**Q & A system**

**Executive Summary**

There is a massive amount of information stored in text files that is not readily available to people without manual parsing for answers. To best take advantage of this information while minimizing human input, I have developed a question and answer system that can identify key components of a question, find sources of potential answers within the text documents, and then elect the best answer from these sources. This system saves a large amount of human input time to answer specific questions that are covered in the text corpus. Right now, the system works to answer a few basic types of questions, but it is built on an infrastructure that could scale up quite well beyond the current corpus and question types. The text corpus is created using articles from BusinessInsider from 2013-2014 and the accepted questions are the following:

-Which companies went bankrupt in month X of year Y?

-Who is the CEO of company X?

-What affects GDP? What percentage of drop or increase is associated with this property?

**Problem Statement**

There is a large amount of information held in text documents, but retrieving the desired information would take a large amount of manual parsing time. This question and answer system seeks to reduce the necessary human input time to extract information from the text corpus

**Assumptions**

* The asked questions are answered/addressed in the corpus
* The information regarding the question in the text corpus is correct
* The answer to the question is constant over the period of time in which the corpus covers

**Methodology**

The first step to searching the text corpus was creating an index of the documents and keywords. This can be done using a tool such as solr. After indexing the documents, we can move onto handling of questions. I created a function that would be able to determine question type for each of the questions. The inputs for the function were the question and the keywords used in answering the last question (so that sequentially asked questions can retain information from the last query).

The first step is to identify the question type. This is very important for identifying what type of answer to search for in the documents. Since we are dealing with a limited number of question types in this trial system, I developed a rules based approach that could identify the desired output of the question by the first (and sometimes second) word of the question. The program would then extract important words from the question using pos tags to determine what role each word played in the question. It is important to recognize that the question may not be asked with wording that is reflected in the corpus, so I used wordnet to extract synonyms for many of these words as well. I then used the keywords and their synonyms to create a proper query in the solr database. This is done by joining the synonyms together, so that they each have equal weight and are interchangeable. The top results are then selected using the internal solr scoring mechanism. The documentation for the scoring mechanism shows that it is a more elaborate/extensive version of the tf-idf score, but also includes important information such as proximity of search terms.

After selecting the top candidate documents, I then extract potential answers from the documents. Since I already identified question/answer type, I know what kind of words/answers to be selected for within the documents. I extract all phrases of this type, identifying them using ne\_chunk in nltk. After extracting the results, I clean them to make sure there are no repetitions or extracted terms that violate certain rules. I then run the terms through a function to rate their relevancy to the question using the corpus.

Instead of creating a custom scoring function for the answers, I wanted to take advantage of the highly efficient and extensive mechanisms within solr. I did this by incorporating the selected answers into the extracted search terms one at a time and then re-querying the corpus. The answer that results in the highest score when combined with the original question is likely to be the correct answer because it yields the most relevant information from the corpus. There was a good amount of hyperparameter optimization necessary for this such as constraints on the proximity between certain keywords, determining which keywords were necessary and which were optional, and the weighting of terms. That being said, this method resulted in an answer selection much faster than a custom function would have and has strong precision and recall on given questions. This is especially true for the first two questions on CEOs and bankruptcy, where the questions can be tested and validated on more than the given 3 questions.

After the potential answers are scored using the solr search function, the answers are ordered and cleaned. They are cleaned once again to make sure that all of the answers are the proper answer type when used in other contexts in the solr corpus and there was no mistake in the initial labelling. After cleaning, the answers are selected according to answer type (top answer if one answer desired, all answers if a full list required like in the bankruptcy question, and the top 100 results when no constraints given like in the affecting GDP question).

**Conclusions./Business Insights**

A question and answer system can be incredibly valuable in a commercial environment due to the amount of time it can save. Without this system, a human would need to manually read through the documents to find the answer. The alternative could be googling for the answer and then reading the selected documents, but this still takes a much longer time than being given an exact answer. This is a very useful time saving tool, especially if a large number of repetitions are required. For example, if a company is looking to conduct lead generation for C-suite executives on fortune 500 companies, a q and a type system could be implemented to loop over these queries and store the results for each company. This would save an enormous amount of time over manually googling and parsing through results.

Even without a 100% accuracy rate, this system saves large amounts of time by giving finding a large number of easily found solutions. This saves time for the humans to manually parse thorugh potentially more difficult questions, such as C-suite positions with recent turnover. Time is saved because a recommended name is given that could be easily searched to verify or discount, often saving parsing time.