Machine Learning Approach Towards Automatic Question-Answer Generation

Akhil Eppa
Department of Computer Science
PES University
Bengaluru, India
akhil.e2k@gmail.com

Santosh Vasisht
Department of Computer Science
PES University
Bengaluru, India
santosh.vasisht@gmail.com

Punit Koujalgi
Department of Computer Science
PES University
Bengaluru, India
punit.koujalgi@gmail.com

Varun Tirthani
Department of Computer Science
PES University
Bengaluru, India
vgttir@gmail.com

Dr. Ramamoorthy Srinath
Department of Computer Science
PES University
Bengaluru, India
srinath@pes.edu

***Abstract*—The conventional way of creating question answers from long text has been tedious and time-consuming. This aspect gives an opportunity to automate this task by building an automatic question answer generation pipeline. Furthermore, with the abundance of information around us, it is essential to draw full benefit of the given data and learning through questions and answers is one promising method for this task. The proposed architecture performs an end to end processing of the input text, right from span prediction to ranking the relevant questions before presenting them. While preparing the question answer pairs, the coherence factor between the sentences is taken into account to link information across sentences. Also, to enhance the information presented to the user, a knowledge graph is utilized to gain additional domain knowledge related to text in the passage. A full fledged tool utilising the given architecture is also presented for ease of use. All these factors assist in producing a wide range of question answer pairs which will prove to be beneficial when used in different fields.**

Keywords—Transformers, SpanBERT, SQuAD, Knowledge Graph, Coherence, Question Answering

I. Introduction

An automatic question answer generation system presents information given in a passage in a manner such that it is easier to assimilate. With the advent of machine learning models and particularly the introduction of transformer networks[1] to tackle complex tasks related to natural language processing, the construction of a pipeline that can generate question answer pairs in a manner similar to what a human would do is now feasible. That is, it is possible to generate a wide range of question answer pairs from a given text without any human intervention. The main challenge lies in identifying the critical information in the text and at the same time linking information across various sentences so as to add value to the user through the generated questions and answers. Further, it becomes important to use external domain knowledge, to generate additional question-answer pairs so that it enables discovery of new information which in turn results in better learning.

Recent work [2-4] has shown the effectiveness of machine learning models at the question answering task. Further, the availability of open source datasets like SQuAD [5] has made it possible to train complex models on a large corpus. Many of the existing pipelines use an answer-aware approach for generating questions. This limits the scope of the pipeline to generate a wide variety of questions and it does not consider the aspect of linking information across sentences. To overcome these shortcomings, we utilize a coherence metric and further use a separate model just for the purpose of span extraction.

We present a complete end to end pipeline to generate a wide variety of question answer pairs given an input passage. We utilize a model to extract spans before generating any questions. The coherence between sentences is taken into account by encoding individual sentences present in a given piece of text and calculating a similarity score which is stored into a coherence matrix. These scores are utilized to merge sentences before extracting the spans. Questions are generated using the available spans and additional domain knowledge is gained by passing the span text to a knowledge graph. Finally the answers are prepared to the generated question, and the answers are ranked according to their relevance probabilistic outcome.

The contributions of this paper include the presentation of an end to end question-answer pairs generator which is composed of various interlinked modules. The primary modules are the span extractor, question generator, answer generator and a ranker for the question answer pairs. Besides this, we also present a novel method to calculate coherence so as to generate more meaningful spans which in turn result in more variety and accurate question answer pairs to be generated. We interlink the various modules and also make use of an additional knowledge graph to make the question answer generation pipeline robust enough to handle a variety of input text and generate meaningful results. An in-depth description of the entire architecture and the working of the individual modules is presented in the further sections.

II. Related Work

In recent years a lot of research is being done towards developing natural language processing techniques for the purpose of question answer generation. In [6], the authors present a review of the various developments in the field of question answer generation. The techniques used, the datasets used and the evaluation methods that are used are discussed in general. This work provides a broad perspective of the research work that is taking place in this field.

In [2], the authors discuss automatic question answer pairs generation along with a question similarity metric. The SQuAD[5] dataset is utilized for building the machine learning models. Spans are extracted as noun and verb phrases prior to generating questions and answers. Finally the machine generated question answer pairs for different passages are presented.

In [3], the authors deal with question generation for scenarios where the answer is not known. Span prediction is discussed and a pipeline to first predict span and then generate questions is presented. In [4], the authors use additional information like clues and style for the purpose of question answer generation. Good insights are presented as to how to extract and use additional information from a given input passage for the purpose of generating question answer pairs.

In [1], the authors discuss the effectiveness of a transformer based architecture and how it outperforms the LSTM [7] based architecture. The self-attention based mechanism and its effectiveness in tasks related to natural language generation are discussed.

III. Methodology

We present an end to end architecture to automatically generate ranked question answer pairs given an input passage. To overcome the limitation of current question answer generation pipelines, we utilize a coherence metric between the individual sentences to improve the process of span generation. Furthermore, we make use of a knowledge graph to obtain domain knowledge and produce additional question answer pairs related to certain important phrases in the given input passage. The models that we utilize are primarily trained on the SQuAD[5] dataset. Details of each of the modules are presented in the following sections.

A. *Span Extraction Module*

Span Extraction plays a critical part in the entire span extraction pipeline. Here we define a “span” as a group of words in a sentence that holds information that is worth being generated into the form of a question and answer. To extract spans using a deep learning model, we are using a pre-trained T5 model[8]. The flow of events was slightly modified to make it suitable primarily for span extraction. Additionally, we are extracting noun phrases as well from the individual sentences which enhances the information gained during this phase as certain groups of text may be missed out by the deep learning model.

Before proceeding towards extracting spans using the above two mentioned methods, we perform a critical operation of coherence measurement between sentences. To measure the coherence between 2 sentences, we first encode the sentences using the Universal Sentence Encoder[9] and then measure the similarity between the two given vectors which represent the two different sentences in a passage. The similarity measure utilized here is the cosine similarity measure. Given 2 encoded sentences S_a and S_b, the similarity measure between 2 vectors is,

$$\frac{\left(S_a, S_b\right)}{\left|S_a\right|\left|S_b\right|}$$

where $\left(S_a, S_b\right)$ denotes the inner dot product of S_a and S_b. The process of calculating the similarity between the encoding of 2 sentences is repeated for every pair of sentence vectors which are extracted for each sentence in the passage. These similarity scores are filled into a matrix of size $N \times N$ where N represents the number of individual sentences in a given passage of text.

Having generated the coherence matrix, the next step to be performed is to pick the appropriate sentences to be grouped. For each sentence in a given passage, except the first and the last sentences, a sentence before and a sentence after are found with the highest coherence from the coherence matrix. Having found the highly coherent sentences that are before and after a given sentence, the sentences are grouped together. To make the groups concise without losing any critical information, the groups have to be succinctly summarized. A pre-trained T5 model [8] was used to perform abstractive summarization on the groups of sentences. From these summarized sentences, information spans are extracted.

Spans are extracted from both the original sentences and from the sentences that were formed after the coherence calculations. To efficiently organize the data, a list of dictionaries containing key value pairs are created. At each stage of the pipeline, information is added to the dictionaries. Each dictionary holds the sentence from which the spans were extracted, the spans that are extracted from the sentence and in the future steps, questions and actual answers will also be added. This aids in the efficient interaction of modules and eases the process of creating an API using the entire module. The entire algorithm for the span extraction module is shown in Figure 1.

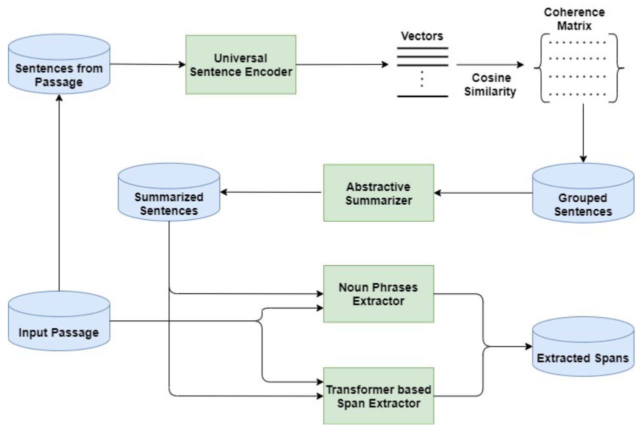

Fig. 1. Algorithmic Flow For Span Extraction

B. *Question Generation Module*

The generated spans are fed to the Question generating module along with the context. We use a Text to Text Transfer Transformer model to generate questions. This model is trained on the SQuAD Dataset. SQuAD dataset is preprocessed to contain only the context, golden answer and the question as the columns. So the T5 model is trained to generate a question given a context and span.

Two kinds of questions are generated in our model. First kind are those whose answer is present in the module. Second kind questions don't contain their answer in the context. So the first variety helps to understand the context better and the second variety helps to understand the domain knowledge of the passage. First kind of Questions are generated with the help of the trained T5 model. The generated spans along with the appropriate context are fed to the Model to generate questions. These kind of questions can overshoot the desired number, so these can be bound by a limit parameter.

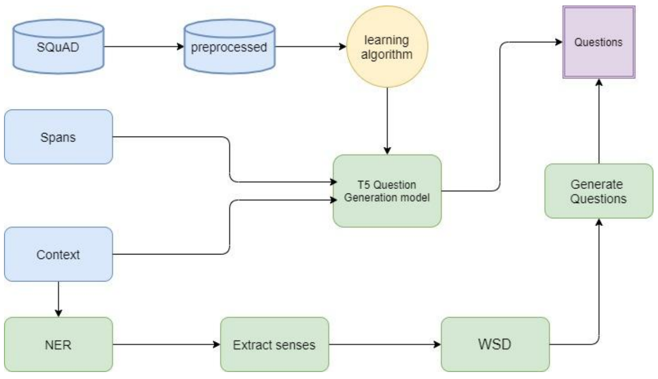

Fig. 2. Algorithmic Flow For Question Generation

Second kind of questions are generated with the help of WordNet or Knowledge graph with the help of techniques like Named entity Recognition(NER) and Word Sense Disambiguation(WSD). First we tag the words in the context as one of PER(abbreviation for 'Person'), ORG(abbreviation for 'Organization'), GPE(abbreviation for 'Geographical') and so on. These tagged words are passed to the WordNet / Knowledge graph to get their meaning/sense. However a word might have several senses(for example Bank). So Word Sense Disambiguation (WSD) is used to make proper sense of the word given the context it's used in. Now we generate simple 'What', 'Who'. and 'Where' questions with the proper sense as the answer.

C. Answer Generation Module

The Answer Generation module is the third stage of the pipeline and it identifies the span of text containing the answer to the generated question. It uses the SpanBERT[10] model fine tuned on SQuAD[5] v1.1 dataset for predicting answer spans from the given context. Google's Knowledge Graph[13] is used to fetch additional information about the extracted answer which is not present in the text passage.

SpanBERT is a masked language model used to encode spans of text (tokens) using a transformer encoder. It consists of a Span Boundary Objective function which is trained to predict the start and end of the answer span for the given question. Based on the predicted span, it generates a score between 0 and 1 which represents the probability that the span identified is the appropriate answer for the given question. The SpanBERT model was fine-tuned on the SQuAD 1.1 dataset and it was able to achieve an F1 score of 91.98% on the answer prediction task.

After the questions are generated from the Question Generator, they are passed to a tokenizer along with the source sentence to generate question and context tokens. The SpanBERT model uses these tokens to identify the relevant context tokens to generate the answer and calculate a probability score. The answer is then searched as a keyword in the Knowledge Graph to retrieve additional domain knowledge required for the FAQ. Finally, the answer along with its probability score and extra information is compiled into a final response and forwarded to the QA Ranker module.

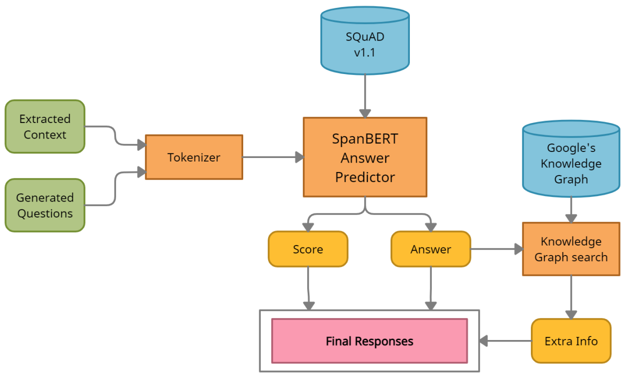

Fig. 3. Algorithmic Flow For Answer Generation

D. Question-Answer Pairs Ranking

This is the final module in the entire pipeline. This module takes as input the list of dictionaries containing the span, question and answer pairs along with the probabilistic score generated by SpanBERT[10].

This module initially segregates and restructures the list of dictionaries such that in the new list of dictionaries, each dictionary contains the question answer pair and the score only. This unsorted list of dictionaries is then sorted in the descending order of their scores into a sorted list.

Based on the user input and the number of good quality question-answer pairs generated (above a certain threshold), the pairs are returned to the user.

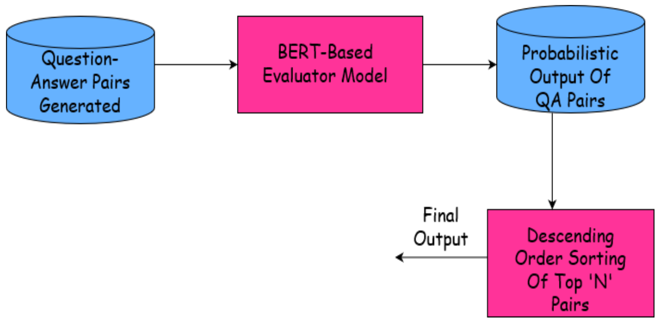
Question-Answer Pairs Generated
BERT-Based Evaluator Model
Probabilistic Output Of QA Pairs
Final Output
Descending Order Sorting Of Top 'N' Pairs

Fig.4. Question Answer Ranker Deployment Diagram

IV. Results And Discussion

A complete pipeline was created by integrating the Span Extraction, Question Generation, Answer Generation and the Question-Answer Pairs Ranking modules. To support efficient integration, a list of dictionaries was created. Additionally, an API interface is provided to interact with the developed software. In order to test the effectiveness of our architecture, question answer pairs were generated for over hundred paragraphs from the SQuAD[5] dataset. The generated question answer pairs were compared to ground truth question answer pairs present in the dataset. Some of the sample question answer pairs generated using the developed pipeline are shown in Table 1.

TABLE I. Sample Question Answer Pairs

Many faults are able to produce a magnitude 6.7+ earthquake, such as the San Andreas Fault, which can produce a magnitude 8.0 event. Other faults include the San Jacinto Fault, the Puente Hills Fault, and the Elsinore Fault Zone. The USGS has released a California Earthquake forecast which models Earthquake occurrence in California.	
Q	Where is the Earthquake forecast from?
A	California
Q	What type of occurrence does the USGS model?
A	Earthquake
Q	What is the name of the Hills Fault?
A	Puente Hills Fault
Q	What fault can produce a magnitude 8.0 earthquake?
A	San Andreas Fault
Q	What type of earthquake can many faults produce?
A	magnitude 6.7+
Q	What is the name of the fault zone?
A	Elsinore Fault Zone

As a first step of the testing process, the outputs generated by the Span Extraction and Answer Generation modules are tested using metrics such as Bilingual Evaluation Understudy Score(BLEU) and Recall-Oriented Understudy for Gisting Evaluation(ROUGE). To test the span extraction module, information spans for every paragraph in the SQuAD[5] test set are generated. For every paragraph, the extracted spans are compared to the actual spans and the average is taken. The BLEU [11] scores are calculated considering 1-grams, 2-grams and cumulative 2-grams. The ROUGE [12] scores are calculated by means of presenting F1, precision and recall for rouge-1, rouge-2 and rouge-l. A similar approach is followed for testing the Answer Generation module independently. Answers are generated for each and every paragraph in the SQuAD[5] test set by using the question and context from the SQuAD test set itself. BLEU [11] and ROUGE [12] scores are calculated and presented in a manner identical to the way the scores are presented for the Span Extraction Module. The BLEU [11] scores and ROUGE [12] scores for the Span Extraction module are shown in Table II and Table III respectively. The BLEU [11] scores and ROUGE [12] scores for the Answer Generation module are shown in Table IV and Table V respectively.

Besides independent testing of the previously mentioned modules, the end to end pipeline was also tested for its effectiveness in the generation of question answer pairs. To ensure the testing results are covered for a wide variety of inputs, question answer pairs were generated for hundred separate paragraphs that are part of the SQuAD[5] test set. BLEU [11] scores were used to measure the quality of answers and questions generated. For the answers BLEU [11] scores were found for individual 1-gram, 2-gram and 3-gram and cumulative 2-gram and 3-gram. Since the word count of individual questions is considerably longer than word count for individual answers, individual 4-gram and 5-gram and cumulative 4-gram and 5-gram scores were also calculated. The measured BLEU [11] scores for the answers and questions generated by the end to end pipeline are presented in table VI and table VII respectively. The standard interpretation of BLEU [11] score ranges is shown in table VIII. During the testing phase we also found that the average number of question-answer pairs generated by our pipeline for a given paragraph of text is 24.58 while the same for the ground-truth question answers pairs present in the SQuAD dataset is 10.28. Furthermore we found that for each individual sentence, our pipeline on an average generated 2.092 question answer pairs while the ground truth on average contained 0.875 question answer pairs for each sentence. This shows that our approach that utilizes coherence metrics is able to correlate the information present in the sentences better which results in higher coverage.

TABLE II. BLEU Scores For Span Extraction

Span Extraction			
BLEU Score	**Individual 1 gram**	**Individual 2 gram**	**Cumulative 2 gram**
	0.3480729	0.1817396	0.1817396

TABLE III. ROUGE Scores For Span Extraction

Span Extraction Rouge Scores			
	F1	**Precision**	**Recall**
rouge-1	0.37112	0.34277	0.54262
rouge-2	0.19887	0.18778	0.29122
rouge-l	0.34039	0.32027	0.45804

TABLE IV. BLEU Scores For Answer Generation

Answer Generation - SpanBERT			
BLEU Score	Individual 1 gram	Individual 2 gram	Cumulative 2 gram
	0.7774395	0.4768997	0.48083012

TABLE V. ROUGE Scores For Answer Generation

Span Extraction Rouge Scores			
	F1	Precision	Recall
rouge-1	0.83700	0.87802	0.84900
rouge-2	0.50882	0.53395	0.51763
rouge-l	0.83842	0.87857	0.84999

TABLE VI. BLEU Scores For Answers Generated By Pipeline

BLEU Score Statistics for Answers generated by the end to end pipeline				
Ind. 1 gram	Ind. 2 gram	Ind. 3 gram	Cum. 2 gram	Cum. 3 gram
0.3562	0.4322	0.4479	0.3810	0.3976

V. Conclusion and Future Prospects

The architecture we have developed has shown very promising results. We have taken into account the factor of coherence between sentences and also the fact that additional domain knowledge can be used to generate better results. So we have used the concepts of coherence measurement and using a knowledge graph to gain additional domain knowledge. These have resulted in more meaningful results as demonstrated through the very promising metrics we have obtained during the testing phase. In comparison to the currently existing question answer generation pipelines, we have gone a step forward and have added our own innovations to the various steps in the question answer generation pipeline. That said, there is scope for further improvement to make the solution more robust and open it to more applications in different avenues.

Domain knowledge can be more tightly knit with the entire architecture and more use of this domain knowledge can be made than what we are doing currently. Different representations can be tried to suitably represent additional information rather than just through a standard knowledge graph. Furthermore, one can experiment with fine tuning the entire pipeline on a particular domain so that the question answer pairs generated related to that domain have a great quality and shown immense robustness.

TABLE VII. BLEU Scores For Questions Generated By Pipeline

BLEU Score Statistics for Questions generated by the end to end pipeline				
Ind. 1 gram	Ind. 2 gram	Ind. 3 gram	Ind. 4 gram	Ind. 5 gram
0.6232	0.5187	0.6496	0.7762	0.8643
	Cum. 2 gram	Cum. 3 gram	Cum. 4 gram	Cum. 5 gram
	0.5307	0.5316	0.5643	0.6046

TABLE VIII. BLEU Scores Interpretation

BLEU Score	Interpretation
< 0.1	Almost Useless
0.1 - 0.19	Hard to get the gist
0.2 - 0.29	Clear but significant grammatical errors
0.3 - 0.4	Understandable to good translations
0.4 - 0.5	High Quality
0.5 - 0.6	Very High Quality, Adequate and Fluent
> 0.6	Quality often better than human

Domain knowledge can be more tightly knit with the entire architecture and more use of this domain knowledge can be made than what we are doing currently. Different representations can be tried to suitably represent additional information rather than just through a standard knowledge graph. Furthermore, one can experiment with fine tuning the entire pipeline on a particular domain so that the question answer pairs generated related to that domain have a great quality and shown immense robustness.

To open up the question answer generation pipeline to more avenues, input formats like videos and audios that contain some speech can be experimented with. For example, given a video as input, the information in the video can be suitably extracted and question answer pairs can be generated for the the content present in the video. The same is applicable for when audio is passed as input as well. Such kinds of different input formats can be trialled and developed further to increase the applications of the automatic question answer generation pipeline. Another aspect that can be looked into, is expanding our pipeline to generate multiple choice questions automatically. This would be really useful in academia and will help in designing various assignments in educational institutes.

Acknowledgment

We would like to thank PES University for facilitating the conduction of this research and in turn aiding in the development of the software. We would like to specifically thank our professor, Dr. Ramamoorthy Srinath for his constant guidance and valuable inputs which have greatly helped in improving the quality of the work while also leading to new developments in our approach.

References

[1] *Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N. and Kaiser, Łukasz and Polosukhin, Illia Attention Is All You Need*

[2] *Aithal, Shivani & Rao, Abishek & Singh, Sanjay. (2021). Automatic question-answer pairs generation and question similarity mechanism in question answering system. Applied Intelligence. 10.1007/s10489-021-02348-9.*

[3] *Ko, Wei-Jen & Chen, Te-Yuan & Huang, Yiyan & Durrett, Greg & Li, Junyi. (2020). Inquisitive Question Generation for High Level Text*

[4] *Bang Liu, Haojie Wei, Di Niu, Haolan Chen, and Yancheng He. (2020). Asking Questions the Human Way: Scalable Question-Answer Generation from Text Corpus. In Proceedings of The Web Conference 2020 (WWW '20). Association for Computing Machinery, New York, NY, USA, 2032–2043. DOI*

[5] *Rajpurkar, Pranav & Zhang, Jian & Lopyrev, Konstantin & Liang, Percy. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. 2383-2392. 10.18653/v1/D16-1264*

[6] *Kurdi, Ghader & Leo, Jared & Parsia, Bijan & Sattler, Uli & Al-Emari, Salam. (2019). A Systematic Review of Automatic Question Generation for Educational Purposes. International Journal of Artificial Intelligence in Education.*

[7] *Hochreiter, Sepp and Schmidhuber, Jurgen (1997). Long Short-Term Memory*

[8] *Raffel, Colin & Shazeer, Noam & Roberts, Adam & Lee, Katherine & Narang, Sharan & Matena, Michael & Zhou, Yanqi & Li, Wei & Liu, Peter. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.*

[9] *Cer, Daniel & Yang, Yinfei & Kong, Sheng-yi & Hua, Nan & Limtiaco, Nicole & John, Rhomni & Constant, Noah & Guajardo-Cespedes, Mario & Yuan, Steve & Tar, Chris & Sung, Yun-Hsuan & Strope, Brian & Kurzweil, Ray. (2018). Universal Sentence Encoder.*

[10] *Joshi, Mandar & Chen, Danqi & Liu, Yinhan & Weld, Daniel & Zettlemoyer, Luke & Levy, Omer. (2019). SpanBERT: Improving Pre-training by Representing and Predicting Spans.*

[11] *Papineni, Kishore & Roukos, Salim & Ward, Todd & Zhu, Wei Jing. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. 10.3115/1073083.1073135.*

[12] *Lin, Chin-Yew. (2004). ROUGE: A Package for Automatic Evaluation of summaries. Proceedings of the ACL Workshop: Text Summarization Branches Out 2004. 10.*

[13] Fensel D. et al. (2020) Introduction: What Is a Knowledge Graph?. In: Knowledge Graphs. Springer, Cham.