Domain Specific Q&A System

Yahia El Bsat
IEMS 308

Executive Summary

A Q&A system was developed in python to answer question of the following types:

1) Which company went bankrupt in [month] [year]?
2) Who is the CEO of company X?
3) What factors affect GDP?
4) For a given factor, what percentage increase or decrease in GDP is associated with it?

The Q&A system uses a mixture of different techniques to extract answers from all news articles of Business Insider in 2013 and 2014. That means that answers are limited to that time period and it is out of date for the most part.

It builds on top of Elastic Search to query and score documents and uses a pipeline from the question asked, to identifying the question type, processing an adequate query, getting highly scored articles through Elastic Search, extracting relevant sentences, and finally returning the most likely answer

This report serves to outline the methodology and techniques used in that system, sample outputs, instructions for use and next steps.

Methodology and Implementation

Elastic Search

Elastic Search which is built on top of Apache is used as the back end for the system. It is preloaded with all articles in the corpus indexed by date, month, year, and file ids to facilitate queries and return useful results.

Elastic Search must be downloaded and be running on port:9200 for the Python script to connect to it.

After cleaning the raw corpus due to unsupported Unicode characters. The daily txt files are then split into article files (multiple articles in a day) with the assumption that a new line character means a new article. This seems mostly correct after evaluating the corpus over many files.

Once all articles are processed (35231 articles), we load them with indexing into Elastic Search.

Question classification

Two alternatives were available for classifying questions into the 4 types outlined in the executive summary. Either a rule based approach or a classification model. Given that we only have 4 question types to work with, and because of the lack of training questions for each type, we opt for the rule based approach.

After preprocessing and tokenizing the question terms and taking out common stop words we create a simple rule for each question type which looks for keywords in the question to be able to classify it. For example, for the first question type, we look for words like ‘bankrupt, ‘bankruptcy’ or ‘chapter’, for the second question we look for ‘ceo’, ‘chief’, ‘executive’, ‘leader’…

The classification of the questions given what we are anticipating is giving good results and there is no need at this stage to move to the machine learning approach.

<u>Query formation</u>

After identifying the question type, we need to generate an adequate query to search the corpus for articles in Elastic Search.

We develop slightly different query formation algorithms for each question type as outlined below.

<u>Type 1 (bankruptcy):</u>

After processing the question and looking for the month and year requested, we generae a query where the article must both include any of bankruptcy keyword as well as the date and year in question.

<u>Type 2 (CEO):</u>

The query looks for a CEO keyword and the name of the company requsted

<u>Type 3 (GDP):</u>

We look for both the terms GDP or gdp and any combination of the words (affect, affects, effects, effect, increase, decrease, rise, drop, growth, drag.

<u>Type 4 (GDP Factor percent):</u>

To identify the factor we are looking for, we look for the last word after a preposition "ADP" (using POS tagging). We then query for documents that include GDP and the factor and any combination of the effect words that are in type 3 above

<u>Searching for articles</u>

Once we have the query ready, we query Elastic Search and preserve the top 50 scoring documents which was tested to yield satisfactory results

<u>Answer Analysis</u>

With our results ready, we need to look for relevant sentences in top 50 documents retrieved from the corpus. That is also dependent on the question type.

<u>Type 1 (bankruptcy):</u>

We look for all sentences in the articles identified that contain a bankruptcy related term as well as the month and year in the question.

With the identified sentences, which are tokenized and preprocessed taking out stopwords and tagged with nltk tagger for POS, we proceed to finding the most common company name in those sentences.

For that, we use our NER classifier from the previous homework which was a random forest classifier yielded very satisfactory results. The random forest is saved as .joblib file through pickles so it can be used without having to train the model again. Each sentence is vectorized using the features of the classifier, and we predict whether the keyword identified is a company name or not. If it is, then we append that name to our list of likely companies and we return the company name with the highest number of occurences.

<u>Type 2 (CEO):</u>

Similar to the bankruptcy answer analysis, we look for sentences that include that both include the company name we are querying and any of the CEO keywords outlined earlier.

With those sentences, we use our CEO NER classifier applying the same preprocessing as in the bankruptcy case to build out the features and predicting whether the names are those of CEOs. With that, we return the CEO name that occurs the most in our sentences.

Type 3 (GDP factors):

The GDP factors answer used a different methodology. We use tf-idf on a series of unigrams and bigrams in the identified articles and score the different n-grams for relevancy. We then pick the most relevant ones. Because of the long run time of tf-idf and given that our corpus is immutable for now, we have tweaked the function to return those terms as hard coded instead of running the scoring every time. The default parameter (run) for the function get_gdp_factors(…) can be changed to True to run it live instead of getting the hard coded answer.

Type 4 (Percent associated with GDP):

This is the most funky part of the implemented system as the results are purely based on nlp and can vary greatly based on the context. But for the purpose of this assignment, we first identify all sentences in the returned query result that include GDP, a GDP change term, and the key factor we are looking for.

With those sentences, we use the regex patterns developed in the previous homework to extract percentage values (which could be either %, percent, percentage, perc. …).

With the extracted percentage values, we return the most occurring one.

A shortcoming here is that we are relying on the string version of the percentage and if one is written in multiple forms it can be over scored by another term that was more consistent. Future work on this system can convert the literals to floats of decimals, then count there, or even return ana averaged value.

Question - Answer Pipeline

The full procedure above is summarized in a pipeline function called answer_question which takes as an input the question, classifies it, get the correct query, searches for articles in Elastic Search, analyze the results and return the most likely answer. It was also developed to take unsupported questions into consideration and a full error fallback so as to not disrupt the runtime environment.

Results and Analysis

Type 1 (bankruptcy):

```
>>> answer_question('Which company went bankrupt in September 2008?')
'Lehman Brothers'

>>> answer_question('Which company went bankrupt in February 2013?')
'Gox Mark Karpeles'

>>> answer_question('Which company went bankrupt in November 2011?')
'MF Global'

>>> answer_question('Which company went bankrupt in May 2012?')
'Energy Future Holdings'
```

We can see from the above examples that we get company names in return to our questions. We can also see that the question must not be limited to the dates of the articles (2013, 2014) but we can go back in time and see if there is information in our corpus about historical events like the Lehman brothers one.

These questions were curated to get nice answers, but the QA system does not always perform well on this task, while it tends to return company names most of the time, we get some other terms sometimes which is related to the quality of our NER classifier.
Further, for some months, we do not get any answer, either because no bankruptcies happened at that time or because we couldn’t identify company names in the returned results.

Type 2 (CEO)

```
>>> answer_question('Who is the CEO of Tesla?')
'Elon Musk'

>>> answer_question('Who is the CEO of Facebook?')
'Mark Zuckerberg'

>>> answer_question('Who is the CEO of Snapchat?')
'Evan Spiegel'
```

CEO questions are the most performant in this implementation and we can constantly get good results. However, like others, we sometimes predict incorrectly that a pronoun is a CEO name when it is not and that is returned instead but for the most part, we get correct answers for the big and well covered companies in the news.

Type 3 (GDP Factors)

The following GDP factors were identified:
'interest rates, credit, investments, inflation, government, growth, debt, labor, markets, economy'

These make sense economically.

Type 4 (GDP Factor percent impact)

```
>>> answer_question('What percentage drop or increase is associated with exports?')
'1 percent'

>>> answer_question('What percentage drop or increase is associated inflation?')
'7 percent'

>>> answer_question('What percentage drop or increase is associated with economy?')
'5 percent'

>>> answer_question('What percentage drop or increase is associated consumer spending?')
'1 percentage points'
```

We are able to retrive percentage values for all the factors identified above. But the shortcoming of these results was outlined previously where the percentages are identified as strings instead of numerical values loosing some relevance along the way. We must also say that answers to those questions should only be used as a rough estimate as the context of where the answer came from is not clear and it can conditional on some other variables or changes and the overall state of the economy. Take this as a proof of concept rather than a trusted and reliable source of information.

Next Steps

- Improve performance of NER classifiers especially with regard to False Positives
- Convert question classifier away from a rule based approach to an appropriate feature based ML model to cater for wider type of questions and question formats relating to the same topic
- Fine tune sentence scoring after query generation
- Topic tagging and modeling of articles as an additional index for a faster and more accurate way of retrieving information

Instructions for Use

Expected setup and run time: 5min

1) Download all python dependencies needed (list on top of main.py)
2) Put the downloaded 2013 and 2014 articles all in the folder 'articles_pre' in the main directory of the project alongside main.py
3) Ensuring directories './days/' and './articles/' in the main directory of the project are created and empty
4) Download and install Elastic Search
5) Running Elastic Search in the background on port 9200 and making sure to empty all index and any data in storage
6) Making sure that features.csv, rf_ceo.joblib, and rf_comp.joblib are available in the main directory
7) Opening main.py and changing ppath variable at the top to the project directory path.
8) Running all code in main.py to set the environment
9) To ask a question, simply use answer_question('your question here') in the console