**PROJECT REPORT**

**Frequently Asked Questions (FAQs) semantic matching application**

**Submitted by:**

**Group Name: We Match**

**Apurva Mithal (axm174531)**

**Sanjana Vijaykumar (sxv163930)**

Table of Contents

[Objective 3](#_Toc500326440)

[System Requirements.. 3](#_Toc500326441)

Programming tools used………………………………………………………………………………………………………………………….3

Language……………………………………………………………………………………………………………………………………………..3

Software………………………………………………………………………………………………………………………………………………3

Packages………………………………………………………………………………………………………………………………………………3

Requirements………………………………………………………………………………………………………………………………………….4

Architectural Diagram……………………………………………………………………………………………………………………………..4

Project Description………………………………………………………………………………………………………………………………….5

Task 1………………………………………………………………………………………………………………………………………………….5

Task 2………………………………………………………………………………………………………………………………………………….5

Task 3………………………………………………………………………………………………………………………………………………….5

Task 4………………………………………………………………………………………………………………………………………………….5

Project Implementation…………………………………………………………………………………………………………………………..6

Task 1…………………………………………………………………………………………………………… ………………………………..6

Task 2.……………………………………………………………………………………………………………………………………………..6

Task 3…………….………………………………………………………………………………………………………………………………..6

Task 4………………………….…………………………………………………………………………………………………………………..7

Issues encountered…………..………………………………………………………………………………………..……………………………8

Potential Improvements………………………………………………………………………………………………………………………….8

Conclusion……………………………………………………………………………………………………………………………………………….8

**OBJECTIVE**

The objective of the project is to implement a Frequently Asked Questions (FAQs) semantic matching application that will produce improved results using NLP features and techniques.

For this project we implemented 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

**SYSTEM REQUIREMENTS**

* **System Details**
  + *Operating System*: Windows 10

**PROGRAMMING TOOLS USED**

* **Language**
  + *Python*: 3.6
* **Software**
  + *Spyder*
  + *Solr:* 
    - An open source enterprise search platform, that includes features like semantic match, semantic search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features and rich document (e.g., Word, PDF) handling.
  + *Pysolr:* 
    - A Lightweight Python wrapper for Apache Solr that provides an interface that queries the server and returns result based on query.
* **Packages**
  + NLTK
    - from nltk.corpus import wordnet as wn
    - from nltk.wsd import lesk
    - from nltk.corpus import stopwords
    - from nltk.tokenize import word\_tokenize
    - from nltk import pos\_tag
    - from nltk.stem.porter import PorterStemmer
    - from nltk.stem.wordnet import WordNetLemmatizer
    - from nltk.parse.stanford import StanfordDependencyParser

**REQUIREMENTS**

* Install Solr
  + Download solr-7.3.0.zip from the following link-

<http://apache.osuosl.org/lucene/solr/7.3.0/>

* + Extract the Zip folder in your machine. Now go to extracted solr folder. Get inside the example folder and execute the command

java -jar start.jar

* + As soon as you run the above command solr will start with default port 8983. That can be accessible on-

<http://localhost:8983/solr/#/>

* + In command prompt go to the bin directory of the solr folder.
  + Create solr core instance by:

solr.cmd create -c collection\_1

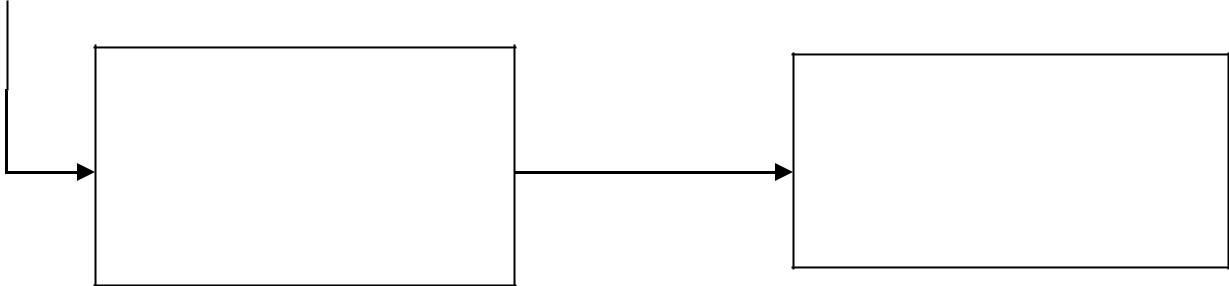
* + To run solr from bin director, give command

solr start

* Install NLTK
  + pip install nltk
* Install PySolr
  + pip install pysolr

**ARCHITECTURAL DIAGRAM**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Training Set |  | Segmented into |  | Sentences |  |
|  |  |  | sentences |  | tokenized to |  |
|  |  |  |  |  |
|  |  |  |  |  | Words, remove stop words |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |



Extract lemmas, Stem, POS

tags, dependency parse

relation, Hypernyms, lemmas

Hyponyms, Meronyms,

Holonyms as NLP features

Add the extracted semantic

features to solr.

Semantically match the user

input with each training FAQ/A

Return top 10 results

**PROJECT DESCRIPTION:**

**Task 1:** Create a corpus of 50 FAQs and Answers.

**Task 2:** Implement a shallow NLP pipeline and bag-of-words matching algorithm

* Bag-of-words creation
  + Tokenize the FAQs and Answers into bag-of-words.
    - Create a bag-of-words for each FAQ
  + Tokenize the user’s input natural language question/statement into a bag-of-words
* Bag-of-words matching
  + Return the FAQ and Answer whose bag-of-words best statistically matches the bag-of-words from the user’s input natural language question/statement
* Evaluate the results of at least 10 user questions/statements for the top-10 returned FAQ matches.

**Task 3:**  Implement a deeper NLP pipeline to extract semantically rich features from the FAQs and Answers

* Tokenize the FAQs and Answers into sentences and words
* Remove stop-words.
* Lemmatize the words to extract lemmas as features
* Stem the words to extract stemmed words as features
* Part-of-speech (POS) tag the words to extract POS tag features
* Perform dependency parsing or full-syntactic parsing to parse-tree based patterns as features
* Using WordNet, extract hypernyms, hyponyms, meronyms and holonyms as features.

**Task 4:**  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

* Run the above described deeper NLP on an user’s input natural language question/statement and extract semantic features
* 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
* Evaluate the results of at least 10 user questions/statements for the top-10 returned FAQ matches.

**PROJECT IMPLEMENTATION**

**Task 1:**

* Our corpus consists of 50 FAQs and Answers on Skullcandy Product. The corpus has been derived from the following link: <https://www.skullcandy.com/support/faq/>
* We collected 50 frequently asked question and answer pairs about SkullCandy product.
* Created a /training/ folder.
* Inside the training folder we created 50 text files. Each file containing 1 FAQ/Answer.

**Task 2: (Shallow NLP Pipeline)**

* We implemented bag-of-words technique to implement shallow NLP faq matching feature.
* Each FAQ/ Answer pair in the training set was tokenized into words using whitespace as a delimiter.
* Then for each of the 50 bags of tokenized words, word frequency count was calculated.
* Similarly, for the user input FAQ, bag of tokenized words was created and word frequency count for each word was calculated.
* Then word overlap technique was used to return the top 10 FAQ / Answer pairs whose bag-of-words best statistically matched with the bag-of-words from the user’s input natural language question/statement.

**Result and Error Analysis**

* We used Mean Reciprocal Rank (MRR) to grade our returned results. We manually looked at the sentences returned and based on the domain knowledge assigned rank to each of the FQA/Answer in result set. Then using MRR formula, we evaluated the results. The calculated MRR was 0.69.

**Task 3:**

* We integrated our code with wordnet using nltk package.
* For each of the 50 FAQ/ Answers in our training set
  + Tokenized the FAQ/Answer pair using nltk.tokenize
  + Tagged each word in the FAQ/Answer tokenized bag of words using nltk (pos\_tag)
  + Implemented dependency parser on the FAQ/Answer tokenized bag of words using nltk.parse.stanford package.
  + Lemmatized the part of speech tagged words using WordNetLemmatizer.
  + Removed the stop words from the FAQ/Answer tokenized bag of words using nltk.corpus (stopwords). This resulted in clean token set.
  + Stemmed each of the cleaned tokens using nltk.stem.porter (PorterStemmer).
  + Extracted the semantic features:
    - For each tagged token, given the word and part of speech tag, extracted the best sense of the word in the context.
    - After finding the best sense,
      * Extracted synonyms using sense.lemmas()
      * Extracted hypernyms using sense.hypernyms()
      * Extracted hyponyms using sense.hyponyms()
      * Extracted hyponyms using sense.meronyms()
      * Extracted hyponyms using sense.member\_holnyms()

**Task 4: (Deeper NLP Pipeline)**

* Extracted the above described deeper NLP (Task 3) on the user’s input natural language question/statement and displayed all the semantic features.
* Used solr to semantically match each of the FAQ/Answer in the training set with the user’s input question/ statement.
* For each of the training FAQ/Answer a JSON object was created containing the extracted semantic features.
* Each of the JSON object for the training example was added to JSON list.
* This JSON list containing semantic features for each training data point was added / indexed in solr using solr.add() function.
* Top 10 semantically matching FAQ/ Answers from the training set corresponding to the user input question/ statement was implemented using solr.search() functionality.

**Improved Deeper NLP Pipeline (Heursitic based)**

* A slight improvement to above technique was implemented.
* We assigned weights to each of the semantic features like giving higher weights to synonyms, hypernyms.
* Though the weights were manually assigned (not calculated) based on domain knowledge and lexical knowledge.

**Result And Error Analysis**

* We used Mean Reciprocal Rank (MRR) to grade our returned results. We manually looked at the sentences returned and based on the domain knowledge assigned rank to each of the FQA/Answer in result set. Then using MRR formula, we evaluated the results. The calculated MRR was 0.85.

**ISSUES ENCOUNTERED**

* While implementing dependency parser, we are getting error:
  + Warning! Out of Memory! There was not enough memory to run all parsers.
  + The code gets terminated.
* It works for 8 files in a batch and then throws memory out of bound error. So, we extracted head words for 8 training files to show that we were able to extract head words from wordnet.

**POTENTIAL IMPROVEMENTS**

* Currently we are manually selecting weights for semantic features.
* Instead we could implement an algorithm for selecting weights.
* Figure out a way to solve memory issues during dependency parsing.

**CONCLUSION**

This project provided us an excellent opportunity to use knowledge gained in class and apply it to a real-world NLP application.

* We learned how to integrate wordnet with our python code.
* We also learned how to use nltk package for various NLP applications.
* We could relate the importance of removing stop words while processing text data.
* Also got idea on word sense dis-ambiguity.