CS 6320: Natural Language Processing

Spring 2018 - Dr. Mithun Balakrishna

University of Texas, Dallas

Course Project: FAQ Semantic Matching

Group Name:
Inquisitors

Group Members:

- Shubham Kothari
- Priyanka Joshi

Index

1. Introduction

The Semantic FAQ finder is an information-access system that retrieves frequently asked question from the given corpus. Based on semantic knowledge of WORDNET to improve its ability to match the given input question to the closest question and answer pair from the corpus. If the question happens to be like one of the frequently-asked ones whose answer has been recorded in a FAQ corpus, FAQ Finder is likely to return the appropriate answer. The system is designed to take an input question from the user and compare it with the corpus given. A lot of features are extracted from both the input question and every question – answer pair, and then the matching happens to get the best result. For the corpus we have used 50 question answer pairs from Yelp frequently asked questions.

2. Problem Statement

To implement a Frequently Asked Questions (FAQs) semantic matching application that will produce improved results using NLP features and techniques. To implement a bag-of-words strategy and an improved strategy using NLP features and techniques.

Input:

- Set of FAQs and Answers
- User's input natural language question/statement

Output:

- One or more FAQs and Answers that match the user's input question/statement

3. Solution Proposed

Approach 1:

Implement a shallow NLP pipeline and bag-of-words matching algorithm
Given - A corpus of a set of 50 Questions and answers
First the corpus was parsed and tokenized. The tokenized corpus was then separated into separate lists of questions and answers. Then a bag of words was created containing each tokenized question answer pair in the corpus. No stop words were removed.
User's input question/statement was also tokenized and converted into a bag of words.
The bag of words from input question/statement were then statistically matched with the FAQs and answers in the corpus using word overlap. Top 10 matched FAQs and answers are displayed and ranked. Evaluation metric used is MRR (Mean Reciprocal Rank).

Approach 2:
Implement a machine-learning, statistical, or heuristic (or a combination) based approach to semantically match the user's input question/statement to one or more FAQs.
Given - A corpus of a set of 50 Questions and answers
To implement the above approach first, a deeper NLP pipeline is used to extract semantically rich features from the FAQs and Answers.
Following features were extracted in the form of tokenized bag of words:

1. tokenized bag of words of tokens of question-answer pairs
2. tokenized bag of words of tokens of question-answer pairs with stop words removed
3. tokenized bag of lemmas of tokens of question-answer pairs
4. tokenized bag of stem of tokens of question-answer pairs
5. tokenized bag of pos tags of tokens of question-answer pairs
6. tokenized bag of head words of sentences of question-answer pairs
7. tokenized bag of hypernyms of words of question-answer pairs
8. tokenized bag of hyponyms of sentences of question-answer pairs
9. tokenized bag of holonyms of sentences of question-answer pairs
10. tokenized bag of meronyms of sentences of question-answer pairs

For removing punctuations, we created our own function to replace certain special characters like " ' " (apostrophes), double quotes.
The nltk library was used for removing stop words, lemmatization, stemming and assigning part of speech tags. For extracting head words from dependency parse trees, we used Stanford corenlp. We used wordnet from nltk to extract hypernyms, hyponyms, holonyms and meronyms.
These features were then converted into a dictionary and indexed into solr in json format. This was used to train the solr. We then extracted the same features for 10 user input natural language question/statement which were used as a query to the solr.
The solr gave a similarity score and gave as output the top – 10 Faq matches for given input query.

Programming tools: -

Language used

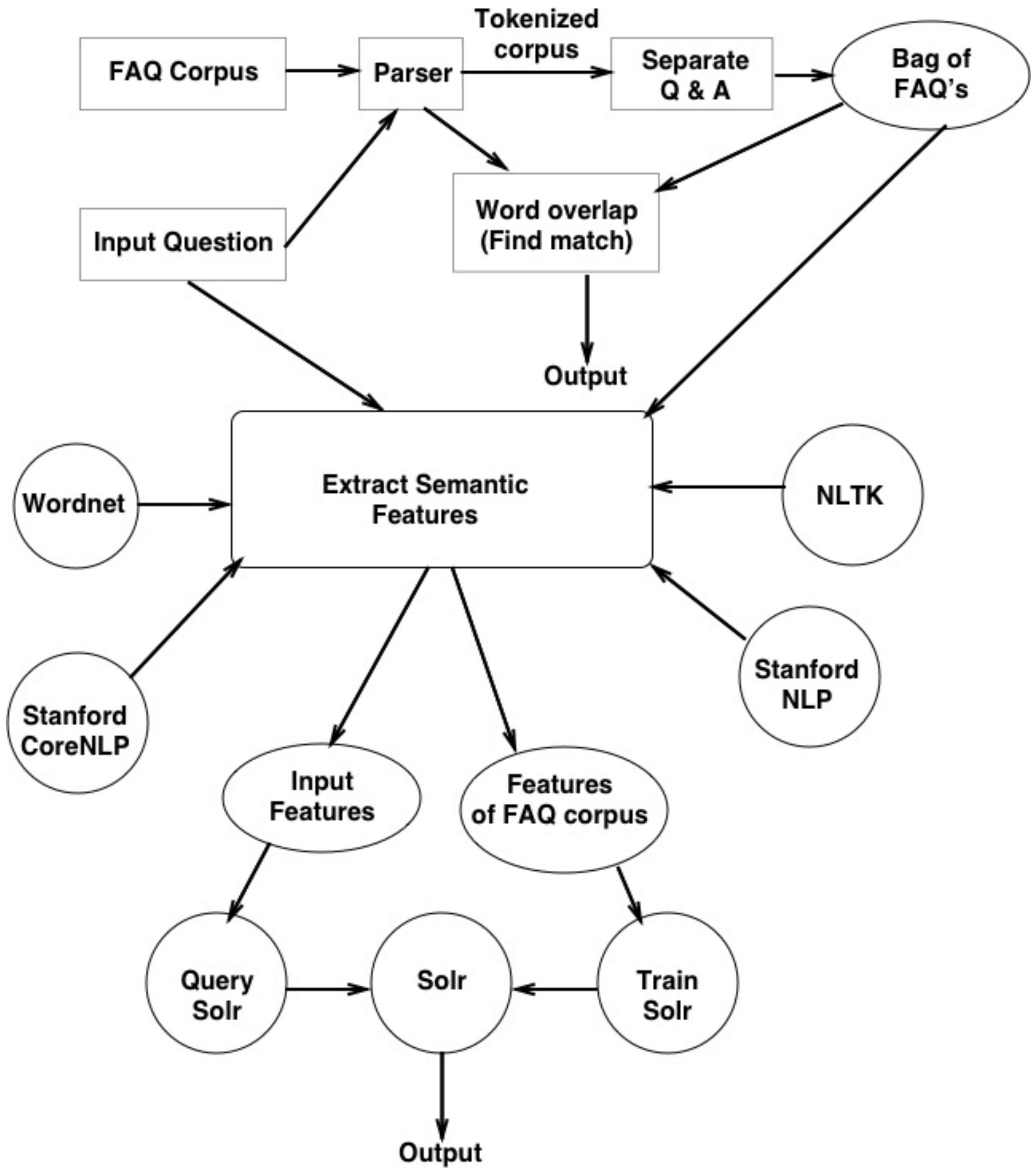
- Python 3.6.4

Libraries Used

- numpy
- scipy
- stanfordcorenlp
- pysolr
- collections
- nltk - sent_tokenize, word_tokenize, PorterStemmer, WordNetLemmatizer

Tools - Sublime text, Anaconda, Solr

Architectural Diagram: -



Output is the top – 10 returned faq matches for user input natural language question/statement.

Evaluation Metric: -

We have used MRR (Mean Reciprocal Rank) to evaluate and compare both approaches – The naïve bag of words matching algorithm and the solr query search.
MRR is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness.

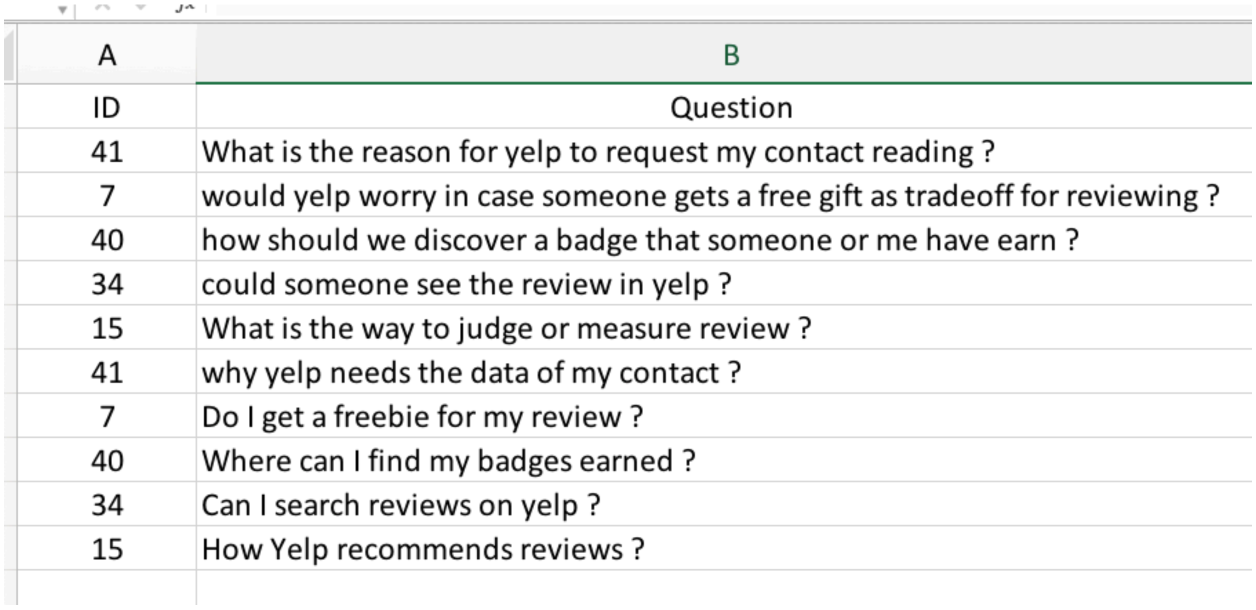
$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\mathrm{rank}_i}.$$

Where,

Rank$_i$ refers to the rank position of the *first* relevant document for the *i*-th query.

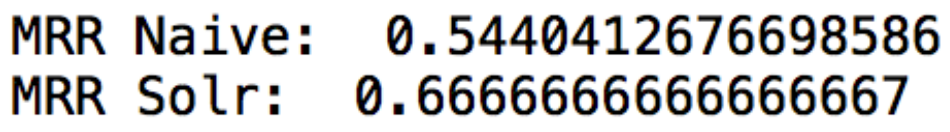
The reciprocal value of the mean reciprocal rank corresponds to the harmonic mean of the ranks.

We created a gold set with 10 test questions which were variations of the original one's and saved the id of the original one that they should ideally match to.



A	B
ID	Question
41	What is the reason for yelp to request my contact reading ?
7	would yelp worry in case someone gets a free gift as tradeoff for reviewing ?
40	how should we discover a badge that someone or me have earn ?
34	could someone see the review in yelp ?
15	What is the way to judge or measure review ?
41	why yelp needs the data of my contact ?
7	Do I get a freebie for my review ?
40	Where can I find my badges earned ?
34	Can I search reviews on yelp ?
15	How Yelp recommends reviews ?

We then calculated reciprocal value of the mean reciprocal rank for all our test questions which was then used to calculate the MRR.



```
MRR Naive:  0.5440412676698586
MRR Solr:  0.6666666666666667
```

Results and Analysis: -

We realized that solr model performs better when the words entered in the input question are not exactly the same since we have used multiple features like hypernyms, holonyms, head words, lemmas, etc.

Cases where the naïve approach worked: -

1. Can I search reviews on yelp? (Missing words example)

Naïve – Rank 1

Original Question - Who can find me and my reviews on Yelp?

2. Where can I find my earned badges? (Rearranged words example)

Naïve – Rank 1

Original Question - Where can I see all the badges that I or another yelper has earned?

This shows that when the user input question/statement has words same as those present in the question-answer pair in the corpus the naïve approach performs well.

Cases where solr performed better: -

1. What is the reason for yelp to request my contact reading?
   (Lemmatization and replaced word example.)

Naïve – Rank 2

Solr – Rank 1

Original Question - Why does Yelp ask for access to my contacts data?

2. would yelp worry in case someone gets a free gift as tradeoff for reviewing? (Hypernyms example)

Naïve – Rank 3

Solr– Rank 1

Original Question - Why Does Yelp mind if I get a freebie in exchange for my review?

Challenges Faced: -

1. Task 2 – Creating separate lists of question – answer pairs after tokenization. Some questions had multiple questions which needed to be included as a single question.

   Resolution – We kept track of the index of last question mark and then separated the tokenized corpus into separate lists of question – answer pairs.
2. Task 3 – Including part of speech tagged words as features for comparison.

   Resolution – We used only the tags and thus got a compatible format for comparison of pos tagged tokens.
3. Task 3 – Dependency parsing too slow.

   Resolution – To make the dependency parsing run faster we ran it on a Stanford corenlp server instead of running on local machine which decreased the time by 10 times. We also extracted the head words of each dependency parse tree.
4. Task 4 – Improving the solr model.

   Resolution – We changed the weights of the different features to improve the MRR score for solr model. Features like lemmas, hypernyms, head words were given more weightage in comparison to other features like removal of stop words and stemming. We discounted the stemming feature because of similarity between lemmatization and stemming.

Pending Issues: -

1. We did not extract hypernyms of hypernyms which would probably give better results.
2. We did not use word sense disambiguation.
3. Implement a machine learning based model.

Potential Improvements: -

1. We could integrate tf-idf.
2. Decision tree model could be used to improve matching.

References: -

1. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.4236&rep=rep1&type=pdf
2. https://pdfs.semanticscholar.org/752d/d8d0cb28b477d9830857915c5300a1d91d99.pdf
3. https://en.wikipedia.org/wiki/Mean_reciprocal_rank
4. http://nlp.stanford.edu/software/corenlp.shtml