	The company also recently announced a quarterly dividend, which was paid on Tuesday, September 3rd.	This represents a $0.12 annualized dividend and a yield of 3.42%.
If you pay the full RAD there is no interest (DAP) pay no RAD and you will pay a DAP which is the interest on the full amount: $22,160.	pay no RAD	you will pay a DAP which is the interest on the full amount: $22,160.
	If you pay the full RAD	there is no interest (DAP)

Table 5: Table representing identical multi-causal chains. Causal chains in the training dataset.

3.6 Ensemble

After each prediction was extracted from different models present in the ensemble, the mode was calculated to find the most frequently occurring label. In the presence of a tie-breaker scenario, we select the label predicted by the best performing single transformer model, BERT Large-Cased. Further, after extracting all the tags, these were aligned with the text to get the actual words bundled together to form the cause-effect pair.

4 Experimentation and Results

Different models along with custom loss functions were trained on the given data and local F1 score, Recall, and Precision were evaluated. Transformer models including RoBERTa (Robustly Optimized BERT Pre-training Approach) (Liu et al., 2019), GPT-2 (Generative Pre-trained Transformer) (Radford et al., 2019), BERT Base (Devlin et al., 2018), BERT Large-Cased Whole Word Masking (Devlin et al., 2018) (BWM), XLNet (Yang et al., 2019) were experimented with different hyper-parameter settings. The best performing settings along with their corresponding scores are reported in Table 3. The results were evaluated locally, and considering those metrics, the model performance was observed. To boost up optimization, ensembles of the aforementioned transformer models were experimented and evaluated.

GPT-2 was trained and experimented with, but due to expensive computational requirements, it was trained for 20 epochs. Loss function while RoBERTa-base transformer model was being trained on the data couldn't converge, resulting in a low metric score; similar behavior was observed in XLNet. BERT large-cased model outperformed all these models due to its large architectural layout when a single shot transformer is concerned. Maximum Sequence Length (MSL) is a critical factor while training a model with limited computational resources, because having a high MSL means most of the memory is wasted for padding and not used for weight update. Subsequently, smaller MSL values are chosen for transformer models with vast architecture. Ensembles mentioned in Table 4 gave a relatively low F1 score when BERT-base was included along with other models indicating that the lower performance of BERT-base single shot experiment could be the prominent dropping factor. The performance metrics of the top approach is shown in Table 4.

5 Conclusion

This paper presents our sequence labeling and modeling approach, combining POS tags with BIO scheme using ensemble optimization strategy comprising BERT-large, RoBERTa, XLNet, GPT-2, and BERT-Large (whole word masking) for causality detection in financial documents which helped us achieve the highest Exact Match score of 0.8777, on the FinCausal-2021 Shared Task leaderboard. Future works can describe an optimization pipeline constituting architecturally larger transformer models. Furthermore, more advanced post-processing strategies can be investigated to extract multiple causal relationships in a text.

References

Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. Nlp in fintech applications: Past, present and future. *ArXiv*, abs/2005.01320.

Qinkai Chen. 2021. Stock movement prediction with financial news using contextualized embedding from bert.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Rushali Dhumal Deshmukh and Arvind Kiwelekar. 2020. Deep learning techniques for part of speech tagging by natural language processing. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 76–81.

Roxana Girju and Dan Moldovan. 2002. Text mining for causal relations.

Elliott Gordon-Rodriguez, Gabriel Loaiza-Ganem, Geoff Pleiss, and John P. Cunningham. 2020. Uses and abuses of the cross-entropy loss: Case studies in modern deep learning.

Pei-Wei Kao, Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. Ntunlpl at fincausal 2020, task 2:improving causality detection using viterbi decoder. In *FNP*.

Shengyu Liu, Buzhou Tang, Qingcai Chen, and Xiaolong Wang. 2015. Drug name recognition: Approaches and resources. *Information*, 6:790–810.

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. 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.

Mayur Patil. 2016. Summarization of customer reviews for a product on a website using natural language processing.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and Michael Hammond. 2016. Creating causal embeddings for question answering with minimal supervision. *CoRR*, abs/1609.08097.

Xiang Wei and Bang Wang. 2019. A survey of event extraction from text. *IEEE Access*, PP:1–1.

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.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

LIORI at the FinCausal 2021 Shared task: Transformer ensembles are not enough to win

Adis Davletov
RANEPA, Moscow, Russia
Lomonosov Moscow State University, Moscow, Russia
davletov-aa@ranepa.ru

Sergey Pletenev
RANEPA, Moscow, Russia
HSE University, Moscow, Russia
pletenev-sa@ranepa.ru

Denis Gordeev
RANEPA, Moscow, Russia
Russian Foreign Trade Academy, Moscow, Russia
gordeev-di@ranepa.ru

Abstract

In this paper, we report on our system for FINCAUSAL 2021 Financial Document Causality Detection Task. In this task, the aim is to identify, in a causal sentence or text block, the causal elements and the consequential ones. We propose a system that uses a pre-trained model, fine-tuned on the extended dataset, and task-specific post-processing of the model's inputs to improve the quality of the results. We tried two types of approaches: 1) a fine-tuned T5-model that generated cause and effect spans 2) and a sequence-to-sequence model based on XLNet that solved the task as token classification. The best result of our XLNet-large is 0.946 F1 on the test set while T5-model got the F1 score of 0.835 which may be due to the lower number of exact matches.

1 Introduction

Causality detection is an important problem as a vital part of natural language understanding. It is especially true for the domain of finance and economics where causes should contribute to the prediction model while effects should either be used as an output or omitted from the model altogether. A major contribution to the field was provided in the workshop FinCausal 2020 (Mariko et al., 2020) where the authors have provided a labelled dataset for causality and effect detection and a platform for the discussion of the results and further aligned measurement of the models. It contained two tracks: the first task was to classify whether a sentence contains causality or not, while the second one was devoted to the extraction of causes and effects from the sentences.

This work is focused on our approach to FinCausal 2021 [1]. Unlike the last year competition, this year shared task consisted of a single track equivalent to the second track from 2020. The dataset consists of texts each containing causes and effects. One text may contain several spans of the same type.

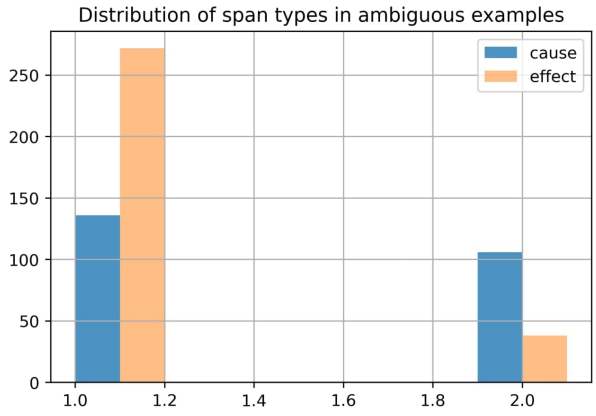


Figure 1: Ambiguous text examples

The winning solution of the 2020 2nd subtask consisted in a BERT-CRF system with a Viterbi decoder (Kao et al., 2020). This year we also tried to implement their solution but were unsuccessful with this approach and got the F1 score of 0.875 on our test dataset derived from 2021 training data.

2 Preprocessing

In this work we experimented on two datasets:

- FinCausal-2021 dataset, which consists of two subtasks, including causal meanings detection (Task 1) and cause-effect detection (Task 2). The numbers of training instances are 22,058

[1] http://wp.lancs.ac.uk/cfie/fincausal2021/

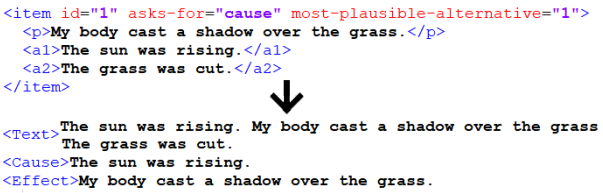


Figure 2: Example of Balanced-COPA dataset transform

and 1,750 for Task 1 and Task 2, respectively. This work only focuses on Task 2.

- Balanced-COPA dataset (Kavumba et al., 2019), The Choice Of Plausible Alternatives dataset (COPA) contains 1000 examples of two question types: a) What is the CAUSE of this? and b) What is the EFFECT of this?. Balanced COPA contains one additional, mirrored instance for each original training instance in COPA. A total of 1,500 examples are given.

Since we are experimenting with two different models, the data sets will be handled differently for both models.

For the Seq2Seq model, the data set is split into two parts. The model is to predict the cause part and the effect part of a text in two steps. In the first step, the model predicts only the Cause part of the text, is the second step – only the Effect part. To control which part of the sentence the model should generate, we use additional prefixes in the text that will be given to the input of the model. At the end of each text, we prepend an additional prefix as shown in 3.

For Sequence Tagging models we used encode_causal_tokens function provided by organizers to reformat all data to BIO format.

3 Models

The task of extracting textual relations can eventually be generalized to the task of selecting a subset of the words or sentences in the text. There are many approaches to solving this task. We have selected two variants: Sequence tagging and Sequence-to-Sequence generation.

3.1 Sequence tagging

In case of sequence tagging models we experimented with BERT(Devlin et al., 2018), RoBERTa(Liu et al., 2019), ALBERT(Lan et al., 2019) and XLNet(Yang et al., 2019) models. Models were asked to predict sequences of labels in BIO format. Every input example was tokenized using 'word_tokenize' function from the NLTK library (Bird et al., 2009). Optionally, we feed the information about the number of input sentences to the models, which we get using PunktTrainer and PunktSentenceTokenizer from NLTK library trained on all provided textual data. We concatenate the one-hot encoded number of sentences to the output from pretrained models. Our models are quite similar to the BERT-base model from (Kao et al., 2020). The differences are in our post-processing steps and different training scheme. Also, we experiment with linear or non-linear classifier layers over pretrained models. In post-processing steps, we apply four transformations similar to the rules worked out by the Workshop organizers to annotate the training data (Mariko et al., 2020). They are:

Text: The politician lost the election. No one voted for him. He ran positive campaign ads.

—

Input: The politician lost the election. No one voted for him. He ran positive campaign ads. Question: cause

Output: No one voted for him

Input: The politician lost the election. No one voted for him. He ran positive campaign ads. Question: effect

Output: The politician lost the election

Figure 3: Example of additional prefixes for text

- If a sentence contained only one fact (cause or effect), we tagged the entire sentence.
- The annotation of sentence-to-sentence causal relationships is prioritized
- When a causal chain is located inside a single sentence, in order to facilitate the extraction process, we chose to span the causal units as much as possible
- If two facts of the same type were located in the same sentence and were related to the same effect or cause, then we annotated these two facts as one unit

Rules	F1	Exact Match
Basic model	86.24	69.58
Rule 1	86.32	69.73
Rule 12	87.25	71.14
Rule 123	86.04	38.53
Rule 1234	86.21	38.53
Rule 124	**87.44**	**71.76**

Table 1: Results for rule combinations on the dev set for the best performing model

We trained our models for 15 epochs making validation every quarter of epoch saving the best models. We tested all models with all sequences of rules and chose the rule combination with the best F1 score and Exact Match. It appeared to be the rule combination 1-2-4 (see Table 1). As our final submitted model, we use a voting ensemble of two ALBERT models and one XLNet model.

3.2 Sequence-to-Sequence

Modern Sequence-to-Sequence models are successful in many tasks. In this paper we use the T5 model (Raffel et al., 2020). The T5 model is trained on several data sets for 18 different tasks, which are split into 8 categories: summarizing text, question answering, translation, etc.

We use HuggingFace Transformers [2] for T5 training and prediction. The model is trained with the following parameters: encoder length 512, decoder length 256, batch size 2, 8 epochs, learning rate 5e-05, after every 1000 steps we evaluate our models with beam size 8.

To get different results in multi-effect or multi-cause cases we use the diverse beam-search(Vijayakumar et al., 2018): If the resulting hypothesis from diverse beam-search starts with a different symbol, they are presented as new results.

4 Results

Model	F1
Viterbi (Kao et al., 2020)	0.875
BARTNER (Yan et al., 2021)	0.7729
T5_large	0.868
T5_large_2	0.8741

Table 2: Viterbi-BERT and T5 analysis on our development set

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

Model	F1
T5 Sequence-to-Sequence	0.835267
ALBERT XXLarge	0.93984
XLNet-base-cased	0.925649
ensemble of 2 ALBERT XXLarge and 1 XLNet models	0.946473

Table 3: Results on the evaluation dataset

The final results of our models are shown in Table 3. As can be seen from the table, our Sequence tagging models outperform T5 Sequence prediction models.

We have also tested the 2020 Fincausal winning solution (Kao et al., 2020) and BARTNER (Yan et al., 2021) as an alternative to T5. BARTNER also solves the token classification problem as a sequence classification task. T5 outperformed BARTNER, while its performance was close to BERT+Viterbi. However, we failed to replicate the results for the Viterbi model. It might be due to hyperparameter tuning or some mistake in processing. All in all, our token classification turned out to be the most successful among the ones that we have tried.

XLNet and ALBERT models have been trained using 2 Nvidia GeForce RTX 2080 TI GPUs. T5 models have been trained using GPUs provided by Google Colab Pro.

5 Error Analysis

Such a low result in T5 compared to regular token tagging is probably due to the fact that the model sometimes has difficulty in predicting perfectly all the tokens in the input text. While token tagging only works with the source text, the t5 model may mix up or miss some tokens due to its seq2seq nature. This is confirmed by the low exact match result in the table.

6 Conclusion

In this work, we describe our results for the FinCausal 2021 dataset. We have tried Sequence Classification and Sequence-to-Sequence models. Sequence classification outperformed Sequence-to-Sequence in our case. We have also tested various sequence post-processing schemes and ensembles of Transformer-based models.

Acknowledgements

We would like to thank the organisers for the task and the workshop.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Pei-Wei Kao, Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. NTUNLPL at FinCausal 2020, task 2:improving causality detection using Viterbi decoder. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 69–73, Barcelona, Spain (Online). COLING.

Pride Kavumba, Naoya Inoue, Benjamin Heinzerling, Keshav Singh, Paul Reisert, and Kentaro Inui. 2019. When choosing plausible alternatives, clever hans can be clever.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

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 preprint arXiv:1907.11692*.

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 *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 23–32, Barcelona, Spain (Online). COLING.

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.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various ner subtasks. *arXiv preprint arXiv:2106.01223*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

The Financial Document Causality Detection Shared Task (FinCausal 2021)

Dominique Mariko
dmariko
@yseop.com

Hanna Aki Akl
habi-akl
@yseop.com

Estelle Labidurie
elabidurie
@yseop.com

Stephane Durfort
sdurfort
@yseop.com

Hugues de Mazancourt
hdemazancourt
@yseop.com

Mahmoud El-Haj
m.el-haj
@lancaster.ac.uk

Abstract

We present the FinCausal 2021 Shared Task on Causality Detection in Financial Documents and discuss the participating systems and results. A total of 6 teams submitted runs across the task and 4 of them contributed with a system description paper. This task is associated with the 3rd Financial Narrative Processing Workshop (FNP 2021), held at Lancaster University, UK, on September 15-16, 2021.

1 Introduction

This shared task is a follow up session of FinCausal 2020 presented at COLING 2020. In this edition we chose to propose only the data and task formerly named Task 2, which is a causality detection task.

2 Data

The data are extracted from a corpus of 2019 financial news provided by Qwam, collected on 14.000 economics and finance websites. The original raw corpus is an ensemble of HTML pages corresponding to daily information retrieval from financial news feed. These news mostly inform on the 2019 financial landscape, but can also contain information related to politics, micro economics or other topics considered relevant for finance information.

For this edition, the training data have been slightly augmented with 643 examples added in the Practice data set, evaluation data remaining the same as in the 2020 edition. For a detailed overview of the corpus creation and 2020 edition systems, see Mariko et al. (2020). Data are released under the CC0 License.

3 Task

The purpose of this task is to extract, from provided text sections, the chunks identifying the causal sequences and the chunks describing the effects.

The trial and practice samples were provided to participants as csv files with headers: *Index; Text; Cause; Effect*

- Index: ID of the text section. Is a concatenation of [file increment . text section index]
- Text: Text section extracted from a 2019 news article
- Cause: Chunk referencing the cause of an event (event or related object included)
- Effect: Chunk referencing the effect of the event

Average statistics on the causes and effects chunks detected in the causal text sections are provided in Table 1. A data sample for the task is provided in Table 2. Interesting results (up to 94.72 F1 score) had been achieved during the 2020 edition, one of the remaining difficulty being the prediction of complex causal chains considered during the annotation process, leading to one text section possibly containing multiple causes or effects.

4 Evaluation

A baseline was provided on the trial samples. Participating systems were ranked on blind Evaluation datasets based on a weighted F1 score, recall, precision, plus an additional Exact Match. Regarding official ranking, weighted metrics from the scikit-learn package were used, and the official evaluation

Metric	Trial	Practice	Evaluation	Total
Total number of text sections	641	1752	638	3031
Total number of unicausal text sections	500	1404	452	2090
Total number of multicausal text sections	141	348	186	941
Average character length of causal chunks	113.73	109.30	112.48	-
Average character length of effect chunks	107.79	102.56	99.66	-

Table 1: Task Data Distribution

Index	Text	Cause	Effect
0009.00052.1	Things got worse when the Wall came down. GDP fell 20% between 1988 and 1993. There were suddenly hundreds of thousands of unemployed in a country that, under Communism, had had full employment.	Things got worse when the Wall came down.	GDP fell 20% between 1988 and 1993.
0009.00052.2	Things got worse when the Wall came down. GDP fell 20% between 1988 and 1993. There were suddenly hundreds of thousands of unemployed in a country that, under Communism, had had full employment.	Things got worse when the Wall came down.	There were suddenly hundreds of thousands of unemployed in a country that, under Communism, had had full employment.
23.00006	In case where SGST refund is not applicable, the state is offering a 15% capital subsidy on investments made in Tamil Nadu till end of 2025.	In case where SGST refund is not applicable	the state is offering a 15% capital subsidy on investments made in Tamil Nadu till end of 2025

Table 2: Three examples from FinCausal 2021 Corpus - Practice dataset

script is available on Github [1]. Participating teams were proposed to submit at most 20 runs, and to choose themselves which run they wished to display on the leaderboard.

Results for the task are provided in Table 3. Last line displays the best 2020 result for the task by Kao et al. (2020) (please note the 2020 edition training set was smaller than the one proposed in 2021) . One of the challenge of this task was to rebuilt the correct span of causal chunks, according to the annotation scheme.

5 Participating systems

Most participants took advantage of the systems proposed in 2020 and worked on augmenting them with different strategies.

The winning system by NUS-IDS adapts Kao et al. (2020)'s BERT-CRF with Viterbi decoder, augmenting the initial BIO-scheme with dependency tree relations, mapping the text dependencies into a directed graph and concatenating them with the BERT and POS embeddings, framing their solution as a Graph Neural Network performing a token classification task.

The system ranked second by DSC-IITISM choose to frame the task as a sequence labelling problem, using additional BIO-scheme with transformers embeddings, the best model averaging ensemble for XLNet + GPT-2 + BWM models. A post processing optimization method is proposed, selecting the longest cause-effect pair when multiple causal chains are present in a given data instance to tackle the Exact Match problem.

The third system by NVJPFSI builds on Becquin (2020), using grid-based ensemble learning among multiple n-best outputs for BERT, RoBERTa and ALBERT models, optimizing the Exact Match metric.

The fourth system by LIORI experiments on sequence to sequence as well as token sequence classification methods, eventually using an ensemble method averaging ALBERT and XLNet.

6 Conclusion

Participants have built interesting augmentations from 2020 edition systems. This and the additional training data proposed for the 2021 shared task allowed to level up the results. Tackling the complex causality problem remains a task in itself though, and would need a dedicated session and specific data to be fully addressed.

[1] https://github.com/yseop/YseopLab/tree/develop/FNP_2020_FinCausal

Team	F1	Recall	Precision	EM	BIO	LM	TF	Ens	HS
NUS-IDS (1)	**95.56**	**95.56**	**95.57**	86.05	x	x			
DSC-IITISM (2)	95.51	95.50	95.54	**87.77**	x		x	x	x
NVJPFSI (3)	94.77	94.78	94.78	87.62			x	x	x
LIORI (4)	94.65	94.67	94.65	78.37			x	x	
NTUNLPL (2020 edition)	94.72	94.70	94.79	82.45	x	x			

Table 3: Task results and approaches adopted by the participating teams. BIO refers to any adaption of the IOB scheme. LM refers to any language model embedding features, excluding the use of Tranformers models. TF refers to Transformers architecture and associated embeddings. Ens corresponds to Ensemble Learning method. HS implies some heuristics has been used in the final computation, mostly to adapt the span.

Acknowledgements

We thank the FNP 2021 Committee for the opportunity to propose this shared task.

7 References

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.

Pei-Wei Kao, Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. NTUNLPL at FinCausal 2020, task 2:improving causality detection using Viterbi decoder. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 69–73, Barcelona, Spain (Online). COLING.

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 *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 23–32, Barcelona, Spain (Online). COLING.

Annotation model and corpus for opinionated economy and finance narrative detection

Jiahui Hu, Patrick Paroubek and Dirk Schumacher

Abstract

Specialized press and professional information channels influence beliefs on economic outlook or prospects for financial markets by drawing attention on particular events, and disseminating domain expert opinions. Analyzing this textual data allows for a better understanding of investors' beliefs and detecting key indicators for market dynamics.

Though considerable efforts have been made to develop data-hungry algorithms on coarse-grained level sentiment analysis on finance-related social media messages, performing fine-grained level target-dependent opinion analysis on documents written by domain experts and journalists is still a relatively unexploited field.

Since some narratives are essentially made of opinions/emotions expressed about economy and finance concepts, we address fine-grained detection of these linguistic markers at an intra-sentential level. We propose, in this paper, a global model extracting from texts terms that are specific to finance and economy or expressing an opinion/emotion in order to address the challenges of the domain-specific language we face: (1) opinions and facts about a given factor may appear at different locations (2) the range of domain-specific concepts is large and opinion may be explicit or implicit (3) syntactic structures and rhetorical relations often carry useful information for detecting market change indicators (4) emotions, like panic, also need to be detected since they are part of the economic and financial market cycle. The proposed model consists of the incorporation of fundamental approaches in natural language processing, language evaluation theory (appraisal theory), and machine learning methods for information extraction and data annotation.

In this paper, we present our annotation model and report on experiments to evaluate the quality of our dataset.

1 Introduction

The processing of information is crucial in determining financial assets' prices. Thus, market participants' opinions can be an essential driver of price dynamics. Recent progress of NLP technologies and access to digitized texts have facilitated automatic sentiment analysis of financial narratives. Most existing corpora for opinion analysis applied to economy and finance focus on the sentence level polarity (Malo et al., 2013) or text level (Cortis et al., 2017), leaving aside opinion targets. (Barbaglia et al., 2020) created a corpus focusing on the polarity of six macroeconomic aggregates, but it is not publicly available as of the time of writing. The corpus of FiQA task 1 [1] for fine-grained opinion analysis of news headlines and tweets is relatively small (1,313 samples) for training supervised learning models, and it contains relatively short sentences. In the texts we will analyze, the sentences are generally longer.

Therefore, we introduce a corpus consisting of labels annotated at the intra-sentential level by humans and algorithms to fill this gap. The novelty of our corpus is that we also consider specific rhetorical modes like financial experts do. Each sentence contains the following annotations: (1) terminologies in economics and finance, (2) opinion and emotion expressions (OEE), (3) name entities, (4) negation patterns and (5) the pair (target, polarity).

2 Methodology

2.1 Dataset

Our corpus is collected from five reputable sources from 1986 to 2021 (see Table 4 for size and sources of the raw dataset, see Figure 4 for corresponding time range). These texts aim at communicating, discussing, or commenting on business and economic activities. On the one hand, contents issued

[1] https://sites.google.com/view/fiqa/home

by the central banks [2] and corporates(MD&A of 10-K filings [3] (Ewens, 2019) and transcripts of earning calls) are first-hand information that is essential for financial markets. On the other hand, when it comes to news articles [4] and tweets, outsider comments on these official contents reflect how financial participants evaluate these events; the popularity of certain narratives in the media also shed some light on the main driver of market dynamics.

2.2 Annotation scheme

Our primary objective is to label all pairs of opinion expressions and their corresponding target, i.e. (target, opinion), at the intra-sentential level. The opinion is classified into three polarities: positive, negative and non-committal; polarities are attributed based on the judgement related to economic norms or the health of business activities. The novelty of our scheme is that we consider how financial experts communicate and analyze the evolution of event trends, namely,

a. Formulations of argumentative constructions
b. Conditional opinions
c. cause-effect relations
d. explicit speculation about the future

Our raw dataset contains both facts and opinions. In TBOA, we focus on **sentences of interest** that contain at least one terminology in economy and finance (called TOI, terms of interest) and at least one opinion & emotion expression (called OEE).

This criterion helps us to separate opinionated sentences from factual ones, because we can use existing NLP technologies to extract appraisal terms (see 2.3) and TOIs. TOIs are extracted as follows: we firstly candidate noun phrases[5] and keep just those containing elements of a domain-specific thesaurus. To extract appraisal terms, we look for exact matches between lemmatized words of each sentence and a pre-defined list of appraisal terms. The pre-annotation pipeline also includes the machine-assisted annotations of names entities and negation patterns.

2.3 How do we detect opinionated sentences

The particularity of opinionated sentences in financial narratives is that authors use evaluative language to monitor and judge an event (i.e. happenings or changes of a business or economic activity) or assess their impact. Authors may:

(1) monitor changes by using language to describe in which direction an event or a concept evolves [a],
(2) express a judgment about these dynamics by clarifying their preference; furthermore their expectations can be diversely grounded in a mix of rationality and/or emotions,
(3) and assess the intensity of these dynamics.

[a] in the DOWN & LOW category, *plummet* and *decrease* convey the notion of scaling *rapid* and *median*, respectively.

Figure 1: Our focus on specific aspects of texts written by financial experts

These elements converge toward the theoretical research about the language of evaluation. We have chosen **appraisal theory** because it provides meaning-making resources to assess the intensity (celled **Graduation**) or the direction of attitudinal expressions (called **Attitude**, i.e. affect, judgment and appreciation) and its author's commitment (called **Engagement**). As illustrated by Figure 1, we propose three axes to regroup opinion expressions about changes in economic and financial activities.

- Variation axis: gain or loss in quantity or volume, or description of stable state
- Attitude axis: recognition of value or loss in value, lack of visibility, anxious awareness of undesirable outcome; or even emotional assessment such as the intense feeling of excitement and strong desire to put ideas into practice, feelings of helplessness, the impression of losing control on the situation.
- Graduation axis is complementary to the two previous axes: high or low intensity.

2.4 Annotations

Our corpus is annotated with an open-source annotation platform called INCEpTION (Klie et al., 2018). As exemplified in Figure 2, the annotator identifies all targets towards which opinions are expressed and their polarities. Following the evaluation campaign DEFT 2018 (Paroubek et al., 2018), the annotator

(i) selects minimal information about the *Opinion & Emotion Expression* (i.e. *"dysfunctional"*, tagged [OEE]),
(ii) selects the most complete information about the target (i.e. *Sovereign bond market* in Fig-

[2] link of ECB's press conferences and speeches, link of FOMC
[3] link of MD&A data source
[4] We randomly choose sentences from Financial PhraseBank dataset (Malo et al., 2013) to apply our annotations.
[5] We use SpaCy (link), an open-source NLP toolkit for its computation efficiency.

ure 2) and attributes polarity (tagged [-] in Figure 2) to it,

(iii) then draws an unlabeled arc from [OEE] toward its corresponding target.

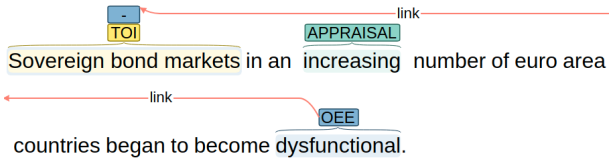


Figure 2: Example of a annotated sentence (explicit opinion)

Our corpus is annotated by one of the authors familiar with the domain terminology. Please refer to Appendix A.1 for more examples of our annotated sentences.

3 Syntactic structure of OEEs

The use of language differs from one speaker to another, depending on culture, profession, personal experience, or target audience. We assume that texts written by journalists and experts tend to use a more diverse vocabulary and syntactic structure to report facts accurately, persuade readers or polish their articles. To verify this assumption, we analyze the syntactic structure of subjective expressions in three types of texts that target different groups of people. We started from this angle because through the analysis of syntactic structure we aim to capture common phenomena in our corpus while detecting domain specificities of subjective expressions in financial narratives.

3.1 Corpora for comparison

SemEval14(Pontiki et al.) corpus is created for the NLP task[6] called *Aspect-based Sentiment Analysis*. This corpus consists of annotations of (target, polarity) of customer reviews on restaurants and laptops separately. OEEs are extracted using the neural model of (Fan et al., 2019).

MPQA(Wiebe et al., 2005) corpus consists of texts collected from a wide range of news sources. Authors annotate expressions related to opinions, beliefs, thoughts, feelings, emotions, goals, evaluations and judgments, called internal states. They divided it into two frames: expressive and direct subjectivity; the latter includes words for subjective speech events(such as say) and explicitly mentioned private states (such as fear). Our annotation scheme does not consider the language used to position a speaker's stance, corresponding to speech event expressions. Thus, we focus on the syntactic structure of expressive subjective elements, which are implicit evaluative expressions.

3.2 Tools

Numerous toolkits have been developed for syntactic analysis. We favoured Stanza(Qi et al., 2020), a state-of-the-art performance toolkit based on a neural NLP pipeline. For our algorithm, we use the universal part-of-speech (POS) tags and the syntactic dependency trees produced by the Stanza parser, focusing on the syntactic constructions of the OEEs themselves and the syntactic dependency relations that link them to other components of the sentence where they are located.

3.3 Result Analysis

We observe different patterns of opinion expressions in these four corpora.

Corpora	Unigram	2-3	4-5	6-10	>10
MPQA	28.16%	30.63%	16.61%	16.38%	8.17%
ECOFIN	49.26%	28.96%	12.11%	7.56%	1.63%
Laptop	94.26%	5.14%	0.23%	0.00%	0.00%
Restaurant	95.76%	3.24%	0.15%	0.00%	0.00%

Table 1 Statistics about number of tokens per OEE

In the SemEval 14 corpora, internauts tend to use unigrams (95.76% and 93.26% of all OEEs, respectively) for writing product reviews. Adjectives (adj) and verbs are the most used for commenting on restaurants and laptops, followed by a small portion of adverbs (adv) and nouns.

When it comes to new articles of the MPQA corpus, the variety of OEEs is the most diversified and balanced; we guess addressing implicit opinions requires more thoughtful expressions. Consequently, 72% of the OEEs are multi-grams.

In our corpus ECOFIN, the top three types of OEEs are unigrams: verbs(24.8%), adj(11.7%) and nouns(7.9%), but the overall portion of unigrams(49.26%) is much smaller than the SemEval 14. These multi-word subjective expressions, such as the combination of adj & nouns, adv & adj, are more frequently used in financial narratives than online comments (see Figure 5). In particular, the combination of verbs with other classes of words (such as adv, adposition[7]) represents at least 10% of OEEs. This observation is in line with the fact that financial experts are more likely to express their subjective opinions around changes and events, which require verbal expressions. We further investigate which are the most used verbs

[6] Semantic Evaluation 2014 Task 4

[7] preposition and postpositions

and how they relate to words in other classes(see 3.4).

Statistics of the five datasets in our experiments confirm our assumption (Table 1 and Figure 5, 6). Opinion expressions are domain-dependent; news articles and financial texts are more likely to employ multi-word opinion expressions composed of a wide range of word classes. Each word inside the OEE can modify the semantic orientation of another word, which complexifies the computation of the overall semantic orientation of the whole OEEs.

3.4 Analysis of verbs inside OEEs

Syntactic structure and word classes of OEEs can be valuable clues for determining where their corresponding target can be found. For example, for a unigram OEE whose word class is adjective, its target is likely to be the noun that follows because adjectives precede the noun they modify in English.

Following this idea, we manually examine 30 sentence of our corpus whose OEEs are in the form of verb+adp and find that most TOI precedes this type of OEEs, but some exceptions can be found in OEEs with the adp "to". Similarly, targets are very likely to be announced before OEEs "remain adj".

Recent studies ((Huang et al., 2020),(Zhao et al., 2021)) have proposed integrating syntax-related information in graph neural networks (GNN) or using GNN for sequence labelling by propagating the labelling information from known to unknown rules (which can be any rules, including syntactic ones). In the future, we want to study how these mechanism can be exploited to analyze our corpus.

3.5 Analysis of dependency relations

We also interest in the dependency relations inside each OEE of our corpus and how it is related to other words in the dependency tree. As exemplified in figure 3, the sentence is separated into three parts: OEEs, words that are above OEE (called *precedent_OEE*) and below OEE (called *posterior_OEE*).

Inside each OEE, the most frequent dependency relations are adjective and adverb modifiers and case-making relations linking adposition with the noun it attaches. The object is the fourth most important type of relation; it is connected to a verb and conveys information about the entity that undergoes a state change(see (1) in Figure 1). For example, in Figure 3, author's evaluative opinion toward *"fragmentation"* is expressed with two OEEs highlighted in orange and purple. Inside these two OEEs, the *"obj"* indicates which financial concepts (i.e., *"costs"* and *"(possibility of) economies of scale"*) are modified.

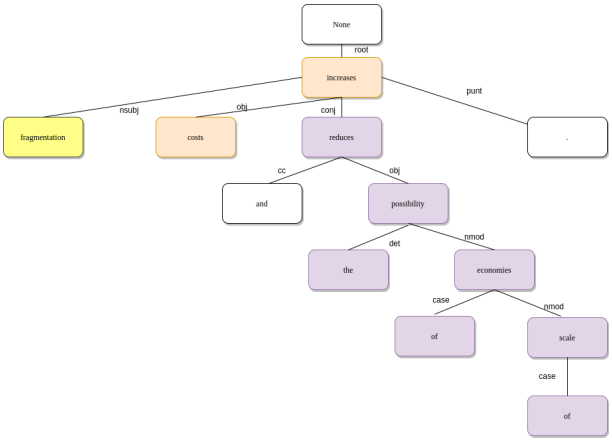


Figure 3: Dependency tree of "This fragmentation increases cost and reduces the possibility of economies of scale."

When it comes to dependency relations on the top of OEEs (posterior_OEE), we can find noun subjects, they can be the receive of an action. other most frequent dependencies of posterior_OEE and precedent_OEE can be found in Table 2.

Corpora	nsubj	obl	amod	advmod	obj	case
posterior_OEE	11.21%	8.56%	6.39%	5.52%	5.48%	3.61%
precedent_OEE	8.21%	7.53%	6.78%	3.91%	3.59%	9.0%

Table 2 Statistics about number of tokens per OEE

4 Conclusion

This paper presents our annotation scheme and the technologies used for pre-annotation. The pre-annotation output allows us to identify candidates for our corpus creation and alleviate the workload of annotators. We also compare the syntactic structure of OEEs of our financial narrative corpus with three corpora of fine-grained sentiment analysis. This comparison underlines the diversity of subjective expressions used by journalists and financial experts and the complexity of their syntactic structure. This result exemplifies why predicting TBOA from financial narratives is challenging. It also helps us understand how financial experts and journalists express opinions and how these subjective expressions in news articles and financial narratives differ from those in online comments.

In the future, we want to develop neural models adapted to our corpus by considering domain-specific knowledge, fundamental approaches in NLP in the neural model architecture to augment the machine's capacity to discover meaningful patterns.

Acknowledgements

This work is supported by the grant CIFRE, a partnership between Natixis CIB Research and the LISN Laboratory (Interdisciplinary Laboratory of Digital Sciences).

References

Luca Barbaglia, Sergio Consoli, and Sebastiano Manzan. 2020. Forecasting with economic news. Available at SSRN 3698121.

Keith Cortis, André Freitas, Tobias Daudert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. 2017. SemEval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pages 519–535, Vancouver, Canada. Association for Computational Linguistics.

Michael Ewens. 2019. Mda statements from public firms: 2002-2018.

Zhifang Fan, Zhen Wu, Xin-Yu Dai, Shujian Huang, and Jiajun Chen. 2019. Target-oriented opinion words extraction with target-fused neural sequence labeling. 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 2509–2518, Minneapolis, Minnesota. Association for Computational Linguistics.

Lianzhe Huang, Xin Sun, Sujian Li, Linhao Zhang, and Houfeng Wang. 2020. Syntax-aware graph attention network for aspect-level sentiment classification. In Proceedings of the 28th International Conference on Computational Linguistics, pages 799–810, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations, pages 5–9. Association for Computational Linguistics.

Pekka Malo, Ankur Sinha, Pyry Takala, Pekka Korhonen, and Jyrki Wallenius. 2013. Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts. arXiv:1307.5336 [cs, q-fin]. ArXiv: 1307.5336.

Patrick Paroubek, Cyril Grouin, Patrice Bellot, Vincent Claveau, Iris Eshkol-Taravella, Amel Fraisse, Agata Jackiewicz, Jihen Karoui, Laura Monceaux, and Juan-Manuel Torres-Moreno. 2018. DEFT2018 : Recherche d'information et analyse de sentiments dans des tweets concernant les transports en Île de France. In DEFT 2018 - 14ème atelier Défi Fouille de Texte, volume 2 of Actes de la conférence Traitement Automatique des Langues, TALN 2018, pages 1–11, Rennes, France.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. page 9.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. Language Resources and Evaluation, 39(2-3):165–210.

Xinyan Zhao, Haibo Ding, and Zhe Feng. 2021. Glara: Graph-based labeling rule augmentation for weakly supervised named entity recognition. CoRR, abs/2104.06230.

A Appendix

Type	CB	EC	MD&A	Tweets
Percentage of sentenced of interest	26%	63%	27%	15%
Randomly chosen sentences	22,259	1,065	4,793	2,628

Table 3 Percentage of sentences of interest from randomly chosen sentences

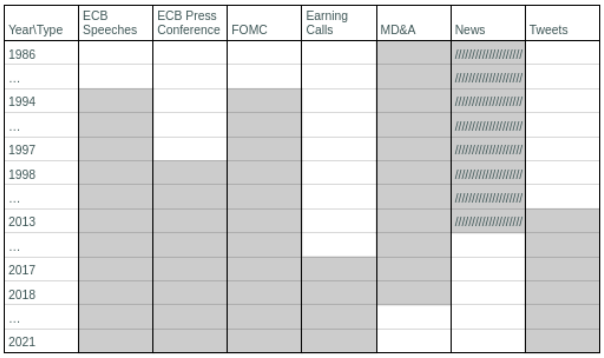


Figure 4: Shaded slashes of the column 'News' indicate that the time range of news sentences from the Financial PhraseBank dataset is incognito.

A.1 Sample sentences

Our ECOFIN corpus

(1) "The fair value of investment properties totalled EUR 2,299.9 mn , compared to EUR 2,229.5 mn in the corresponding period in 2009." [8]

(2) "This fragmentation increases costs and reduces the possibilities of economies of scale."[9]

Sent Num	target	polarity	OEE
(1)	deficit ratio	-	rise
(2)	fragmentation	-	increases costs
(2)	fragmentation	-	reduces the possibilities of economies of scale

Table 4 Our manual annotations: pair(target, polarity) and the corresponding OEE of each sample sentences

[8] from Financial Phrasebank dataset

[9] source: link

MPQA corpus[10]

- 'The criteria set by Rice are the following : the three countries in question are repressive and grave human rights violators , and aggressively seeking weapons of mass destruction.'
- 'The solidarity would bring about an international campaign to dry up the roots of terrorism and expand peace and security in the international community , but , certain countries resort to military power and embark on trampling upon human rights of civilians.'
- 'He explained that both the US and Jordan have different issues to deal with on a national level , including environmental issues.'

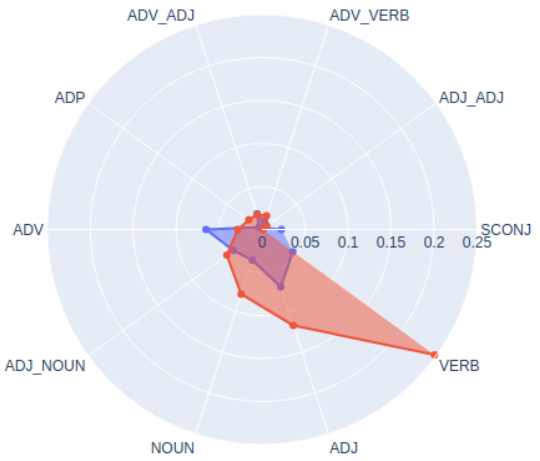


Figure 5: Top 8 universal POS of MPQA (blue) and ECOFIN (red) corpora

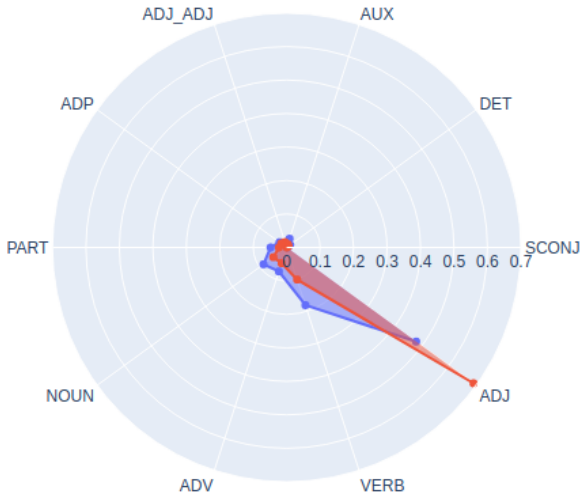


Figure 6: Top 8 universal POS of SemEval 14 corpora, Laptop (blue) and Restaurant(red)

[10]Expressive subjective elements are been underlined.

T5-LONG-EXTRACT at FNS-2021 Shared Task

Mikhail Orzhenovskii
orzhan057@gmail.com

Abstract

This paper describes T5-LONG-EXTRACT system submitted to the Financial Narrative Summarization Shared Task (FNS-2021). We developed a task-specific extractive summarization method based on pre-trained language model T5 and implemented a filtering procedure for the source dataset. The language model was fine-tuned on long sequences (4096 tokens) to identify the beginning of a continuous narrative part of the document. Our system ranks 1st on Rouge-1, Rouge-2, Rouge-SU4, and Rouge-L official metrics.

1 Introduction

Since financial data is constantly growing, automatic summarization is becoming increasingly important. The goal of the Financial Narrative Summarization Shared Task (El-Haj et al., 2021) is to summarize the annual financial reports from UK firms listed on The London Stock Exchange. It is a challenging task since the provided documents are long and the structures seem to vary.

We analyzed the provided reports and gold summaries and identified the non-overlapping matching word sequences between the reports and the corresponding gold summaries. For 99.4% summaries, there was only one matching block containing the entire summary, so the summary is included in the report as a whole subsequence. Further, we refer to those summaries as continuous.

In the vast majority of the reports, there is at least one continuous gold summary. Because of that, we concentrated on extracting one continuous part of the report as a summary. To perform this task, we only need to find two positions in the text: the beginning of the narrative part and its end. Most continuous summaries begin within the first 4000 tokens of the text.

The task could then be framed as an extractive summarization task, where we used the first 64 tokens of the selected gold summary as the target sequence and the first 4096 tokens of the report as the source sequence. After locating the beginning of the narrative part in the report, we select 60 sentences (but no more than 1000 words) as the system's summary.

2 Dataset

The dataset (Table 1) includes annual reports produced by UK firms listed on The London Stock Exchange. The texts can be 80 pages long which makes it challenging to analyze them manually. In the training and validation set, there were from 3 to 7 gold summaries for each report. We used the training set to fine-tune the model and select the number of training epochs, and used the validation set to choose the best performing model configuration.

We explored continuous gold summaries, which are included in the corresponding report as a whole subsequence. 3761 gold summaries (38.3%) are longer than 1000 words (maximum length required by the shared task). The average length is 1086 words. Only 5% of the summaries are longer than 3214 words. The average number of sentences in the gold summaries is 36, and 80% of the summaries have 57 or less sentences.

The position of the continuous gold summaries is on average 1695 words from the beginning. Half of those summaries have a position less than 1003, so most of such summaries are located at the beginning of the report. For 85.1% of the continuous summaries the position in the report is less than 2775 words and less than 4000 T5 tokens.

3 System

We selected only one gold summary for each annual report which had at least one continuous gold summary located not more than 4096 tokens from the beginning. Because the target metric was calculated using the average ROUGE score between the system's summary and all gold summaries, we

Dataset	Reports	Summaries
Train	3000	9873
Valid	363	1250
Test	500	N/A

Table 1: Dataset size

picked the summary that had the most intersections with the other summaries. The score for summary Sc_i was calculated based on intersections of words between summary S_i and other summaries S_j:

$$Sc_i = \frac{\sum_j |x \in S_i \cap S_j|}{|x \in S_i|}$$

This schema improved the results of the system, compared to selecting just the first summary or using all summaries.

We picked only summaries that were at least 100 words long. Out of 2940 resulting training samples, 2740 were inputted into the model, and the remaining 200 samples were used to determine the number of training epochs.

T5 (Raffel et al., 2020) is a sequence-to-sequence language model, pre-trained on multiple tasks, including abstractive text summarization. It is configured for 512 input tokens; however, the model is based on relative position embeddings, which allows to scale it to longer input sequences. Because of the complexity $O(n^2)$ of the Transformer's self-attention mechanism such scaling significantly increases memory consumption. As a countermeasure, we truncated the output sequence length to 64 tokens.

We fine-tuned the T5 model on the filtered dataset. During the inference, the model outputs 64 tokens. We locate the closest match of these tokens in the report's full text (in most cases, there is an exact match). The summary is created from 60 sentences starting from this location (the number of the sentences was selected based on gold summary statistics). After that the system's summary is truncated to 1000 words. To keep the output readable, we preserve white space while splitting the text into words.

4 Experiments

For fine-tuning we used HuggingFace's Transformers (Wolf et al., 2020) with the following parameters (Table 2).

We explored the effect of language model size on the system's performance. In an experiment, t5-base achieved ROUGE-2 F1 0.3808 on the validation set, and t5-small scored 0.3846 in the same configuration. Due to the increased source length, the models require more GPU memory to train even with batch size 1 (16 GB for t5-small and 40 GB for t5-base), so we were not able to experiment with the larger models.

It is critical to select a source length of 4096 in this task, as reducing it results in significantly poorer performance. The best ROUGE-2 F1 score of the t5-base model with input size 2048 was 0.3621. Other scores were also lower than those of the original model.

5 Results

The official metrics are shown in Table 3. On the validation set the system achieved Rouge-2 F1 0.38, with precision 0.34 and recall 0.47. We examined the summaries, produced by the system, on the validation set and compare them to the gold summaries. Including extra content in the summary lowers the precision. The system's summary is 1.72 times longer than the corresponding gold summary. Using a more accurate method of identifying the end of the summary (instead of the heuristics) can further improve the system's precision and F1 score.

The position of the narrative part identified by the system exactly matched one of the gold summaries in 24.5% of the validation samples. These samples typically begin with the words "chairman's", "highlights" and "strategic". For 47.3% samples, the system's summary was not more than 50 words away from one of the gold summaries. The accuracy of locating the summary is negatively correlated with the position: the farther the summary is placed in the text, the more difficult it is identify.

6 Conclusion and Future Work

In this paper, we describe T5-LONG-EXTRACT system for the Financial Narrative Summarization Shared Task. The proposed method achieved the highest score in all metrics.

Parameter	Value
Base model	t5-small
Maximum source length	4096
Maximum target length	64
Batch size	1
Warm-up ratio	0.1
Learning rate	5e-05
Training epochs	12

Table 2: Training parameters

System	ROUGE-1 F1	ROUGE-2 F1	ROUGE-L F1	ROUGE-SU4 F1
orzhan (T5-LONG-EXTRACT)	0.54	0.38	0.52	0.43
MUSE (TOPLINE)	0.5	0.28	0.45	0.32
POLY-baseline	0.37	0.12	0.26	0.18
LEXRANK (baseline)	0.26	0.12	0.22	0.14
TEXTRANK (baseline)	0.17	0.07	0.21	0.08

Table 3: Official scores of the system and the baselines. ROUGE Evaluation on the test set

T5-LONG-EXTRACT's source code and the weights of the fine-tuned model are available[1] and can be run using the provided instructions.

In the future, the language model can be trained in a multi-task setting to locate the end of the narrative section as well as the beginning. To reduce the memory requirements, sparse attention models (Longformer (Beltagy et al., 2020) or BigBird (Zaheer et al., 2020)) can be used as the encoder of the sequence-to-sequence model.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Mahmoud El-Haj, Nadhem Zmandar, Paul Rayson, Ahmed AbuRa'ed, Marina Litvak, Nikiforos Pittaras, and George Giannakopoulos. 2021. The Financial Narrative Summarisation Shared Task (FNS 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

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. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33.

[1] https://github.com/orzhan/t5-long-extract

Daniel@FinTOC-2021: Taking Advantage of Images and Vectorial Shapes in Native PDF Document Analysis

Emmanuel Giguet and Gaël Lejeune

Abstract

In this paper, we present our contribution to the FinTOC-2021 Shared Task "Financial Document Structure Extraction". We participated to the tracks dedicated to English and French document processing. We get results for Title detection and TOC generation performance which demonstrates a good precision. We address the problem in a fairly unusual but ambitious way which consists in considering simultaneously text content, vectorial shapes and images embedded in the PDF document, and to structure the document in its entirety.

1 Introduction

In the Fintoc-2021 Financial Document Structure Extraction competition (El Maarouf et al., 2021), two tasks have been proposed by the organizers : *ToC Structure Extraction* and *Title Detection*.

Extracting a table of content (ToC) is a way to improve information access in large documents, it can also be a mean to improve the results of some Natural Language Processing or Document Analysis tasks, for instance document classification and clustering (Doucet and Lehtonen, 2007; Ait Elhadj et al., 2012), converting a document in a format in which the structure is important (Marinai et al., 2010) or navigating through electronic documents(Déjean and Meunier, 2005). A bigger motivation lies in the fact that having a proper Table of Contents is a way to handle documents like a structured set of paragraphs rather than a simple sequence of strings or sentences where the logical structure disappeared. This logical structure often happens with generic OCR tools whose output is simply a character string where the only structural aspect that has been kept is the pages and other structural properties have disappeared during the automatic processing (Lecluze and Lejeune, 2014). Two different subtasks lie behind ToC extraction : (i) extracting a logical structure that has been explicitly marked in a ToC and map it to the titles present in the document, and (ii) creating a ToC when no one is present in the document by spotting the titles and their hierarchy. In both cases, systems need to be able to perform Title Detection in order to get candidates to populate the ToC. Detecting titles has also an interest on its own since it has also been used as a feature for different tasks like text classification (Lejeune et al., 2013), Terminology Acquisition (Daille et al., 2016) or Keyphrase Extraction (Florescu and Caragea, 2017). The paper is organized as follows. In Section 2 we quickly present the datasets that were given for this edition of FinTOC and in Section 3 we describe our method for both tasks. the method we designed. In Section 4 we present a discussion about our results, and we draw some perspectives for future work.

2 Data

The training set and test set of the shared tasks are composed of financial prospectuses written in French and English. The documents are distributed as native PDF documents.

The structure of the prospectuses is not standardized, which helps to have a real-world case where there is variation in the data. However, we expect the presence of some particular sections and an overall structure which will not vary too much. The format and layout varies greatly from one document to another and is often complex, with tables, nested lists of numbered and bulleted items, framed content, graphs, columns. The majority of prospectuses are published without a table of content (ToC), which means you can not rely on a ToC detection and parsing module to achieve the tasks. All this makes the challenge interesting.

3 Method

The experiment is conducted on native PDF documents. In line with the work presented in FinSBD-2 task by (Giguet and Lejeune, 2021), we choose to implement an end-to-end pipeline from the PDF file

itself to a fully structured document. This approach allows to control the entire process. Titles and Table of Contents that we generate for the shared tasks are derivative outputs of the system.

3.1 Document Preprocessing

The document content is extracted using the `pdf2xml` command (Déjean, 2007). Three useful types of content are extracted from the document: text, vectorial shapes, and images.

Text Preprocessing

`Pdf2xml` introduces the concepts of token, line and block, as three computational text units. We choose to only rely on the "token" unit. In practice, most output tokens correspond to words or numbers but they can also correspond to a concatenation of several interpretable units or to a breakdown of an interpretable unit, depending on character spacing. We choose to redefine our own "line" unit in order to better control the coherence of our hierarchy of graphical units. We abandon the concept of "block" whose empirical foundations are too weak.

Vectorial Shapes Preprocessing

Using `pdf2xml` allows to rely on vectorial information during document analysis. Text background, framed content, underline text, table grid are crucial information that contributes to sense making. They simplify the reader's task, and contribute in a positive way to automatic document analysis.

Most vectorial shapes are basic closed path, mostly rectangles. Graphical lines or graphical points do not exist: lines as well as points are rectangles interpreted by the cognitive skills of the reader as lines or points. In order to use vectorial information in document analysis, we implemented a preprocessing stage that builds composite vectorial shapes and interprets them as background colors or borders. This preprocessing component returns shapes that are used by our system to detect framed content, table grids, and text background. It improves the detection of titles which are presented as framed text and it avoids considering table headers as titles.

Images Preprocessing

`Pdf2xml` extracts images from the pdf. They may be used in different context such as logos in the title page, figures in the document body. An other interesting feature lies in the fact that certain character symbols are serialized as images, in particular specific item bullets such as arrows or checkboxes. They are indistinguishable from a standard symbol character by the human eye.

We choose to handle images as traditional symbol characters, so that they can be exploited by the structuration process, in particular by the list identification module. Identical images are grouped, and a virtual token containing a fake character glyph is created. The bounding box attributes are associated to the token and a fake font name is set. These virtual tokens are inserted at the right location by the line builder module thanks to the character x-y coordinates. This technique significantly improves the detection of list items and, as a consequence, the recognition of the global document structure.

3.2 Document Structure Parsing

Page Layout Analysis

Page Layout Analysis (PLA) aims at recognizing and labeling content areas in a page, e.g., text regions, tables, figures, lists, headers, footers. It is the subject of abundant research and articles (Antonacopoulos et al., 2009).

While PLA is often achieved at page scope and aims at bounding content regions, we have taken a model-driven approach at document scope. We try to directly infer Page Layout Models from the whole document and we then try to instantiate them on pages.

Our Page Layout Model (PLM) is hierarchical and contains 2 positions at top-level: the *margin area* and the *main content area*. The *margin area* contains two particular position, the *header area* located at the top, and the *footer area* located at the bottom. *Aside areas* may contain particular data such as vertically-oriented text. The *main content area* contains *column areas* containing text, figures or tables. *Floating areas* are defined to receive content external to column area, such as large figures, tables or framed texts.

The positions that we try to fill at document scope are header, footer and main columns. First, pages are grouped depending on their size and orientation (i.e., portrait or landscape). Then header area and footer area are detected. Column areas are in the model but due to time constraints, the detection module is not fully implemented in this prototype yet.

Detecting Header and Footer Areas

Header and footer area boundaries are computed from the repetition of similar tokens located at simi-

lar positions at the top and at the bottom of contiguous pages (Déjean and Meunier, 2006). We take into account possible odd and even page layouts. The detection is done on the first twenty pages of the document. While this number is arbitrary, we consider it is enough to make reliable decisions in case of odd and even layouts.

A special process detects page numbering and computes the shift between the PDF page numbering and the document page numbering. Page numbering is computed from the repetition of tokens containing decimals and located at similar positions at the top or at the bottom of contiguous pages. These tokens are taken into account when computing header and footer boundaries.

Detecting the Table of Contents

The TOC is located in the first pages of the document. It can spread over a limited number of contiguous pages. One formal property is common to all TOCs: the page numbers are right-aligned and form an increasing sequence of integers.

These characteristics are fully exploited in the core of our TOC identification process: we consider the pages of the first third of the document as a search space. Then, we select the first right-aligned sequence of lines ending by an integer and that may spread over contiguous pages.

Linking TOC Entries and Headers

Linking Table of Content Entries to main content is one of the most important process when structuring a document (Déjean and Meunier, 2010). Computing successfully such relations demonstrates the reliability of header detection and permits to set hyperlinks from toc entries to document headers.

Once TOC is detected, each TOC Entry is linked to its corresponding page number in the document. This page number is converted to the PDF page number thanks to the page shift (see section 3.2). Then header is searched in the related PDF page. When found, the corresponding line is categorized as header.

Table Detection

Table detection to exclude table content from the main text stream. It allows to exclude tables when searching for list items, sentences or titles.

The table detection module analyzes the PDF vectorial shapes. Our algorithm builds table grids from adjacent framed table cells. The framed table cells are built from vectorial shapes that may represent cell borders. The table grid is defined by the graph of adjacent framed table cells.

Unordered List Structure Induction

Unordered lists are also called *bulleted lists* since the list items are supposed to be marked with bullets. Unordered lists may spread over multiple pages.

Unordered list items are searched at page scope. The typographical symbols (glyphs) used to introduce items are not predefined. We infer the symbol by identifying multiple left-aligned lines introduced by the same single-character token. In this way, the algorithm captures various bullet symbols such as squares, white bullets... Alphabetical or decimal characters are rejected as possible bullet style type. Images of character symbols are transparently handled thanks to virtual tokens created during the preprocessing stage.

The aim of the algorithm is to identify PDF lines which corresponds to new bulleted list item (i.e., list item leading lines). The objective is not to bound list items which cover multiple lines. Indeed, the end of list items are computed while computing paragraph structures: a list item ends when the next list item starts (i.e., same bullet symbol, same indentation) or when less indented text objects starts.

Ordered List Structure Induction in PDF Documents

Ordered list items are searched at document scope. We first select numbered lines thanks to a set of regular expressions, and we analyse each numbering prefix as a tuple $\langle P, S, I, C \rangle$ where P refers to the numbering pattern (string), S refers to the numbering style type (single character), I refers to the numbering count written in numbering style type (single character), and C refers to the decimal value of the numbering count (integer).

The numbering style types are defined as follows: Decimal (D), Lower-Latin (L), Upper-Latin (M), Lower-Greek (G) Upper-Greek (H), Lower-Roman (R), Upper-Roman (S), Lower-Latin OR Lower-Roman (?), Upper-Latin OR Upper-Roman (!).

To illustrate, the line “A.2.c) My Header" is analysed as $\langle$ A.2.L), L, c, 3 $\rangle$.

Lines are grouped in clusters sharing the same numbering pattern. A disambiguation process assigns an unambiguous style type to ambiguous lines. The underlying strategy is to complement unambiguous yet incomplete series in order to build coherent, ordered series.

Paragraph Structure Induction

The aim of paragraph structure induction is to infer paragraph models that are later used to detect paragraph instances. The underlying idea to automatically infer the settings of paragraph styles.

Paragraphs are complex objects: a canonical paragraph is made of a leading line, multiple body lines and a trailing line. The leading line can have positive or negative indentation. In context, paragraphs may be visually separated from other objects thanks to above spacing and below spacing.

In order to build paragraph models, we first identify reliable paragraph bodies: sequences of three or more lines with same line spacing and compatible left and right coordinates. Then, leading lines and trailing lines are identified considering same line spacing, compatible left and/or right coordinates (to detect left and right alignments), same style. Paragraph lines are categorized as follows: L for leading line, B for body lines, T for trailing line. Header lines are categorized H. Other lines are categorized as ? for undefined.

In order to fill paragraph models, paragraph settings are derived from the reliable paragraphs that are detected. When derived, leading lines of unordered and ordered list items are considered to create list item models.

Once paragraph models and list item models are built, the models are used to detect less reliable paragraphs and list items (i.e., containing less than three body lines). Compatible models are applied and lines are categorized L, B (if exists) or T (if exists). Remaining undefined lines are categorized considering line-spacing.

4 Results and discussion

The document-wise approach we presented was evaluated on both tasks of FinTOC 2021 : *Title Detection* and *Table of Content extraction*.

In table 1 and 2 we present the results we obtained respetively on the *Title Detection* and the *ToC Extraction* tasks. The results we obtain shows an overall good precision on both languages but a quite low recall. These results are encouraging when we consider all the structures that we want to handle, though our system gives too much False Negatives for title detection and consequently for ToC extraction.

The rationale of our method is to have an end-to-end pipeline from the PDF file itself to a fully structured document, it seems that this is a good way to avoid false positives. The steps comprised in our method (layout analysis, header/footer detection, list detection and paragraph induction) seem to act as filters to avoid an over structuration of the document. The consequence is that the results we obtain are very encouraging in terms of precision, but the recall remains quite low. Still, we believe there is a great interest in representing a fairly unusual but ambitious way to deal with the document structure as a whole.

Table 1: Results for Title Detection

	Prec	Rec	F1
fr	0.842	0.485	0.606
en	0.913	0.338	0.465

Table 2: Results for ToC Extraction

	Prec	Rec	F1
fr	49.7	28.6	35.8
en	52.8	18.6	25.1

References

Ali Ait Elhadj, Mohand Boughanem, Mohamed Mezghiche, and Fatiha Souam. 2012. Using structural similarity for clustering XML documents. *Knowledge and Information Systems*, 32(1):109–139.

Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. 2009. A realistic dataset for performance evaluation of document layout analysis. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 296–300.

Béatrice Daille, Evelyne Jacquey, Gaël Lejeune, Luis Felipe Melo, and Yannick Toussaint. 2016. Ambiguity Diagnosis for Terms in Digital Humanities. In *Language Resources and Evaluation Conference*, Portorož, Slovenia.

Hervé Déjean. 2007. *pdf2xml open source software*. Last access on July 31, 2019.

Hervé Déjean and Jean-Luc Meunier. 2005. Structuring documents according to their table of contents. In *Proceedings of the 2005 ACM symposium on Document engineering*, pages 2–9.

Hervé Déjean and Jean-Luc Meunier. 2006. A system for converting pdf documents into structured xml format. In *Document Analysis Systems VII*, pages 129–140, Berlin, Heidelberg. Springer Berlin Heidelberg.

Hervé Déjean and Jean-Luc Meunier. 2010. Reflections on the inex structure extraction competition.

In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS '10, page 301–308, New York, NY, USA. Association for Computing Machinery.

Antoine Doucet and Miro Lehtonen. 2007. Unsupervised classification of text-centric xml document collections. In *Comparative Evaluation of XML Information Retrieval Systems, Fifth International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2006*, volume 4518 of *Lecture Notes in Computer Science*, pages 497–509. Springer.

Ismail El Maarouf, Juyeon Kang, Abderrahim Aitazzi, Sandra Bellato, Mei Gan, and Mahmoud El-Haj. 2021. The Financial Document Structure Extraction Shared Task (FinToc 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

Corina Florescu and Cornelia Caragea. 2017. PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115, Vancouver, Canada. Association for Computational Linguistics.

Emmanuel Giguet and Gaël Lejeune. 2021. Daniel at the FinSBD-2 task : Extracting Lists and Sentences from PDF Documents: a model-driven end-to-end approach to PDF document analysis. In *Second Workshop on Financial Technology and Natural Language Processing in conjunction with IJCAI-PRICAI 2020*, Proceedings of the Second Workshop on Financial Technology and Natural Language Processing, pages 67–74, Kyoto, Japan.

Charlotte Lecluze and Gaël Lejeune. 2014. Deft2014, automatic analysis of literary and scientific texts in french (deft 2014, analyse automatique de textes littéraires et scientifiques en langue française)[in french]. In *TALN-RECITAL 2014 Workshop DEFT 2014: DÉfi Fouille de Textes (DEFT 2014 Workshop: Text Mining Challenge)*, pages 11–19.

Gaël Lejeune, Romain Brixtel, Charlotte Lecluze, Antoine Doucet, and Nadine Lucas. 2013. Added-value of automatic multilingual text analysis for epidemic surveillance. In *Artificial Intelligence in Medicine (AIME)*, pages 284–294.

Simone Marinai, Emanuele Marino, and Giovanni Soda. 2010. Table of contents recognition for converting pdf documents in e-book formats. In *Proceedings of the 10th ACM symposium on Document engineering*, pages 73–76.

Summarization of financial documents with TF-IDF weighting of multi-word terms

Sophie Krimberg
Shamoon College of Engineering (SCE)
Beer-Sheva
Israel
krsofi@gmail.com

Natalia Vanetik
Shamoon College of Engineering (SCE)
Beer-Sheva
Israel
natalyav@sce.ac.il

Marina Litvak
Shamoon College of Engineering (SCE)
Beer-Sheva
Israel
marinal@ac.sce.ac.il

Abstract

Financial documents, such as corporate annual reports, are usually very long and may consist of more than 100 pages. Every report is divided into thematic sections or statements that have an inner structure and include special financial terms and numbers. This paper describes an approach for summarizing financial documents based on a Bag-of-Words (BOW) document representation. The suggested solution first calculates the Term Frequency-Inverse Document Frequency (TF-IDF) weights for all single-word and multi-word expressions in the corpus, then finds the sequence of words with a maximum total weight in each document. The solution is designed to meet the requirements of the Financial Narrative Summarization (FNS 2021) shared task and has been tested on FNS 2021 dataset shared-task dataset.

1 Introduction

Corporate annual reports and financial statements are challenging to summarize due to their length, format, structure, and contents. An annual report is a document of tens and often hundreds of pages. Sometimes annual report includes a table of contents, but there are a lot of reports that do not. Usually, reports have several thematic sections, but the order, the quantity, and the structure of sections differ from one report to another. Financial documents use specialized financial terms. Additionally, every company that publishes a report operates within its field, and this field's lexicon can appear in the report and be an important part of it, while all of the other documents in the corpus do not use that lexicon at all.

The 1st Joint Workshop on financial Narrative Processing and MultiLing financial Summarisation (FNP-FNS 2020) (El-Haj et al., 2020a) ran the financial narrative summarisation (FNS) task, which resulted in the first large-scale experimental results and state-of-the-art summarization methods applied to financial data. The task focused on annual reports produced by UK firms listed on the London Stock Exchange. Because companies usually produce glossy brochures with a much looser structure, this makes automatic summarization of such reports a challenging task. A total number of 9 teams participated in the FNS 2020 shared task with a total of 24 system submissions. All teams were ranked by several ROUGE-based measures and compared to the four topline and baseline summarizers—MUSE (Litvak et al., 2010), POLY (Litvak and Vanetik, 2013), TextRank (Mihalcea and Tarau, 2004) and LexRank (Erkan and Radev, 2004)—in (El-Haj et al., 2020b).

The participating systems used a variety of techniques and methods ranging from rule based extraction methods (Litvak et al., 2020; Vhatkar et al., 2020; Arora and Radhakrishnan, 2020; Azzi and Kang, 2020) to traditional machine learning methods (Suarez et al., 2020; Vhatkar et al., 2020; Arora and Radhakrishnan, 2020) and high performing deep learning models (Agarwal et al., 2020; Singh, 2020; La Quatra and Cagliero, 2020; Vhatkar et al., 2020; Arora and Radhakrishnan, 2020; Azzi and Kang, 2020; Zheng et al., 2020). The text representation was also very diverse among the participating systems—very basic morphological and structure features (Li et al., 2020; Suarez et al., 2020), syntactic features (Vhatkar et al., 2020), and semantic vectors using word embeddings (Agarwal et al., 2020; Suarez et al., 2020) were applied. In addition, some teams (Litvak et al., 2020; Zheng et al., 2020) investigated the hierarchical structure of reports. Different ranking techniques, such as Determinantal Point Processes (Li et al., 2020), a combination of Pointer Network and Text-to-text transfer Transformer algorithms (Singh, 2020) were used for extractive approaches, together with deep language models (La Quatra and Cagliero, 2020; Zheng et al., 2020), hierarchical summarization under different discourse topics (Litvak et al., 2020), and ensem-

ble based models (Arora and Radhakrishnan, 2020) have also been reported. The main challenge of this task, as reported by its participants, was the average length of a document, which made the training process extremely inefficient. In addition, participants argued that extracting text and then structure from PDF files with numerous tables, charts, and numerical data resulted in a lot of noise.

This year FNS-2021 (El-Haj et al., 2021) shared task asks to provide summaries of annual company reports. The dataset is supplied with 2-4 gold standard summaries per document. These gold standard summaries are complete sections of the original document selected by human financial experts as the most important sections of the documents.

Term Frequency-Inverse Document Frequency (TF-IDF) (Sammut and Webb, 2010) is a term weighting scheme, commonly used for making relevant decisions and discover the strength of the relationship of words with the document they appear in (Ramos et al., 2003).

In this paper, we propose a TF-IDF weighing method that helps to determine the most successful candidate for the extractive summary among the possible continuous document parts of the required length. This approach is based on the fact that all of the gold standard summaries in the data provided by the organizers are in fact sections of the original documents that did not undergo any rewriting. We use the TF-IDF score to detect the most important sequence of up to 1000 words in a document. While the classic implementation is based on the evaluation of single words, we calculate the TF-IDF values for single-word and multi-word terms, mainly to recognize the specific financial terminology.

2 The method

On purpose to find sequences of up to 1000 most important words in every document of the corpus, we do the next steps:

1. define the value of the maximal length of multi-word term (See Section 2.1);

2. find all the existing multi-word terms in the corpus and calculate the TF-IDF score for every one of them;

3. compute summarized TF-IDF scores for all continuous sequences with 1000 words in a document;

4. select the highest-ranking sequence as a summary for the specific document.

The pipeline of our approach is depicted in Figure 1.

2.1 Multi-word terms

Because classic TF-IDF is computed for single-word terms only, and we want to extend it to multi-word terms, we introduce a parameter that defines a maximal number of words in such a term. The aim of evaluating multi-word terms is to recognize the set of important document-specific phrases from their TF-IDF weights.

2.2 Preprocessing

The original files are preprocessed using Python *nltk* library (Bird et al., 2009). The preprocessing includes text splitting, tokenization, special symbols removal, removing of phone numbers, emails etc. Stopwords are not removed, but we use a custom stopword list containing the words ['and', 'the', 'is', 'are', ' this','at', 'of', 'to', 'in', 'on', 'for', 'or','a', 'an', 'as', 'page', 'by', 'with', 'our', 'we', 'that', 'may']. All multi-word terms that contain stopwords only get zero TF-IDF values.

2.3 Creating the TF-IDF matrix of a document

When the maximal number of words in term is defined (denoted by TL), the system finds in the corpus all the existing word sequences of length 1 to TL and calculates the TF-IDF score for every one of them. The following steps are performed:

1. Generate multi-word terms of length 1 to TL as follows. For a document with DL words, there are DL single-word terms, $DL-1$ two-word terms, and so on. Finally, we have $DL-TL+1$ terms with TL words.

2. Let T be a multi-word term with $TL := |T|$ words in a document D_i having $DL_i := |D_i|$ words in total, and let T appear DR_i times in the document D_i. Term frequency of T in D_i is calculated as

$$TF(T, D_i) = \frac{DR_i}{DL_i - TL + 1} \quad (1)$$

3. Let T appear in CR documents in the corpus of size N. Then the IDF score of T is:

$$IDF(T) = \log \frac{N}{CR} \quad (2)$$

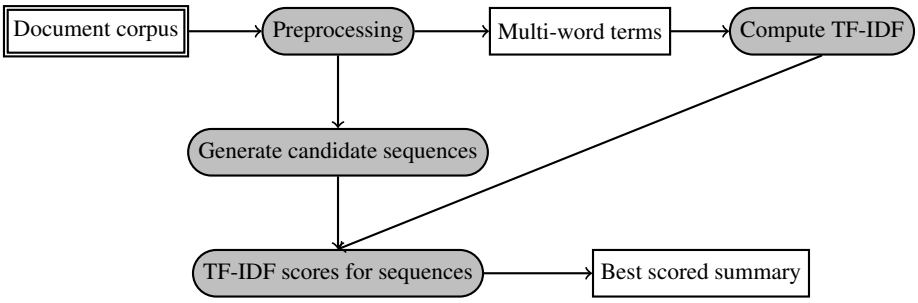


Figure 1: Pipeline of our approach

4. Finally, the TF-IDF score of a term T in document D_i is:

$$TF\text{-}IDF(T, D_i) = TF(T, D_i) \cdot IDF(T) \quad (3)$$

2.4 Most important sequence in a document

In every document D_i, we find all the sequences of up to 1000 words (there are $DL_i - 999$ such sequences in a document with more than a 1000 words), and calculate the sum of TF-IDF values for all the multi-word terms of any length that appear in every such sequence S:

$$TF\text{-}IDF(S) = \sum_{k=1}^{TL} \sum_{T \in S, |T|=k} TF\text{-}IDF(T, D_i) \quad (4)$$

We rank the sequences by their TF-IDF scores and select the highest-ranking sequence as our summary. Implementation of calculating the totals for multiple sequences is based on the idea that given total of sequence $W_1 W_2 \ldots W_n$ we can calculate the total of sequence $W_2 W_3 \ldots W_{n+1}$ by subtracting the values of terms that can include W_1 and adding the values of terms that can include W_{n+1} (according to the maximum number of words in a term). This approach allows the system to calculate and compare thousands of such sequences in each document in a very short time.

3 Experiments

FNS 2021 Shared Task provides a dataset that contains companies' annual reports and 3-4 gold standard summaries for each report. The gold standard summaries were created by extracting whole sections (one or more) from the original document, according to a human financial expert's decision. The selected summaries sections are considered by the experts as most important and informative. Table 1 describes the dataset contents. The training dataset, which contains 3,000 reports and 9,873 gold summaries, was randomly divided by us into 3 groups of 1,000 documents each to facilitate the tf-idf computation. Furthermore, every one of those three groups was divided into two subgroups of 500 documents each. We three variants of our system using values 1, 2, and 3 as the multi-word term size TL.

3.1 Tools and runtime environment

For preprocessing such as sentence splitting and tokenization we used *nltk* package (Bird et al., 2009); We have used the MUSEEC tool (Litvak et al., 2016) to compute MUSE summaries to be used as a baseline a with 1000-word limits, respectively. We used the ROUGE 2.05 java package (Ganesan, 2018).

3.2 Methods and baselines

For evaluation of the results of this approach, we applied it on the validation part of the FNS 2021 shared task dataset and compared the results to the results of Muse (Litvak et al., 2016) on the same set of documents. As an additional reference, we use the results of a trivial TOP-K baseline that includes the first 1000 words of a document. The results are reported in table 2, the results of our approach appear as **TFIDF-SUM-N**, where the number N is the maximal number of words in a term. [1] Experiments were performed on Google Colab with the default configuration.

3.3 Evaluation results

Four ROUGE (Lin, 2004) metrics—ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU4 were applied on the validation set. Table 2 shows the results, with recall, precision, and F-measure for each metric. It can be seen that as the maximum number of words in the term increases, the results

[1] The results on the test set, provided by the FNS organizers, can be seen in the Appendix and on the leaderboard: https://www.lancaster.ac.uk/staff/elhaj/docs/fns2021_results.pdf.

dataset	# documents	# gold summaries	avg sentences	avg words	avg characters
Train	3,000	9,873	2,700	58,838	291,014
Validation	363	1,250	3,786	82,906	416,040
Test	500	NA	3,743	82,676	412,974

Table 1: FNS 2021 dataset statistics.

System	R1 R	R1 P	R1 F	R2 R	R2 P	R2 F
TOP-K	0.266	0.241	0.221	0.040	0.038	0.034
MUSE	0.261	0.297	0.243	0.042	0.052	0.040
TFIDF-SUM-1	0.353	0.317	0.322	0.153	0.110	0.121
TFIDF-SUM-2	0.450	0.396	0.410	0.244	0.156	0.183
TFIDF-SUM-3	**0.477**	**0.415**	**0.433**	**0.279**	**0.177**	**0.209**
System	**RL R**	**RL P**	**RL F**	**RSU4 R**	**RSU4 P**	**RSU4 F**
TOP-K	0.264	0.239	0.220	0.081	0.076	0.069
MUSE	0.255	0.292	0.238	0.084	0.100	0.079
TFIDF-SUM-1	0.263	0.279	0.258	0.218	0.141	0.164
TFIDF-SUM-2	0.374	0.332	0.343	0.312	0.188	0.227
TFIDF-SUM-3	**0.411**	**0.362**	**0.374**	**0.344**	**0.207**	**0.250**

Table 2: ROUGE results for FNS-2021 validation set.

System	R1 F	R2 F	RL F	RSU4 F
BASE	0.45	0.24	0.42	0.27
MUSE	0.50	0.38	0.52	0.43
TFIDF-SUM-1	0.33	0.12	0.27	0.17
LexRank	0.31	0.12	0.27	0.16

Table 3: ROUGE results for FNS-2021 test set.

improve, but even with a single term (TFIDF-SUM-1), the system outperforms the baselines. Due to time constraints, only the TFIDF-SUM-1 system was submitted to the FNS-2021 shared task competition and it appears in its results as an SCE-3 system.

It is important to note that increasing the maximum number of words in a multi-word term increases their amount drastically, and the memory usage increases as well. Therefore running the system with 3-word terms on Colab required us to divide the dataset into two parts and to compute the tf-idf scores for them separately. This approach reduces the precision of tf-idf, but because every run is still performed on almost 200 documents, we can see from the resulting ROUGE scores that an additional term compensates for the lack of tf-idf precision. Table 3 shows the results for the same ROUGE metrics, F-measure, obtained on the test set (provided by the FNS organizers).

3.4 Performance

Our system works very fast while producing hundreds of summaries in several minutes. For example, for 363 annual reports from Validation dataset, execution on Google Colab with default configuration was completed in 2 minutes 54 seconds with TFIDF-SUM-1, 6 minutes 22 seconds with TFIDF-SUM-2 and 10 minutes 50 seconds with TFIDF-SUM-3. Times may differ as the performance of Colab itself changes. But as the maximum number of words in a multi-word term increases, more possible terms exist and more memory is required. Using multi-word terms with more than three words resulted in an out-of-memory error.

4 Conclusions and Future Work

This paper introduces a method for summarization of financial documents. The method implements the TF-IDF technique with optimization for multi-word terms. The system is fast, simple, and outperforms baselines. The evaluation results show that (1) evaluating multi-word terms vs single-word ones improves the quality of the summaries and (2) that extracting continuous sequence from the document provides the results.

Future work may include modifying the current method to extract the most important sentences instead of extracting the whole sequence. In addition, combining the multi-term TF-IDF weighting scheme with machine learning algorithms and FinBERT (Yang et al., 2020) embedding may provide interesting results.

References

Raksha Agarwal, Ishaan Verma, and Niladri Chatterjee. 2020. Langresearchlab_nc at fincausal 2020, task 1: A knowledge induced neural net for causality detection. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 33–39.

Piyush Arora and Priya Radhakrishnan. 2020. Amex ai-labs: An investigative study on extractive summarization of financial documents. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 137–142.

Abderrahim Ait Azzi and Juyeon Kang. 2020. Extractive summarization system for annual reports. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 143–147.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Mahmoud El-Haj, Vasiliki Athanasakou, Sira Ferradans, Catherine Salzedo, Ans Elhag, Houda Bouamor, Marina Litvak, Paul Rayson, George Giannakopoulos, and Nikiforos Pittaras. 2020a. Proceedings of the 1st joint workshop on financial narrative processing and multiling financial summarisation. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*.

Mahmoud El-Haj, Marina Litvak, Nikiforos Pittaras, George Giannakopoulos, et al. 2020b. The financial narrative summarisation shared task (fns 2020). In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 1–12.

Mahmoud El-Haj, Nadhem Zmandar, Paul Rayson, Ahmed AbuRa'ed, Marina Litvak, Nikiforos Pittaras, and George Giannakopoulos. 2021. The Financial Narrative Summarisation Shared Task (FNS 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Kavita Ganesan. 2018. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. *arXiv preprint arXiv:1803.01937*.

Moreno La Quatra and Luca Cagliero. 2020. End-to-end training for financial report summarization. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 118–123.

Lei Li, Yafei Jiang, and Yinan Liu. 2020. Extractive financial narrative summarisation based on dpps. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 100–104.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Marina Litvak, Mark Last, and Menahem Friedman. 2010. A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 927–936.

Marina Litvak and Natalia Vanetik. 2013. Mining the gaps: Towards polynomial summarization. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 655–660.

Marina Litvak, Natalia Vanetik, Mark Last, and Elena Churkin. 2016. Museec: A multilingual text summarization tool. In *Proceedings of ACL-2016 System Demonstrations*, pages 73–78.

Marina Litvak, Natalia Vanetik, and Zvi Puchinsky. 2020. Sce-summary at the fns 2020 shared task. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 124–129.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242(1), pages 29–48. Citeseer.

Claude Sammut and Geoffrey I. Webb, editors. 2010. *TF–IDF*, pages 986–987. Springer US, Boston, MA.

Abhishek Singh. 2020. Point-5: Pointer network and t-5 based financial narrative summarisation. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 105–111.

Jaime Baldeon Suarez, Paloma Martínez, and Jose Luis Martínez. 2020. Combining financial word embeddings and knowledge-based features for financial text summarization uc3m-mc system at fns-2020. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 112–117.

Amit Vhatkar, Pushpak Bhattacharyya, and Kavi Arya. 2020. Knowledge graph and deep neural network for extractive text summarization by utilizing triples. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 130–136.

Yi Yang, Mark Christopher Siy Uy, and Allen Huang. 2020. Finbert: A pretrained language model for financial communications. *CoRR*, abs/2006.08097.

Siyan Zheng, Anneliese Lu, and Claire Cardie. 2020. Sumsum@ fns-2020 shared task. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 148–152.

ISPRAS@FinTOC-2021 Shared Task: Two-stage TOC generation model

Ilya Kozlov, Oksana Belyaeva, Anastasiya Bogatenkova and Andrew Perminov

Abstract

We propose a two-stage approach for TOC generation from financial documents. This work connected with participation in FinTOC-2021 Shared Task: "Financial Document Structure Extraction". The competition contains two subtasks: title detection and TOC generation. Our model consists of two classifiers: the first binary classifier separates title lines from non-title, the second one determines the title level. In the title detection task, we got 0.813 and 0.787 F1 measure, in the TOC generation task we got 37.9 and 42.1 the harmonic mean between Inex F1 score and Inex level accuracy for English and French documents respectively. With these results, our approach took third place among all submissions. As a team, we took second place in the competition in all categories.

1 Introduction

Currently, electronic documents have become widespread. A large number of documents are presented in a PDF format, but only a few of them contain an automatic table of contents (TOC). However, there may be the need for a quick search of information and it may be a problem for large documents. One example is financial documents, which can be over 100 pages long. Financial documents contain a lot of important information and can have different appearances and structures. The task of automatically extracting the table of contents from financial documents seems to be relevant and its solution is not obvious.

FinTOC-2021 (El Maarouf et al., 2021) offers to solve the problem of extracting structure from financial documents in two languages: English and French. The results of solving two subtasks are evaluated:

- **Title detection (TD)** - selection from all lines of the document only those that should be included in the table of contents;
- **Table of contents (TOC) generation** - identification nesting depths of selected titles.

The competition is held for the third time. Similar tasks were solved at FinTOC-2019 (Juge et al., 2019) and FinTOC-2020 (Bentabet et al., 2020); in 2020, documents in French were added.

In FinTOC-2019, the best solution (Tian and Peng, 2019) for title detection is based on the LSTM with augmentation and attention. The best solution (Giguet and Lejeune, 2019) for the TOC generation task relies on the decision tree classifier DT 10 and TOC page detection.

In FinTOC-2020, the best solution (Hercig and Kral, 2020) for title detection (French) was obtained with the maximum entropy classifier. For title detection in English documents (Premi et al., 2020) LSTM, CharCNN, and a fully connected network with some handcrafted features were used. The best approach for TOC generation (Kosmajac et al., 2020) consisted in extracting linguistic and structural information and using the Random Forest classifier.

In this paper, we describe our solution to the shared task. This work is a continuation of (Bogatenkova et al., 2020). We make a list of features for each document line and use two classifiers for the consequent solution of both title detection and TOC generation tasks.

The paper is organized as follows. We describe in detail the given dataset for the competitions in Section 2. We present our approach to solving the task in Section 3. Results and a discussion are given in Section 4 and 5 respectively. Section 6 contains a conclusion about our work.

2 Dataset

2.1 Train dataset

The training data of the FinTOC-2021 shared task consists of 49 English and 47 French financial PDF documents with a textual layer. The documents

	English	French
Number of documents	49	47
Mean number of pages	64	30
Number of extracted lines	191373	79071
Number of TOC	43	4
Mean number of titles	181	142
Max title depth	9	6

Table 1: Training dataset statistics

	English	French
Number of documents	10	10
Mean number of pages	66	26
Number of extracted lines	42100	13027
Number of TOC	9	0

Table 2: Test dataset statistics

are very heterogeneous, both groups contain documents with and without TOC. Moreover, not only existing TOC should be included in the result, but also smaller titles of each document.

The main information about the training dataset is in the Table 1. The mean of pages' numbers is 64 and 30 for English and French documents respectfully. But the size of documents varies greatly from 3 to 285 pages. The dataset contains one-column, two-column, and even three-column documents. At the same time, a different number of columns may occur within one document. Moreover, documents are different in their appearance (e. g. the appearance of titles or existing TOC) and logical structure. So, there is no way to extract a complete TOC using regular expressions and we need to use machine learning techniques.

There is a set of annotations for each document. Annotations include only titles with the text and the depth for each title. The number of titles and maximum title depth are also different for each document. The number of titles varies from 20 to 1004 and from 33 to 527 for documents in English and French, respectively. Maximum title depth is from 3 to 9 for English documents while it equals from 4 to 6 for French documents. Thus, a sample of very different documents is presented at the competition.

2.2 Test dataset

The test dataset is similar to the training dataset. It contains 10 documents in each English and French set. The documents are also very diverse, none of the French documents contain a table of contents. More statistics are shown in the Table 2.

3 Proposed approach

We proposed the 2-stage method (see figure 1) for solving the both tasks TD and TOC generation. Each stage includes classification using the XGBoost classifier. In the first stage, the binary classifier classifies each line as title or non-title. Thus, the first stage is a process of filtering all lines of the document. In the second stage of our method hierarchy levels are found for each filtered title from the first stage.

Text and metadata extraction. Since the input PDF documents have a textual layer, we extracted text, bold and italic font, and colors of the text with help of PDFMiner (Yusuke Shinyama, 2019). PDFMiner has different layout reading modes. To read the entire document we have chosen the universal layout mode for multi-column documents with parameters *LAParams(line_margin=1.5, line_overlap=0.5, boxes_flow=0.5, word_margin=0.1, detect_vertical=False)* . Thus the list of lines with text and metadata is extracted from the input documents. To obtain lines with labels we matched the provided labelled titles and the extracted lines using a Levenshtein distance with 0.8 threshold.

As preprocessing, we remove footers and headers from a document using the method (Lin, 2003). It helps to improve the quality of the binary classifier and the TOC extration module. The problem with headers and footers is that they are similar to titles and can predicted as the element of TOC.

Existing TOC extraction. As additional information, we separately extract a table of content (TOC) for each document. We look for the keywords of the TOC heading in the document (for example, "Table of contents", "CONTENT") as the beginning of TOC. Then, we detect the TOC's body using regular expressions.

Most tables of contents in the given documents are one-column regardless of the number of columns in the whole document. The TOC extraction module requires PDFMiner to be run in the single column mode because the TOC text may be read automatically as a multi-column. In this case, PDFMiner should be run with the parameters *LAParams(line_margin=3.0, line_overlap=0.1, boxes_flow=-1, word_margin=1.5, char_margin=100.0, detect_vertical=False)*.

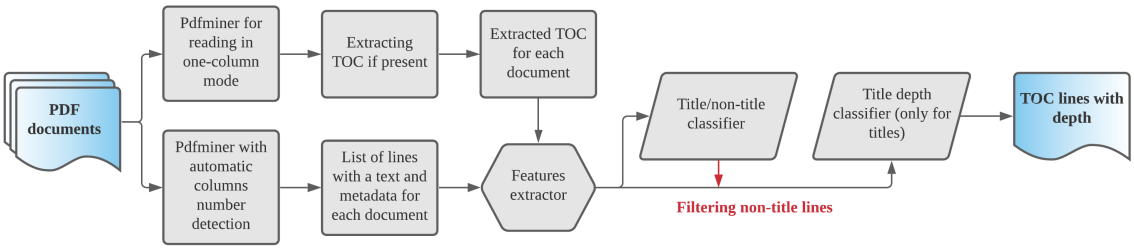


Figure 1: Full pipeline description

Features extraction. The list of extracted lines and extracted TOCs (if present) are processed to obtain a vector of features for each extracted line. We formed a vector from 184 features some of which are enlisted below:

- **visual features**: bold or italic text, indentation, spacing between lines, normalized font size, text color;
- **features from letter statistics**: the percentage of letters, capital letters, numbers, the number of words in line, normalized line's length;
- **features from regular expressions for lines beginning**: indicators of matching some regular expressions for list items;
- **features from regular expressions for lines end**: indicators of ending with a dot, colon, semicolon, comma;
- **features connected with lines depth**: the level of numbering for list with dots (like 1.1.1), relative font size and indentation;
- **TOC features**: indicator of being in the existing TOC (we extracted it automatically), indicator of being the TOC header;
- **other features**: normalized page number and line number;
- **context features**: the same features for 3 previous and 3 next lines.

Training process and experiments. For both tasks we experimented with two models: one-stage and two-stage classifiers. Under the one-stage model we call the model without the first stage (without the binary classifier). In this case the input lines are not pre-filtered. We use the XGBoost classifier in both models. The training process ran with parameters 0.1 learning rate and 100 estimators. We use 3-fold cross-validation for evaluate the results of each model. The mean results for English documents are given in the Table 3. The evaluation script is provided by the organizers (El Maarouf et al., 2021).

Model type	F1 (TD)	Inex08-F1 (TOC)
XGBoost 1-stage	0.77	50.5
XGBoost 2-stage	0.81	55.7

Table 3: The results from cross-validation on the training dataset (English)

Team run	F1 (English)	F1 (French)
Christopher Bourez2	0.830	0.818
Christopher Bourez1	0.822	0.817
ISP RAS (our)	**0.813**	**0.787**
Yseop Lab	0.728	0.639
Cilab_fintoc2	0.514	–
NovaFin	0.507	0.562
Daniel	0.465	0.606
Cilab_fintoc1	0.456	–

Table 4: Title Detection Competition results

The two-stage model performed better than the single-stage one. Thus, we've chosen two-stage model for solving the task on the test dataset.

4 Results

The competition results on test dataset (see table 2) are presented in the table 4 (Title Detection), and tables 5, 6 (TOC generation). Our approach ranks third among 8 and 6 submitted solutions for English and French documents respectfully. As a team we took the second place in all categories.

5 Discussion

The two-stage model demonstrates high scores for both tasks. But the model has disadvantages. Pri-

Team run	Inex08-P	Inex08-R	Inex08-F1	Inex08-Title acc	Inex08-Level acc	harm mean
Christopher Bourez2	55.4	52.6	53.6	60.3	30.6	53.6
Christopher Bourez1	53.3	52	52.5	59	36.5	52.5
ISP RAS (our)	**51.1**	**45.3**	**47.6**	**55.6**	**31.5**	**37.9**
Yseop Lab	61.1	50.3	53.4	68.2	12.4	20.1
Cilab_fintoc2	30.5	18.6	22.6	38.9	31.4	26.3
NovaFin	22.2	21.5	21.2	35.7	31.4	25.3
Daniel	52.8	18.6	25.1	54.3	0	–
Cilab_fintoc1	26.6	14.4	17.6	34.2	34.8	23.4

Table 5: TOC Generation Competition on English documents

Team run	Inex08-P	Inex08-R	Inex08-F1	Inex08-Title acc	Inex08-Level acc	harm mean
Christopher Bourez2	60.8	54.3	57.3	63.5	38.7	57.3
Christopher Bourez1	60.9	54.2	57.3	63.6	39	57.3
ISP RAS (our)	**52.6**	**38.8**	**44.5**	**53.6**	**39.9**	**42.1**
NovaFin	29.7	24.7	26.7	34.6	32	29.1
Yseop Lab	46.8	28.1	34.4	47.3	16.6	22.4
Daniel	49.7	28.6	35.8	53.7	7.1	11.8

Table 6: TOC Generation Competition on French documents

marily, the model misclassifies questionable titles, the ground truth of which are interpreted differently for different documents. For example, one document has a line with some features (color, font size, style and etc.) as a title, but the equivalent line in another document is not a title. Also, we don't combine adjacent titles together as in the ground truth of the data sets.

As well, a two-stage model accuracy in the title detection task is limited by the binary classifier. If the model filters out the title lines in the first step, it will not be able to determine their depths in the second one. Therefore, the accuracy of the two-stage model will not exceed the accuracy of the binary classifier.

As a development of the work, we propose to consider more advanced and complicated models, e. g. the LSTM model. This model can give greater accuracy through the use of long-term memory. Thus, we will be able to remember the previous predictions made up to this point in the document.

6 Conclusion

We proposed the approach for automatic title detection and TOC generation for PDF financial documents with a textual layer. We extracted lines with metadata using Pdfminer and found existing TOCs using the regular expressions. This information we transform to the feature matrix with the vector of predefined features for each document line. Then we use a two-stage model for solving both title detection and TOC generation problems. First, we filter titles from all document lines using the XGBoost binary classifier. Then, we find the depths of the filtered lines using the second XGBoost classifier. The described approach does not depend on the presence of the table of contents in the document and can be used for both English and French financial documents. As a result, our team has taken second place in all categories in the FinTOC-2021 competition.

References

Najah-Imane Bentabet, Remi Juge, Ismail El Maarouf, Virginie Mouilleron, Dialekti Valsamou-Stanislawski, and Mahmoud El-Haj. 2020. The Financial Document Structure Extraction Shared Task (FinToc 2020). In *The 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation (FNP-FNS 2020)*, Barcelona, Spain.

Anastasiya Olegovna Bogatenkova, Ilya Sergeyevich Kozlov, Oksana Vladimirovna Belyaeva, and Andrey Igorevich Perminov. 2020. Logical structure extraction from scanned documents. *Proceedings*

of the Institute for System Programming of the RAS, 32(4):175–188.

Ismail El Maarouf, Juyeon Kang, Abderrahim Aitazzi, Sandra Bellato, Mei Gan, and Mahmoud El-Haj. 2021. The Financial Document Structure Extraction Shared Task (FinToc 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

Emmanuel Giguet and Gaël Lejeune. 2019. Daniel@ fintoc-2019 shared task: toc extraction and title detection. In *Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019)*, pages 63–68.

Tomáš Hercig and Pavel Kral. 2020. UWB@FinTOC-2020 shared task: Financial document title detection. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 158–162, Barcelona, Spain (Online). COLING.

Rémi Juge, Imane Bentabet, and Sira Ferradans. 2019. The fintoc-2019 shared task: Financial document structure extraction. In *Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019)*, pages 51–57.

Dijana Kosmajac, Stacey Taylor, and Mozhgan Saeidi. 2020. DNLP@FinTOC'20: Table of contents detection in financial documents. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 169–173, Barcelona, Spain (Online). COLING.

Xiaofan Lin. 2003. Header and footer extraction by page association. In *Document Recognition and Retrieval X*, volume 5010, pages 164–171. International Society for Optics and Photonics.

Dhruv Premi, Amogh Badugu, and Himanshu Sharad Bhatt. 2020. AMEX-AI-LABS: Investigating transfer learning for title detection in table of contents generation. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 153–157, Barcelona, Spain (Online). COLING.

Ke Tian and Zi Jun Peng. 2019. Finance document extraction using data augmentation and attention. In *Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019)*, pages 1–4.

Philippe Guglielmetti Pieter Marsman Yusuke Shinyama. 2019. Pdfminer.six documentation.

Not All Titles are Created Equal: Financial Document Structure Extraction Shared Task

Anubhav Gupta and **Hanna Abi Akl** and **Hugues de Mazancourt**
Yseop
{agupta, habi-akl, hdemazancourt}@yseop.com

Abstract

This paper presents a multi-modal approach to *FinTOC-2021 Shared Task*. With help of a fine-tuned `Faster-RCNN` our solution achieved a Precision score comparatively better than other participants.

1 Introduction

Heading or title is a phrase that either represents an oeuvre or demarcates a text into chapters, sections etc. It serves as a milestone that helps readers find their way through a long text. Its appearance is governed by style guides. For example, AER[1] mandates that the title of a section begins with roman numerals and that of a subsection with capital letters. APA[2] and MLA limit the number of levels of titles to 5. Their guide ensures that each level is visually distinct from another. Titles at level 4 or below are "run-on heads[3]" / "run-in[4]" / in-line headings i.e. they appear along with the text.

Unfortunately no such guide is available for the prospectuses provided as part of FinTOC 2021 Financial Document Structure Extraction Shared Task (El Maarouf et al., 2021). In other words the documents do not follow the same style guide, assuming they are respecting one. As a result, the task of identifying a heading and its correct level is daunting. This is proved by the fact that the best score for the previous year's shared task (Bentabet et al., 2020) was **0.37**.

[1] https://www.aeaweb.org/journals/aer/submissions/accepted-articles/styleguide#IVA

[2] https://apastyle.apa.org/style-grammar-guidelines/paper-format/headings

[3] https://projects.iq.harvard.edu/crea-lit/headings-and-subheadings

[4] http://www.creativeglossary.com/graphic-design/run-in-heading.html

2 Data

The training set consisted of 47 prospectuses in `PDF` format along with the annotations in `json` format. The annotations had depth (level), page number, file name and raw text of each title. The test set had 10 prospectuses for each language and the task is to generate a json file as described before for each of the files.

The first challenge was to find these texts in the PDFs in order to extract more metadata viz. position, font, size etc. This process is detailed in Section 3.

The second challenge was that the number of levels that the titles can have was not defined. Also, there were quite a number of documents having multiple depth one title i.e. main heading!

A cursory glance reveals that the title level 1 is always present in the first page and is either the name of the fund or the key phrase **Prospectus** for English and **Informations clés pour l'investisseur** for French. If both the fund's name and the key phrase exist it is generally the one that appears first irrespective of the style of the sentence. There were certain documents where this wasn't the case.

We argue that **Prospectus** or **Informations clés pour l'investisseur** should always be the first level title if present in the first page since it describes the document and is consistent with other main titles of documents such as **Status**, **Reglement**, **Key Information Document**, etc.

Language	P	R	F1
English	*0.858*	0.670	0.728
French	**0.911**	0.510	0.639

Table 1: Title detection results.

Language	Inex08					Harmonic
	P	R	F1	Title Acc	Level Acc	Mean
French	46.8	28.1	34.4	47.3	16.6	22.4
English	**61.1**	50.3	*53.4*	**68.2**	12.4	20.1

Table 2: TOC extraction results.

3 System

We used `pdfminer.six`[5] to parse the files. We extracted `LTTextLine` and matched it against the annotations. If

- it is an exact match, we extract features.
- `LTTextLine` is a subtext of an annotation, we find all such subtexts, then merge them and finally, extract features.
- a annotation is a subtext of `LTTextLine`, it is ignored.

In short we ignored quite a few inline titles. This might have led the models to treat them as normal text and may have been the cause of low Recall score.

From each `LTTextLine` we collected:

- coordinates (normalized: divided by page dimensions)
- percentage of characters in **bold**
- percentage of characters in *italics*
- mode of character sizes (min-max normalized)
- height (min-max normalized)
- page number (min-max normalized)
- inverse length
- percentage of capital letters
- does it start with numbering
- does it start with a capital
- is it all capital letters
- average character area
- MinHash[6] encoding of the first 10 characters
- normalized text (only alphabets without accents) in lowercase to compute `tf-idf`

The scikit-learn (Pedregosa et al., 2011) was used to get TFIDF with the following arguments:

```
analyzer = 'char'
ngram_range = (3, 3)
max_df = 0.93
max_features = 3000
sublinear_tf = True
```

As mentioned above, some `LTTextLines` were needed to be combined to match a title in the annotations. This was done as follows:

- find `LTTextLine` that matches the beginning of the annotation
- if this subtext along with previously matched `LTTextLines` has the least area then keep it
- update the annotation by removing the prefix that matched `LTTextLine`
- if annotation is an empty string then stop else repeat

Once we identified the titles along with features in the PDFs we converted the documents into images with the help of `pdf2image`[7]. The coordinates of the titles were multiplied by 4 to get the bounding boxes and then saved in COCO format[8]. This was used to fine-tune the PubLayNet (Zhong et al., 2019) Faster-RCNN model as explained on their github repository[9] with the hyperparameters of Table 3.

The fine-tuned model was used to obtain IoU and probability of being title for each `LTTextLine`.

[5] https://pdfminersix.readthedocs.io/en/latest/

[6] https://dirty-cat.github.io/stable/generated/dirty_cat.MinHashEncoder.html#dirty_cat.MinHashEncoder

[7] https://github.com/Belval/pdf2image

[8] https://cocodataset.org/#format-data

[9] https://github.com/ibm-aur-nlp/PubLayNet/tree/master/pre-trained-models

Hyperparameters	English	French
BASE_LR	0.005	0.001
MAX_ITER	36000	50000
STEPS	[0, 24000 , 32000]	[0, 30000 , 40000]

Table 3: Hyperparameters to finetune PubLayNet.

These two values along with the features listed above was fed to a Gradient Boosting Classifier with parameters:

```
n_estimators = 200
learning_rate = 0.2
max_leaf_nodes = 10
min_samples_leaf = 15
max_depth = 20
random_state = 10
```

We trained one model for each language.

At test time, the fine-tuned PubLayNet was used to merge `LTTextLines` and then extract features for classification.

After classification the titles were sorted by their size. The largest titles were attributed a depth of 1. The next in order were given 2 as depth and so forth.

4 Results

Our model, despite having the highest precision for French and second-highest precision for English, came second in the title detection task (see Table 1).

In case of TOC extraction, the English model had the highest Inex08 Precision and Inex08 Title Accuracy among the competing methods. We could not achieve the same performance for French due to less time allotted for fine-tuning the PubLayNet model. Since the models scored low on Inex08 Level Accuracy, we were nowhere near the top performing team that achieved the Harmonic Mean greater than **0.5**.

5 Discussion

We can further improve the Inex08 scores related to title detection for French by better fine-tuning the PubLayNet model.

We would also like to compare this model with LayoutLM[10,11](Xu et al., 2020), also based on `Faster-RCNN`.

[10]https://github.com/microsoft/unilm/tree/master/layoutlm

[11]https://huggingface.co/transformers/model_doc/layoutlm.html

However, a model that can correctly identify the title levels and be ported to other domains remains elusive.

6 Conclusion

We feel that lack of an annotation guide makes it difficult to analyse the errors related to title levels and as a result improve the results. The use of a vision-based model improves the title detection and can be generalized to other domains.

Acknowledgements

We would like to thank Fortia Financial Solutions for organizing this task and thus contributing to the advancement of document structure analysis.

References

Najah-Imane Bentabet, Remi Juge, Ismail El Maarouf, Virginie Mouilleron, Dialekti Valsamou-Stanislawski, and Mahmoud El-Haj. 2020. The Financial Document Structure Extraction Shared Task (FinToc 2020). In *The 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation (FNP-FNS 2020)*, Barcelona, Spain.

Ismail El Maarouf, Juyeon Kang, Abderrahim Aitazzi, Sandra Bellato, Mei Gan, and Mahmoud El-Haj. 2021. The Financial Document Structure Extraction Shared Task (FinToc 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

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.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. 2019. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE.

FinTOC2021 - Document Structure Understanding

Christopher Bourez
christopher.bourez@gmail.com

Abstract

Most machine learning applications to documents have been focused on analysing the semantic components of the texts without their formatting. Nevertheless, formatting contains information in a number of ways. First, it encodes the semantic structure of the documents and extracting the Table of Contents (TOC) is a way to summarize, categorize and search into documents. Second, machine learning models, such as for instance segmentation in computer vision, have always been imprecise at boundaries and been the subject of many years of research (conditional random fields, models in U, etc). For texts, understanding the formatting gives precise boundaries to text sections. This document discusses the different aspects of structure understanding through style formatting, and describes the method that gives state-of-art results at FINTOC 2021 challenge.

1 Introduction

Document structure understanding has specifities that need to be taken into consideration for the tasks of title detection and TOC extraction.

1.1 Emphasis and background

Annotating headers is an ambiguous task. Humans will disagree whether the underlined word "Abstract" in the first paragraph of every publication is a header or not, in particular when it is inline. Any emphasis in formatting (bold font, italics, all caps, ...) is a marker of the structure and of a higher level node in the structure tree. But despite being a higher-level node, it is ambiguous to determine if it should be included into the final TOC (Table of Contents).

Emphasis is relative and requires a contrast in style with the locally most common style, that we could call "background style", used to format the "paragraphs", the stream of characters that expresses the content with semantics and no highlight. A font size of 12 points might be an attribute of the paragraph style in one document, but of header style in another document. Frequency and contrast are at the heart of the header inference. Headers detach themselves from paragraphs using a different style, with more emphasis, most of the time.

Note also the difference between the structure tree and the TOC. The structure tree is like the HTML/Word DOM, with nodes both for headers and paragraphs. Paragraphs are found at any depth in the structure tree, for example between H1 and H2 titles, or after a H2 title. A paragraph between H1 and H2 titles in the text is a child of the H1 node as the H2 node is, and in consequence, at the same level in the structure tree as the H2 node. Although a paragraph between H1 and H2 titles will have a different level in the structure tree as a paragraph after a H2 title, in most cases both paragraphs will have the same style. Contrary to headers in the TOC, paragraphs in the structure tree do not necessarily have a different style for different depths. The TOC only includes headers and different depth in the TOC is usually linked to different styles.

1.2 Lonely headers and enumerations of headers

Headers of same level in the structure tree are usually printed in a same style. Sometimes, they are numbered with the same numbering pattern ("Article 1", "1.", "I"...), with continuous or constant numbers. Let's call them "enumerations", even when their numbering is implicit.

Among all emphasis, some headers are unique in their style. This is usually the case of the main title of the document, with an emphasis superior to all other headers. It can also be the "Abstract" header in the introduction, for example. Once the clustering into enumerations is made, it might be good to check if lonely headers are not the result of an error in formatting or numbering, which usually happens

and re-attach them to previous enumerations.

If confirmed, a lonely header is a source of uncertainty. Since it appears alone, one can only compare its emphasis with other headers' emphasis to decide of its depth in the TOC. To the contrary, enumerations are more robust, due to the fact that multiple headers in the same styling or numbering format give confirmation clues about the position of the header in the structure tree, with their "wavelength", the average number of blocks (paragraph or header) inbetween headers of the enumeration. For example, the paragraphs most of the time follow each other, while headers are usually separated by paragraphs and do not follow. So, an enumeration of blocks in the same style will be more likely to be paragraphs if they follow each other (continuous paragraph numbers), while enumeration of headers at higher level (lower depth) will be separated by more blocks.

Contrary to lonely headers, enumeration of headers is a robust concept, where multiple headers confirm the fact they are structuring the document with a particular style and belong to the same depth in the TOC or are paragraphs. It is complementary to the inference by "most of the time"/frequency described in previous section.

Reasoning with the concept of enumerations or clusters of headers is also a more interesting concept for evaluation, than indexing the depth of the header in the TOC by an absolute integer. First, any error in previous levels in the TOC (adding one more header or missing one header on the path) will trigger a different depth for all children headers. In particular, the first page of each document contains numerous "lonely headers", sub-titles, making it arbitrary to decide at which of these sub-headers we should attach the main enumeration of headers structuring the whole document. Any wrong subtitle on the path before will consider as false an enumeration of multiple child headers that structure correctly and unambiguously the document because their absolute depth relies on the previous subtitles' depth and will have changed. Absolute depth is ambiguous and not as important as recognizing the main unambiguous enumerations that structure the document.

1.3 Validation

Though a high number of headers, one might note the relatively low number of documents. Splitting the train dataset into 2 new splits, train and val and adding more features to my models gave higher and higher accuracies on the dataset, while the accuracy on the official validation dataset was not getting better. In fact, the model was probably learning the different styles in each document that were common to the two splits of the train dataset. So, the headers in the dataset are not randomly distributed headers, but headers belonging to few documents. And evaluation on the validation dataset gives an entirely new distribution to test the generalization capability of the features.

Increasing the number of documents in the dataset would require to ensure that they do not follow the same document template (same styles).

2 Method

2.1 Block extraction

ABBYY is used to convert PDF to paragraphs, in particular because of its good layout extraction technology that works well in multi-column layout as well as provides usually accurate reading order of blocks. Indeed PDF format contains only the position and styling of the characters, but do not enforce any grouping of characters into words, lines, paragraphs, and reading order of paragraphs.

ABBYY Finereader is also used for table detection to exclude paragraphs inside tables.

Paragraph attributes are:

- Page, page height, page width
- Number of lines
- Text
- Line spacing (not used, but could)
- Formatting attributes:
- Font name
- Font color
- Font weight
- Underline
- Italics
- Font size
- First line:
  - Text
  - Formatting attributes

While a paragraph might contain characters in different formatting, always the most common formatting attribute is used, meaning that first histograms are computed for each attribute and do not average the value but take the most common. Two paragraphs with most common font size 12 could have for example averaging values such as 11.3234 and 12.3545, making not much sense, losing the fact they belong to the background/paragraph style because of a formatting style change occuring in the middle of the paragraphs.

The first line attribute does not exactly exactly correspond to the first line, but to all starting characters in the first line that share the same formatting.

2.2 Predictions

A style is defined by the following features:

- Font name
- Font color
- Font weight boolean
- Underline boolean
- Italics boolean
- Font size float in pixel (converted from the point size value)
- A boolean indicating if string is all capitalized

A serialization method computes a hash of each style. Style is defined either at paragraph level or first line level.

Prediction takes as input the following features:

- Font size ratio with most common font size of the document
- Indentation of the paragraph's first line (difference between the first line's start abscissa and paragraph's abscissa), in pixels.
- The local frequency of the paragraph style, where local means on a window of 20 paragraphs after the current one.
- The local frequency of each style feature
- The global frequency of the paragraph style, where global means all paragraphs in the document
- Style feature equality between current paragraph p_n and paragraphs p_{n-2}, p_{n-1}, p_{n+1}, p_{n+2} as well as first line of paragraph p_n: a boolean for each style feature to indicate equality
- Position feature difference between current paragraph p_n and next and previous paragraphs (p_{n-1}, p_{n+1}) where position features are paragraph start abscissa, paragraph's first line start abscissa, indentation and paragraph width and are normalized to the page width
- Paragraph vertical (top and bottom) margins: distance to previous and next paragraph in the same column, if any

The local frequency features, for each style as well as for each style features, is an approximation of the concept of enumerations. The same remark is true for style feature comparison with surrounding paragraphs.

Other features, such as normalized font size, vertical margin, and global frequency of a style feature in the document are measures of the emphasis. That means that the model can consider boldness or italics as emphasis attribute only if their global frequency in the document is low. In a document completely written in italics, 'no-italics' is an emphasis attribute.

2.3 Title extraction

Once the block is predicted as header, spaces and bullet characters are stripped from the text of the block. If not void, the text of the first line is considered, otherwise the paragraph's text, and, if still void, the next paragraph's text. If the string contains a colon mark, only the text before the colon mark is kept. During extraction, we consider as "first line" not the full line, but first characters of the first line that share the same formatting.

2.4 Numbered enumeration averaging

It is a small feature that gives a gain of 1 or 2 points in accuracy. It is easy to parse the numbering at the start of each block, and build enumerations of headers with same numbering format. For blocks part of a numbering enumeration, the average prediction probability for blocks inside this enumeration instead of the prediction for the single block.

Run	Precision	Recall	F1
Training 1	0.825	0.828	0.822
Training 2	0.851	0.823	0.830

Table 1: Title detection English

Run	Precision	Recall	F1
Training 1	0.870	0.772	0.817
Training 2	0.873	0.771	0.818

Table 2: Title detection French

2.5 Training

The best working model is a XGBoost classifier of maximal depth 7. Two trainings are provided for each task: in each first training, use of the validation dataset (private 2020) to choose the datasets to train on. On the English task, the French dataset does not help, while on the French task, the English dataset does. In second training, use of all datasets, both english+french, train+validation, to provide the results, but without being able to run a validation since the validation dataset is used in training.

Results are ranked first on all tasks on FINTOC 2021. We see on these results that more data (training 2) provide better results on all tasks. During development of the features, best results are also achieved on FINTOC 2020, except for French title detection, probably due to other aspects such as block extraction, or text matching in the evaluation metric due to French accents (being French, I'm supposed to deal with that better).

3 Discussion

Despite a theory of enumerations not fully demonstrated, the local frequency of style features translates it as an approximation. The comparison to neighbor paragraphs enables to capture a local context as well, which is important to decide if it is a header.

The current metric to evaluate was developed for books, where there are only a few, usually one header per page. In the case of financial documents, there a multiple headers per page and a mis-alignment in early headers in the page will disqualify all following predicted headers, even if they were true positive. It should match headers per page, inside each page, and not consider the order to match. This will also help when the block extraction technology makes error in the block reading order.

The current metric is also comparing absolute depth described by an integer, while early titles in on the first page of the document are ambiguous and have an impact on the depth. Clustering metrics to compare the groundtruth list of enumerations and the predicted enumerations of headers (headers child of a header) will be good alternatives.

Last, it would be more comfortable if best machine learning models would be rule based and predict rules that one could understand. Though, feature importance is used to search for features, and manual rules to numbered enumerations.

References

Marco Aiello, Christof Monz, Leon Todoran, and Marcel Worring. 2002. Document understanding for a broad class of documents.

Najah-Imane Bentabet, Remi Juge, Ismail El Maarouf, Virginie Mouilleron, Dialekti Valsamou-Stanislawski, and Mahmoud El-Haj. 2020. The Financial Document Structure Extraction Shared Task (FinToc 2020). In *The 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation (FNP-FNS 2020)*, Barcelona, Spain.

Ismail El Maarouf, Juyeon Kang, Abderrahim Aitazzi, Sandra Bellato, Mei Gan, and Mahmoud El-Haj. 2021. The Financial Document Structure Extraction Shared Task (FinToc 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

Emmanuel Giguet and Gaël Lejeune. 2019. Daniel@FinTOC-2019 Shared Task : TOC Extraction and Title Detection. In *The Second Financial Narrative Processing Workshop (FNP 2019)*, Proceedings of the Second Financial Narrative ProcessingWorkshop (FNP 2019), pages 63–68, Turku, Finland.

Yufan Guo, Roi Reichart, and Anna Korhonen. 2013. Improved information structure analysis of scientific documents through discourse and lexical constraints. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 928–937, Atlanta, Georgia. Association for Computational Linguistics.

Yufan Guo, Roi Reichart, and Anna Korhonen. 2015. Unsupervised declarative knowledge induction for constraint-based learning of information structure in scientific documents. *Transactions of the Association for Computational Linguistics*, 3:131–143.

Run	Inex08-P	Inex08-R	Inex08-F1	Inex08-Title acc	Inex08-Level acc	harm mean
Training 1	53.3	52	52.5	59	36.5	52.5
Training 2	55.4	52.6	53.6	60.3	30.6	53.6

Table 3: TOC extraction English

Run	Inex08-P	Inex08-R	Inex08-F1	Inex08-Title acc	Inex08-Level acc	harm mean
Training 1	60.8	54.3	57.3	63.5	38.7	57.3
Training 2	60.9	54.2	57.3	63.6	39	57.3

Table 4: TOC extraction French

Yutong Jiang, Ning Li, and Yingai Tian. 2019. Understanding the structure of streaming documents based on neural network. In *ICCPR '19: 8th International Conference on Computing and Pattern Recognition, Beijing, China, October 23-25, 2019*, pages 30–38. ACM.

Stefan Klampfl, Michael Granitzer, Kris Jack, and Roman Kern. 2014. Unsupervised document structure analysis of digital scientific articles. *Int. J. Digit. Libr.*, 14(3-4):83–99.

Extractive Financial Narrative Summarisation using SentenceBERT Based Clustering

Tuba Gokhan, Phillip Smith and **Mark Lee**
School of Computer Science
University of Birmingham, United Kingdom
{txg857|smithpm|m.g.lee}@cs.bham.ac.uk

Abstract

We participated in the FNS-2021 Shared Task: "Financial Narrative Summarisation" organized by at 3rd Financial Narrative Processing Workshop (FNP-2021). We build an unsupervised extractive automatic financial summarisation system for the specific task. In our approach to the FNS-2021 shared task, the documents are first analyzed and an intermediate bespoke document set created containing the most salient sections of the reports. Next, vector representations are created for the intermediate document set based on SentenceBERT. Finally, the vectors are then clustered, and sentences from each cluster are chosen for the final generated report summaries. The achieved results support the proposed method's effectiveness.

1 Introduction

With the growth of financial sector along with the economy's growth and development, there are quite a few new companies emerging and going public. Investors often find long-form annual reports of various companies difficult to deal with because their content may be tedious or redundant, and it can be arduous to filter out effective key information by human inspection alone, hence an automatic summarisation system would be useful to help investors effectively understand important company information.

The Financial Narrative summarisation Shared Task for 2021 (FNS-2021)(El-Haj et al., 2021) aims to evaluate the performance of automatic summarisation methods applied to annual reports from UK corporations listed on The London Stock Exchange (El-Haj et al., 2020). Compared to reports prepared by U.S companies, these reports have a notably less rigid structure making summarisation particularly challenging. These reports can be divided into two main sections. The first section is a "narrative" section which is also known as a "front-end" section containing textual information and reviews by the company's management and board of directors; the second section is the "back-end" section which contains financial statements, which tend to consist of tables of numerical data. The FNS-2021 shared task entails determining which are the most important narrative sections and then summarising these to achieve a summary of approximately 1000 words.

In this paper, we will discuss the solution that we developed for the FNS-2021 shared task. The data set used is the annual reports in the financial field provided by the organizer. Since there is often a lot of redundant information in the annual reports, it is planned to choose the most useful parts first as the intermediate documents to be summarized. Thus, to identify the salience of the sentences in order to extract the most relevant ones from the original financial report our system uses a combination of sentence embedding and clustering algorithms.

2 Data

For this shared task, the data that we used consisted of 3,863 United Kingdom annual reports for corporations listed on the London Stock Exchange (LSE) between 2002 and 2017. Annual reports in the UK are long papers that average approximately 80 pages, with some exceeding 250 pages. For the FNS-2021 shared task, these annual reports were separated into three sections: training, testing, and validation.

The complete text of each yearly report, as well as the gold-standard summaries, are included in the training and validation sets. Each annual report has at least three gold-standard summaries on average, with some reports including up to seven gold-standard summaries. The task participants were only provided access to the full texts for the testing data set. Further details are shown in Table 1.

Data Type	Training	Validation	Testing
Report Full Text	3000	363	500
Gold Summaries	9873	1250	1673

Table 1: FNS-2021 Shared Task Dataset

3 System

3.1 Pre-processing

Before development of our system the gold standard summaries were examined in detail. In the provided training set, we found that the main narrative sections were mostly under four headings: "Chief Executive's review", "At a glance", "Highlights" and "Chairman's Statement". These sections typically include summarized financial topics. The other sections contain a significant amount of statistical data, tables, graphs, and diagrams, therefore they are not included in the organizer's gold summary. The focus of the study is to find only on narrative segments to build our summarisation system. For this reason, a condensed intermediate document covering only the sections we wish to appear in the final summaries is required. As a result, both the processing speed and the success percentage of the generated summaries is considerably improved.

We also noted that these sections were generally in the first 10% of each report. We therefore extracted these and a new data set containing only these sections was created. On this new, condensed dataset, sentences were tokenized using the '`tokenize`' package from the NLTK library (Loper and Bird, 2002).

3.2 Sentence Embedding

One of the main problems of natural language processing is the encoding and symbolising of words or characters in a text so that the machine can, to a degree, understand them. Embedding is a mathematical method of mapping an object in one domain to another object in another. Sentence embeddings converts a sentence into a vector. The BERT architecture is chosen due to its high performance over other NLP algorithms for Sentence Embedding. BERT is built on transformer architecture (Vaswani et al., 2017). Two BERT models are published by Google for public use, one containing 110 million parameters and the other 340 million parameters. (Devlin et al., 2019).

Sentence-BERT (SBERT) is a modification of the pre-trained BERT network that uses Siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. This reduces the effort for finding the most similar pair from 65 hours with BERT / RoBERTa to about 5 seconds with SBERT, while maintaining the accuracy from BERT (Reimers and Gurevych, 2019). Due to its performance in the larger pre-trained BERT model, SBERT was ultimately chosen for our experiments. In our studies, the sentences are vectorised with SBERT. Different models are used in the vectorisation phase. The highest performing models and their results are shown in Table 2.

3.3 Clustering

Since the BERT model outputs sentence embeddings, these sentences can be clustered with a size of K, allowing dynamic summaries of the FNS-2021 shared task dataset. A maximum of 1000 words is expected in the output summaries. When the data set is examined, the N value must consist of a dynamic value less than 25 in order to create 1000-word summaries.

During our experiments, the Scikit-Learn Clustering library [1] was examined in detail. The K-means algorithm is an unsupervised clustering algorithm which partitions a set of data, usually termed dataset into a certain number of clusters. K-Means is chosen as the appropriate model for clustering sentence embeddings from the BERT model, since the algorithm approves the amount of dynamic clustering and is scalable on large data. For the final summary, sentences closest to the centroid are selected from the clusters. Euclidean distance is used to measure the distances to centroids.

4 Results

Following the development of the aforementioned system we evaluate as follows. The FNS-2021 Shared Task contest decided to use the ROUGE2 package[2] to evaluate the system outputs. The ROUGE2 package is a multilingual tool that implements ROUGE (Lin, 2004) metrics.

[1] https://scikit-learn.org/stable/modules/clustering.html
[2] https://github.com/kavgan/ROUGE-2.0

	Model Name	R-1/F	R-2/F	R-L/F	R-SU4/F
Our System 1	nli-mpnet-base-v2	0.48	0.25	0.41	0.29
Our System 2	distiluse-base-multilingual-cased-v2	0.48	0.25	0.40	0.29
Our System 3	nli-distilroberta-base-v2	0.47	0.25	0.40	0.29

Table 2: Results on validation set.

System	R-1/ F	R-2/ F	R-L /F	R-SU4 /F
TEXTRANK	0.17	0.07	0.21	0.08
LEXRANK	0.26	0.12	0.22	0.14
PointT-5	0.46	0.28	0.45	0.28
SumTO	0.42	0.24	0.39	0.26
HULAT	0.44	0.26	0.38	0.26
MUSE	0.5	0.28	0.45	0.32
Our System 3-1	**0.47**	0.25	0.4	**0.29**
Our System 2	**0.48**	0.26	0.4	**0.29**

Table 3: Results measured by the organizers for the test set.

The method we developed to measure system performance in our experiments was applied on the validation dataset. The summaries we created were evaluated using Rouge-1, Rouge-2, Rouge-SU4 and Rouge-L metrics. The results obtained as a result of our measurements of the three systems with the highest performance among the results are shown in Table 2. The results of our systems' summaries, 363 documents in validation dataset, produced scores very similar to each other.

In the FNS-2021 Shared Task contest, the gold summaries of the test set in which the results will be evaluated were not shared with the participants. The performance of the generated summaries was measured by the organizers over the test set. Table 3 shows the organizers' calculated scores for the three systems we provided. In Table 3, the results of our systems, the results of the baseline TEXRANK (Mihalcea and Tarau, 2004) and LEXRANK (Erkan and Radev, 2004) algorithms, the results of PointT-5 (Singh, 2020), SumTO (La Quatra and Cagliero, 2020) and HULAT (Baldeon Suarez et al., 2020), which are the systems with the highest performance in the FNS-2020 Shared task, and the results of the topline MUSE (Litvak et al., 2010) algorithm are presented.

The overall ranking of systems varies depending on the evaluation metric considered. When our results were compared to baseline algorithms, we achieved relatively successful performance in all metrics. Furthermore, when compared to FNS-2020 Shared Task, our results indicate good outcomes in the Rouge-1 and Rouge-SU4 metrics. And, when we compare it to the MUSE method, which is based on topline, we see that the results are fairly similar.

5 Discussion

In this study, a SentenceBERT-based clustering approach is proposed as an unsupervised method for the FNS Shared task. As a result of this approach, extractive summaries of less than 1000 words were created. In order to create high quality summaries in this dataset, first and foremost, it is necessary to define the "Chief Executive's review", "At a glance", "Highlights" and "Chairman's Statement" sections that form the basis of gold summaries. However complex text documents are produced as a result of converting the dataset from PDF, makes this definition difficult. For this reason, the preprocessing phase is extended in our work.

Another challenge in this task is producing 1000-word summaries.The basis of our proposed approach is clustering. In order to create summaries of 1000 words, we need to limit the number of sets to a maximum of 25. However, in clustering approaches, it is necessary to determine the ideal number of clusters according to the data distribution. This number of clusters varies depending on the documents, and the restriction of the number of clusters causes sentences that do not have similar meanings to be included in the same cluster.

In addition, when creating sentence vectors, our method employs pre-trained language models.

These models were created using various datasets. We believe that the use of fine-tuned language models using financial documents and terms to improve the performance of the study will improve the performance of the summary system.

6 Conclusion

The paper describes an extractive summarisation approach to summarizing textual financial reports for the Financial Narrative Summarisation Shared Task (FNS-2021). The proposed approach relies on clustering sentence vectors created with sentence embedding. Firstly, an intermediate document dataset covering the most important parts of the documents is prepared. Then, pre-trained language representation model Bidirectional Encoder Representations from Transformers (BERT) is utilized to generate sentence embeddings. Finally, the k-means clustering algorithm is applied to find similar sentences and from each cluster, a sentence vector representing the set is selected for the final summary. Three systems are created using different sentence embedding models are submitted. The performance of the obtained summaries is measured with the ROUGE metric. Our approach outperforms the baseline algorithms in terms of performance when is compared to the literature, whereas the topline algorithm produce partially identical results.

References

Jaime Baldeon Suarez, Paloma Martínez, and Jose Luis Martínez. 2020. Combining financial word embeddings and knowledge-based features for financial text summarization UC3M-MC system at FNS-2020. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 112–117, Barcelona, Spain (Online). COLING.

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.

Mahmoud El-Haj, Ahmed AbuRa'ed, Marina Litvak, Nikiforos Pittaras, and George Giannakopoulos. 2020. The financial narrative summarisation shared task (FNS 2020). In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 1–12, Barcelona, Spain (Online). COLING.

Mahmoud El-Haj, Nadhem Zmandar, Paul Rayson, Ahmed AbuRa'ed, Marina Litvak, Nikiforos Pittaras, and George Giannakopoulos. 2021. The Financial Narrative Summarisation Shared Task (FNS 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Moreno La Quatra and Luca Cagliero. 2020. End-to-end training for financial report summarization. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 118–123, Barcelona, Spain (Online). COLING.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Marina Litvak, Mark Last, and Menahem Friedman. 2010. A new approach to improving multilingual summarization using a genetic algorithm. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 927–936.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Abhishek Singh. 2020. PoinT-5: Pointer network and T-5 based financial narrative summarisation. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 105–111, Barcelona, Spain (Online). COLING.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Joint abstractive and extractive method for long financial document summarization

Nadhem Zmandar[1] **Abhishek Singh**[2] **Mahmoud El-Haj**[1] **Paul Rayson**[1]

[1]Lancaster University, UK
n.zmandar@lancaster.ac.uk

[2]Samsung R&D Bangalore, India
abhishek.s.eee15@iitbhu.ac.in

Abstract

In this paper we show the results of our participation in the FNS 2021 shared task. In our work we propose an end to end financial narrative summarization system that first selects salient sentences from the document and then paraphrases extracted sentences. This method generates an overall concise summary that maximises the $ROUGE$ metric with the gold standard summary. The end to end system is developed using a hybrid extractive and abstractive architecture followed by joint training using policy-based reinforcement learning to bridge together the two networks. Empirically, we achieve better scores than the proposed baselines and toplines of FNS 2021 (LexRank, TextRank, MUSE topline and POLY baseline) and we were ranked 2nd in the shared task competition.

Keywords: Summarization, Neural networks, Reinforcement learning, sequence to sequence learning; actor-critic methods; policy gradients.

1 Introduction

The task of text summarization is to condense long documents into short summaries while preserving the content and meaning. It can be performed using two main techniques: extraction and abstraction. The extractive summarization method directly chooses and outputs the salient phrases in the original document (Jing and McKeown (1999); Knight and Marcu (2002)). The abstractive summarization approach involves rewriting the summary (Rush et al. (2015); Liu et al. (2015)); and has seen substantial recent gains due to neural sequence-to sequence models (Chopra et al. (2016); Nallapati et al. (2016a); See et al. (2017a); El-Haj et al. (2018); Paulus et al. (2017)).

In the general case, extractive summarization approaches usually show a better performance compared to the abstractive approaches especially when evaluated using $ROUGE$ metrics (Kiyoumarsi, 2015). One of the advantages of the extractive approaches is that they can summarize source articles by extracting salient snippets and sentences directly from these documents, while abstractive approaches rely on word-level attention mechanism to determine the most relevant words to the target words at each decoding step. Several studies ((Widyassari et al., 2020) ; (Tretyak and Stepanov, 2020)) proposed to combine extractive and abstractive techniques in order to improve performance.

Abstractive models can be more concise by generating summaries from scratch in a context where the gold summaries were deleted from the original annual reports. However, this method suffers from slow and inaccurate encoding of very long documents which is the case with financial annual reports (above 50,000 tokens per report). Abstractive models also suffer from redundancy, especially when generating summaries of long documents. (Cohan et al., 2018) .

Therefore, the proposed summarizer follows a hybrid extractive-abstractive architecture, with policy-based reinforcement learning (RL) to bridge together the two networks. The model first uses an extractor agent to select salient phrases, and then employs an abstractor network to rewrite (compress and paraphrase) each of these extracted sentences. We then use actor critic policy gradient with sentence-level metric rewards to jointly train these two summarization models in order to perform Reinforcement Learning and learn sentence saliency.

2 Background

Recurrent models typically take in a sequence in the order it is written and use that to output a sequence. Each element in the sequence is associated with its step in computation time. These models generate a sequence of hidden states, as a function of the previous hidden state and the input for current position.

The sequential nature of models (RNNs, LSTMs or GRUs) does not allow for parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. In order to compute current outputs, the model needs to process previous outputs and inputs, therefore outputs cannot be calculated using parallel computation. This method is not appropriate if text is too long since it takes long time to process the outputs and calculate the loss after several time steps. Therefore, attention mechanisms have become critical for sequence modeling in various tasks, allowing modeling of dependencies without caring too much about their distance in the input or output sequences (Chen and Bansal, 2018).

Long sequence NLP presents many challenges for current models. In fact, long range dependencies often require complex reasoning and forces models to both locate relevant information and combine it. Models need to ignore a lot of irrelevant text. Many popular algorithms are designed to work in short sequence setting, and have limitations in long setting. RNN/LSTM: process input sequentially and stores relevant information from previous states therefore it is slow for long sequences. Transformers are based on self-attention and cannot process long input with current hardware. (e.g. BERT pre-trained Language model is limited to 512 tokens).

3 Data description

The dataset is composed of UK annual reports in English from the financial summarization shared task (FNS 2021) (El-Haj, 2019; El-Haj et al., 2020, 2021). The dataset contains 3,863 annual reports for firms listed on the London Stock Exchange (LSE) covering the period between 2002 and 2017. The average length of an annual report is 52,000 tokens. The dataset is randomly split into training (75%), testing and validation (25%). Data is further described and analysed in Appendix A.

4 Methodology

4.1 Financial word embeddings

The financial summarization task requires embeddings of domain-specific vocabulary that embeddings pre-trained on a generic corpus may not be able to capture.

Financial documents include words that appear in any general purpose pre-trained word embedding such as Glove (Pennington et al., 2014). However the usage of these words will be different and therefore the representation in the vector space should be different as well. The jargon used in financial disclosures is different from 'general' language. For example, corporate earnings releases use nuanced language not fully reflected in GloVE vectors pre-trained on Wikipedia articles. For all these reasons, working on training custom word embedding for financial domain is helpful in our case.

To implement a financial word embedding model using word2vec model, we used the Gensim[1] library. We perform pre-processing using the NLTK[2] library. We deleted non alphanumeric values, and replaced some special characters by their equivalent (e.g. "m" is replaced "million". Moreover, we convert all words into lowercase. Finally, we extract tokenized sentences of the dataset using the NLTK tokenizer and created a vocabulary of the training dataset in the form of dictionary where keys are words and values are number of occurrence. The tokenized sentences were passed as input to the word2vec model from the Gensim library which produced the word vectors as output. We limit the Vocab size to 20,000 (most frequent words) and the maximum number of words in a sentence to 60. The parameters we used to train word2vec model are shown in Table 2:

4.2 Model

We train a reinforcement learning model based on standard policy gradient method to form an end-to-end trainable computation graph which is divided into extraction and abstraction phases. In fact, it is infeasible to start a randomly initialized neural network to train the whole summarization model. The extractor would often select sentences that are not relevant. On the other hand, without a well-trained abstractor the extractor would get noisy reward (bad $Rouge - 2$, which leads to a bad estimate of the policy gradient and a sub optimal policy.

We should work on optimizing each sub-module (extractor and abstractor) separately using maximum likelihood objectives. Train the extractor machine learning model to select salient sentences and the abstractor model to generate shortened summary. Finally, reinforcement learning is applied to train the full end to end model.

[1] https://radimrehurek.com/gensim/
[2] https://pypi.org/project/nltk/

4.2.1 Extractor agent:

The extractor agent is designed to model the extraction function, which can be thought of as extracting salient sentences from the document. We exploit a hierarchical neural model to learn the sentence representations of the document and a 'selection network' to extract sentences based on their representations.

In extraction process we assume that for every summary sentence there is matching sentence in the annual report. To train extraction model we need these corresponding sentences in the reports. Since, annual reports are not marked explicitly with sentences we followed $ROUGE$ scores to extract these sentences as done in (Nallapati et al., 2016b) ; (Chen and Bansal, 2018). For every summary sentence we calculate $ROUGE$ with every sentence in the report and then choose the sentence with maximum $ROUGE-2$ value.

$$j_t = argmax_i(\text{ROUGE}_L F1(d_i, s_t))$$

where d_i represents i^{th} document sentence and s_t represents t^{th} summary sentence. We extract sentences from the annual reports that maximize $ROUGE$ score with the gold summaries. These sentences are used as labels for training the machine learning extractor model.
In fact, for every annual report, we calculate summary level $ROUGE$ scores for each of the provided summaries. We greedily match summary sentences to article sentences with higher $ROUGE$ score (Nallapati et al., 2016a). Selected sentences should greedily maximise the global summary-level $ROUGE$. For each summary sentence exactly one document sentence is matched, based on the individual sentence-level score to avoid redundancy in the summary, since summary is limited to 1000 words. Eventually summary level $ROUGE$ scores are calculated and summary with maximum score is chosen for further processing and training.

Once labels are generated using the above described method, extractor model is trained to extract salient sentences from the reports. The ML extractor model uses attention mechanism (Bahdanau et al., 2016) based Pointer Networks (Vinyals et al., 2015) which is different from the copy mechanism used in (See et al., 2017a). Given these proxy sentences extracted in the previous step as ground truths and sentences extracted using pointer network, we train it to minimize cross-entropy loss.

The parameters used to train the ML extractor model are shown in Table 3 in the appendix section. The ML model training took 4 hours. The model converged to the optimal value after 56 Epochs reducing the loss to 0.779927.

4.2.2 Abstractor agent:

The abstractor network approximates the function that paraphrases an extracted document sentence to a concise summary sentence. We use an encoder-decoder model based on RNN and Attention mechanism (Bahdanau et al., 2016) ; (Luong et al., 2015). Copy mechanism is adopted to help directly copy some out-of-vocabulary (OOV) words (See et al., 2017a).

For the abstractor training, training pairs are created by taking each summary sentence and pairing it with its extracted document sentence. The network is trained as an usual sequence-to-sequence model to minimize the cross-entropy loss. First sentences are encoded using the financial word embedding vectors and passed to Convolutional Neural Network layer for encoding and further passed to Long Short Term Memory layers for sequence modelling. Final output of the encoder is passed to LSTM based decoder to generate paraphrased summary sentences.

4.2.3 Reinforcement Learning

The Markov Decision process property states that the future depends only on the present and not on the past. It is a probabilistic model that depends on the current state to predict the next state. The future is conditionally independent of the past states. In other words, we could predict P_{t+1} using only P_t.

The goal of reinforcement learning models is to learn using an agent that interacts with a stochastic environment. Reinforcement learning optimizes the agent's decisions by learning the value of states and actions from a reward function. The main goal is to define a policy function that maps states to actions. Reinforcement learning helps to maximise $ROUGE$ score by rewarding good sentences that are extracted and penalising bad sentences.

Once the extractor and abstractor models are trained individually, final complete model is trained using policy gradient algorithm with similar process as in (Chen and Bansal, 2018). At every extraction step agent samples an action to extract document sentence an receive reward $r(t+1)$ which is $ROUGE$-2 F1 score between output after abstraction and ground truth summary sentence.

The reinforcement learning training works as follow: The extractor starts by choosing a relevant sentence from the report, then the abstractor rewrites it. If the $ROUGE$ 2 F1 score match would be high the action is encouraged. If an irrelevant sentence is chosen and the abstractor still produces a compressed version of it, the summary would not match the ground truth and therefore low $ROUGE$ 2 F1 score discourages this action.

In the actor-critic approach, the actor takes the state of the environment as the input, then returns the best action, or a policy that refers to a probability distribution over the actions. In our case we use Pointer Network to perform the actor job.

On the other hand, the critic evaluates the actions returned by the actor neural network and returns a score representing the value of taking that action given the state.

The figure 1 gives a concise description of the end to end summarizer system.

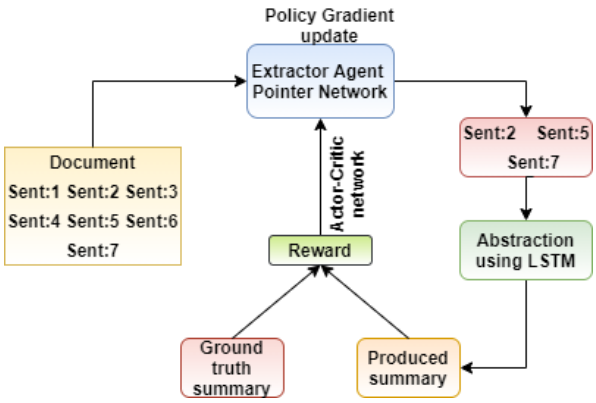


Figure 1: The end to end summarizer

5 Experimental setup

In order to train our extractor, abstractor and RL models, we use a Tesla P100-PCIE GPU with accelerated high RAM of gigabytes with batch size of 16 and check point frequency of 16 batches.

Please refer to appendix for full training setup. Hyperparameters details are shown in Table 3, Table 4 and Table 5.

6 Results

6.1 Metrics

The $ROUGE$ measure finds the common unigram ($ROUGE$-1), bigram ($ROUGE$-2), and largest common substring (LCS) ($ROUGE-L$) between the ground-truth text and the output generated by the model and calculates respective precision, recall, and F1-score for each measure.[3] For the entire dataset, we evaluate standard $ROUGE1$, $ROUGE$-2, and $ROUGE$-L and $ROUGE$-SU4 (Lin, 2004) on full length F1 (with stemming) following previous works (See et al. (2017a); Nallapati et al. (2016a)). The $ROUGE$ 2.0 package Ganesan (2015) is used for calculations.

6.2 Scores

In this section, we present results from our experiments and compare with different baselines MUSE (Litvak et al., 2010), Text-rank (Mihalcea and Tarau, 2004), Lex-Rank (Erkan and Radev, 2004), and Polynomial Summarisation (Litvak and Vanetik, 2013).

Overall, our model achieves better results than all the proposed baselines with $ROUGE1$: 0.52, $ROUGE$-2 : 0.30, $ROUGE$-L : 0.46 and $ROUGE$-SU4 : 0.32

Metric	R-1/F	R-2/F	R-L/F	R-SU/F
TextRank	0.17	0.07	0.21	0.08
LexRank	0.26	0.12	0.22	0.14
Polynomial	0.37	0.12	0.26	0.18
MUSE	0.5	0.28	0.45	0.32
rnn-ext + abs + RL	**0.52**	**0.3**	**0.46**	**0.32**

Table 1: FNS shared task results

7 Conclusion and Future Work

In this paper, we have reported on our solution for the Financial Narrative Summarisation (FNS2021) shared task using actor critic reinforcement learning approach. It is a combination of both extractive and abstractive methods using Pointer Network. With these methods we are able to achieve the second highest F1 score in every evaluation metric and were able to beat the baseline and topline models.

In our future work we would like to address several limitations of our method such as factual correctness in summaries which is very important in financial domain as done in Zhang et al. (2020b) in summarizing radiology reports. To improve precision of our generated summaries under 1000 words we would formulate a penalty if system generates more than 1,000 words during training of RL algorithm rather than restricting algorithm to fixed number of words.

[3] https://github.com/google-research/google-research/tree/master/rouge

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Ahsaas Bajaj, Pavitra Dangati, Kalpesh Krishna, Pradhiksha Ashok Kumar, Rheeya Uppaal, Bradford Windsor, Eliot Brenner, Dominic Dotterrer, Rajarshi Das, and Andrew McCallum. 2021. Long document summarization in a low resource setting using pretrained language models.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of ACL*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

Mahmoud El-Haj. 2019. Multiling 2019: Financial narrative summarisation. In *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources*, pages 6–10.

Mahmoud El-Haj, Ahmed AbuRa'ed, Marina Litvak, Nikiforos Pittaras, and George Giannakopoulos. 2020. The Financial Narrative Summarisation Shared Task (FNS 2020). In *The 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation (FNP-FNS 2020)*, Barcelona, Spain.

Mahmoud El-Haj, Paul Rayson, Paulo Alves, Carlos Herrero-Zorita, and Steven Young. 2019a. Multilingual financial narrative processing: Analyzing annual reports in english, spanish, and portuguese. In *Multilingual Text Analysis: Challenges, Models, And Approaches*, pages 441–463. World Scientific.

Mahmoud El-Haj, Paul Rayson, Paulo Alves, and Steven Eric Young. 2018. Towards a multilingual financial narrative processing system.

Mahmoud El-Haj, Paul Rayson, Martin Walker, Steven Young, and Vasiliki Simaki. 2019b. In search of meaning: Lessons, resources and next steps for computational analysis of financial discourse. *Journal of Business Finance & Accounting*, 46(3-4):265–306.

Mahmoud El-Haj, Nadhem Zmandar, Paul Rayson, Ahmed AbuRa'ed, Marina Litvak, Nikiforos Pittaras, and George Giannakopoulos. 2021. The Financial Narrative Summarisation Shared Task (FNS 2021). In *The Third Financial Narrative Processing Workshop (FNP 2021)*, Lancaster, UK.

G. Erkan and D. R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Kavita Ganesan. 2015. Rouge 2.0: Updated and improved measures for evaluation of summarization tasks.

Alexios Gidiotis and Grigorios Tsoumakas. 2020. A divide-and-conquer approach to the summarization of long documents.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization.

Hongyan Jing and Kathleen R. McKeown. 1999. The decomposition of human-written summary sentences. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 129–136, New York, NY, USA. Association for Computing Machinery.

Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K. Reddy. 2019. Deep reinforcement learning for sequence to sequence models.

Farshad Kiyoumarsi. 2015. Evaluation of automatic text summarizations based on human summaries. *Procedia - Social and Behavioral Sciences*, 192:83–91. The Proceedings of 2nd Global Conference on Conference on Linguistics and Foreign Language Teaching.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artif. Intell.*, 139(1):91–107.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Marina Litvak, Mark Last, M. Friedman, S. Kisilevich, and Sami Shamoon. 2010. Muse – a multilingual sentence extractor.

Marina Litvak and Natalia Vanetik. 2013. Mining the gaps: Towards polynomial summarization. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 655–660, Nagoya, Japan. Asian Federation of Natural Language Processing.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016a. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Guchere, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016c. Abstractive text summarization using sequence-to-sequence rnns and beyond.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017a. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017b. Get to the point: Summarization with pointer-generator networks.

Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. 2020. Neural abstractive text summarization with sequence-to-sequence models.

Vladislav Tretyak and Denis Stepanov. 2020. Combination of abstractive and extractive approaches for summarization of long scientific texts.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, and De Rosal Ignatius Moses Setiadi. 2020. Review of automatic text summarization techniques methods. *Journal of King Saud University - Computer and Information Sciences*.

Wen Xiao and Giuseppe Carenini. 2019. Extractive summarization of long documents by combining global and local context.

Senci Ying, Zheng Yan Zhao, and Wuhe Zou. 2021. LongSumm 2021: Session based automatic summarization model for scientific document. In *Proceedings of the Second Workshop on Scholarly Document Processing*, pages 97–102, Online. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2021. Big bird: Transformers for longer sequences.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.

Yuhao Zhang, Derek Merck, Emily Bao Tsai, Christopher D. Manning, and Curtis P. Langlotz. 2020b. Optimizing the factual correctness of a summary: A study of summarizing radiology reports.

Appendices

sg	min_count	window	size	sample
1	3	2	300	6e-5
alpha	negative	workers	epochs	—
0.05	20	16	15	—

Table 2: Word2Vec Parameters

Parameter	Value	Description
lr	1e-3	learning rate
decay	0.5	learning rate decay ratio
clip	2.0	gradient clipping rate
batch	16	training batch size
net_type	rnn	network type
vsize	20000	vocabulary size
n_hidden	256	number of hidden units of LSTM size
emb_dim	300	dimension of word embedding
n_layer	2	the number of layers of LSTM
conv_hidden	100	number of hidden units of LSTM size
lstm_hidden	256	Number of hidden layers in LSTM network
max_art	100	maximun words in a single article sentence
max_abs	50	maximun words in a single abstract sentence

Table 3: Hyperparameters for the ML extractor

Parameter	Value	Description
vsize	20000	vocabulary size
emb_dim	300	dimension of word embedding
n_hidden	256	number of hidden units of LSTM size
lr	1e-3	learning rate
decay	0.5	learning rate decay ratio
clip	2.0	gradient clipping rate
batch	16	training batch size
n_layer	2	the number of layers of LSTM
max_art	100	maximun words in a single article sentence
max_abs	50	maximun words in a single abstract sentence

Table 4: Hyperparameters for the abstractor

Parameter	Value	Description
lr	1e-4	learning rate
decay	0.5	learning rate decay ratio
clip	2.0	gradient clipping rate
batch	1	training batch size
lr_p	0	patience for learning rate decay
gamma	0.95	discount factor of RL
reward	ROUGE-2	reward function
stop	1.0	stop coefficient for ROUGE-2
patience	5	patience for early stopping

Table 5: Hyperparameters for the RL extractor

CILAB@FinTOC-2021 Shared Task: Title Detection and Table of Content Extraction for Financial Document

Hyuntae Kim, Soyoung Park, Seongeun Yang, Yuchul Jung*

Department of Computer Engineering,
Kumoh National Institute of Technology (KIT), Gumi, Korea
{ hyuntaekim09, haluna8836, banilla03090, enthusia77}@gmail.com

Abstract

This paper describes Cilab's system for extracting the title and table of contents (TOC) of FinToc-2021 Shared Task's financial documents. We use FinTOC2021-trainset, FinTOC2020-testset data provided by the organizer, and 444 documents obtained from four different publication organizations. In terms of the training algorithm, we selected a random forest classifier for title detection and considered font style, bold, font size, and text to determine the level of the title. Among English and French tracks, we participated only in English tracks and ranked fourth (F1-score, 51.4%) among the six teams that participated in title detection. In TOC extraction, we achieved third place among the six teams that participated in Harmonic mean 26.3.

1 Introduction

Many recent studies (Pappagari, R. et al., 2019; Martha O. Perez-Arriaga, 2016; Adhikari et al., 2019) have attempted to grasp the structure of PDF documents. However, most of these studies are on thesis-type or receipt-type documents with formal structures. Few studies identify the structure of financial documents, detect titles, and extract tables of contents (TOC).

The goal of FinTOC-2021 is to extract specific texts that act as the title, section, and title of the document in the financial document and generate a table of contents of the extracted titles.

Towards a robust TOC extraction from financial documents, we start by finding a good feature set required for distinguishing title text from non-title texts. Besides, to find & train a best-performing title classifier, a total of five models (e.g., Random Forest, SVM (Jiu-Zhen Liang, 2004), Naïve Bayes, Logistic Regression, and BERT based fine-tuning) were tested. Moreover, to overcome the lack of training data, 400+ financial documents were newly collected and used for building the title classifier. Finally, based on an in-depth investigation of the given training data, a set of heuristics was designed for post-processing, allocating appropriate depth levels based on the given title texts.

This paper shortly summarizes recent approaches of title detection and TOC extraction in Section 2, analyzes the data sets provided and additionally collected data in Section 3, talks about the structure and internal experiments of the system that detects title and extracts TOC in Section 4, describes official results in Section 5, and present our future work in Section 6.

2 Related Work

In FinTOC-2020 shared tasks on structure extraction from financial documents, various approaches on feature selection, algorithm, and procedures had been attempted.

Title Detection and TOC extraction tasks are highly co-related, and influential features can be obtained basically from preprocessing like removing the header or footer or erasing the table. After preprocessing, the text of the document is classified into title or non-title by applying various machine learning techniques.

As a combination approach, Daniel@FinToC'2 shared task (Emmanuel et al., 2020) combined TOC itself, wording of the document, and lexical domain knowledge.

Although there can be numerous machine learning approaches for title detection and TOC extraction, interestingly, the random forest algorithm showed the best performance (Kosmajac, Detal, 2020). As a different solution, others adopted maximum entropy classifier (Hercig et al., 2020) and multilingual BERT (Hase et al., 2020)

3 Datasets

FinTOC-2021 provided with two datasets: FinTOC2021-trainset and FinTOC2020-testset. However, due to the lack of training data, we analyzed the most frequent publication agencies of the data provided. Finally, we chose four different publication organizations, BISCIAV, USB, DNB, and SEB. To obtain documents similar to those included FinTOC2021-trainset and FinTOC2020-testset, we selected more than 400 PDF documents by crawling the four publication organizations. The newly constructed dataset was named FHFO (**F**our **H**undred documents from **F**our different **O**rganizations). In our experiments, a total of three datasets were used. Table 1 shows the data statistics for Title/Non-Title labels and the PDF documents included for the three datasets.

Dataset Name	Num. of Titles	Num. of Non-Titles	Num. of PDF Docs.
FinTOC2020 –testset	6.5k	154k	22
FinTOC2021 -trainset	6.5k	144k	42
FHFO Data	52.2k	3.6M	444

Table 1: Data statistics for title detection and table of contents extraction

In the case of FinTOC2021-trainset used 42 of the 47 documents as educational materials, excluding files that could not be opened and those that could not be processed with PDF-miner[1].

Four hundred forty-four financial PDF documents of FHFO data were used to train models that separated titles and non-titles using pseudo-labeling.

In addition, the distribution of the level of the table for contents in the two prepared datasets is shown in Table 2.

Depth Level	2020 Test	2021 Train
Level 1	193	45(0.5%)
Level 2	1149	99
Level 3	1405	975
Level 4	1450	2291
Level 5	430	2652
Level 6	241	980
Level 7	27	645
Level 8	2	211
Level 9	0	62
Level 10	0	0
Total	4897	7960

Table 2: Depth level distribution

4 Our Proposed System

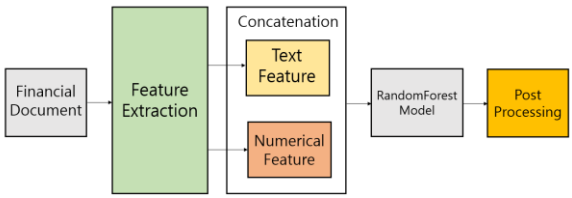


Figure 2: System Architecture

The core components of our proposed financial document structure extraction system consist of three sequential steps, such as feature extraction, random forest model (L. Breiman, 2001) for title detection, and post-processing for resolving depth levels as in Figure 1.

4.1 Feature Extraction

The feature extraction relies on PDF-Miner to extract features from financial documents. The extracted feature is as follows.

- Text: Extract the text of the document by row.
- Font Size: Extract the font size of the extracted text.

[1] https://github.com/pdfminer/pdfminer.six

- Font Style: Extract the font style of the text.
- Bold: Extract 0 or 1 whether the extracted text is bold.
- Text Coordinate: Extract the bounding box area (left upper, right lower) of the extracted text.

Text features and numerical features of text, font size, bold, and text coordinate are used to train the model for title detection, and the features of the font style are added to the features above.

4.2 Random Forest Model for Title Detection

To find the most appropriate dataset combination for building our title detection model, we divided the datasets in Section 3 into D1, D2, D3, and D4. Table 3 lists the dataset combinations used in our experiments.

	Dataset Combinations
D1	2021-trainset
D2	2021-trainset+2020-testset
D3	2021-trainset+FHFO data
D4	2021-trainset+2020-testset+FHFO data

Table 3: Dataset description

To decide the best classification algorithm for title classification, we tried a total of five models (e.g., Random Forest, SVM (Jiu-Zhen Liang, 2004), Naïve Bayes, Logistic Regression, and BERT based fine-tuning) among various machine learning algorithms as in Table 4. Experiments were conducted on the D1 dataset with the five models, and the Random Forest model showed the best performance.

Model	Precision	Recall	F1
SVM	0.84	0.84	0.84
Naïve Bayes	0.82	0.82	0.82
Logistic Regression	0.83	0.83	0.83
BERT based Fine-tuning	0.79	0.70	0.67
Random Forest	**0.87**	**0.86**	**0.86**

Table 4: Title classification performances of five different algorithms for the D1 dataset

Table 5 summarizes the results of applying the random forest algorithm from D1 to D4 datasets. We achieved an F1 score of 96% with the D4 dataset. The second-best performance was gotten with the D2 dataset.

Dataset	Precision	Recall	F1
D1	0.87	0.86	0.86
D2	0.94	0.91	0.92
D3	0.89	0.87	0.88
D4	**0.97**	**0.95**	**0.96**

Table 5: Title classification performances for D1~D4 datasets with random forest algorithm

4.3 Post-Processing for Resolving the Depth Levels

After that, post-processing procedures were carried out to determine the depth level for each title detected. To this ends, FinTOC2021-trainset and FinTOC2020-testset were verified semi-automatically in order to establish our internal criteria for allocating depth levels.

Through an in-depth analysis, we delineated the following heuristics for resolving depth levels for each title.

1) From depth 1 to 2, they mainly deal with the contents of titles and sub-headings and the large font size.

2) From depth 3 to 5, they mostly contain figures which represent the levels of titles.

3) Depths of 6~7 have a smaller font size and are rarely less than depth 8. Thus, our heuristics are target to allocate between depths 1 ~ 7 entirely.

Our internal post-processing criteria are generally based on the font style and font size. Therefore, bold or Italic was set as an essential criterion among the font styles, and the Font size was divided by the most significant or most minor Font size.

Since the depths of most titles were 3 to 5 levels, it was difficult to distinguish only by manual verification, and in this case, the font style was first screened out and through a regular expression to check whether there are specific rules in the text.

To derive a set of rules for the post-processing, we further subdivided and generalized the above

findings by applying them to FinTOC2021-trainset and FinTOC2020-testset.

5 Official Results

In this paper, the system of CILAB who participated in the FinTOC 2021 sharing work was explained. The proposed system ranked 4th in Title Detection and 3rd in TOC Extraction. The official evaluations were shown in Table 6 (Title Detection) and Table 7 & 8 (TOC Extraction), respectively.

Team	Precision	Recall	F1
Cilab_fintoc1	0.702	0.376	0.456
Cilab_fintoc2	0.708	0.422	0.514

Table 6: Title Detection results of Caleb team

Team	Precision	Recall	F1
Cilab_fintoc1	26.6	14.4	17.6
Cilab_fintoc2	30.5	18.6	22.6

Table 7: Precision, Recall, and F1 score of TOC Extraction

Team	Title acc.	Level acc.	Harm. mean
Cilab_fintoc1	34.2	34.8	23.4
Cilab_fintoc2	38.9	31.4	26.3

Table 8: TOC Extraction results from "Inex08-result"

Among the two submitted runs, the 2nd run (i.e., Cilab_fintoc2) performed better. It ranked fourth in title detection and third in TOC extraction out of six teams, respectively.

The model used a random forest classifier and PDF-Miner to distinguish the title from various financial documents and extracted features such as text, font size, font style, and bold. Then, we went through assigning depth to the extracted feature based on the depth arbitrarily determined by us.

The difference between the two models lies in the difference in learning data. In the case of Cilab_fintoc1, the training was performed with FinTOC2021-trainset and FinTOC2020-testset. On the other hand, Cilab_fintoc2's training data includes our constructed FHFO data additionally. As a result, it was confirmed that more training data from similar publication organizations could produce better results.

However, there are many noises in detecting the only title in the financial documents, and the data of non-title is much more biased, indicating that the performance, when applied to actual data, is much lower than that of the training stage.

6 Future Work

As our future work, we are interested in building a more robust TOC extraction model using multi-modal machine learning techniques specialized in financial documents that better combine visual features and text features.

7 Reference

El Maarouf, Ismail and Kang, Juyeon and Aitazzi, Abderrahim and Bellato, Sandra and Gan, Mei and El-Haj, Mahmoud "The Financial Document Structure Extraction Shared Task (FinToc 2021) " (2021).

Kosmajac, D. et al. "DNLP@FinTOC'20: Tableof Contents Detection in FinancialDocuments." Fin ancial Narrative Processing Workshops (2020).

Hase, Frederic and Steffen Kirchhoff."Taxy.io@FinTOC-2020: Multilingual DocumentStructure Extraction using Transfer Learning." Financial Narrative Processing Workshops (2020).

Hercig, Tomás and P. Král. "UWB@FinTOC-2020 Shared Task: Financial Document Title Detection." Financial Narrative Processing Workshops (2020).

Emmanuel Giguet ,Gaël Lejeune ,Jean-Baptiste Tanguy" Daniel@FinTOC '2 Shared Task: Title Detection and Structure Extraction" Financial Narrative Processing Workshops (2020).

L. Breiman. Random forests. Machine Learning, 45:5–32, 2001. Text Extraction using Document Structure Features and Support Vector Machines

Jiu-Zhen Liang, "SVM multi-classifier and Web document classification," Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), 2004, pp. 1347-1351 vol.3

Martha O. Perez-Arriaga, Trilce Estrada, and Soraya Abad-Mota "TAO: System for Table Detection and Extraction from PDF Documents"

Adhikari, Ashutosh & Ram, Achyudh & Tang, Raphael & Lin, Jimmy. (2019). DocBERT: BERT for Document Classification.

Pappagari, R. et al. “Hierarchical Transformers for Long Document Classification.” 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU) (2019): 838-844.