Dynamic QA Generator for Research Papers

CSC 447 Natural Language Processing - Project Report

Harshal Talele
MS Data Science
University of Rochester

Sindhu Kishore
MS Data Science
University of Rochester

Shubham Shailesh Tamhane
MS Data Science
University of Rochester

I. Abstract

Researchers spend a great deal of time studying research papers. Comprehending an entire paper in a single pass is highly difficult. A study suggests that the most efficient way to understand the paper would be using the three-pass approach, which involves reading the paper three times with a different focus each time. However, this process is tedious and time-consuming. A simpler way to understand the paper would be just to go through the abstract. However, a lot of key aspects like evaluation metrics, implementation details, mathematical explanations, etc. may be missed out in the abstract. There is a need for an efficient summarization method which helps interpret the paper in a more detailed way. Our proposed solution uses the Search and extract behaviour of the question-answering model to help in a better interpretation of the paper. For this purpose, we use a pre-trained model T5 and fine tune it on QASPER dataset modified using OpenAI.

II. Introduction

An easy way to understand the paper would be through the paper being compressed into a question and an answer form. The Search and extract behaviour of the question-answering model helps in a better interpretation of the paper. Hence the need for a QA model arises. This project aims to design and develop a Dynamic QA model for research papers. Our model will extract relevant information from research papers and provide questions and answers pertaining to it. This will help save time and effort as compared to the conventional method. The model will identify the takeaways and present them in a concise format helping the researchers get a quick comprehension.

Additionally, the model will be trained to answer user-generated questions related to the paper, further improving its effectiveness.

III. Dataset

A. QASPER Dataset

QASPER is an information-seeking question answering (QA) dataset over academic research papers. It consists of 5,049 questions over 1,585 Natural Language Processing papers. Each question is written by an NLP practitioner who read only the title and abstract of the corresponding paper, and the question seeks information present in the full text. The questions are then answered by a separate set of NLP practitioners who also provide supporting evidence to answers.

Each paper has an average of 3.2 questions, up to a maximum of 12 questions for a single paper. In addition to providing answers when the questions are answerable,the annotators were asked to select text, tables, or figures as evidence required for answering the questions. 55.5% of the questions require evidence from multiple paragraphs in the paper and 13% require tables or figures.

The annotators tasked with creating the QASPER dataset were randomly assigned papers along with their associated questions, full text, and figures and tables. They were asked to make a binary decision as to whether each question was answerable, and if so, to select the minimal set of evidence snippets necessary to answer it. In addition, annotators were asked to provide a concise answer to each question, and to indicate whether the answer was extracted from the evidence, a yes or no response, or an abstractively written summary. The annotators were instructed to prioritize text over figures and tables when selecting evidence, unless the necessary information was only present in the figures or tables.

B. QASPER Data Cleaning

In order to prepare the QASPER dataset for use as an input for the model, it was necessary to flatten the data structure and map only the relevant fields needed for training. However, the dataset did contain a certain number of unanswerable questions, which needed to be removed from consideration during preprocessing. Additionally, some of the section and evidence fields in the original research paper contained missing values, which were labeled as 'NaN' and had to be excluded from the final dataset. We utilized the selected evidence snippets as the answers to the associated questions. Notably, more than half of the questions (55.5%) required multiple evidence snippets to provide a complete answer. In these cases, we selected the first occurring evidence snippet as the answer to the corresponding question.

While the QASPER dataset offers many benefits for training and evaluating question-answering models, there are some notable limitations to consider. Oneof which is the reduced amount of data available after cleaning the dataset. Additionally, some of the questions in the dataset are too generic and lack the nuance and specificity required for the research papers. Moreover, a few of the answers provided in the dataset

are incomplete or out-of-context, which could affect the accuracy of models trained on the dataset. These limitations suggest the need for further research to improve the dataset and address these issues.

C. OpenAI Integration with QASPER

To address these limitations, we integrated OpeanAI's API services with QASPER dataset. We passed on the sections from QASPER dataset to the OpenAI's Chat Completion API built on the "gpt-3.5-turbo" model and generated question answer pairs on each section. We stored this dataset and gave this as an input to fine tune the model. To overcome the limitations of the QASPER dataset, we leveraged the OpenAI API services by integrating the dataset with the Chat Completion API, which is built on the "gpt-3.5-turbo" model. We extracted the sections from the QASPER dataset and generated question-answer pairs using the API. The resulting dataset was then used to fine-tune the model to increase its accuracy and relevance.

IV. Implementation

A. Model

Attention is All you need introduces the transformer architecture which gave birth to the Large language models like BERT, GPT, PaLM etc. Our model will be trained on a large corpus of data using a combination of supervised and unsupervised learning. We will use transfer learning to fine-tune the model T5 and RoBERTa to generate the QA model for the research papers. The model will be evaluated on BLEU score, ROUGE and QAEval etc

We used state of the art T5 model for question generation and answer generation tasks seperately. T5 (Text-to-Text Transfer Transformer) is a state-of-the-art pre-trained model developed by Google AI Language. In order to use T5 for question and answer generation, we fine-tune the model on a dataset of question-answer pairs. The fine-tuned T5 model can then be used to generate questions for a given text or generate answers to given questions.

The T5-base model used in this project is pre-trained on the C4 dataset, which was created and released along with the T5 research paper. The model was also trained on unsupervised denoising objectives using the C4 and Wiki-DPR datasets. For question answering, the model was fine-tuned on three datasets: MultiRC, ReCoRD, and BoolQ. MultiRC was developed by Khashabi et al. in 2018, ReCoRD was developed by Zhang et al. in 2018, and BoolQ was developed by Clark et al. in 2019. These datasets were chosen as they offer a range of challenges for the model, including answering questions based on multiple paragraphs of text, identifying answers in tables and figures, and determining whether a given question can be answered based on the information available. By fine-tuning the T5 model on these datasets, we aim to improve its performance on question generation and answer generation tasks.

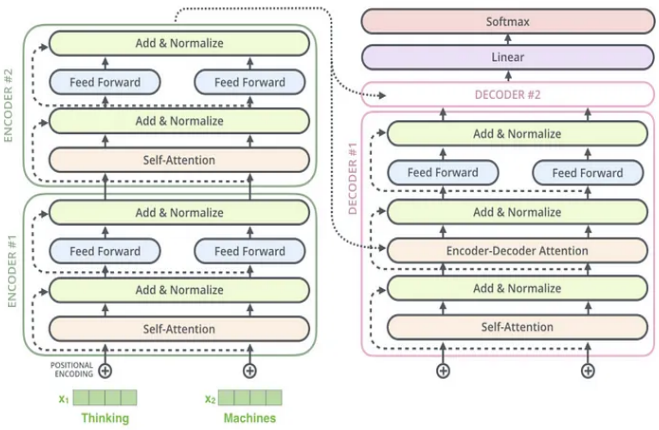

Fig. 1. T5 Architecture

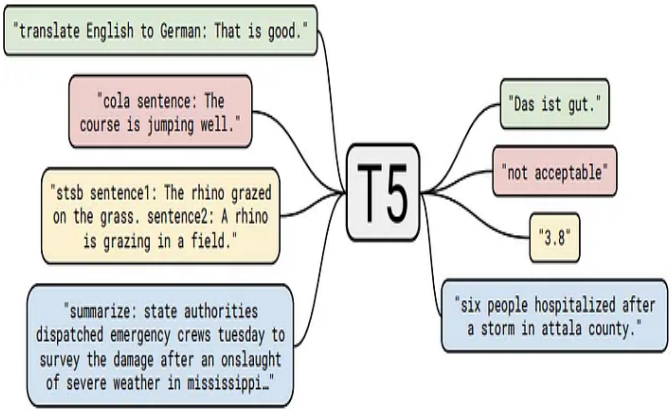

Fig. 2. T5 Architecture

B. Question Generation on QASPER only

We tried training the T5 model on QASPER dataset. However, due to the limitations of the dataset, we did not get the desired outputs. One such instance can be found in the example below

```
Generic question from QASPER -
What metric was used?

Illogical Output from model after training on QASPER -
What is the difference between the metric of metric and the metric
```

Fig. 3. T5 Architecture

C. Question Generation on QASPER with OpenAI

In our study, the final dataset consisted of 20153 question-answer pairs. For the task of question generation, we utilized the T5Model available in the simpletransformers library of Python. The T5 model requires three inputs, namely, "input_text", "target_text", and "prefix". In our case, we used "ask_question" as the prefix, while the answer was fed as

”input_text” and the resulting question was used as ”target_text”. We then fine-tune this model on this data. The model arguments can be found below -

```
model_args = {
    "reprocess_input_data": True,
    "overwrite_output_dir": True,
    "max_seq_length": 128,
    "train_batch_size": 8,
    "num_train_epochs": 1,
    "save_eval_checkpoints": True,
    "save_steps": -1,
    "use_multiprocessing": False,
    "fp16": False,
}
model = T5Model("t5", "t5-base", args=model_args, overwrite_output_dir = True)
```

Fig. 4. Arguments for Question Generation T5 model

For prediction, this fine tuned model is given the section from the research paper and it gives the respective question as the predicted output. These questions are then fed in as an input to the answer generator model to get the answers for each question.

D. Answer Generation

For the task of answer generation, we utilized the T5Model available in the simpletransformers library of Python. The T5 model requires three inputs, namely, ”input_text”, ”target_text”, ”context” and ”prefix”. In our case, we used ”question:” as the prefix, while the question was fed as ”input_text” and the resulting answer was used as ”target_text”. The sections form research paper were fed as ”context” to the model. We then fine-tune this model on this data. The model arguments can be found below -

```
model_args = {
    "reprocess_input_data": True,
    "overwrite_output_dir": True,
    "max_seq_length": 512,
    "train_batch_size": 4,
    "num_train_epochs": 2,
    "save_eval_checkpoints": True,
    "save_steps": -1,
    "use_multiprocessing": False,
    "fp16": False,
}
model2 = T5Model("t5", "t5-base", args=model_args)
```

Fig. 5. Sample output

For prediction, this fine tuned model is given the section from the research paper as its context and the previously generated questions as input and eventually, it gives the respective answer as the predicted output.

V. Output

```
[23] preds = model.predict(

        ["ask_question: " + "Figure 1 illustrates an overview of our model, which is mainly consisted of \
        three parts: word-level module, sentence-level module, and pair-level module. Token sequences of \
        sentence pairs (Arg1 and Arg2) are encoded by word-level module first and every token becomes a \
        word embedding augmented by subword and ELMo. Then these embeddings are fed to sentence-level \
        module and processed by stacked encoder blocks (CNN or RNN encoder block). \
        Every block layer outputs representation for each token. Furthermore, the \
        output of each layer is processed by bi-attention module in the pair-level module,\
        and concatenated to pair representation, which is finally sent to classifiers \
        which are multiple layer perceptrons (MLP) with softmax. \
        The model details are given in the rest of this section."]
    )

    print(preds)

Generating outputs: 100% 1/1 [00:00<00:00, 5.08it/s]
Decoding outputs: 100% 1/1 [00:00<00:00, 1.18it/s]
['What is shown in Figure 1?']
```

Fig. 6. Sample output

```
[25] preds = model.predict(

        ["ask_question: " + "The decoder is also composed of a stack of N=6 identical layers. \
        In addition to the two sub-layers in each encoder layer, the decoder inserts a third \
        sub-layer, which performs multi-head attention over the output of the encoder stack.\
        Similar to the encoder, we employ residual connections around each of the sub-layers,\
        followed by layer normalization. We also modify the self-attention sub-layer in the \
        decoder stack to prevent positions from attending to subsequent positions. \
        This masking, combined with the fact that the output embeddings are offset by one \
        position, ensures that the predictions for position i can depend only on the known \
        outputs at positions less than i"]
    )

    print(preds)

Generating outputs: 100% 1/1 [00:00<00:00, 1.26it/s]
Decoding outputs: 100% 1/1 [00:02<00:00, 2.14s/it]
['How does the decoder layer differ from the encoder stack?']
```

Fig. 7. Sample output

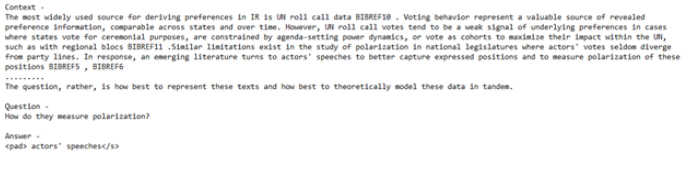

Fig. 8. Sample output

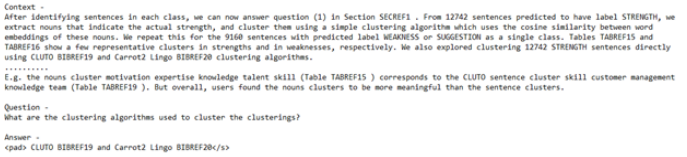

Fig. 9. Sample output

VI. Limitation

Due to limited computational resources, we used the T5-base model for our question generation and answer generation tasks, although more advanced versions such as T5-large and T5-11b are also available. The OpenAI API we used generated abstractive answers, making it difficult to determine the exact starting position of the answer in the context, which limited our ability to use certain models. Since multiple questions can be generated from the same context, there is no clearly defined ground truth, making it challenging to evaluate the generated questions. As a result, we relied on human evaluation for assessing the quality of the generated questions. It is worth noting that the accuracy of the generated answers is not always precise, which can be a limitation in some contexts.

VII. Future work

In future, we plan to support user generated queries and also provide summarization of the entire paper. Also, the system can be modified to use techniques such as sentence or paragraph indexing to specify the location of the content that has been extracted

References

[1] Deutsch, D., & Roth, D. (2020). Towards Question-Answering as an Automatic Metric for Evaluating the Content Quality of a Summary. ArXiv. /abs/2010.00490

[2] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv. /abs/1907.11692

[3] Kaggle Dataset for fine tuning - https://www.kaggle.com/datasets/Cornell-University/arxiv

[4] Narayan, S., Simoes, G., Ma, J., Craighead, H., & Mcdonald, R. (2020). QURIOUS: Question Generation Pretraining for Text Generation. ArXiv. /abs/2004.11026

[5] T5 model - https://huggingface.co/t5-base

[6] Poliak, A., Haldar, A., Rudinger, R., Hu, J. E., Pavlick, E., White, A. S., & Van Durme, B. (2018). Collecting Diverse Natural Language Inference Problems for Sentence Representation Evaluation. ArXiv. /abs/1804.08207

[7] Reisinger, Drew & Rudinger, Rachel & Ferraro, Francis & Harman, Craig & Rawlins, Kyle & Durme, Benjamin. (2015). Semantic Proto-Roles. Transactions of the Association for Computational Linguistics. 3. 475-488. 10.1162/tacl_a_00152

[8] Deutsch, D., Dror, R., & Roth, D. (2022). Re-Examining System-Level Correlations of Automatic Summarization Evaluation Metrics. ArXiv. /abs/2204.10216

[9] Wolfson, T., Geva, M., Gupta, A., Gardner, M., Goldberg, Y., Deutch, D., & Berant, J. (2020). Break It Down: A Question Understanding Benchmark. ArXiv. /abs/2001.11770