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.

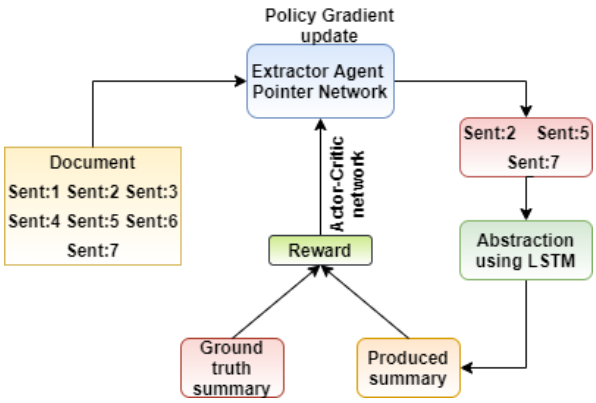

Figure 1: The end to end summarizer

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