PROJECT REPORT

SEMANTIC MATCHING APPLICATION TO PRODUCE FAQs RESULTS USING NLP FEATURES AND TECHNIQUES

Submitted by
NITHIN VASIREDD(nxv160230)
SOMYA SINGH(sxs161331)

Description

To implement a Frequently asked questions(FAQs) semantic matching application using bag of words strategy and improved strategy using NLP features and techniques. The main objective is to implement a shallow NLP pipeline and a deeper NLP pipeline to perform semantic matching application.

Proposed Solution:

Task1: To create a set of 50 Frequently asked questions as the corpus. Domain of our FAQs is Amazon Web Services EC2.

Task2: To implement a shallow NLP pipeline by performing below steps

Bag of words creation:

- Split the input file into FAQs and segment them into sentences
- Tokenize each sentences into words to create a word vector
- Index Each word vector per sentence into search index in SOLR

Query parsing and search by:

- Run the search query using user input against the sentence word vector present in the SOLR search index created from the corpus
- Evaluate results using 10 search queries for the top 10 returned sentences

Task3: This phase included implementation of a deeper NLP pipeline. The main purpose of this pipeline from FAQs and answers to do the following:

Feature extraction:

- Split the input file into FAQs and segment them into sentences
- The Tokenize each sentences into words
- Remove the stop words from the tokenized words
- Lemmatize the words to extract lemmas as features
- Stem the words to extract stemmed words as features
- POS tag the words to extract POS tag as features using pos_tag.
- Syntactically parse each sentence to extract dependency parse relations
- Extract Hypernyms, Hyponyms, Meronyms, Holonyms of each words as features using WordNet

Query parsing and search by:

- Run above deeper NLP on user's input to extract search query features
- Run Search against separate or combination of search index fields created from corpus

Task4: This phase included implementation of a statistical model to semantically match the user's input question/statement to FAQs using combination of deeper NLP pipeline features. The main purpose of this pipeline is to do the following:

Keyword search index creation and Querying:

- Index each of the extracted features as search fields in SOLR
- Run above deeper NLP on user's input to extract search query features
- Run Search against separate or combination of search index fields created from corpus

Implementation Details:

Task1: This phase included creating a corpus with 50 frequently asked questions from https://aws.amazon.com/ec2/faqs/

Task2: This phase included implementation of a shallow NLP pipeline. The main purpose of this pipeline is to do the following:

Bag of words creation:

- The input file is split into FAQs
- FAQs were segmented into sentences using sent_tokenize of NLTK
- The sentences were tokenized into words using word_tokenizer of NLTK to create a bag of words
- Each bag of words per FAQ were indexed into search index in SOLR using **pysolr**

Query parsing and search by:

- Run the search query using user input against the sentence word vector present in the SOLR search index created from the corpus
- The results were evaluated using 10 search queries for the top 10 returned sentences

Task3: This phase included implementation of a deeper NLP pipeline. The main purpose of this pipeline from FAQs and answers to do the following:

Keyword search index creation:

- The input file is split into FAQs
- FAQs were segemented into sentences using sent_tokenize of NLTK
- The sentences were tokenized into words using wordtokenizer of NLTK
- The stop words were removed using stopwords of NLTK
- The words were lemmatized to extract lemmas as features using WordNetLemmatizer of NLTK
- The words were stemmed to extract stemmed words as features using PorterStemmer of NLTK
- The words were POS tagged to extract POS tag as features using pos_tag of NLTK.

- Each sentence were syntactically parsed to extract dependency parse relations using StanfordDependencyParser
- Hypernyms, Hyponyms, Meronyms, Holonyms of each words were extracted as features using WordNet

Task4: This phase included implementation of a statistical model to semantically match the user's input question/statement to FAQs using combination of deeper NLP pipeline features. The main purpose of this pipeline is to do the following:

Experiments:

- Implemented Multinomial Naïve Bayes from the extracted features to retrieve the probabilities for the FAQs and getting the top matching FAQs based on the probabilities.
- When we analysed the probabilities, they were so close and we were not getting clear good results.

Keyword search index creation and Querying:

- We Merged all the features into one bag per FAQ and indexed into SOLR using pysolr
- Run above deeper NLP on user's input to extract search query features
- Run Search against SOLR indexed fields created from corpus

Programming tools:

Python(V 3.6): All of the tasks are implemented using python.

Pysolr : A Lightweight Python wrapper for Apache Solr that provides an interface that queries the server and returns result based on query.

SOLR(V 7.2.1): An open source enterprise search platform, that includes features like full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features and rich document (e.g., Word, PDF) handling.

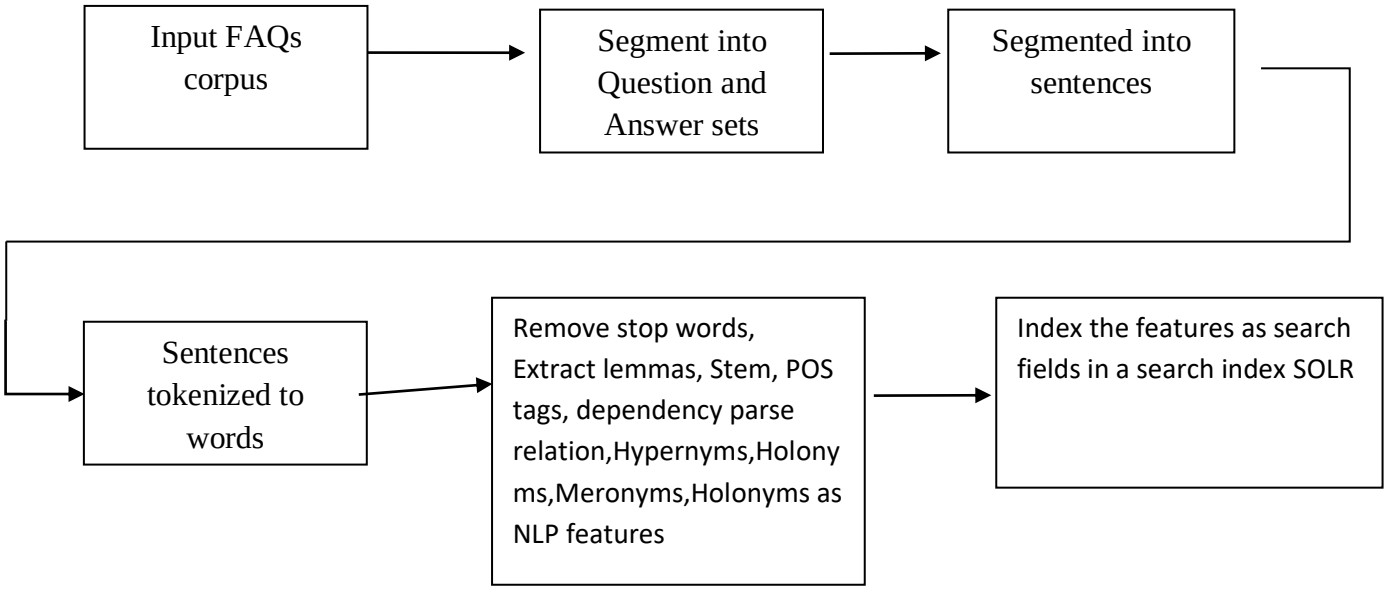
NLTK (V 3.2.5): Processing libraries used

nltk.corpus, nltk.tokenize, nltk.stem.wordnet, nltk.stem, nltk.wsd, nltk.corpus

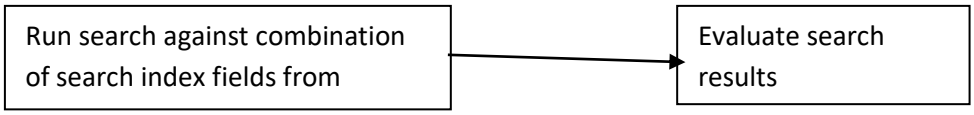
Stanford NLP(Verson 3.8): stanford-corenlp, stanford-parser

Architectural Diagram:

Search Index Creation:



Query Parsing and Search:



Results and Error Analysis:

All the results are evaluated by calculating **MRR**. The mean reciprocal rank 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.

Task1: Created corpus with 1003 articles and 100,000 word

Task2:

Following are the list of sentences that worked as the words of the sentences and article sentences matched

- What new can developers do now(Rank=1)
- How my system can be accessed (Rank=3)
- Scaling capability both up and down how much quick (Rank=1)
- Operating system environment supported (Rank=1)
- How my usage get bill by amazon(Rank =6)

- Monthly bill per second vs per hour shown(Rank=1)
- How will I be charged and billed for my use of Amazon EC2?(Rank =1)
- How is EC2 compute unit introduced(Rank =1)
- What all regions amazon EC2 is there(Rank=5)

Sentences that didn't work:

- Is pricing taxed

Above sentences has worked after modifying them as below

- Are prices taxed(Rank=1)

MRR = ((1/5) + (1/3) + (1/1) + (1/6) + (1/1) + (1/1) + (1/1) + (1/1) + (1/1) + (1/1)) /10 =0.66

Task4:

Following are the list of sentences that worked as the words of the sentences and article sentences matched

- How will I be charged and billed for my use of Amazon EC2?(Rank=1)
- Are prices taxed(Rank=1)
- How is EC2 compute unit introduced(Rank =1)
- What all regions amazon EC2 is there(Rank=2)
- How can I get informed about aws security(Rank=2)
- How do I know if ec2 is running in more than 1 regions(Rank=1)
- How to restrict other users to access my system(Rank=1)
- How my usage get bill by amazon(Rank =1)
- How my system can be accessed (Rank=3)
- Operating systems amazon supporting(Rank=1)

MRR = ((1) + (1/2) + (1) + (1/1) + (1/2) + (1/1) + (1/1) + (1/1) + (1/1) +(1/3)) /10=0.83

Challenges:

Following are the issues that were faced:

- Changing the indexing strategy in solr.
- Handling different features for task 4 and how to efficiently use it.
- Finding appropriate questions so that we could show the difference between task2 and task4.

Pending Issues:

Improving Task 4 to give even better search results.

Potential Improvements:

- Handling task features efficiently