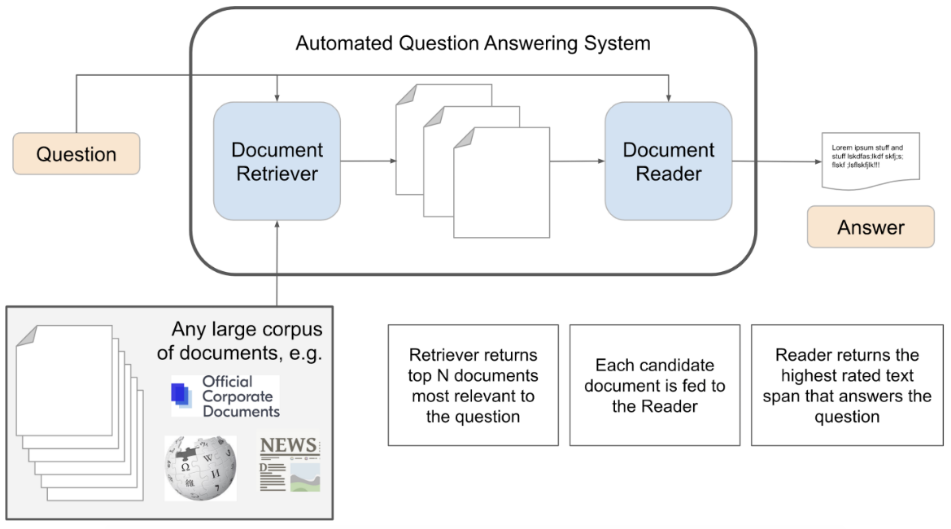
Introduction

Question Answering models are able to retrieve the answer to a question from a given text. This is useful for searching for an answer in a document. Depending on the model used, the answer can be directly extracted from text or generated from scratch.

In this approach, we have experimented with **Document Retrieval** - so given a question, the document retriever will return the **k** most likely documents that contain the answer to the question.



Automated Question Answering System
Question
Document Retriever
Document Reader
Lorem ipsum stuff and stuff lskdfas;lkdf skfj;s; flskf ;lsflskfjk!!!
Answer
Any large corpus of documents, e.g.
Official Corporate Documents
NEWS
Retriever returns top N documents most relevant to the question
Each candidate document is fed to the Reader
Reader returns the highest rated text span that answers the question

Generic IR QA system

Fig. from https://qa.fastforwardlabs.com/methods/background/2020/04/28/Intro-to-QA.html

Overall Procedure for TF-IDF based Document Retrieval

- The data is converted into an appropriate format (Pandas dataframe) from JSON.
- Documents are encoded and questions are transformed into vectors using TF-IDF.
- TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus and is often used as a weighting factor in searches of information retrieval, text mining, and user modeling.
- Search for a question is performed by comparing with the document vectors.
- The **k (**here, k = 3**)** most similar documents (NearestNeighbors) are returned.
- The actual **context_id** is searched for amongst the k returned contexts - if it's present, then the process is marked as a success, otherwise a failure.

Going by the above process, the accuracies in the following components of SQuAD (Stanford Question And Answering Dataset) are as follows :-

Partition of Dataset	**Observed Accuracy**
Train	0.7182
Dev	0.8488
Train + Dev	0.7126

This is a difficult problem, because we have multiple documents and the answer can be in one or more documents. Thus, the retriever usually returns k documents, since its not complete / fair to return only one document.

TF-IDF has some problems though :-
(1) Its only able to compute similarity between questions and documents that present the same words, so it can not capture synonyms.
(2) It cannot understand the question context or the meaning of the words.

<u>Overall Procedure for Word2Vec</u>

The overall process stays the same, but just that, instead of TF-IDF, Word2Vec is used to encode the documents and to transform the questions into vectors, WikiPedia says this about Word2Vec -> *Word2vec is a technique for natural language processing published in 2013. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence.*

The entire corpus is scanned, and the vector creation process is performed by determining which words the target word occurs with more often. In this way, the semantic closeness of the words to each other is also revealed.
For example, let each letter in the sequences **..x y A z w..** , **.. x y B z k..** and **..x l C d m..** represent a word. In this case, word_A will be closer to word_B than word_C. When this situation is taken into account in vector formation, the semantic closeness between words is expressed mathematically.

But the accuracy achieved **()** through using Word2Vec embeddings doesn’t turn out to be that much with using a vector_size of 25,000 (the vocabulary may be so much more, in which case, taking a vector_size of 25k isn’t going to be that useful, in contrast, the vector_size used in case of TF-IDF was 50,000 - thus it becomes quite clear that there isn’t enough context for Word2Vec for proper training) - but using a greater vector_size, we run into memory considerations, as this experiment was performed in Google Colab. Also, the time taken for computing the embeddings of all the words is quite a lot.

Other Approaches to be Explored

- Use of some Pre-Trained models such as ***word2vec-google-news-300***.
- Use of other methods of embeddings for document retrieval such as **Doc2Vec**, **FastText** from the **gensim** library.
- Use of other robert embedding techniques such as transformer models (BERT).
- Testing the performance of all the above models on other relevant QnA datasets.