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

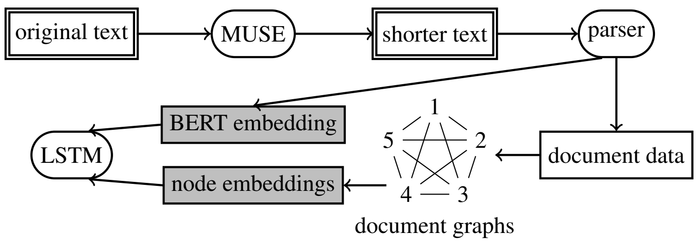

Figure 1: Pipeline of our approach

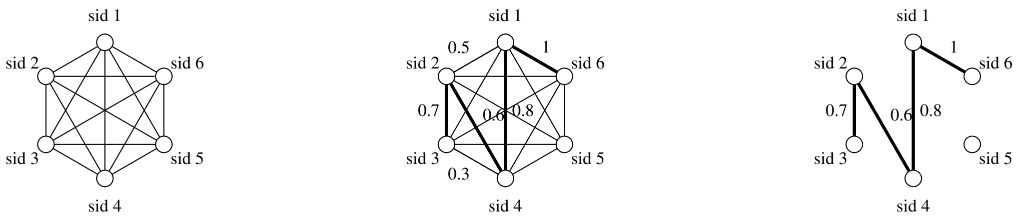

Figure 2: Pruning example for a graph on 6 sentences and median edge weight 0.5

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). .