QA System

Anand Raman

March 2020

1 Executive Summary

In this assignment, Business Insider articles were used to answer three types of questions.

- Who is the CEO of company X?
- What affects GDP?
  - What percentage of drop or increase is associated with X?
- Which companies went bankrupt in month X of year Y?

The tool is a python script that runs in the command line. It leverages ElasticSearch to query the 730 Business Insider articles from 2013 and 2014. For each question, relevant documents are selected. For CEO and company related questions, the top scored document is broken into sentences and queried again, after which the top sentences are identified. Companies and CEOs are then extracted. For the GDP question, the top 10 most relevant articles are broken into chunks of text surrounding any GDP related sentences. The top 10 chunks of text are returned as the result. The full index of text chunks containing GDP related content are used for subqueries such as "What percentage of drop or increase is associated with investment?". The top 3 chunks of text containing information about factors and their marginal (percentage) impact on GDP are returned.

2 Introduction

The goal of this project was to create a script capable of answering the questions listed in the Executive Summary. The corpus of text was Business Insider articles from 2013 and 2014. Each text file corresponds to a day in each year. Business Insider specializes in technology and financial news among various other topics. To query the data and return relevant results, ElasticSearch was used.

3 Methods

The python script has distinct paths for each question type. But some avenues were shared. Since we are intended to answer three types of questions, and one sub-question, question type was always classified by the first token in the question. 'Who' corresponds to CEOs, 'Which' coresponds to companies, and 'What' corresponds to a GDP query. In addition, before the user is allowed to ask a question, the articles are indexed in ElasticSearch. They had two fields, the date the article was written and the content of the article.

3.1 CEOs

First the question is processed with spaCy and the name of the company is extracted. The full article database is then queried for articles with the company name and 'CEO' as keywords. The top matching document is identified after sorting the matches by ElasticSearch's built in score. The top article is broken into sentences with spaCy's sentencizer and queried again using the same keyword search. The relevant sentences are scored with ElasticSearch's built in score function and the person entity is extracted from the top scoring sentence using spaCy NER. This method was more accurate than my own NER model and was faster but equally accurate to the method that TF-IDF vectorizes sentences and then ranks by score. This was true for all of questions that I was required to answer.

3.2 Companies

The question is processed with spaCy and the date entity is extracted. Using a dictionary of month name, month number key-value pairs, I was able to extract a date of the format YYYY-MM from a given question. The article index was queried for the keyword 'bankrupt' with a filter for articles published in the relevant month and year. The top document was selected and broken into sentences. These sentences were queried for the term 'bankrupt' and then using spaCy NER, organizations were extracted from the text.

3.3 GDP

The article database was queried for the term "GDP" using ElasticSearch. Since there are likely many articles with GDP related content, it would be inaccurate to only select the top article. So the top ten articles were selected and processed with spaCy. The text split into sentences was searched for the term "GDP". Upon matching the term in a sentence, both the previous sentence and the following sentence were joined to the relevant sentence in the correct order. This was due to the fact that many sentences about GDP contain description of factors affecting GDP in the previous or following sentence. The result is evident in my sample output. Below is an example of sorted scores that were used for key document selection.

```
[0.098375306,
 0.098163806,
 0.09812673,
 0.09780524,
 0.097567536,
 0.09752772,
 0.09748583,
 0.09748288,
 0.09745113,
 0.09714462]
```

Figure 1: GDP: Top Document Scores

4 Results

The quality of results varied by question type.

4.1 CEOs

CEO identification was reasonably successful. For larger companies like Facebook, Oracle, Apple, and Google, the finished product works quite well. Sometimes, however, a different permutation of the name will return a more accurate result. For example, "Who is the CEO of Google?" returned an inaccurate result, which "Who is the CEO of Google, Inc.? returns the correct answer: Larry Page. If I were to take on the project again, without the guideline of returning just a first and last name, I would probably choose to return 3-5 candidate sentences alongside a list of candidate names. This way, a user would be able to identify the context surrounding an extracted entity. However, since the requirement was to just return a the top ranked name, I did not use this approach.

4.2 Companies

Overall, identifying companies that went bankrupt in a given month was the least successful. There are three primary cases that I identified. First is the false positive case, where the system simply returns an incorrect answer. I included one of these in my sample output. The QA system returned [Zames, JPMorgan]. JPMorgan never went bankrupt and Zames turns out to be one of their employees. I identified the candidate document and it appears that the article was talking about how JPMorgan managed to make a lot of money buying bankrupt firms. The second case is more sensitive to the presence of bankrupt firms, but less precise. Since I am extracting all the company names from candidate sentences, a lot of extra, non-bankrupt companies can end up included. In my sample output, the query correctly identified Trump Entertainment as a bankrupt entity but it also included US Bankruptcy court, because of the match on 'bankrupt'. The third case problem I identified is a problem related to dates. While my query can successfully find articles written in a certain month of a certain year, there is no guarantee that the company filed for bankruptcy in that month or year. However, the fix for this is not simple. It

is not feasible to just include a date in querying the content of the article itself because sometimes articles will say things like "This morning company X filed for bankruptcy." So in either case, items can be missed or incorrectly identified. In my test cases, on average, it seemed that there was at least one bankrupt company included in the list of identified candidates.

4.3 GDP

The query about what affects GDP was quite successful because the formulation of the query was very precise. After discussing options for returning candidate answers with the professor, he told me I should return top ranked passages of text containing explanations of the factors that impact GDP. The user is required to do some brief reading to extract factors. The top candidates chunks are quite good and contain items such as "trade", "exports", "personal consumption" etc. The sub-queries were also quite successful and fast as they queried a database that had already been populated with candidate sentences. The answers were all quite accurate as well and the result is returned as a list of 3 chunks of text containing candidate answers.

5 Conclusion

Querying text data effectively is difficult and can vary considerably based on input from the user. Oftentimes the best solution will not return a single answer but many candidate answers that also include the context of the selection. Herein lies a major opportunity for improvement for any use case where users are extracting data from a text database. By automating the document query process and giving many candidate answers to a user, then allowing the user to select the "correct" answer, a company could simplify it's text extraction process while also building a "training" dataset that specifies queries and answers. Then, potentially, a machine learning algorithm could be deployed in combination with the query to refine the result further.