Factoid Question Answering System

Evgeniia Shcherbinina Hasnah Said Jorie Fernandez

---

1. Abstract

The paper is focused on a factoid question answering system that uses Wikipedia as the knowledge source. The project is implemented using a Document Reader and Retriever approach with the aid of natural language processing techniques and Bi-directional Attention Flow network. The system produced satisfactory answer but it will still require further improvements to get a more accurate and precise result.

2. Introduction

Question Answering (QA) System is a fundamental application in Natural Language Processing. The QA system must have the ability to read questions and answer them in human-like language. This is a challenging task for computers because it requires them "to understand" natural language, find the answer by searching in a large amount of information and then present the result in a human-like language.

The system that this paper is focused on answers open-domain factoid questions, such as *"What is the biggest ocean in the world?"* or *"What is the capital city of Washington state?"*. Factoid questions are questions that have exactly one correct answer which is extracted from text segments [3]. Our system is going to use Wikipedia articles as its primary knowledge source.Wikipedia contains detailed English articles about more than five million topics. It is constantly being updated with new knowledge, and new articles are being added. To answer questions using Wikipedia, our system will first retrieve a relevant article from Wikipedia and then find the answer based on that article [1]. QA systems have many interesting and important applications. They can be used in the medical field by providing answers to doctors and patients. They can also be used in schools to help students to get short accurate answers to their questions quickly.

There have been previous researches that used Wikipedia as a knowledge base in QA systems. For example, Chen and others utilized "a search component based on bigram hashing and TF-IDF matching with a multi-layer recurrent neural network model, which are trained to detect answers in Wikipedia paragraphs." They have also mentioned

other related works on Wikipedia-based question and answer systems, where other authors used a combination of "article contents with multiple other answer matching modules based on different types of semi-structured knowledge such as infoboxes, article structure, category structure, and definitions" [1].

One challenge in the implementation of the system is the sufficiency of the amount of information that is available in the Wikipedia to answer the open domain questions. There are also credibility issues on Wikipedia as a source. Recently, a group of Microsoft researchers led by Yang have introduced WikiQA dataset, which consisted of sentence pairs and answers. However, he stated that "it is still inferior to the standard QASent dataset. " According to Yang and his team, the questions with incorrect answers were not filtered and still considered in the dataset and in modeling [10].

Hence, a thorough analysis and model improvement should be considered to identify the answers. The improved model will focus on the factual and correct information to maintain its reliability by using Bi-Directional Attention Flow. This is based on the research by Seo and others using the neural networks [8]. The Stanford Question Answering Dataset (SQuAD) set will also be considered to train the Document Reader. SQuAD is "a reading comprehension dataset consisting of over 100, 000 question-answer pairs on 500 articles". The articles in this dataset comes from a variety of sources like Wikipedia, and the answers are directly based on the contents of the passage and can be directly picked up from a portion of the passage [6].

3. Background

Question Answering System is a computer system that has the ability to read natural language questions and answer them in a human-like language quickly and accurately. Question Answering has become an important application of natural language processing because it can provide users with answers they are looking for, usually in a word or a sentence, without having them manually extract information from a large amount of text. The answers that the users get are backed with articles that validates the answer. Current search engines can return a ranked list of documents that are related to the question the user is asking, but they do not deliver an answer to the question.

QA systems are not a new area of research in Natural Language Processing. Two of the earliest questions answering systems are BASEBALL [4], which was developed in the

1960s to answer questions about dates, location and results of baseball matches. And LUNAR [9], which was developed in the 1970s to answer questions about lunar geology, it was able to answer 90% of the questions in its domain asked by people who were not trained on the system. The main limitation of these of systems is that they were only capable of answering questions within a specific domain. Closed-domain QA systems, such as BASEBALL, use structured databases that store the knowledge required to answer the questions.

Newer QA systems are mostly open-domain, they are capable of answering questions in any domain using a large collection of unstructured data (e.g. the web). One of the first open-domain QA systems is START [7]. START was developed in 1997 and it uses the web as its knowledge source. The system applies its own heuristics to store the information from web documents into a local knowledge database that is later accessed when a question is asked, it then uses NLP techniques to generate the answers to any question [2]. Another famous open-domain QA system is IBM's Watson, it won the *Jeopardy!* in 2011 Man vs. Machine Challenge by defeating two former grand *Jeopardy!* champions. It was answer complex natural language question in an extremely broad domain of knowledge [5].

4. Problem Statement

Watson, chatbots, information retrieval, these are just some of the question and answering systems, which is one of the applications in Natural Language Processing. Furthermore, question and answering is part of the oldest research in the field that is focused on answering textual question.

As information becomes more available via web, sources for question and answering system also expands. Wikipedia, for instance, is a popular source online since it allows information exchange from different communities. Thus, more information is gathered, and the knowledge repository becomes accessible to anyone.

These impacts on the information retrieval motivated the group to investigate and implement a Wikipedia-based question and answering system. The project is concerned with open domain factoid question answering system, where it uses Wikipedia as the source of data with an aid from the SQuAD dataset. The model will follow the Bi-Directional Attention Flow network, which was proposed by Seo and others [8]. Our group aims to

learn and apply natural language processing techniques and deep learning concepts, focused on neural networks, to implement the system. Moreover, the project also helps explore the effectiveness of the combined Wikipedia and SQuAD dataset in producing automated answers to the provided questions.

5. Technical Details

5.1 Dataset

Our work relies on three types of data:

1. Wikipedia that serves as our knowledge source for finding answers. We use a Python package wikipedia 1.4.0 that makes it easy to access and parse data from Wikipedia.

2. The Stanford Question Answering Dataset (SQuAD) which is used to train Document Reader. SQuAD is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. The dataset contains 87k examples for training and 10k for development.

Source: https://rajpurkar.github.io/SQuAD-explorer/.

Format: JSON files.

3. Pre-trained Global Vectors for Word Representation (GloVe) which are trained on the non-zero entries of a global word-word co-occurrence matrix and tabulate how frequently words co-occur with one another in a given corpus (Wikipedia 2014 + Gigaword 5).

Source: https://nlp.stanford.edu/projects/glove/.

Format: text files.

5.2 Model / Implementation Details

Our QA system consists of two components: Document Retriever module for finding a relevant article and a machine comprehension model, Document Reader, for extracting an answer from a single document (Figure 1).

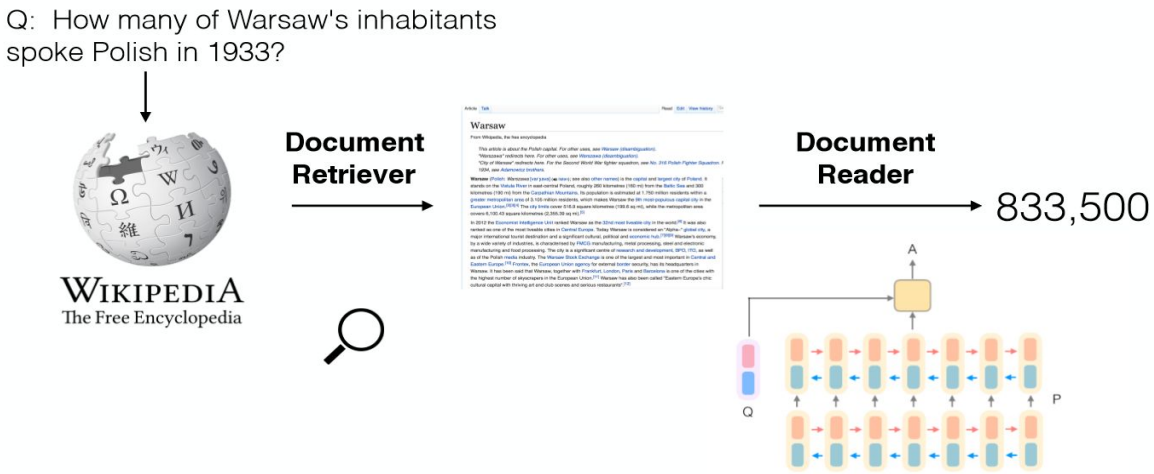

Figure 1. Question Answering System [1]

First, the QA system asks a user to enter a question, checks the input for spelling errors, and if this is the case, suggests spelling corrections (Fig. 2).

```
Enter your question below
>> What do robots that resemble humans attempt to do?
Would you like to replace robots with roots? (Y/N)
>> n
```

Figure 2. Sample output of the Document Retriever step 1: Asking a question and spelling suggestion.

Then, using Rapid Automatic Keyword Extraction algorithm (RAKE) the system tries to determine key phrases in a body of text by analyzing the frequency of word appearance and its co-occurrence with other words in the text. Extracted words are used to find the five relevant documents in Wikipedia by utilizing wikipedia 1.4.0 - a Python library that makes it easy to access and parse data from Wikipedia (Fig. 3). We limit each article to top 10 sentences, where the answer is likely to be.

```
 resemble humans attempt robots
['Robotics', 'Humanoid robot', 'Robot', 'Outline of robotics', 'History of robots']
```

Figure 3. Sample output of the Document Retriever step 2: Extracting keywords and retrieving relevant articles from Wikipedia.

Finally, all extracted summaries of the most relevant documents are converted to a matrix of TF-IDF features (Fig. 4). We use TfidfVectorizer from scikit-learn Python library to represent the question and summaries in high-dimensional vector space and compute the similarity score between each pair of vectors. The document with the highest score is selected as the most relevant.

```
[[1.         0.09180945 0.07036179 0.0847684  0.0337378  0.0265198 ]
 [0.09180945 1.         0.10164115 0.19477773 0.24279278 0.07372873]
 [0.07036179 0.10164115 1.         0.06498419 0.04552158 0.05732289]
 [0.0847684  0.19477773 0.06498419 1.         0.20494998 0.06437706]
 [0.0337378  0.24279278 0.04552158 0.20494998 1.         0.02578029]
 [0.0265198  0.07372873 0.05732289 0.06437706 0.02578029 1.        ]]
[0.09180945 0.07036179 0.0847684  0.0337378  0.0265198 ]
```

Figure 4. Sample output of the Document Retriever step 3: Converting a collection of retrieved documents to a matrix of TF-IDF features.

Along with the question, the summary of the most relevant document becomes the input of the Bi-Directional Attention Flow (BiDAF) network described by Seo et al. [8]. The network consists of six layers (Fig. 5):

1. Character Embedding Layer maps each word to a vector space using character-level CNNs.

2. Word Embedding Layer maps each word to a vector space using a pre-trained word embedding model. We used pre-trained GloVe vectors to obtain the 100-dimensional fixed word embedding of each word.

3. Contextual Embedding Layer utilizes contextual cues from surrounding words to refine the embedding of the words. These first three layers are applied to both the query and context.

4. Attention Flow Layer couples the query and context vectors and produces a set of query aware feature vectors for each word in the context.

5. Modeling Layer employs a Recurrent Neural Network to scan the context.

6. Output Layer provides an answer to the query.

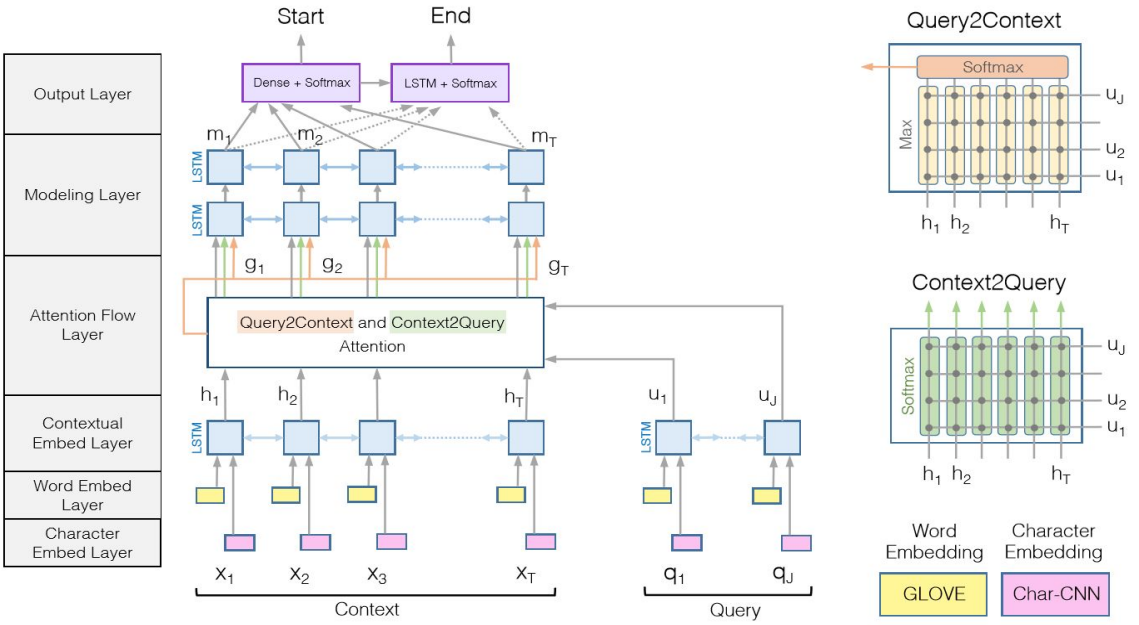

Figure 5. BiDirectional Attention Flow Model [8]

A single model and the ensemble model allowed authors to achieve 77.3% and 81.1% F1 score on SQuAD respectively [8]. For our project, we used the AllenNLP Machine Comprehension model, which is a reimplementation of BiDAF (Seo et al. [8]). Our model achieved 77.9% F1 score on SQuAD development set (Fig. 6).

```
2018-03-13 18:36:27,284 - INFO - allennlp.commands.evaluate - Metrics:
2018-03-13 18:36:27,284 - INFO - allennlp.commands.evaluate - start_acc: 0.64210
02838221381
2018-03-13 18:36:27,284 - INFO - allennlp.commands.evaluate - end_acc: 0.6710501
41911069
2018-03-13 18:36:27,284 - INFO - allennlp.commands.evaluate - span_acc: 0.552696
3103122043
2018-03-13 18:36:27,284 - INFO - allennlp.commands.evaluate - em: 0.683727530747
3983
2018-03-13 18:36:27,284 - INFO - allennlp.commands.evaluate - f1: 0.778573652867
3448
```

Figure 6. Evaluation Metrics of the Document Reader

Our Document Reader finds a sub-phrase of the summary to answer the question. The phrase is derived by predicting the start and the end indices of the phrase in the summary (Fig. 7).

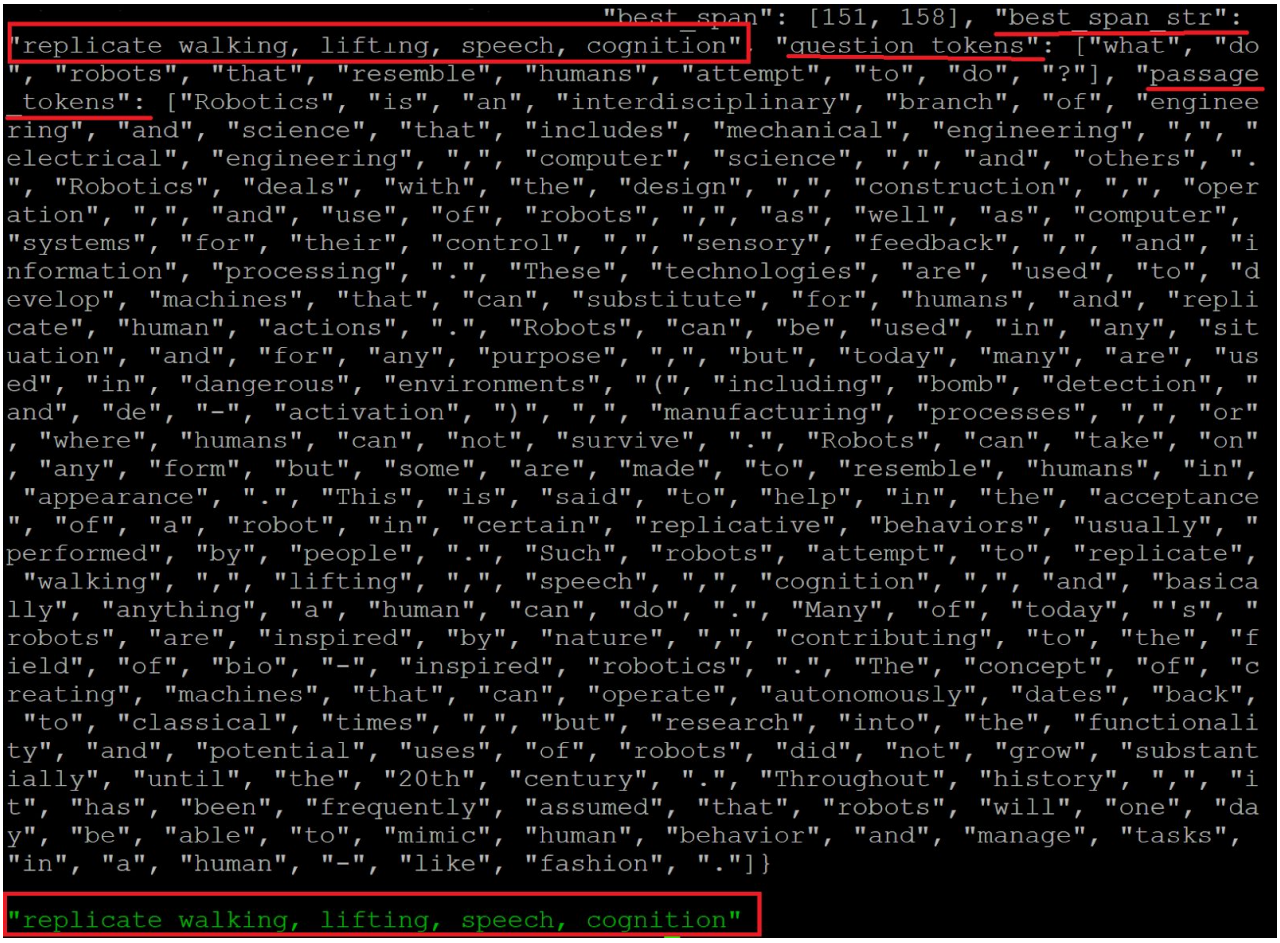

Figure 7. Sample Output of the Document Reader

6. Future Step and Conclusion

In conclusion, we have described the approach we took to build a factoid question answering system that used Wikipedia as its main knowledge base. Our system is split into two parts: Document Retrieval and Document Reader. In the first part we used RAKE (Rapid Automatic Keyword Extraction) to extract the keyword from the question asked after processing it. It then retrieved the relevant documents from Wikipedia using their Python library. The documents are then ranked using TF-IDF (Term Frequency-Inverse Document Frequency), the document with the highest score is put in a JSON file together with the question and it's passed to our Document Reader. The document reader will use Bi-Directional Flow Attention to read the document and extract the answer to the question.

One important step we could take to improve our QA system is to change the way we find the documents by better understanding the question. The system in this paper uses Rapid Automatic Keyword Extraction (RAKE) algorithm to extract keywords from questions. Many English words have several meanings and pinpointing the exact meaning of words is challenging task for the computer. The step we to improve our system is to use question classification. The main point of question classifications is to categorize questions into different semantic classes based on the possible semantic types of the answer [8]. By classifying the questions, we could narrow down the number of documents we are searching which will improve the speed and accuracy of the answers. Furthermore, Wikipedia is considered an insufficient source of knowledge. Thus, it is recommended to explore other corpus or data sources to answer factoid questions.

7. References

[1] Chen, Danqi, et al. “Reading Wikipedia to Answer Open-Domain Questions.” 2017.

[2] Dwivedi, Sanjay K., and Vaishali Singh. "Research and Reviews in Question Answering System." *Procedia Technology*10 (2013): 417-424.

[3] Er, Nagehan Pala, and Ilyas Cicekli. "A factoid question answering system using answer pattern matching." *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. 2013.

[4] Green Jr, Bert F., et al. "Baseball: an Automatic Question-answerer." *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*. ACM, 1961.

[5] High, Rob. "The Era of Cognitive Systems: An Inside Look at IBM Watson and How It Works." *IBM Corporation, Redbooks* (2012).

[6] Jain, Atishay, and Faraz Wasim. "Answering SQuAD."

[7] Katz, Boris. "Annotating the World Wide Web Using Natural Language." *Computer-Assisted Information Searching on Internet*. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 1997.

[8] Li, Xin, and Dan Roth. "Learning question classifiers: the role of semantic information." *Natural Language Engineering* 12.3 (2006): 229-249.

[9] Seo, Minjoon, et al. “Bi-Directional Attention Flow for Machine Comprehension.” arXiv: 1611.01603[cs.CL] (2017). Web. 3 Feb. 2018.

[10] Woods, William A. "Progress in Natural Language Understanding: an Application to Lunar Geology." *Proceedings of the June 4-8, 1973, national computer conference and exposition*. ACM, 1973.

[11] Yang, Yi, et al. “WIKIQA: A Challenge Dataset for Open-Domain Question Answering.” https://aclweb.org/anthology/D15-1237.