Project 4
Question Answering

CS535 – Fall 2016

Team Illuminati

Anand Prashant Popat | apopat | 50203882

Divya Mishra | divyamis | 50208069

Nitika Chaudhary | nitikach | 50209474

Raj Jayasukh Patel | rajjaysu | 50208278

1. Overview

Question Answering is a major topic in the field of Information Information Retrieval and implements Natural Language Processing (NLP) and Data processing to generate short and the most relevant answer to the question asked. It's one of the most researched topics currently in the field of IR and its implementation can be seen in the form of Apple's smartphone assistant Siri and Google's Allo or Google Now. As it reduces the amount of work the user has to put in order to find an answer on the internet and for all the time it saves, it is seen as the future of Information Retrieval.

2. Key Features

What differentiates Question Answering system from something like a chat bot is that it aims at producing short and correct answers to the question asked rather than retrieving a paragraph or in our case, the whole tweet.

The key features include:

- Clean UI. Just type the query in a simple search box
- Suggestions. The text box will show suggestions on the type of questions that can be answered
- Freedom to ask any question as long as it start with an interrogative word
- Exact answer in the shortest form possible will be shown in the box unless it doesn't exist
- Below the box on the result page will be the tweets from which the result was derived

As the purpose of this project is to devise a strategy to extract data from the database which makes sense and the database is the tweets that we collect, the system will not be able to answer something that is out of topic (i.e. other than demonetization) and we are not searching the internet or Wikipedia to produce answers as we believe that extracting the answer from the tweets that we collect is the central idea of the project.

3. Implementation

We broke down the project into four parts:

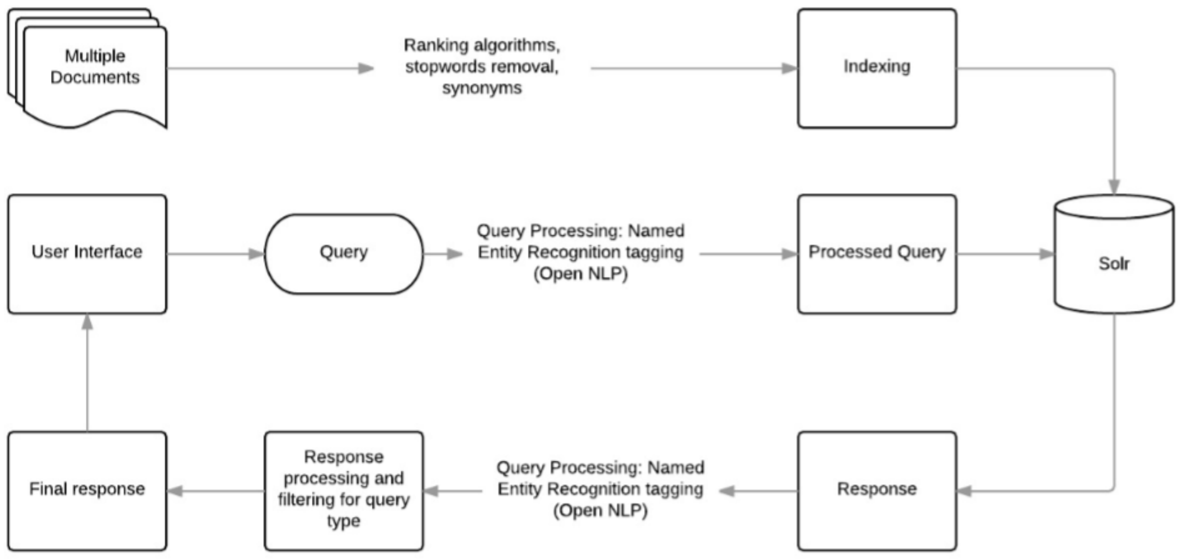
- **Database:** The amount of tweets and the quality of the tweets that we collect is really important so we collected tweets using different hashtags, mentions and handles to collect information related to all the subtopics. We collected more than 80,000 tweets (without retweets) from different time frames to make sure we collect as much information as possible.
- **Indexing the database:** Ranking algorithms like DFR, VSM and BM25 were used to see which one suited our system the best. We tweaked the parameters to produce the best results. Stopwords removal and synonyms were also implemented to get the most relevant tweets at the top of the retrieved list.
- **Understanding the query:** One limitation of this project is that the question should start with an interrogative word. The purpose is to make it easy to understand what kind of question it is. We use Part-Of-Speech (POS) tagger (openNLP) to break down the query and understand it. Depending on the structure, the program understands the context of the query and uses the figure of speech to implement the strategies explained in the next step.
- **Implementing the strategy:** We have devised five strategies depending on the interrogative word used and the context of the query. The table below describes them in detail.

Question Type	Strategy applied
Who	*Who* type questions expect *person name* as an answer **Query Processing:** • "NNP", "NN" and "IN" tags are retrieved from the query and are then searched against Solr in the form *NNP+NN+IN.* **Response Processing:** • For each document retrieved, the "NNP" and "NN" tags are picked from the Solr response. • Each word is ranked as per its occurrence in the response and simultaneously irrelevant words are removed (for e.g. co, t). • Depending on the question, the top term or the top 5 terms will be returned.
When	*When* type questions expect *time* as an answer **Query Processing:** • If there is "did" in sentence, the word present in "VBD" will be converted to "VB" form (Gets the present Verb form from Past tense Verb form). • The words present in "NNP" and "VB" are searched against Solr in the form NNP+VB+"on". Here "on" is used as the sentence formation of the answer will always contain "on" before the date which is the answer. E.G: Demonetisation policy is implemented **on** 8th November, 2016.

	Response Processing: • The word/number following "on" in the retrieved results is first searched for a number in the form (dd/mm/yyyy) or (dd/mm) in CD (Cardinality Number) • If no such format is found, the word following "on" will be matched with January to December or Jan to Dec or Monday to Sunday or Mon to Fri (Sat and sun are not used as they have other meanings). • If any of such word matches, the word along with the number before or after the word (e.g. 9 Feb or Feb 9) will be taken as the answer. • If no such pattern is found, the number present in the CD will be taken as answer.
How	*How* type questions are complex to answer and expect an explanation. But its sub-category *How many* expects a number. **Query Processing:** • If "How" is followed by "many" in the query, "WRB" and "many" from "JJ" tags are removed. • If "many" is not found, the query is stripped of "how" and the remaining query is searched against Solr. **Response Processing:** • If numerical response is required, then for each document, the numbers in the "CD" tags are picked from the top 10 Solr response. The number with most recurrences is returned. • Else, all documents are returned from the response.
Where	*Where* type questions expect *place* as an answer. In our case, we have assumed that the questions having "when" will be related to demonetization and hence the result query would be containing a place along with a date E.G.: "Where was demonetization policy announced?" The result queries would have a format like "The demonetization policy was announced in New Delhi on 8th November". Based on the topic, use of "where" type of questions is quite less and hence its implementation is also limited. **Query Processing:** • The query is stripped of "where" and is searched against Solr. **Response Processing:** • For each document in the top 10 Solr response, the one containing both "NNP" and "CD" tags are picked and returned.

What	*What* type questions are difficult to answer and hence the results best matched for the query are shown. **Query Processing:** • Query is stripped of "what" and searched against Solr. **Response Processing:** • Top tweets found by the Solr are returned.

The flow of the project can be better understood by the following flow-chart:


Multiple Documents
Ranking algorithms, stopwords removal, synonyms
Indexing
User Interface
Query
Query Processing: Named Entity Recognition tagging (Open NLP)
Processed Query
Solr
Final response
Response processing and filtering for query type
Query Processing: Named Entity Recognition tagging (Open NLP)
Response

4. Team Work

All four of us contributed equally in researching and deciding the strategies used in this project. All of us worked on all the domains of the project bringing flexibility in work distribution.

5. Results

- The results that the system gets are NOT at all hardcoded. Each time the user fires a query, it goes through the general algorithm mentioned in the sections above and calculates the result. But Question Answering is such a vast topic that we cannot expect a particular algorithm to give correct answers every time making it an ever evolving algorithm.
- The question types suggested in the suggestion bar will produce correct outputs along with other queries that are of the same type.
- We have not used Wikipedia or the web and hence it will not produce relevant answers to questions that are out of scope of the topic.
- As we did not intend to make it function like a chat bot, it will not produce simple answers to questions like "How are you?". Rather it will search the database for the phrase and will give relevant tweets.
- We tried implementing speech recognition but due to limitations put by Google on using Web Browser's speech recognition and discontinuing x-webkit toolkit, it doesn't support speech recognition yet. This is one of the things that can be implemented later to improve the project.

6. References

1. https://piazza-resources.s3.amazonaws.com/isg3aqua4r76my/iw2pgri5sby3w6/Rec13Project4.pdf?AWSAccessKeyId=AKIAIEDNRLJ4AZKBW6HA&Expires=1481340588&Signature=k3FJ4W95ZiQZy0X8VLOAyROb8Rc%3D

2. https://piazza-resources.s3.amazonaws.com/isg3aqua4r76my/iwdiqzok5rr5we/Rec14FinalReview.pdf?AWSAccessKeyId=AKIAIEDNRLJ4AZKBW6HA&Expires=1481340613&Signature=oWxhP0Bl2Tc7XgJM4d0G1X49FPI%3D

3. https://en.wikipedia.org/wiki/**Question_answering**

4. https://web.stanford.edu/class/cs124/lec/qa.pdf

5. https://web.stanford.edu/~jurafsky/slp3/28.pdf

6. https://lucidworks.com/blog/2013/10/22/road-to-revolution-shrinking-the-haystack-using-Solr-and-opennlp/

7. research.microsoft.com/en-us/um/people/sdumais/EMNLP_Final.**pdf**

8. https://perso.limsi.fr/bg/pageWebPHP/fichiers/GrauKO02.pdf

9. http://www.cs.columbia.edu/~kathy/NLP/ClassSlides/Slides09/Class18-QA/myQA.pdf

10. https://cs.uwaterloo.ca/~jimmylin/publications/Lin_Katz_EACL2003_tutorial.pdf

11. http://stackoverflow.com/questions/4795807/question-answering-with-lucene

12. http://trec.nist.gov/pubs/trec11/papers/pohang.seunghoon.pdf