Financial Text Summarization using RL policies

Maggio Leonardo
Politecnico di Torino
s292938@studenti.polito.it

Palmisano Vito
Politecnico di Torino
s288859@studenti.polito.it

Zingarelli Valerio
Politecnico di Torino
s281586@studenti.polito.it

***Abstract*—In this paper we present a solution to the problem of financial long documents summarization, as presented in the FNS2021 competition. Our model first uses an extractor to select salient sentences and then rewrites them using an abstractor to generate fluent summaries. The two components are well combined thanks to the use of a Reinforcement Learning policy based on a sentence-level metric reward (ROUGE) as in Zmandar et al. [1] and Chen et al. [2]. We extend our experiments also on the FNS2022 dataset, which is composed by long financial texts written not only in English, but also in Spanish and Greek. To deal with the computational complexity of this task, we compute the distributional analysis of the documents, in order to catch their most relevant parts and work only with them. For the same reason, we introduce the Numba library to improve computational performances. We perform a domain-task adaptation of our architecture on the CNN news dataset [3] as well. It is also a task change, not only a domain adaptation, because our purpose is to generate headlines for the articles. In the end, we decided to conduct an ablation study, cutting-off the abstractor. Concerning summarization task, we obtained comparable results with respect to Zmandar et al. [1]. We performed quite well also considering the FNS2022 competion. Moreover, the analysis of distributions displays that the most important sections inside the documents differ for the different languages and also for different documents lengths. As regards headlines'generation task, we have been able to get quite good results. Concerning the ablation study, we understood English summaries are really extractive compared with other languages ones. All the code is available at https://github.com/z216z/NLP_Project.**

I. Introduction

In Natural Language Processing, Text Summarization is the process of shortening long texts. Texts are getting longer and longer and saving time avoiding reading useless parts of them could state difference between success and failure. That's why text summarization is getting more and more important. More specifically, the task consists in giving the model a long text, and it gives us back a shortened version of it: its summary or abstract.

In our case, we have to follow a Deep Learning approach. Usually, this task can be performed basically using 2 techniques: extraction (Jin et al. [4] and Knight et al. [5]) and abstraction (Rush et al. [6] and Liu et al. [7]). More specifically, the former approach consists in enlightening the most salient sentences from the original article and then concatenating them. Unfortunately, this leads to a summary which is not fluently readable by a human because the phrases are not well connected using conjunctions or adversary prepositions. On the contrary, the abstractive approach rewrites an abstract by scratch trying to capture the idea of the text in order to create a fluent summary. In general, extractive summarization shows to have better performances with respect to abstractive one especially when the chosen score is the $ROUGE$, according to Kiyoumarsi et al. [8]. In the past, different works have tried to merge together the extractive and the abstractive paradigms.

In this work, we try to accomplish the task of long financial documents summarization, reproducing the work done by Nadhem Zmandar et al. [1]. We also try to add some extensions to it. To do this, we start exploring the method proposed by Chen et al. [2]. We train an Extractor and an Abstractor, and combine them using a Reinforcement Learning Policy. Our effort aims to demonstrate the value and challenges of applying automatic text summarization to financial texts written in English, Spanish and Greek [9]. Before starting with the preprocessing steps, we try to analyze the dataset in order to provide a distributional analysis of documents'sections importance. This has been done to spot the most important ones for our task and use only them instead of considering only the first N articles'rows.

After that, we generate the labels using a greedy approach. Since the generation of such labels was really time-demanding, we decided to use Numba library to avoid such a waste of time. By doing so, we managed to save 60% of the time we would have wasted without Numba's `@jit`. Then we train our extractor-abstractor model and obtain, as we will see, quite well results, considering our computational limitations.

Furthermore, we provide a task-domain adaptation procedure using CNN dataset to generate headlines for newspaper's articles.

In the end, we perform an ablation study cutting off the abstractor. That is because, during the data-exploration step, we have noticed that English summaries are quite "extractive" with respect to other languages. The experiments we conducted confirms our hypothesis. In section IV, all the results we managed to obtain are provided.

II. Related Works

A. Summarization

During past years, a lot of studies concerning text-summarization have been conducted and all of them had as common denominator: recurrent models. These networks are really capable of processing sequential data taking them in a sequence and respecting the order they are written. One of the most relevant drawbacks of this kind of approach is the one related to memory constraints. As a matter of fact, the sequential nature of the models (especially LSTMs and GRUs) and the memory constraints do not allow for

parallelization within training examples. Hence, these methods are not suitable for long texts, because it takes too much time to process and compute the loss after several steps. That is why "Attention Mechanisms" have been introduced [10]. Attention mechanism allows to model dependencies without taking care about their distance in input or output sequences [2]. Reinforcement Learning has been explored to mitigate the problem of "exposure bias" that occurs in supervised learning contexts: Henß et al. [11] apply RL for extractive summarization, while Paulus et al. [12] use RL in an abstractive summarization task. The work of Chen et al. [2] builds upon these ideas and use sentence-level rewards to optimize an extractor, while an abstractor rewrites summary sentences.

B. Headline Generation

With the rapid proliferation of online media sources and published news, users are overwhelmed by the number of articles. Therefore, headlines have become increasingly important for attracting readers to news articles. The headline generation task can be seen as a variant of the summarization task with one or two sentences, standardized in the DUC-2004 competitions [13]. A new challenge for AI is to generate attractive headlines while still being faithful to the content. Narayan et al. generate extreme news summarization by creating a one-sentence summary answering the question "What is the article about?" (Narayan, Cohen, and Lapata 2018 [14]). However, none of the existing approaches has considered generating attractive headlines with data-driven approaches. Zhang et al. [15] formulate the attractive headline generation task as QHG (Question Headline Generation), according to the observation that interrogative sentences attract more clicks. In the end, Song et al. [16] propose an approach based on Extractor, Abstractor and Reinforcement Learning (similar to Chen et al. [2]) to generate headlines starting from CNN's news.

III. Methodology

The pipeline we adopt in this study is composed by four different steps:

- **Preprocessing**: texts are prepared to be processed by the extractor and the abstractor.
- **Extractor Training**: the extractor agent is pre-trained in order to select the most salient sentences.
- **Abstractor Training**: the abstractor network is pre-trained to generate shortened summaries.
- **Reinforcement Learning**: RL training is performed to optimize the whole model.

A. Preprocessing

Financial documents are characterized by a different jargon from common language and therefore the financial summarization task requires embeddings of domain-specific vocabulary. To this end, we build the pipeline depicted in Fig. 1 to process the documents before feeding them to the neural networks.

Due to the limits imposed by the environment in which we run the code, we initially cut the documents, selecting only the first n sentences for the following steps.

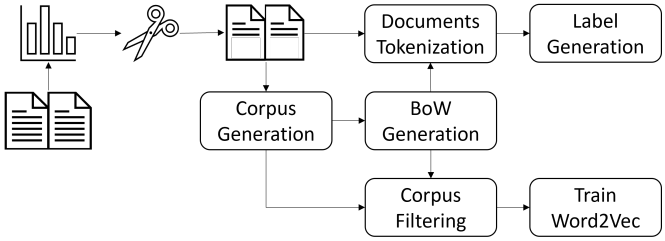

Fig. 1. A sketch of the preprocessing pipeline.

Initially we create a full corpus for the training set, which we use to obtain the "common Bag of Words", that is a dictionary containing the 20,000 most frequent words. On the one hand, it is used to train the word2vec model. On the other hand, individual documents are pre-processed to convert all words to lowercase, remove non-alphanumeric values, replace abbreviations with full names[1], and finally tokenize sentences by filtering out uncommon words.

In order to train the extraction model, it is necessary to have a dataset that matches each sentence of the summary with a sentence of the report. "Proxy" target labels are generated using a Greedy Algorithm that for each ground-truth summary sentence finds the most similar document sentence. Adopting an approach similar to Nallapati et al. [17], we use as labels for training the extractor model the report sentences that maximize the ROUGE-L score with the gold summaries:

$$j_t = argmax_i(ROUGE\text{-}L(d_i, s_t)) \quad (1)$$

where d_i represents the i^{th} report sentence and s_t represents the t^{th} golden summary sentence.

B. Extractor Agent

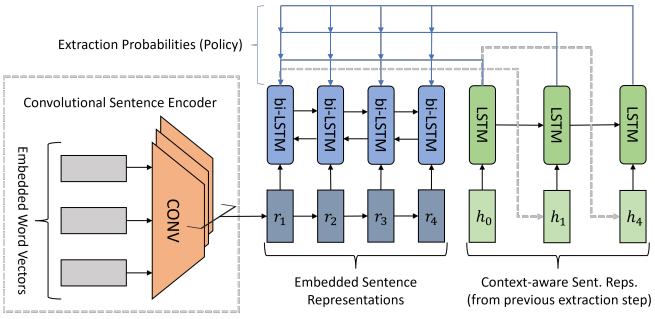

Fig. 2. The convolutional encoder computes representation r_j for each sentence. The RNN encoder computes context-aware representation h_j and then the RNN decoder selects sentence j_t at time step t. With j_t selected, h_{j_t} will be fed into the decoder at time $t+1$.

Extraction step is performed by extractor which architecture is displayed in Fig.2. The extractor agent is designed to model a function which main purpose is to extract salient

[1]Since in financial texts, authors often use acronyms and/or abbreviations, we decided to apply a regex filter in order to "unpack" these abbreviations. Some naive examples could be the transformations $b \rightarrow billions$, $CDS \rightarrow credit\ default\ swap$ or $\$ \rightarrow dollars$.

sentences from the documents. We exploit a hierarchical neural model to learn the sentence representations of the document and a 'selection network' to extract sentences based on their representation, according to Chen et al. [2]. The neural model converts the embeddings created by word2vec into a sentence representation using convolutional filters of sizes 3, 4 and 5 and, then, it uses ReLU and max-pooling in order to obtain the embedding of the sentence h_j. Then, the representation of the sentence goes into a bidirectional LSTM and, in the end, another LSTM is added to train a Pointer Network to extract sentences (Vinyals et al. [18]). The probability a sentence has to be extracted is computed as follows:

$$u_j^t = v_p^T tanh(W_{p1} h_j + W_{p2} e_t) \tag{2}$$

$$P(j_t|j_1, ..., j_{t-1}) = softmax(u_t) \tag{3}$$

where j_t is the selected sentence at time t, while W and v are trainable parameters and e_t is the context vector that is computed as follows:

$$a_j^t = v_g^T tanh(W_{g1} h_j + W_{g2} z_t) \tag{4}$$

$$\alpha^t = softmax(a^t) \tag{5}$$

$$e_t = \sum_j \alpha_j^t W_{g1} h_j \tag{6}$$

where z_t is the output at step t of Pointer Network (Vinalys et al. [18]).

C. Abstraction

The Abstraction Network approximates the function that compresses and paraphrases sentences. Model used is a standard encoder-decoder, according to Bahdanau et al. [19]. In addition to that, a Copy Mechanism (as seen in See et al. [20]) has been implemented to extend the decoder to predict over the extended vocabulary of words in the input document. Then, generation probability is computed as follows:

$$p_{gen} = sigmoid(w_{h*}^T h_*^T + w_s^T s_T + w_x^T x_t + b_{ptr}) \tag{7}$$

where all the parameters are learnable parameters. In the end, the probability distribution over all the *extended vocabulary* (the union of the vocabulary, and all words appearing in the source document) is computed:

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} \alpha_i^t \tag{8}$$

Where P_{vocab} is the probability over the Word2Vec vocabulary and α_i^t is the ith element of the attention distribution. Note that if w is an out-of-vocabulary (OOV) word, then $P_{vocab}(w)$ is zero; similarly if w does not appear in the source document, then $\sum_{i:w_i=w} \alpha_i^t$ is zero. In the end, the abstractor is trained as an usual sequence-to-sequence model. Specifically, the loss-function we aim to minimize is a *cross entropy loss* according to the following:

$$L = -\frac{1}{M} \sum_{i=1}^{M} (log(P_{\theta_{abs}}(w_m|w_{1:m-1})) \tag{9}$$

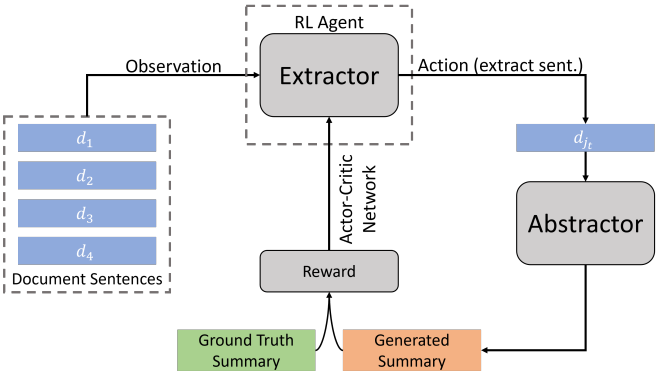

Fig. 3. Reinforced training of the extractor (for one extraction step) and its interaction with the abstractor.

Where M is the output size, θ_{abs} is the set of trainable parameters of the abstractor and w_m is the m-th extracted word.

D. Reinforcement Learning

Reinforcement Learning is a technique based on Markov Decision Process. At each step t, the agent takes into consideration the current state c_t and choices an action j_t to perform from the set of all possible actions. Between all the document sentences d_j, it extracts a sentence d_{j_t} and receives a reward $r(t+1)$ depending on the paraphrased version of the extracted sentence generated by the abstractor $g(d_{j_t})$.

$$r(t+1) = ROUGE\text{-}2_{F_1}(g(d_{j_t}), s_t) \tag{10}$$

To solve high variance problem which is intrinsic to the model, a critic network is necessary. In fact, the former, computes the prediction of expected reward for all possible action at a certain moment t. This baseline is then used to get the advantage function. More specifically, this network's aim is to minimize MSE loss between the reward in input at time t and the baseline we computed as explained before. In other words, actions leading to high ROUGE scores are encouraged while, on the other side, actions which had lead to poor ROUGE scores are discarded and not chosen anymore. An entire sketch of the process is provided in Fig. 3.

IV. Experiments

A. Datasets

Two large datasets are used to conduct experiments on our model. The former is the FNS2022 dataset, which is an extention of the FNS2021 dataset, while the latter is the CNN/DM-HD dataset (CNN/Daily Mail-Document with Headlines). The details are shown in the Table I.

1) FNS2022 dataset: has been proposed at the 2022 Financial Summarization Shared Task [9]. In this competition, the task is to generate summaries of financial annual reports written in 3 different languages: English, Greek, and Spanish. Only the training and validation sets of the competition have been published, so we decide to use the validation set provided

as a test set, while another validation set is extracted from the train set (20%).

2) CNN/DM-HD dataset: was built by Song et al. [16] based on the CNN/Daily Mail dataset [3], which contains online newspaper articles coupled with multi-sentence summaries, and to which article headlines have been added.

Dataset	Language	N. of documents	AVG N. of sentences
FNS2022	English	3,363	1382.71
	Greek	212	904.3
	Spanish	212	1295.66
CNN/DM-HD	English	304,512	33.28

TABLE I
DATASETS DETAILS.

B. Experimental Setup

We build our neural networks using the PyTorch framework [21]. The training of the extraction, abstraction, and reinforcement modules is carried out in the Google Colab environment, using a 12GB Tesla P100 GPU.

We use the NLTK library to perform preprocessing [22], while we use the word2vec model from the Gensim library to implement a custom word embedding model [23].

We adopt the `@jit` decorator from the Numba library to optimize the preprocessing phase. Numba is an high performance Python compiler that translates Python's functions to optimized machine code at runtime [24]. In particular, we are able to reduce the computation time of the label generation and distribution analysis steps by 60%.

Metrics used for evaluating the performances of our method are ROUGE-1, ROUGE-2 and ROUGE-L scores.

C. Distributional Analysis

Since the financial reports collected in FNS2022 are quite long, we decide to perform a distributional analysis. The idea is to have, for each language, a global understanding of the distribution of the most salient sections, so that we can later cut the documents according to that. This work has to be carried out to train a model that is as general as possible, but which is also able to perform with the limited resources available. Recognizing the portions of the documents that are on average the most relevant in the training set, avoids losing information and allows us to generalize the cutting of documents on the test set.

The assumption underlying our analysis is that sections lengths are proportional to the length of documents, thus, we can study sections importance. For instance, if we have a document made of 500 sentences and a document made of 5000 sentences, our assumption allows us to say that, if the introduction section of the former document contains x sentences, then the introduction section of the latter document will contain $10x$ sentences. More specifically, the steps we implement are the following:

1) Each document is divided in 100 "buckets" (this number is chosen arbitrarily, not knowing the structure of the reports.).
2) For each bucket, we compute the total score of the bucket. In particular, for each gold summary, we compute ROUGE-L between its sentences and the sentences of the corresponding article. For each gold sentence, the 10 article sentences having the highest rouge scores (eq.12 are taken using $top\_M = 10$), and for each "top" sentence, the score is added to the one of the corresponding bucket.

$$score_i = \sum_{d \in D} \sum_{s \in d \cap b_i} score_s \quad \forall i = 1, ..., 100 \tag{11}$$

$$score_s = \sum_{g \in G_d} \mathbb{1}_{\{s \in top\_M_g\}} ROUGE\text{-}L(s, g) \tag{12}$$

where d is a document belonging to the set of documents D, s is a sentence in the document d and g is a sentence of the document gold summary G_d. In the end, $\mathbb{1}_{\{s \in top\_M\}}$ is a function which value is 0 if the sentence is not in the top_M sentences and 1 otherwise.

Plotting the obtained scores distribution, we notice that distributions change their behaviour based on the lengths of the documents we were considering. Hence, to train our model, documents are cut according to three different distributions, based on the different documents lengths (0-500, 500-1000, 1000+). Appendix A describes all the reasons behind our decisions and show all the analysis and the distribution plots.

D. Extractive Labels Generation Using Numba

According to what has been written formerly, using Numba's `@jit` decorator allow us to save a lot of time. To be able to increase computation speed we had to convert all the functions used for the numerical computations needed for the labels generation. More precisely we had to use only the numpy library, because we have to use a low level library to allow numba to process data fast, as a consequence we can not use dictionaries, sets and lists. The huge amount of time needed is due to the ROUGE-L computation, more precisely to the computation of the LCS (Longest Common Subsequence), which has a computational complexity O(m*n), where m and n are the lengths of the article and of the gold.

Since label creation is the preprocessing step that wastes most of the time, adopting such a decorator allowed us to obtain a time saving up to 60%. In Table II, the average times to create a label with and without `@jit` are displayed.

Documents	Numba	No Numba
Entire document	7.37s	12.18s
cut 100	0.55s	1.04s
cut 500	4.34s	11.23s

TABLE II
THE AVERAGE TIME TO GENERATE A LABEL IN DIFFERENT SET UPS. THE EXPERIMENTS HAVE BEEN CONDUCTED ON SPANISH DATASET.

To understand better what is the real time saving we have, let's consider the English dataset, where we have 3363 labels to generate. If we cut documents to the first 500 rows, using the decorator `@jit` we pass from 10 and half hours to generate labels to only 4 hours.

E. Results

1) FNS 2022 Dataset: Table III reports the best scores obtained on the FNS2022 dataset, divided by language and obtained cutting documents on the first N rows, while in Table IV we can see the results obtained cutting documents according to the distributions. Due to the computational limitations of our environment, we adopt parameters different from the original ones to adapt the model to each subset of the dataset. Reinforcement Learning is optimized to maximize the ROUGE-2 [25]. Following Chen et al. [2] and Zmander et al. [1] we use ROUGE-2 F1, but we also experimented using ROUGE-2 Recall, since we have to generate summaries of limited length to 1000 words.

Language	HypPar	RType	R-1	R-2	R-L
English	n_hidden=128 n_LSTM=1 batch_size=4 cut_size=500	F1 Recall	0.333 0.319	0.078 0.056	0.324 0.310
Greek	n_hidden=256 n_LSTM=2 batch_size=2 cut_size=1000	F1 Recall	0.432 0.254	0.256 0.142	0.422 0.250
Spanish	n_hidden=128 n_LSTM=1 batch_size=2 cut_size=500	F1 Recall	0.372 0.255	0.102 0.087	0.360 0.249

TABLE III
MODEL HYPERPARAMETERS WITH THE ASSOCIATED ROUGE SCORES USING DOCUMENTS CUT TO THE FIRST N ROWS, WHERE N=CUT_SIZE.

Language	HypPar	RType	R-1	R-2	R-L
English	n_hidden=128 n_LSTM=1 batch_size=4	F1 Recall	0.332 0.317	0.118 0.123	0.326 0.310
Greek	n_hidden=256 n_LSTM=2 batch_size=4	F1 Recall	0.489 0.227	0.311 0.140	0.479 0.224
Spanish	n_hidden=128 n_LSTM=1 batch_size=4	F1 Recall	0.340 0.292	0.094 0.081	0.334 0.286

TABLE IV
MODEL HYPERPARAMETERS WITH THE ASSOCIATED ROUGE SCORES USING DOCUMENTS CUT ACCORDING TO THE DISTRIBUTIONS.

The performance of the model is significantly different depending on the dataset. As a matter of fact, using the distribution cut method allow us to improve the performances for English and Greek languages. On the other side, Spanish's performances got a little downgrade using distribution cut and this could be due to the fact that Spanish's distribution is uniform. As a consequence, using the first 500 rows catches more information than the ones captured by the distribution cut.

Concerning the differences between the usage of the ROUGE-2 F1 and the ROUGE-2 Recall for the RL agent training, we can notice that the F1 performances are slightly better than the Recall ones, except for the Greek, where the model trained with F1 outperforms the one with Recall.

2) CNN Dataset: In table V, the results we got for CNN's headline generation task has been displayed.

Language	R-1	R-2	R-L
English	0.100	0.031	0.092

TABLE V
RESULTS FOR CNN HEADLINE GENERATION TASK.

To try to understand the reasons behind these low scores, we analyzed the generated headlines and compared them with the gold ones. In the following we can see some examples:

News text 10013:

- **Gold**: "teacher paul johnson posed as young girl online to trick pre teen student"
- **Generated**: "gym teacher accused of posing as young girl on a social media site"

News text 114:

- **Gold**: "jeremy clarkson talks himself out of a parking ticket in london"
- **Generated**: "jeremy clarkson talked himself out of a ticket after leaving his green lamborghini laughing with a traffic warden who had tried to put a ticket on his borrowed supercar"

News text 9996:

- **Gold**: "watford players party on the team bus following fairytale return to premier league"
- **Generated**: "watford dramatically clinched to the premier league"

It's possible to notice generated titles are quite good for the articles they refer to. Nevertheless gold and generated titles have a very similar meaning, the texts are not composed by the same words, so an evaluation made using a ROUGE score could not lead to numerical high performances.

F. Ablation Study

During the data exploration part, we noticed that the gold-summaries of the English dataset were more "extractive" with respect to other languages. So we tried to train a model without the abstractor to see if it could be the cause of the high differences between the results obtained for the different languages. The following tables VI and VII displays the results we managed to obtain.

As we expected, if we use only the extractor to train our model, then we obtain quite well results for the English dataset, while we drop down the performances over the Greek and the Spanish datasets. This is true for both the cut methods we use, first N rows or distribution.

V. CONCLUSION

In this study, through a review of the works of Zmandar et al. [1] and Chen et al. [2], we managed to provide a solution to the task of summarizing long financial documents. The proposed approach, a combination between Extractive Summarization and Abstractive Summarization through a Reinforcement Learning policy, performed quite well, with obtained results comparable with the ones of the papers we cited before.

Language	HypPar	RType	R-1	R-2	R-L
English	n_hidden=128 n_LSTM=1 batch_size=4 cut_size=500	F1 Recall	0.163 0.302	0.115 0.124	0.162 0.294
Greek	n_hidden=256 n_LSTM=2 batch_size=2 cut_size=1000	F1 Recall	0.219 0.229	0.079 0.082	0.209 0.218
Spanish	n_hidden=128 n_LSTM=1 batch_size=4 cut_size=500	F1 Recall	0.020 0.021	0.013 0.013	0.020 0.020

TABLE VI
MODEL HYPERPARAMETERS WITH THE ASSOCIATED ROUGE SCORES USING ONLY EXTRACTOR. TRAINING HAS BEEN PERFORMED CUTTING DOCUMENTS TO THE FIRST N ROWS, WHERE N=CUT_SIZE.

Language	HypPar	RType	R-1	R-2	R-L
English	n_hidden=128 n_LSTM=1 batch_size=4	F1 Recall	0.185 0.235	0.152 0.171	0.184 0.232
Greek	n_hidden=256 n_LSTM=2 batch_size=2	F1 Recall	0.162 0.083	0.065 0.040	0.157 0.081
Spanish	n_hidden=128 n_LSTM=1 batch_size=4	F1 Recall	0.052 0.077	0.025 0.038	0.050 0.074

TABLE VII
MODEL HYPERPARAMETERS WITH THE ASSOCIATED ROUGE SCORES WITHOUT USING ABSTRACTOR. TRAINING HAS BEEN PERFORMED ON THE FNS2022 DATASET CUT ACCORDING TO THE DISTRIBUTIONS.

Moreover the analysis over financial reports we have performed, allowed us to understand how the importance of the sentences is distributed over the documents. Furthermore, this analysis showed also how the distributions varies based on the length of the documents and not through the different languages. Combining all these information we were able to increase the performances of our model by far.

We conducted experiments to analyze the usage of a different Reinforcement Learning reward policy that made us understand how different metrics can perform differently for different languages. In addition to that, we decided to perform an ablation study, indeed, we decided to cut-off the abstractor. Such an analysis underlined the difference between the "quality" of the summaries. In fact, the summaries we were provided with are very "extractive" in English dataset.

Finally, we applied our method to produce headlines for news articles that are faithful to the content of the article. Despite the results seem to be poor, different titles have been provided in order to demonstrate our model generates quite representative headlines.

Future works

Since we have a dataset that gathers financial reports in several languages, a multilingual model could be applied in the future in order to exploit the knowledge of a model trained on the English dataset to documents written in other languages. For this purpose, multilingual frameworks such as sBERT can be applied [26].

Further studies on this model can be conducted on algorithmic texts, such as recipes, or to summarize medical documents, such as patients' medical reports.

In this study we decided to follow previous works and to adopt the ROUGE as our reinforcement learning policy. However, other types of reward scores may be adopted, such as BERT-score or BLEU-ROUGE F1, which could lead to improvements in the quality of the summaries.

Finally, the work of Song et al. could be further investigated [16]since they propose to use Popular Topic Attention for guiding the extractor to select attractive phrases from the article. Moreover, an idea to valorise the work of our model in headlines' generation task, could be to perform sentence embeddings on both generated title and label and compute a distance metrics between them (e.g. cosine distance/similarity) instead of using a ROUGE metric.

REFERENCES

[1] 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

[2] Y. Chen and M. Bansal, "Fast abstractive summarization with reinforce-selected sentence rewriting," *CoRR*, vol. abs/1805.11080, 2018. [Online]. Available: http://arxiv.org/abs/1805.11080

[3] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," 2015. [Online]. Available: https://arxiv.org/abs/1506.03340

[4] H. Jing and K. R. McKeown, "Cut and paste based text summarization," in *1st Meeting of the North American Chapter of the Association for Computational Linguistics*, 2000. [Online]. Available: https://aclanthology.org/A00-2024

[5] K. Knight and D. Marcu, "Statistics-based summarization - step one: Sentence compression," in *AAAI/IAAI*, 2000.

[6] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 379–389. [Online]. Available: https://aclanthology.org/D15-1044

[7] F. Liu, J. Flanigan, S. Thomson, N. Sadeh, and N. A. Smith, "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*. Denver, Colorado: Association for Computational Linguistics, May–Jun. 2015, pp. 1077–1086. [Online]. Available: https://aclanthology.org/N15-1114

[8] F. Kiyoumarsi, "Evaluation of automatic text summarizations based on human summaries," *Procedia - Social and Behavioral Sciences*, vol. 192, pp. 83–91, 2015, the Proceedings of 2nd Global Conference on Conference on Linguistics and Foreign Language Teaching. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877042815034849

[9] M. El-Haj, N. Zmandar, A. AbuRa'ed, P. Rayson, N. Pittaras, M. Litvak, G. Giannakopoulos, A. M. Sandoval, B. C. Coronado, and A. Kosmopoulos, "The Financial Narrative Summarisation Shared Task (FNS2022)," in *Proceedings of the 4th Financial Narrative Processing Workshop*, Lancaster, United Kingdom, 2022.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates,

Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
[11] S. Henß, M. Mieskes, and I. Gurevych, "A reinforcement learning approach for adaptive single- and multi-document summarization," in *GSCL*, 2015.
[12] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017. [Online]. Available: https://arxiv.org/abs/1705.04304
[13] P. Over, H. Dang, and D. Harman, "Duc in context," 2007-11-21 2007. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=50955
[14] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 1797–1807. [Online]. Available: https://aclanthology.org/D18-1206
[15] R. Zhang, J. Guo, Y. Fan, Y. Lan, J. Xu, H. Cao, and X. Cheng, "Question headline generation for news articles," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 617–626. [Online]. Available: https://doi.org/10.1145/3269206.3271711
[16] Y.-Z. Song, H.-H. Shuai, S.-L. Yeh, Y.-L. Wu, L.-W. Ku, and W.-C. Peng, "Attractive or faithful? popularity-reinforced learning for inspired headline generation," 2020. [Online]. Available: https://arxiv.org/abs/2002.02095
[17] R. Nallapati, B. Zhou, C. N. d. santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," 2016. [Online]. Available: https://arxiv.org/abs/1602.06023
[18] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," 2015. [Online]. Available: https://arxiv.org/abs/1506.03134
[19] 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
[20] 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
[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[22] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.
[23] R. Rehurek and P. Sojka, "Gensim–python framework for vector space modelling," *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.
[24] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.
[25] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013
[26] N. Reimers and I. Gurevych, "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, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

Appendix A
Distribution Analysis

Starting from what we previously said in the Distributional Analysis subsection, here we further analyze the documents distributions for the different languages.

Fig.5 shows that, for the English documents, the scores distribution over the 100 buckets changes as we change the number of the article sentences more similar to the target abstract sentence. The more are the sentences we consider for each abstract sentence, the more the distribution becomes similar to a uniform distribution. The number of sentences considered for each abstract sentence is represented by top_M. We decide to use $top\_M = 10$ to carry out this analysis in order to obtain a fairly general distribution, but without considering excessive information.

Fig.6 is a comparison between the distribution we just talked about (the one with $top\_M = 10$) and its weighted version. In this weighted version, the score of each bucket b_i is no more the sum of the scores of the sentences s belonging to it, but the weighted average of them, where the weight of each sentence score is the length of the sentence document.

$$score_i = \frac{\sum_{d \in D} \left(|d| \sum_{s \in d \cap b_i} score_s \right)}{\sum_{d \in D} |d|} \quad \forall i = 1, ..., 100 \tag{13}$$

where $score_s$ is computed as in 12.

From this comparison, we notice that the two distributions are slightly different, so we decide to analyze deeper if the distribution is influenced by the length of the documents. In Fig.4 we can see as it is true that the length of the considered documents influences differently the distribution based on 100 sections. This is why we decided to use the different distributions of figures 4, 7 and 8 (English, Greek and Spanish distributions by documents length) to cut documents based on their length.

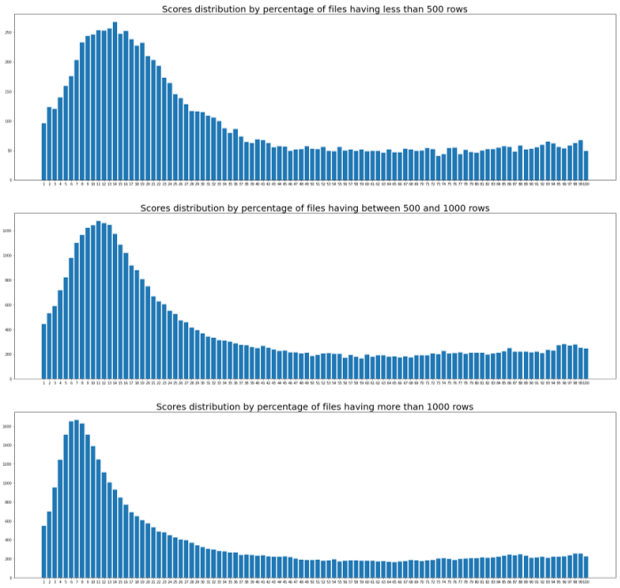

Fig. 4. English scores distributions comparison based on the length of documents, with top_M=10

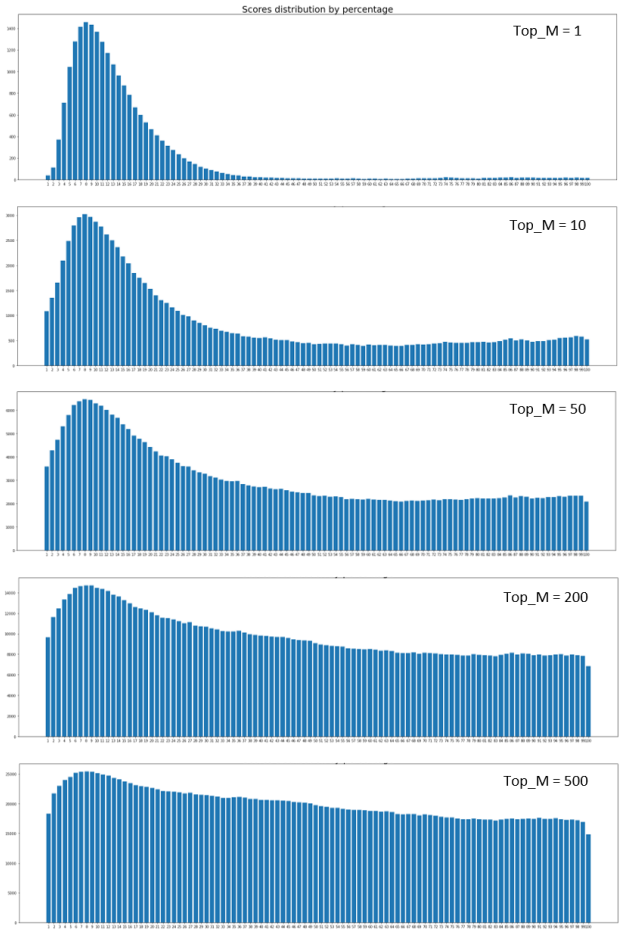

Fig. 5. English scores distributions by percentage: comparison with different top_M values

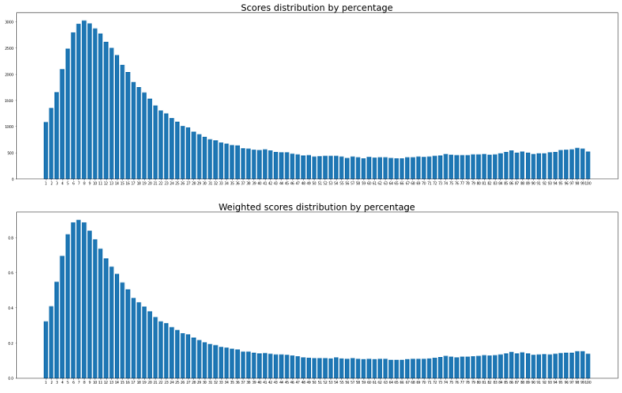

Fig. 6. English scores distributions by percentage and by weighted percentage, with top_M=10

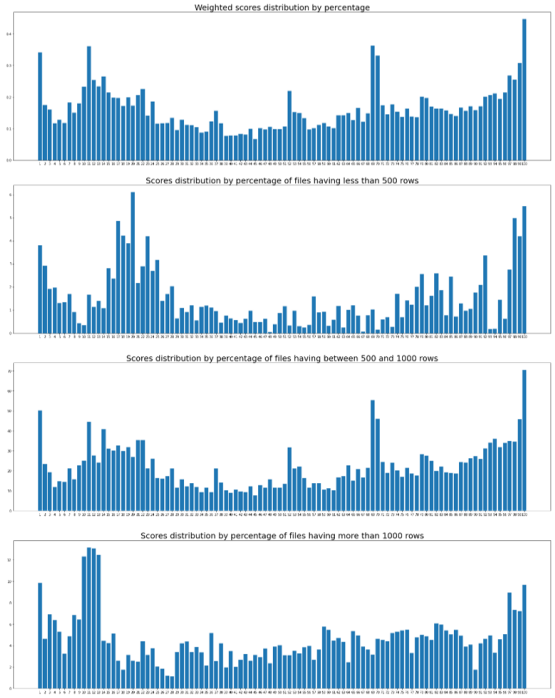

Fig. 7. The first plot represents the Greek scores distributions by weighted percentage, while the other three plots represent the scores distributions comparison based on the length of documents. We use top_M=10

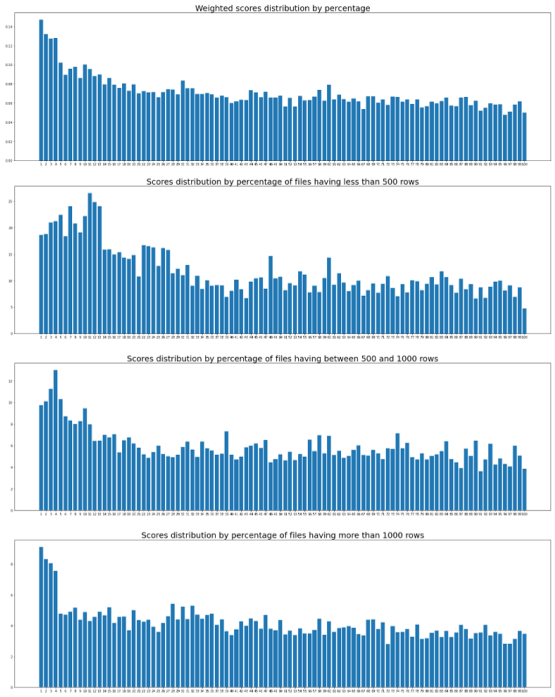

Fig. 8. The first plot represents the Spanish scores distributions by weighted percentage, while the other three plots represent the scores distributions comparison based on the length of documents. We use top_M=10