Hybrid Text Summarization through Reinforcement Learning

Francesco Di Salvo
francesco.disalvo@studenti.polito.it

Gianluca La Malfa
gianluca.lamalfa@studenti.polito.it

***Abstract*—This paper explores an hybrid neural summarization task, where an extractor agent filters the most salient information and an abstractor agent will paraphrase them. Then, a reinforcement learning agent will reward the produced output for jointly learning both agents. The proposed architecture was tested on the FNS dataset, containing financial reports, and on the CNN/Daily Mail dataset, covering a wider number of news domains. The experiments on CNN/Daily Mail achieved a lower Rouge-L score compared to ones obtained on the financial reports, justified by the variety of topics and by the significantly smaller number of summary sentences proposed. Moreover, on two randomly selected subsamples, we showed that despite a lower Rouge-L score we obtain comparable results on BERTScore, that takes into account the context and the semantic of the produced summaries. The code [1] and the main pre-trained models [2] used for the following experiments are publicly available.**

I. Introduction

In 2021, it was forecast that the overall amount of data created worldwide would reach 79 zettabytes. By 2025, this amount is expected to double. The maintenance of this enormous amount of data is definitely expensive and time consuming. This is one of the reasons why the research has been always more and more focused on automatic text summarization, the task of capturing the most salient information from a given document or a collection of documents. This task can be split into two main categories: extractive and abstractive text summarization. In extractive summarization approaches, the summary is created by copying and concatenating the most relevant sentences from the given text corpora. A basic pipeline requires then an adequate text preprocessing, feature extraction, sentence scoring and ranking and finally, sentence extraction. Abstractive summarization is the process of generating a summary containing words not necessarily present on the given text corpora. This involves paraphrasing salient text through Natural Language Understanding and Generation techniques. While in extractive approaches we greedily match sentences from the text corpus, in abstractive approaches the system needs to first understand the overall document meaning and then apply sentence compression and fusion techniques. At the time of writing, extractive approaches typically perform better, obtaining concise and grammatically correct summaries. In fact, abstractive approaches still suffer from out-of-vocabulary (OOV) words, sentence repetition and inaccurate or grammatically incorrect sentences. Moreover, since abstractive approaches need to encode the whole text, they typically obtain poor performances on long documents, which is the case of financial documents.

Over the years some hybrid approaches were proposed [1], [2], trying to combine both extractive and abstractive summarization pipelines in order to maximize the performances. This work aims to replicate the work of Zmandar et al [3], inspired again from Chen et al [4]. The proposed approach follows a hybrid extractive-abstractive architecture with a policy-based reinforcement learning for combining both networks. In particular, the extractor filters the most salient information that will be paraphrased by the abstractor agent. Finally, the reinforcement agent has a sentence-level metric reward for jointly training both architectures.

[1]https://github.com/francescodisalvo05/nlp-financial-summarization-rl
[2]shorturl.at/iMYZ7

II. Related works

Abstractive summarization reached more and more interest along the years, bringing up to light a lot of interesting approaches. Many Natural Language Processing applications relies on encoder-decoder architectures, whose encoder converts variable-length sequences into fixed-length sequences and the decoder do the same thing on the other way around. They both employed RNNs [5] for the proposed conversions, but lately they were replaced with LSTMs [6] and GRUs [7] in order to capture and model long range dependencies. Alternatively, bi-direction RNNs/LSTMs go one step ahead, modeling both backward and forward language models. A couple of years later we faced the new era of the attention mechanism [8] that allows to weight each token with its importance with respect to the overall sentence. One of the main issues of abstractive summarization is the comprehension of long documents, this is why some hybrid approaches were proposed [1], [2], [9]. The idea is to first extract the most salient information through extractive approaches and then to paraphrase them through abstractive ones. Novel approches integrates Reinforcement Learning agents [4], [10], [11] in order to improve the the cross-learning between extractive and abstractive approaches.

III. Methodology

The reinforcement learning model is trained starting from both pre-trained extractor abstractor. In fact, a poor extractor would have passed to the abstractor not meaningful sentences, obtaining noisy rewards. With that to be said, both extractor and abstractor are trained with the same training data. A more

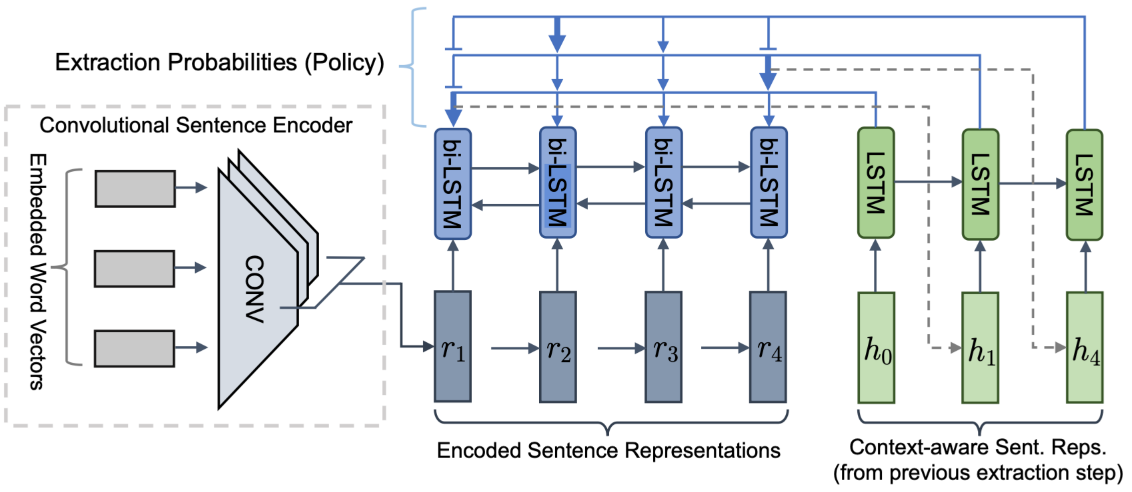
Extraction Probabilities (Policy)
Convolutional Sentence Encoder
Embedded Word Vectors
CONV
bi-LSTM
bi-LSTM
bi-LSTM
bi-LSTM
LSTM
LSTM
LSTM
r_1
r_2
r_3
r_4
h_0
h_1
h_4
Encoded Sentence Representations
Context-aware Sent. Reps.
(from previous extraction step)

Fig. 1. Overview of the Extractor Architecture [4]

robust approach may train the abstractor model with the output sentences predicted by the extractor, as it happens at inference time. However, due to the dimension of the training data, we may assume to train a robust abstractor that may not necessary require the actual outputs of the extractor.

A. Extractor

The extractor agent (Figure 1) models the extraction function, that aims to extract the most salient sentences from the text. To train the extractor model we need to extract first the potential most salient sentences to learn from. Therefore, given a set of reports and associated golden summaries, for every summary sentence we extract one matching sentence from the report. The ranking of the sentences was performed according to the ROUGE-L score [12]:

$$j_t = \underset{i}{\operatorname{argmax}}\, ROUGE_L F1(d_i, s_t) \tag{1}$$

where d_i represents the i^{th} document sentence and s_t represents the t^{th} summary sentence. This allows to greedily match summary sentences against report sentences. Moreover, for computational reasons we decided to select only one summary for each report, but due to the extractive nature of the process, it would be interesting to evaluate how much impact this choice has on the overall performances.

Once the labels have been generated, they were fed to an extractor network for learning the extraction patterns. The sentence representation is leveraged through a hierarchical neural model, converting the word level embeddings to sentence level ones. On top of the encoded sentence representations, a bidirectional LSTM-RNN [13] is employed for capturing long range dependencies. Then, in order to recurrently extracting the sentences, another LSTM-RNN is employed for training a Pointer Network [14]. At each time step, the decoder performs a 2-hop attention mechanism: it attends to the hidden representation to get a context vector and then it attends it again for the extraction probabilities, classifying all the sentences at each iteration step.

B. Abstractor

The abstractor compresses and paraphrases the extracted sentences, namely, the most representative ones. As mentioned before, this was trained in parallel to the extractor with the labels, assuming to have inject enough noise and redundancy with the provided datasets. A standard encoder-aligner decoder [15] is employed, together with the so called copy-mechanism [16] in order to copy some out of vocabulary (OOV) words. The network is then trained to minimize the cross entropy loss

$$L(\theta_{abs}) = -\frac{1}{M}\sum_{m=1}^{M} \log P_{\theta_{abs}}(w_m | w_{1:m-1}) \tag{2}$$

of the decoder language model at each generation step, where θ_{abs} is the set of trainable parameters of the abstractor and w_m is the m^{th} generated word.

C. Reinforcement Learning

Reinforcement Learning (RL) frameworks uses an agent that interacts with a stochastic environment. In order to make the extractor under RL settings, we may start from the formulation of the Markov Decision Process, a probabilistic model that depends only on the current state for predicting the next one. Hence, likewise [4], at every extraction step t, the agent observes only the current state c_t, samples an action j_t to extract a document sentence and finally receives a reward $r(t+1)$, that is simply the ROUGE-2 F1 score between output after abstraction and ground truth summary sentence:

$$r(t+1) = ROUGE_L F1(g(d_{j_t}), s_t) \tag{3}$$

where d_{j_t} is the extracted sentence based on the action j_t at step t, and s_t is the summary sentence that should have been extracted at the given step. Therefore, once the extractor starts selecting relevant sentences, the abstractor will try to

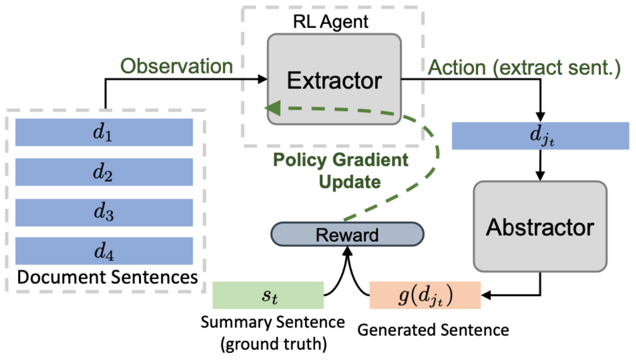

Fig. 2. Reinforcement learning pipeline

paraphrase them and the ROUGE-L F1 will be used to evaluate the outcome. If this is high, this action will be encouraged, otherwise it will be discouraged.

It is well known that the vanilla policy gradient algorithm suffers from an high variance, therefore a critic network was added. The introduced actor has to calculate the prediction of the expected reward across all possible actions at the given step. This is called ”baseline” and it is used for calculating the advantage function that we aim to maximize, minimizing on the other hand the square loss between the critic value and R_t, that is an estimate of action-value function.

Finally, in order to determine how many summary sentences it is required to produce, a ”stop” action was introduced as a policy in the action space. To do so, a further set of trainable parameters were added v_{EOE} (end of extraction). Formalizing the stop action in this way will allow the pointer-network decoder treats v_{EOE} as one of the extraction candidates and it results in a stop action in the stochastic policy.

IV. Dataset

The proposed approach was evaluated on two completely different datasets: Financial Narrative Summarisation [17] and the CNN/Daily Mail [18].

A. FNS

The Financial Narrative Summarisation contains financial reports extracted from UK annual reports. As provided from the FNS Challenge, there are $3,000$ training samples and 363 validation samples. Due to the lack of an external test data, the provided validation set has been used as an internal test set, whereas a subsample of 450 pairs of reports and summaries has been extracted from the training sample in order to validate the model performances.

Each sentence was normalized (lowercase) and all the non alpha numerical tokens were removed. Then, each sentence was cropped at 60 tokens for computational reasons. Moreover, since we are dealing with financial statements, the integer numbers representing ”millions” and ”billions” were translated into their representative words. Finally, once the whole dataset was preprocessed, as staten in [3], only the $20,000$ most frequent tokens were filtered. This was possible because the whole data collection was homogeneous. After that, Gensim Word2Vec [19] was employed for gathering word-level embeddings to feed the first convolutional network.

B. CNN/Daily Mail

The CNN/Daily Mail dataset contains human generated abstractive summary bullets generated from news stories in CNN and Daily Mail websites. Three versions are available and we used the not anonymized one. The full datasets has $286,817$ training pairs, $13,368$ validation pairs and $11,487$ test pairs. However, due to the limited computational resources, for each split we extracted a random subsample of $10,000$, $1,000$ and $1,000$ pairs, respectively. The disproportion with FNS dataset is motivated by the fact that CNN/Daily has on average only 1-2 sentences for each provided summary, therefore it requires way more samples for learning how to properly extract them.

The dataset was normalized (lowercase) and all the non alpha numerical tokens were removed. Then, as proposed by [18], the full text was cropped at 800 tokens whereas the summary sentences were cropped at 100 tokens. Contrary as before we did not filter the most frequent tokens because the dataset is open-domain, therefore based on the ”imbalance” of the topics, this would be quite risky, especially considering that just a small sample has been extracted for this study.

V. Experiments

In order to fully explore the potentialities and performances of the proposed pipeline, we performed several tests, to investigate the results both with divergent application domain and metrics computed in different phase of the process. In particular we first used the traditional pipeline for `FNS` and `CNN/Daily Mail` datasets. Then, for computational reasons we selected a random subsample from both datasets and we evaluated their performances with different scoring functions, for both extraction and reinforcement learning policies. All the experiments were performed on an NVIDIA Tesla P-100, provided by Google Colaboratory Pro.

A. Metrics

This section explains the main metrics that we used for extracting the labels and as Reinforcement Learning policies.

- Rouge score [12]: the metric compares an automatically produced summary or translation against a reference or a set of references (human-produced) summary or translation. It is actually a set of metrics, in fact it is possible to use different versions of rouge depending on the desired evaluation. Rouge-L measures the longest common sub-sequence between the ground-truth text and the output generated by the model. More precisely, on our study we focused on Rouge-L F1-Score, in order to have a reliable measure of our model performance that relies not only on the model capturing as many words as possible (recall) but doing so without outputting irrelevant words (precision).

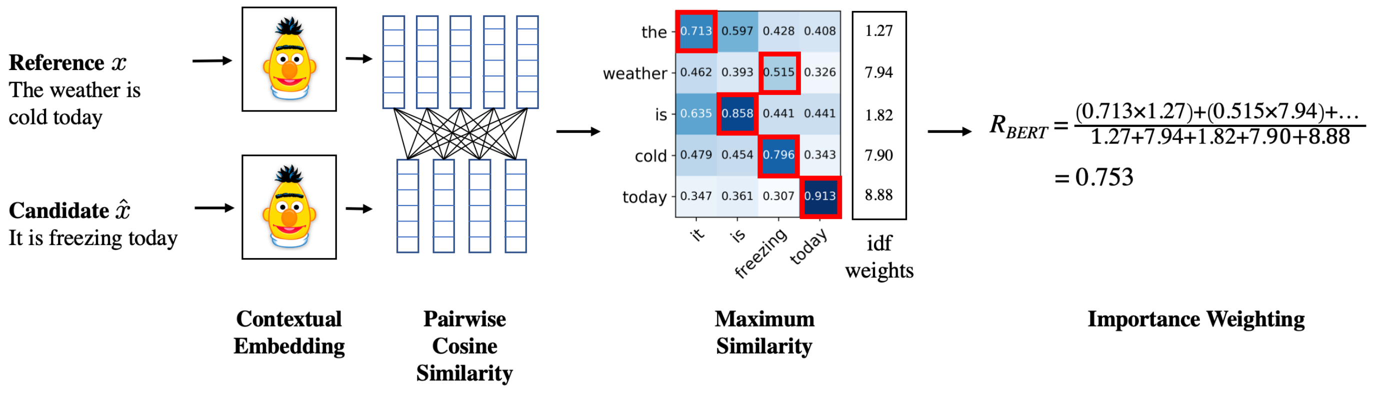
Reference x
The weather is cold today
Candidate $\hat{x}$
It is freezing today
Contextual Embedding
Pairwise Cosine Similarity
Maximum Similarity
Importance Weighting
idf weights
$R_{BERT} = \frac{(0.713\times1.27)+(0.515\times7.94)+\ldots}{1.27+7.94+1.82+7.90+8.88} = 0.753$

Fig. 3. Overview of BERT score [20]

- BERT score [20]: it represents a step forward in the direction of the correctness of semantic evaluation of the obtained result and the ground truth annotation. One of the main weaknesses [12] of Rouge is that it considers words overlap between reference and candidate, but this does not guarantee semantic correctness, especially in abstractive settings, where it is very likely that the output documents has different words compared to the one in the input documents. With BERT score, in fact, our aim is to not only consider syntactic overlapping between hypothesis and reference, but its focus is deeper in the context, trying to understand the meaning of what have been generated and what was supposed to be generated and then perform comparison. The process of evaluation reported in Figure 3 consists of taking both reference (ground truth) and candidate (generated one) sentences and pass them through the pre-trained BERT model (`roberta-large`) in order to obtain contextual embedding for each sentence. Once the final embeddings are produced for each of these tokens, the pairwise cosine similarity is calculated and the ones having the highest one, according to precision, recall or f1-score. In our study case we focused on the F1-Score. At the moment, the main weak points of the BERT score are related to its heaviness, in fact, for the standard English model it is necessary to download a 1.4GB file and not all languages are supported.

In the early stages of our work, we also tried to study the behavior of the BLEU score [21], but we found out that the performances were considerably lower than Rouge, above all in the extraction phase, so we decided to work only with the aforementioned metrics which we considered could lead us to the more meaningful results.

B. Dataset

The main limitation of using the BERT score is the computational time. Therefore, we decided to select a smaller subsample for both FNS and DailyCNN, keeping the same proportions used for the main baseline experiment (Table V-B). Due to the length of each report and summary in FNS, even with a limitation of 300 sentences per report, the extraction of the labels through BERT score required more than 30 hours on our machines.

TABLE I
OVERVIEW OF THE DATASETS SPLITS EMPLOYED FOR THE EXPERIMENTS.

	FNS		**DailyCNN**	
Split	Large	Small	Large	Small
Train	$2,550$	300	$10,000$	$1,200$
Val	450	50	$1,000$	100
Test	363	50	$1,000$	150

VI. Results

For the sake of clarity we reported all the quantitative results in the following tables. As it is possible to observe, we used two different notations. `Small` is used in reference to the subsample extracted for evaluating the overall performances obtained using BERTScore for generating the labels and as a RL policy. `Large`, on the other hand, is used in reference to the complete FNS dataset and the bigger subsample extracted from Daily/CNN, where its objective was to assess and evaluate the performance of the models across different domains. Assuming to retrieve two subsamples with the same distribution as in their relative large datasets, the best model hyperparameters (Tables V and VI) obtained on the large one will be used also on the small one.

TABLE II
ROUGE SCORE ACHIEVED BY THE EXTRACTOR ONLY AND THE FULL PIPELINE ON THE LARGE CONFIGURATION

FNS		**DailyCNN**	
Only Extractor	Full pipeline	Only Extractor	Full pipeline
0.36	0.38	0.20	0.23

Regarding the performances of the model on the big samples (Table III), we noticed a significant drop in performances according to the Rouge-L score. Interestingly, in both cases

TABLE III
COMPARISON BETWEEN THE RESULTS OBTAINED AFTER REINFORCEMENT LEARNING ON DATASET SAMPLES (SMALL) WITH MAXIMUM 100 SELECTED SENTENCES FROM THE REPORTS. A CROSS EVALUATION HAS BEEN USED TO MEASURE THE EFFECTIVENESS OF THE TWO APPROACHES

	FNS		DailyCNN	
Extracted labels & RL Policy	**Rouge-L**	**BERT score**	**Rouge-L**	**BERT score**
Rouge-L	0.27	0.80	0.09	0.78
BERT score	0.26	0.81	0.10	0.78

the full performances obtained are comparable with the results obtained using only the extraction model. Therefore, there is still room for improvement for the abstractive model. An analysis focused only on the abstractor would have not produced meaningful results because according to the proposed pipeline, it would have paraphrased the whole text.

Moreover, as it is possible to observe from Table IV, we explored the behavior of our small pipeline on the FNS dataset with a different number of selected sentences for the reinforcement learning and evaluation phase. This experiment could not be applied to the CNN/Daily Mail dataset because it contains on average less samples than 100 samples. To compare the two different metrics that we used a cross-evalution with both Rouge-L and BERT score. As the number of maximum sentences increases, the performances of the pipeline trained with Rouge tend to be stable whereas for the one trained with BERT there is a huge degradation of the Rouge-L score.

VII. CONCLUSIONS

This work explored the implementation of Chen et al. [4] developed for solving the FNS competition. The proposed pipeline was then tested on two completely different domains in order to understand its robustness. Moreover, we compared the performances in both domains with two differents metrics for the extraction and the reinforcement learning policy. The Rouge-L score shows a significant drop in performances on the CNN/Daily Mail dataset, that is broader than the FNS one and it contains just a couple of summary sentences per news. However, thanks to BERTScore we were able to evaluate not only the co-occurrence between reference and candidate summaries but also their context, fundamental in abstractive scenarios. In fact, under BERTScore the difference in performances is much lower than Rouge. It would be interesting to evaluate our pipeline on other domains, in order to understand which can be its strengths and criticalities. Moreover, the proposed results may be more meaningful if applied to the whole datasets rather than just a single subsample, or even better, including new languages other than English.

REFERENCES

[1] D. Sahoo, A. Bhoi, and R. C. Balabantaray, "Hybrid approach to abstractive summarization," *Procedia Computer Science*, vol. 132, pp. 1228–1237, 2018, international Conference on Computational Intelligence and Data Science. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918307701

[2] Y. Jaafar and K. Bouzoubaa, "Towards a new hybrid approach for abstractive summarization," *Procedia Computer Science*, vol. 142, pp. 286–293, 2018, arabic Computational Linguistics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918322026

[3] N. Zmandar, A. Singh, M. El-Haj, and P. Rayson, "Joint abstractive and extractive method for long financial document summarization," in *Proceedings of the 3rd Financial Narrative Processing Workshop*. Lancaster, United Kingdom: Association for Computational Linguistics, 15-16 Sep. 2021, pp. 99–105. [Online]. Available: https://aclanthology.org/2021.fnp-1.19

[4] Y.-C. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting," 2018. [Online]. Available: https://arxiv.org/abs/1805.11080

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] H. Dave and S. Jaswal, "Multiple text document summarization system using hybrid summarization technique," in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, 2015, pp. 804–808.

[10] S. Li, D. Lei, P. Qin, and W. Y. Wang, "Deep reinforcement learning with distributional semantic rewards for abstractive summarization," 2019. [Online]. Available: https://arxiv.org/abs/1909.00141

[11] L. Liu, Y. Lu, M. Yang, Q. Qu, J. Zhu, and H. Li, "Generative adversarial network for abstractive text summarization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://doi.org/10.1609/aaai.v32i1.12141

[12] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[13] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[14] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," 2015. [Online]. Available: https://arxiv.org/abs/1506.03134

[15] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014. [Online]. Available: https://arxiv.org/abs/1409.0473

[16] A. See, P. J. Liu, and C. D. Manning, "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)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1073–1083. [Online]. Available: https://aclanthology.org/P17-1099

[17] M. El-Haj, "MultiLing 2019: Financial narrative summarisation," in *Proceedings of the Workshop MultiLing 2019: Summarization Across Languages, Genres and Sources*. Varna, Bulgaria: INCOMA Ltd., Sep. 2019, pp. 6–10. [Online]. Available: https://aclanthology.org/W19-8902

[18] R. Nallapati, B. Zhou, C. dos Santos, Ç. Gulçehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 280–290. [Online]. Available: https://aclanthology.org/K16-1028

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[20] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2019. [Online]. Available: https://arxiv.org/abs/1904.09675

[21] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

Appendix

TABLE IV
Study applied on the FNS dataset to determine how the number of selected sentences affects the results in terms of Rouge score and Bert score with the two different reinforcement rewards

Sentences		**Rouge-L**	**BERT score**
200	**Rouge-L**	0.20	0,80
	BERT score	0.19	0.80
300	**Rouge-L**	0.13	0.79
	BERT score	0.15	0.81
400	**Rouge-L**	0.21	0.79
	BERT score	0.12	0.81

TABLE V
Extractor hyperparameters

Hyperparameter	**FNS**	**Daily/CNN**
Embedding size	300	100
Vocab size	20,000	74,930
LSTM layers	2	2
LSTM hidden layers	256	256
Conv hidden layers	100	80

TABLE VI
Abstractor hyperparameters

Hyperparameter	**FNS**	**Daily/CNN**
Embedding size	300	100
Vocab size	20,000	74,930
LSTM layers	2	2
LSTM hidden layers	256	256