paht_nlp @ MEDIQA 2021: Multi-grained Query Focused Multi-Answer Summarization

Wei Zhu[1] *, **Yilong He**[2], **Ling Chai**[2], **Yunxiao Fan**[2],
Yuan Ni[2], **Guotong Xie**[2], **Xiaoling Wang**[1]
[1] East China Normal University, Shanghai, China
[2] Pingan Health Tech, Shanghai/Beijing, China

Abstract

In this article, we describe our systems for the MEDIQA 2021 Shared Tasks. First, we will describe our method for the second task, Multi-Answer Summarization (MAS). For extractive summarization, two series of methods are applied. The first one follows Xu and Lapata (2020). First a RoBERTa model is first applied to give a local ranking of the candidate sentences. Then a Markov Chain model is applied to evaluate the sentences globally. The second method applies cross-sentence contextualization to improve the local ranking and discard the global ranking step. Our methods achieve **the 1st Place** in the MAS task. For the question summarization (QS) and radiology report summarization (RRS) tasks, we explore how end-to-end pre-trained seq2seq model perform. A series of tricks for improving the fine-tuning performances are validated.

1 Introduction

Automatic summarization is an essential task in the medical domain. It is time consuming for users to read a lot of medical documents when they use a search engine like Google, Medline, etc, about some topic and obtain a list of documents which are potential answers. First, the contents might be too specialized for layman to understand. Second, one document may not answer the query completely, and the users might have to summarize the conclusions across multiple documents, which may lead to waste of time or misunderstanding. In order to improve the users' experiences when using medical applications, automatic summarization techniques are required.

The MEDIQA 2021 shared tasks are held to investigate the current state of the art summarization models, especially how they perform in the medical domains. Three tasks are held. The first one is Question Summarization (QS), which summarizes long and potentially complex consumer health questions into simple ones, which are proven to be beneficial for automatic question answering. Empirical QA studies based on manual expert summarization of these questions showed a substantial improvement of 58% in performance (Abacha and Demner-Fushman, 2019). The second task is Multi-Answer Summarization (MAS) (Savery et al., 2020). Different answers can bring complementary perspectives that are likely to benefit the users of QA systems. The goal of this task is to develop a system that can aggregate and summarize the answers scattered in multiple documents. The third task is Radiology Reports Summarization (RRS) (Zhang et al., 2018, 2020b), which is to generate radiology impression statements by summarizing textual findings written by radiologists. which have several applications. First, it can speed up the technicians' workflow. Second, a system can extract the information in the reports and summarize into sentences that a layman can understand.

In the MAS task, we improve upon (Xu and Lapata, 2020) via three methods. First, during the coarse ranking of a sentence in one of the given documents, we also add the surrounding sentences as input and use two special tokens marking the positions of the sentence. This modification improves the coarse ranking with a large margin. Second, during fine-grained re-ranking, instead of incorporating a inverse sentence frequency (IFS) score based similarity matrix between sentences in the Markov chain model, we find that directly using semantic similarity scores to form the similarity matrix performs better. Third, due to the low resource settings of this task, we find that applying a RoBERTa (Liu et al., 2019) model which is already fine-tuned on the MS-MACRO task (Campos et al., 2016) can be beneficial.

For the other two tasks, we mainly explore how the pre-trained seq2seq model like BART (Lewis et al., 2020), PEGASUS (Zhang et al., 2020a), etc, can perform in these tasks. Two take-aways can

*Contact: 52205901018@stu.ecnu.edu.cn.

Proceedings of the BioNLP 2021 workshop, pages 96–102

be made. First, for tasks with small dataset size, freezing a part of the transformer blocks can be beneficial. Second, for the RRS task, we find that controlling the maximum output sequence length can improve the ROUGE score on the test set.

Our team PAHT_NLP participate in all the three tasks, and won the 1st place in the MAS task. Experiments will show that our modifications are beneficial for both stage of the MAS task. We also report extensive experiments for task 1 and task 3.

2 Multi-grained Multi-Answer Summarization

2.1 problem formulation

Let Q denote a query, and $D = \{d_1, d_2, ..., d_M\}$ a set of documents returned by the search engine or a question answering system (e.g., the ChiQA system ((Demner-Fushman et al., 2020))). It is often assumed (e.g., in our MAS task) that Q consists of a short question (e.g., Will influenza be the next pandemic?).

We implement the multi-grained MDS following Xu and Lapata (2020). We first decompose documents into segments, i.e., sentences. Then, a trained RoBERTa model quantifies the semantic similarities between a selected sentence and the query, which give importance estimations of the sentences based the sentence itself or their local contexts (Local Estimator). Third, to give a global estimations of the importance of the segments to the summary, we apply a Markov Chain (Erkan and Radev, 2004) based estimator (Global Estimator).

2.2 Local Estimator

We leverage fine-tuned pretrained language models as our evidence estimator, and use the trained estimators to rank the answer candidates.

Let Q denote a query sequence and $\{S_1, S_2, ..., S_N\}$ the set of candidate answers. Our training objective is to find the correct answers within this set. We leverage RoBERTa as our sequence encoder. We concatenate query Q and candidate sentence S into a sequence $< s >, Q, < /s >, < /s >, S, < /s >$ as the input to the RoBERTa encoder (we pad each sequence in a mini-batch of L tokens). The starting $< s >$ token's vector representations t serves as input to a single layer feed forward layer to obtain the distribution over positive and negative classes:

$$p_k = \frac{1}{Z} \exp\left(t^T W_{:,k}\right), \tag{1}$$

where $k = 0, 1, 1$ denoting that a sentence contains the answer and 0 otherwise. Z is the normalizing factor, and matrix $W = [W_{:,0}; W_{:,1}] \in \mathbf{R}^{d \times 2}$ is a learn-able parameter. We use a cross entropy loss as the training objective:

$$L = -\sum_{i=1}^{N} (y \log p_1^i + (1 - y) \log p_0^i). \tag{2}$$

After finetuning, the probability of the positive class is regarded as the local evidence score and we will use it to rank all the sentences for each query.

2.3 Global Estimator

Although our local estimator measures the semantic relevance between the query and the candidate segments, these estimation is done locally. To obtain a global estimation of the scores for each segment, we apply a Global Estimator following (Xu and Lapata, 2020). The centrality estimator essentially is an extension of the well-known LexRank algorithm (Erkan and Radev, 2004).

For each document cluster, i.e., the collections of documents for each query in our tasks, LexRank builds a graph $G = (V; E)$ with nodes V corresponding to sentences and undirected edges E whose weights are computed based on a certain similarity metric. The original LexRank algorithm uses TF-IDF (Term Frequency Inverse Document Frequency). (Xu and Lapata, 2020) proposes to use TF-ISF (Term Frequency Inverse Sentence Frequency), which is similar to TF-IDF but operates at the sentence level.

Following ((Xu and Lapata, 2020)), we integrate our evidence estimator into the similarity matrix E, that is,

$$\tilde{E} = w * [\tilde{q}; ...; \tilde{q}] + (1 - w) * E, \tag{3}$$

where $w \in (0, 1)$ controls the extent to which the evidence estimator can influence the final summarization, and $\tilde{q}$ is obtained by normalizing the evidence scores,

$$\tilde{q} = \frac{q}{\sum_v^{|V|} q_v}. \tag{4}$$

Note the similarity matrix E can be seen as the transition probabilities. If the similarity score $E_{i,j}$ between sentence i and j is higher, it is more likely that sentence i and j are both selected in the finally summary or are discarded at the same time. We can see selecting the sentences into summaries as

a Markov chain process, and we will leverage the final stationary distribution $\tilde{q}^*$ of this Markov chain as the final scores of each segment. $\tilde{q}^*$ is obtained by solving this equation:

$$\tilde{q}^* = \tilde{q}^* \tilde{E} \quad (5)$$

Note that with our evidence estimator and centrality estimator, $\tilde{q}^*$ can simultaneously expresses the importance of a sentence in the document and its semantic relation to the query. Thus, to formulate the final summary, we rank the sentences based on $\tilde{q}^*$ and select the top k^{sum} ones.

3 Contextualized evidence estimation

The previous section describe a two-step method for extractive MDS. However, it does not fully exploit the advantages of pretrained sentence encoders, since it only compares the query to single sentences which suffers from losing the contexts. In this section, we provide a simple method to conduct extractive MDS in one step, and promote the performances.

Let Q denote a query sequence and $\{S_1, S_2, ..., S_N\}$ the set of candidate answers. And we put each sentence S_i back into its contexts by concatenating the sentences surrounding it. Denote the S_i with its contexts as $C_i = [N_i^L; S_i; N_i^R]$. For implementation, we limit the sequence length of N_i by L_{max}, which is 512 for RoBERTa. For formulating the input of RoBERTa, we concatenate C_i following its sequential order, so that its contexts is not corrupted. Thus the sequence input should be like $< s >, Q, < /s >, < s >, N_i^L, < /s >, < s >, S_i, < /s >, < s >, N_i^R, < /s >$.

The above operation adds the contextual information of S_i, but the position of S_i is not emphasized, and the model might focus on N_i^R or N_i^L instead of S_i. Thus, we add a pair of special tokens before and after S_i to address the position of the sentence we are concerning. Thus, the input sequence becomes $< s >, Q, < /s >, < s >, N_i^L, < /s >, < s >, < t1 >, S_i, < t2 >, < /s >, < s >, N_i^R, < /s >$.

The RoBERTa will encode the above sequence and outputs the semantic relevance score, which we will use as the final semantic score of the sentence regarding summarization.

4 End-to-end abstractive summarization

Pre-trained models. In this section, we experiment on applying pretrained Seq2Seq models to obtain abstractive summarizations, after finetuning their on our datasets. We mainly investigate two types of models, BART ((Lewis et al., 2020)) and PEGASUS ((Zhang et al., 2020a)).

In terms of architecture, BART adopts a standard transformer seq2seq architecture ((Vaswani et al., 2017)) with some small changes. It uses GeLU (xxx,) rather than ReLU (xxx,) as activation function and initiates paramaters with normal distribution. For pre-training tasks, BART allows arbitrary noising transformations of input texts and learns a model to rebuild original text. BART achieves the state-of-the-art (SOTA) results on a wide range of tasks, including summarization and machine translation.

PEGASUS uses pre-training objectives tailored for abstractive text summarization. During pre-training, the text inputs are documents with several important missing sentences and the output is the predicted missing sentence sequences. PEGASUS can perform quite well on summarization tasks with low resources, e.g., when the training sets only contains only hundreds of samples.

Finetuning techniques. For finetuning the pre-trained seq2seq models, we experiment a few methods/techniques which can improve the downstream task performances:

- Freezing parameters. For tasks like QS and MAS, the training dataset is quite small and the large pre-trained models can be easily overfitting. We alleviate the overfitting problem by freezing the lower layers of the models.
- We use the advarsarial training method, i.e., Projected Gradient Descent (PGD, (Madry et al., 2018)) for more robust fine-tuning.
- Back translation from English to Chinese, and Chinese to English is applied for data augmentation.

5 Experiments on MAS

In task 2, We used two methods to deal with the problem of low resource data. The first method is to add muti-ext-summary and single-ext-summary as targets to the training data. Since some sentences in the summary are not exactly the same as the sentences in the article, the Jaccard similarity is used to align the sentences in article to the sentences in the extractive summary. Because the final target is multi-text-summary, in order to increase its

model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
		dev set		
roberta-large	56.95	48.11	41.36	56.29
+marco	57.08	48.15	42.10	56.33
+marco+reverse	57.62	49.47	41.90	56.99
+marco+lexrank	57.06	48.31	42.04	56.07
+marco+context	57.57	48.62	42.06	56.75
electra-large+marco	58.53	49.46	42.35	57.84
ensemble-model	59.29	51.09	43.80	58.88
		test set		
ensemble-model	58.5	50.8	43.5	-

Table 1: Comparison of different models on dev set in Task 2. Marco means using ms-marco data pretrain model, reverse means inverting Q and S on the input refer to (Su et al., 2020) , lexrank means using lexrank to get the global score of the sentence described in section 2.3, context means adding Contextual information described in section 3

weight, we repeatedly sampled sentences in multi-ext-summary and added it to the training set. The second method, public dataset ms-marco is used to pre-train the RoBERTa model.

Finally, the top 20 sentences based on the model score are selected and we restore their relative positions by recording the position of each sentence in the article in advance as the target. The result is shown in Table 1. As roberta-large as a baseline model, both resampling and pretraining by ms-marco have slightly improved the result of the model because of the increasing of training set. Although the lexRank method described in section 2.3 has made a improvement, the weight of model score must be a large value compared to the TF-ISF, for example 0.99 in our model. For contextualized evidence estimation described in section 3, we selected the two sentences before and after as the context and this method greatly improves the model. Referring to (Su et al., 2020), we tried to concat the question and the sentence like <s>, S, </s>,</s>,Q,</s>, this method has achieved competitive results in validation set, but the result in test set has slightly decreased. In addition, we also tried the ELECTRA (Clark et al., 2020) model and achieved a competitive results in validation set compared to RoBERTa. Ensemble model uses all models mentioned above, and weighted sum all scores of model for one sentence based on the results normalized ROUGE-2 score in validation set. The ensemble model achieves the best results on the validation set.

Our model is optimized with Adam on one Tesla V100 GPU using the following parameters: learning rate = 1e-5 batch size = 16, maximum length = 128. The learning rate is warmed up over the first 1 epoch. Early stopping strategy for 5 epoch is used to select the optimal model

In the end, we submitted the results of ensemble model and achieved the first place, as shown in Table 1

6 Experiments on QS

At first, we compare the end-to-end abstractive methods on an 8:2 split at the train set, shown in Table 2. The result shows that the PEGASUS-large model with 3-freezed-layer encoder and 3-freezed-layer decoder gains the highest score. Training on the whole training set and evaluating on the official validation set, the model performs shown in Table 3, without the question type nor question focus given. We try to do data augmentation, like translating the train data to Chinese and German and then translating back to English, but have failed to improve the result. When concatenating the two kinds of information with the original message, we find that the result has been improved (Table 3).

Over CHQA datasets, we train a span prediction model based on the pointer networks and a question type classification model to predict the question focus and question type, respectively. The span prediction model obtains the performance of 83% exact match F1, and the question type classification model achieves 78% F1. Based on those two models, we process train, valid and test set to the same pattern as the input: "SUBJECT:{question_focus};{question_type} MESSAGE:{message}". Table 4 indicates the results with different parameters.

By checking the generated sentences, we find

model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
BART-base	52.33	34.93	49.91	49.90
BART-large	54.25	36.28	51.56	51.51
PEGASUS-large	51.30	34.28	49.33	49.37
PEGASUS-large(freeze=3)	56.97	38.74	54.03	54.07

Table 2: Comparison of different end-to-end models on 80% train set in Task 1

valid set	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
NO type&focus(baseline)	36.17	16.39	35.23	35.32
data augmentation	34.50	13.73	34.03	33.85
WITH type&focus	38.58	12.47	38.42	38.42

Table 3: Results of PEGASUS-large model on valid set in Task 1

the questions are highly like to be predicted as two sentence patterns: "what the treatments for ..." and "where can I find information on ...". We find these patterns appear more than 300 of 1000 train data, so we do the re-sampling for train data according to the frequency of the first four word of target questions. We train model on this re-sampled train set and get the result on valid set (Table 5). Although the score on valid set has decreased but the final score in the test set has increased. We conclude that the improvement are due to the higher diversity of the sentence patterns.

7 Experiments on RRS

Table 6 reports the main results on 80% training set with the most popular end-to-end models for summarization task currently. When using a 8:2 split at official training set, we find that PEGASUS-large model outperforms all other models with a 2% difference of ROUGE-1. We also test PEGASUS-pubmed but find suprising low performances, indicating that pubmed corpus does not fit to our tasks.

Table 7 analyses how different freezing strategies influence model performances. We consider freezing two different kinds of layers in structure: embedding layers and encoder layers. So, there are four combinations of strategies. As for BART-base model, we can see that models with frozen encoder layers fall far behind models freezing none of encoder layers, indicating that encoder layers are more important than embedding layers. It is interesting that freezing embeding layers sometimes helps BART models perform better while other models worse. As a result, We than use stratgies of freezing embedding layers or freezing no layers to our subsequent trainging settings.

According to the results of table1, we choose PEGASUS as our best model. PEGASUS models stand out from other popular models due to their specially designed pretrain tasks. We test how different optimizers influence performances. Table 8 also reveals that using adafactor will raise the ROUGE-2 metric by 2%. From the data we have, private information of patients will be replaced by token "___", which absolutely will not appear in the vocabulary of PEGASUS. Considering the fact that summaries also contain this special token, we test whether adding this to vocabulary will help models perform better. The results show that this operation decreases the performance a little bit, possibly because of not having a good initial value for the added token in embedding space.

By analysing data carefully, we find that almost half of the summaries start with pattern like "No acute ..." or "No evidence of ...". A simple idea is that we can separate the data according to the pattern into two kinds, one with pattern of starting from "No", one with other patterns, and train models separately. When predicting, we also need a classifier to classify samples and send samples into according models. We label samples of which summaries start with "No ..." as label 1, and label other samples as label 0. We than train PEGASUS-large models to generate summaries and BERT-base model to classify. The results are shown on Table 9.

Considering our classifier does make mistakes when predicting, we set a threshold of 0.75. Only when the classifier give samples probabilities higher than this, will we use the separately trained models. Otherwise, we will use the wholly trained model to predict.

model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
PEGASUS-large(freeze=3)	42.83	23.50	41.47	41.33
PEGASUS-large(freeze=0)	42.97	23.93	41.73	41.57

Table 4: Results of PEGASUS-large model on valid set with question type and focus in Task 1

model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
PEGASUS-large(freeze=0)	38.30	19.68	36.68	36.94

Table 5: Results of PEGASUS-large model fine-tuned on re-sampled data in Task 1

model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
BERT-abs	49.79	35.51	46.68	46.72
BART-base	61.90	49.39	58.86	60.29
BART-large(freeze)	60.10	47.38	57.01	58.55
PEGASUS-large	**63.61**	**51.86**	**60.51**	**62.28**
PEGASUS-pubmed	30.61	19.28	26.91	29.12
T5-small	57.08	45.13	54.65	55.47
T5-base	61.77	49.30	58.72	60.34
T5-large	61.85	50.81	59.19	60.56

Table 6: a comparison of different end-to-end models on 80% training set in Task 3.

model	freeze encoder	freeze embedding	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
BART-base	yes	yes	48.68	33.78	45.88	47.37
BART-base	no	yes	**61.90**	**49.39**	**58.86**	**60.29**
BART-base	yes	no	57.48	45.57	54.75	56.10
BART-base	no	no	61.30	49.31	58.45	60.01
PEGASUS-large	no	yes	53.68	42.58	51.57	52.45
PEGASUS-large	no	no	**63.61**	**51.86**	**60.51**	**62.28**
PEGASUS-pubmed	no	yes	26.83	15.83	23.79	24.41
PEGASUS-pubmed	no	no	**30.61**	**19.28**	**26.91**	**29.12**

Table 7: a comparison of same models using different freezing strategies

model	optimizer	add vocab	ROUGE -1	ROUGE -2	ROUGE -L	ROUGE -Lsum
PEGASUS-large	adam	no	62.29	49.15	59.30	60.62
PEGASUS-large	adafactor	no	**63.07**	**51.18**	**60.06**	**61.42**
PEGASUS-large	adafactor	yes	62.99	51.10	59.97	61.34

Table 8: a comparison of PEGASUS using different optimizer and adding special token in Task 3.

pipeline part	model	acc	ROUGE-1	ROUGE-2
classification	BERT-base	88.2		
label 0	PEGASUS-large		54.02	37.34
label 1	PEGASUS-large		76.81	69.73
ensemble			61.97	50.02

Table 9: pipeline results on task3

8 Conclusion

In this work, we elaborate on the methods we employed for the three tasks in the MEDIQA 2021 shared tasks. For the extractive summarization of MAS task, we build upon Xu and Lapata (2020), and achieve improvements by adding contexts and sentence position markers. For generating abstractive summaries, we leverage the pre-trained seq2seq models. To improve the fine-tuning performances on the downstream tasks, we implement a few techniques, like freezing part of the models, adversarial training and back-translation. Our team achieves the 1st place for the MAS task.

References

Asma Ben Abacha and Dina Demner-Fushman. 2019. On the role of question summarization and information source restriction in consumer health question answering. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, 2019:117–126.

Daniel Fernando Campos, T. Nguyen, M. Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, L. Deng, and Bhaskar Mitra. 2016. Ms marco: A human generated machine reading comprehension dataset. *ArXiv*, abs/1611.09268.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555.

Dina Demner-Fushman, Yassine Mrabet, and Asma Ben Abacha. 2020. Consumer health information and question answering: helping consumers find answers to their health-related information needs. *Journal of the American Medical Informatics Association : JAMIA*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res.*, 22:457–479.

M. Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, A. Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ArXiv*, abs/1910.13461.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

A. Madry, Aleksandar Makelov, L. Schmidt, D. Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083.

Max E. Savery, Asma Ben Abacha, Soumya Gayen, and Dina Demner-Fushman. 2020. Question-driven summarization of answers to consumer health questions. *Scientific Data*, 7.

Dan Su, Yan Xu, Tiezheng Yu, Farhad Bin Siddique, Elham Barezi, and Pascale Fung. 2020. CAiRE-COVID: A Question Answering and Query-focused Multi-Document Summarization System for COVID-19 Scholarly Information Management. *ArXiv*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Yumo Xu and Mirella Lapata. 2020. Coarse-to-fine query focused multi-document summarization. In *EMNLP*.

Jingqing Zhang, Y. Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *ICML*.

Yuhao Zhang, D. Ding, Tianpei Qian, Christopher D. Manning, and C. Langlotz. 2018. Learning to summarize radiology findings. In *Louhi@EMNLP*.

Yuhao Zhang, Derek Merck, E. Tsai, Christopher D. Manning, and C. Langlotz. 2020b. Optimizing the factual correctness of a summary: A study of summarizing radiology reports. In *ACL*.