Question and Answer System

Enzo Profli

1 Executive Summary

In this report, we look into a corpora of articles extracted from Business Insider in 2013 and 2014, in order to process relevant text information contained in those articles. The objective is to create a Question & Answer system, that takes an user question as input and, given Business Insider articles, returns a relevant answer.

The following steps were taken to generate the model:

- Create an answer type prediction machine learning model
- Input the question and parse it, generating keywords
- Run a TF-IDF analysis and return the 10 most adequate documents, given keywords
- Run another TF-IDF analysis and return the 10 most adequate sentences
- Score potential answers, and return final answer

These steps generated a competent Q&A system, that is particularly good at returning person-type answers (CEOs, presidents, etc.). However, it has a few shortcomings that prevent it from being extremely precise on its answers. The following sections present the methodology and circumstances in more detail.

Finally, to run the model, run the Profili_Enzo_QA_System.rmd chunks, and then run the QA_System() function, with no arguments. You will be prompted to enter a question.

2 Problem Statement

In this report, we look into a corpora of articles extracted from Business Insider in 2013 and 2014, in order to process relevant text information contained in those articles. The objective is to create a Question & Answer system, that takes an user question as input and, given Business Insider articles, returns a relevant answer.

3 Methodology

The first step in the process is generating a machine learning model able to determine the type of answer required, given the question. To this end, I have used the University of Illinois Urbana-Champaign Experimental Data for Question Classification dataset, which contains over 5,000 questions, along with a descriptive identifier. I translated these identifiers into Name Entity Recognition tags, so that they can be used later on. Given this dataset, I generated features based on a word + POS tag combo (for example, What_DET is one of the columns). I did not add bigrams since their combination would generate too many columns, and would not be feasible in a regular computer. I then applied a Random Forest model to this data, and prediction was relatively good: it got the correct type of NER tag 75% of the time. This modeling process can be seen in helper_function.Rmd, and, for completeness, the confusion matrix of this model on an unseen testing subset of the UIUC dataset is available in Appendix A.

The next step was receiving the question and adequately parsing it - the parse_question() function in python_functions.py performs this role. It takes a question as an input from the user, tokenizes the question, applies POS tags and returns keywords. The next function (question_class_prediction()) predicts the type of answer required by the user. For example, “Who” questions will probably be predicted as PERSON-type answers.

Now that the question is in adequate form, we look for the most adequate documents to answer this question. To this end, we perform a TF-IDF analysis of documents to select the n most adequate documents. The function score_docs() takes the TF-IDF matrix and sums keyword TF-IDF values for each document, and returns the n documents with highest value - this ensures that rarer keywords have a larger weight in the decision. Once we have the documents, we conduct sentence segmentation and a TF-IDF analysis on sentences, so that we can perform a similar procedure with the score_docs() function - this time with sentences. The product of this section are the n most adequate sentences in the selected documents.

Once we have the sentences, the retrieve_answer() function returns the answer to the question. It does so by applying a NER model, using the Spacy library in Python, to the selected sentences. Once we have NER tags, we count the most common “correct” NER tag expression, as determined by question_class_prediction(), and returns that as the answer. If no suitable answer is found, it returns “I don’t know!”.

Finally, the QA_System() function wraps this entire procedure in a fast function. To run the function, you should run the Profili_Enzo_QA_System.rmd chunks, and then run the QA_System() function. You should have python_functions.py, question_classification_model.rds and model_features.rds in the same directory.

4 Analysis

For the purposes of this report, I modified the QA_System() function so that it shows not only the answer, but the type of answer it is looking for, as well as all possible candidates and their count in the selected sentences. The answer would be the first name in the list, so, for example, in the first query below the answer would be Steve Ballmer. Below I present some examples of results from running QA_System().

- Enter question here: Who is Microsoft’s CEO?

PERSON Counter({‘Steve Ballmer’: 6, ‘Satya Nadellarecently’: 1, ‘Mark Zuckerberg’: 1, ‘Facebook’: 1, ‘Thinkorswim’: 1, ‘Bill Gates’: 1})

- Enter question here: Who is Apple’s CEO?

PERSON Counter({‘Tim Cook’: 4, ‘Carl Icahn’: 2, ‘Carl Icahnissued’: 2, ‘Leon Cooperman’: 1, ‘Cooperman’: 1, ‘Michael Brown’: 1})

- Enter question here: Who is Google’s CEO?

PERSON Counter({‘Hotel Finder’: 1, ‘Glass’: 1, ‘Facebook’: 1, ‘Steve Ballmer’: 1, ‘Patrick Pichette’: 1})

- Enter question here: Who is the president of China?

PERSON Counter({‘PayPal’: 1, ‘Zong’: 1, ‘Thomson ReutersChina s’: 1, ‘Xi’: 1})

- Enter question here: Who is the president of Brazil?

PERSON Counter({‘Dilma Rousseff’: 4, ‘Petrobras’: 3, ‘Dilma Rouseff’: 1})

- Enter question here: Who is federal reserve governor?

PERSON Counter({‘Ben Bernanke’: 2, ‘William C. Dudley’: 1, ‘Christine Cumming’: 1, ‘Charles I. Plosser’: 1, ‘Jeffrey M. Lacker’: 1, ‘Sandra Pianalto’: 1, ‘Charles L. Evans’: 1, ‘Richard W. Fisher’: 1, ‘Dennis P. Lockhart’: 1, ‘John C. Williams’: 1, ‘Janet L. Yellen’: 1})

- Enter question here: Which company filed for bankruptcy?

ORG Counter({‘GT Advanced’: 1, ‘GM’: 1, ‘Opel’: 1})

*Enter question here: Which company filed for bankruptcy in April 2014?

ORG Counter({‘Fed’: 1, ‘Residential Investment’: 1, ‘REUTERS’: 1, ‘Rolling Stone’: 1})

*Enter question here: what affects GDP?

PERSON Counter()

As we can see above, the system is somewhat competent in determining CEOs, presidents, etc. There are a couple of strange answers, such as Mr. Hotel Finder as Google CEO and Mr. Paypal as the Chinese president, but it got the other answers correctly. The random forest model got it right that a person should be the answer every time. As for bankruptcies, it got correctly that both GT Advanced and GM/Opel filed for bankruptcy, but I am not sure if recall is high in this question - it is possible that many other mentioned companies slipped by. I also tried being more specific, asking for companies bankrupt in April 2014, and answers were less satisfying. I believe the other keywords ("April" and "2014") may have shifted the system's focus away from bankruptcy. It is also possible that the question is too specific, and articles from April/2014 don't mention bankruptcies.

Finally, I tried asking what affects GDP, and the random forest model incorrectly guessed that the answer should be a person, which barred the system from doing a good job. If I had not altered the function, it would say "I don't know!", as discussed in the methodology section. I manually checked the selected sentences and they looked interesting - although they discussed GDP figures, not factors behind growth.

This brings us to some shortcomings in the model. The first one is that the question type prediction model is not extremely accurate, as the number of questions it was trained on is not very large, and they are broad questions, not focused on business. For example, GDP is not a unigram feature in the model, so the model just assigns the question to PERSON, as it is the most common type. This factor reduces the system's precision.

Another shortcoming of the model is that it requires that keywords be in sentences, which is not always necessary. For example, an article might mention a synonym of bankrupt, and the system would not catch this deviation - this definitely reduces recall. Finally, this system does not distinguish between keywords. For example, in the last question, some of the selected sentences may include "April" and "2014", but not bankruptcy, even though "bankruptcy" (or its lemma) should be a necessary condition for the sentences, while "2014" is not necessarily essential, depending on how the sentence flows.

5 Appendix A: Random Forest Model Confusion Matrix

```
##                Reference
## Prediction      CARDINAL DATE EVENT GPE LANGUAGE LOC MISC MONEY NORP ORG PERSON
##   CARDINAL            82    0     0   0        0   1    4     4    0   0      0
##   DATE                 1   40     0   1        0   1    0     1    0   2      2
##   EVENT                0    0     3   0        0   0    0     0    0   0      0
##   GPE                  0    0     1  61        0   0    0     0    0   1      2
##   LANGUAGE             0    0     0   0        2   0    0     0    0   0      0
##   LOC                  0    0     0   2        0  80    6     1    0   3      3
##   MISC                 4    1     3   1        0   3   14     2    0   6      4
##   MONEY                0    0     0   0        0   0    0     5    0   0      0
##   NORP                 0    0     0   0        0   0    0     0    0   0      0
##   ORG                  0    0     0   0        0   0    1     0    0  17      2
##   PERSON               2    2     4   4        1   5   13     2    0  17    175
##   PRODUCT              0    0     0   1        0   2    4     0    0   4      1
##   TIME                 0    0     0   0        0   0    0     0    0   0      0
##   WORK_OF_ART          1    0     0   0        0   0    1     0    0   0      3
##                Reference
## Prediction      PRODUCT TIME WORK_OF_ART
##   CARDINAL            0    1           0
##   DATE                1    0           0
##   EVENT               0    0           0
##   GPE                 1    0           0
##   LANGUAGE            0    0           0
##   LOC                 9    1           3
##   MISC                8    0           1
##   MONEY               0    0           0
##   NORP                0    0           0
##   ORG                 0    0           0
##   PERSON             12    1          12
##   PRODUCT            12    1           3
##   TIME                0   11           0
##   WORK_OF_ART         1    0          22
```