Semantic Search Application

Natural Language Processing

12/07/2017

Team: The Decipherables

Akhilesh Joshi (adj160230)

Karan Bharat Motani (kbm160230)

Raveena Singh (rxs159830)

Problem Description:

We need to design and develop a Natural Language Processing application for semantically searching text by implementing a keyword-based strategy and an improved strategy using NLP features and techniques. The application should produce improved accuracy with good results at each step using the NLP features and techniques.

Proposed Solution:

For this Semantic Search application, we input a search query, that looks up and finds the relevant articles for our query from the corpus. The results should show improvement at each task using the NLP features and techniques. For this application, we are using a corpus that contains news articles. We then segment and index the text from the corpus and search on this indexes giving some results. Then, we implement features such as lemmatization, stemming, POS tagging, hypernym, hyponyms, etc. and index on the respective features giving better results than before. Later on, we add the weights to the attributes depending on their significance and query again over these indexes. We observe that we get better results at each iteration, giving almost exact answers to our query.

FULL IMPLEMENTATION DETAILS:

Task 1:

Corpus:

The Reuters Corpus contains 10,788 news documents totaling 1.3 million words. The documents have been classified into 90 topics. It contains new related topics. A news story often covers multiple topics. We can ask for the topics covered by one or more documents, or for the documents included in one or more categories. For convenience, the corpus methods accept a single file id or a list of file ids.

Accessing the Data-Set:

To access Reuter’s documents: reuters.fileids (document name =‘training/01’)
To access Reuters categories: reuters.categories ('training/456')

Data Set Statistics:

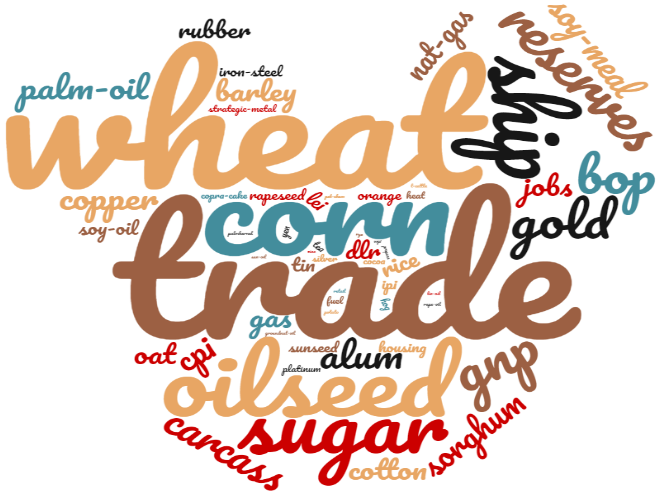
The Reuters document has the following

· Documents: 3000

· Sentences: 15172

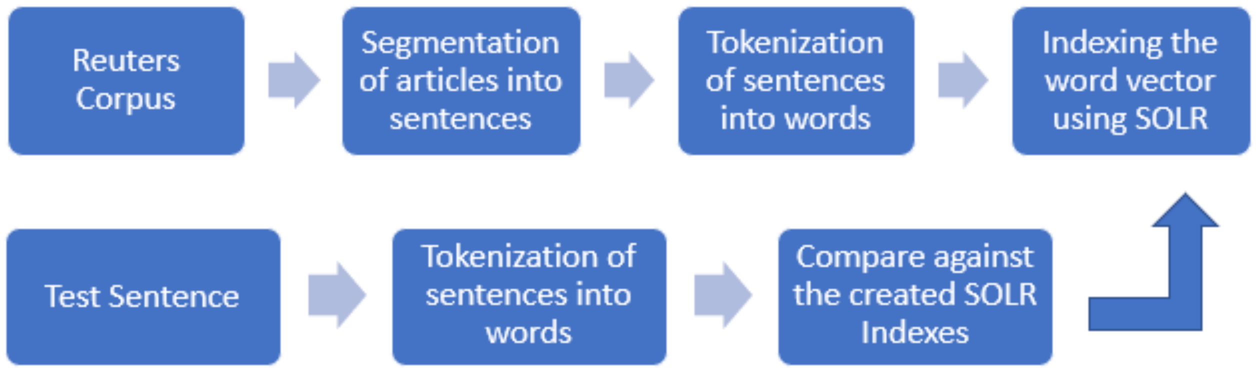
· Words: 373803

Below is the **Word Cloud** we found from our dataset. It clearly shows the most frequently used ***topics*** in the corpus.



Task 2:

Implementation of Shallow NLP pipeline:


Reuters Corpus
Segmentation of articles into sentences
Tokenization of sentences into words
Indexing the word vector using SOLR
Test Sentence
Tokenization of sentences into words
Compare against the created SOLR Indexes

There are 2 tasks in Shallow NLP Pipeline:

- Indexing
- Searching

Indexing:

The following tasks are performed in indexing:

- Segment the entire article into sentences
- Tokenize the sentence
- Index the sentence and tokens and add the data to Solr instance.

The indexing has been done using Solr. It is an open source enterprise search platform, written in Java, from the Apache Lucene project. Solr has its own tokenization function and carries the tokenization internally, thus, the input to Solr is just a text sentence.
We are using PySolr package that is used to run Solr with Python. Solr.add() is used to add the data to the Solr instance.

Schema:

- url: http://localhost:8983/solr/nlp-core1
- Port Number: 8983
- Core: nlp-core1
- Index Fields:
    - id: [document_Number]-[sentence_Number]
    - text: ‘Contains the entire sentence’

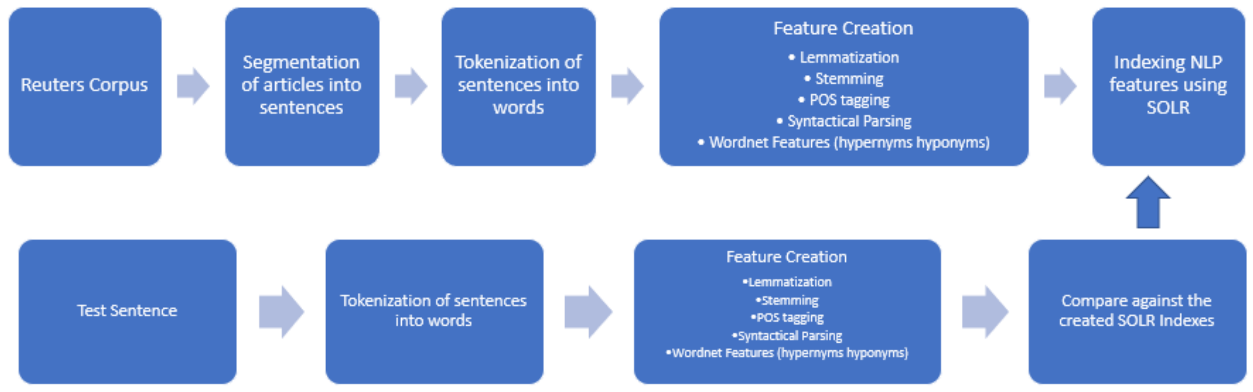
Searching:

We use Solr.search() to query the indexes in solr. The input to the Solr is a string of data which the Solr then uses its inbuilt function to search from the indexed data.
Solr.search(‘input string’, sort=’score desc’, fl='*,score')
Where input string is of the format
text: (‘query’)
fl: The fl parameter gives the information included in a query response to a specified list of fields, where * signifies that all the fields should be displayed on the console output.

Task 2 performs a basic naïve search based on the input data that does not include any other attributes for contextual search.

Task 3:

Implementation of Deeper NLP pipeline:


Reuters Corpus
Segmentation of articles into sentences
Tokenization of sentences into words
Feature Creation
• Lemmatization
• Stemming
• POS tagging
• Syntactical Parsing
• Wordnet Features (hypernyms hyponyms)
Indexing NLP features using SOLR
Test Sentence
Tokenization of sentences into words
Feature Creation
•Lemmatization
•Stemming
•POS tagging
•Syntactical Parsing
•Wordnet Features (hypernyms hyponyms)
Compare against the created SOLR Indexes

Deeper NLP Pipeline uses additional sematic attributes that give us some context when searching for data. The features includes:

- Lemmatization
  - WordNetLemmatizer() is used for this purpose, which correctly identifies the intended POS and meaning of a word in the sentence.
- Stemming
  - nltk.stem package is used for Stemming that consists of various kinds of Stemmer.
  - Porter's Stemmer is used for this application which gives the root of the word in the sentence.
- POS Tagging
  - We are using nltk to get the POS tags for the words in the corpus.
  - POS tagging is a key feature that helps us understand the context and the definition for the word in the sentence.
- Syntactically parsing the sentence and extracting phrases
  - Using Spacy, we are extracting the Noun phrases from the sentence that give the highlight of the sentence.
- WordNet is used for other features that includes:
  - Hypernyms
  - Hyponyms
  - Holonyms
  - Meronyms

Schema:

- url: http://localhost:8983/solr/nlp-core2
- Port Number: 8983
- Core: nlp-core2
- Index Fields:
    - ID: [document_Number]-[sentence_Number]
    - Text: Consists of the entire sentence
    - Tokens: Consists of the Tokens of entire sentence,
    - POS Tags: Consists of the POS tags for the words in the sentence,
    - Stem: Consists of the Stems of each word,
    - Lemma: Consists of the Lemmas of each word,
    - Noun Phrases: Consists of the Noun Phrases of the sentence,
    - Hypernym: Consists of the Hypernyms of each word,
    - Hyponym: Consists of the Hyponyms of each word,
    - Meronym: Consists of the Meronyms of each word,
    - Holonym: Consists of the Holonyms of each word,

Searching:

- We use Solr.search() to query the indexes in solr. The input to the Solr is a string of data which the Solr then uses its inbuilt function to search from the indexed data.
- We are using solrq Python Solr query package to simplify the searching of data in Solr. The searching is represented as follows:
    - Solr.search('input string', sort='score desc', fl='*,score')
      where input string is of the format
        - Q(tokens=tokens) & Q(stem=stems) …
    - fl: The fl parameter gives the information included in a query response to a specified list of fields, where in * signifies that all the fields should be displayed on the console output.

Task 3 implements a deeper NLP Pipeline that uses the NLP features which results in significantly better results than in Task 2. This concludes that Task 3 output results that are more relevant to the query using the above features.

Task 4:

Task 4 implements the same features as that in Task 3. Here, we give weights to different features based on their priority so as to improve the accuracy of our search results.
The weights assigned to the features are:

- Text = 0.5
  - Solr searches for the text but it searches for exact match and does not work with a little modification of words such as plurals.
- Stems = 0.5
  -
- Lemmas = 4
  - Lemmas correctly identifies the intended POS and meaning of a word in the sentence and returns the words to base or dictionary form
- posTags = 0.02
  - As the POS tags can be different for the same word, it does not contribute to improve the accuracy of the result.
- Phrases = 5
  - Phrases highlights the important context of the sentence, which can match the context of the search query giving better results.
- Hypernym = 2
  - Hypernym gives the superclass of the word which can be used to for searching generically giving related data to the search query.
- We are not considering Hyponyms as it gives the sub class for the words that can reduce the accuracy of the search.
- Similarly, Holonyms and Meronyms does not provide any better results and do not contribute in improvement of the accuracy.

Programming Tools:

- Solr-7.1.0 with Python 3.5
- Packages such as:
  - NLTK
  - WordNet
  - PySolr
  - Spacy
  - solrq
  - re

Results and Error Analysis:

Observation:
Task 2 gives very less precision that revolves around 3/10.
Task 3 gives significantly better results than Task 2, around 5/10
Task 4 gives similar or even more relevant results than Task 2 giving a precision around 7/10

Let’s look at an example and the precision for all the tasks.
The top 5 results for the following query are:

Input sentence : coffee exports dropped over the years

TASK 2: (Precision 2/5)

- The target is to raise exports to 20 pct of Gross Domestic Product over the next four years compared to 15 pct now," he said.
- Exports to Western Europe eased 3.5 pct and Far East exports, due to an economic revival in Japan, dropped 5.2 pct.
- THAI COFFEE EXPORTS RISE IN 1986 Thai coffee exports rose to 22,068 tonnes in 1986 from 20,430 a year earlier, the Customs Department said.
- He said exporter registrations dropped from an average weekly 500 tonnes in March to 45 tonnes last week, with exports in coffee year 1986/87, ending September, forecast to total about 8,000 tonnes against 48,000 in 1985/86.
- PHILIPPINE COFFEE EXPORTS SEEN FALLING SHARPLY Philippine coffee exports are expected to fall sharply due to a combination of the International Coffee Organisation's (ICO) decision not to revive export quotas and higher local prices, ICO Certifying Agency official Dante de Guzman told Reuters."]'.

TASK 3: (Precision 3/5)

- He said exporter registrations dropped from an average weekly 500 tonnes in March to 45 tonnes last week, with exports in coffee year 1986/87, ending September, forecast to total about 8,000 tonnes against 48,000 in 1985/86.
- I definitely see 90 cents and would not rule out a brief drop to 85 cents," said Debra Tropp, a coffee analyst with Prudential Bache.
- There, the producers expressed their political will to negotiate basic quotas, particularly in the face of the damaging drop in coffee prices after the council failed to agree quotas, Montes said.
- Coffee production was expected to drop slightly to about one mln bags of 60 kg each in the 1986/87 crop year ending June from 1.1 mln bags last year, he said.
- COFFEE PRICE DROP NOT AFFECTING COLOMBIA'S DEBT the sharp fall in international coffee prices will not affect colombia's external credit situation, finance minister cesar gaviria told reuters.

TASK 4: (Precision 4/5)

- He said exporter registrations dropped from an average weekly 500 tonnes in March to 45 tonnes last week, with exports in coffee year 1986/87, ending September, forecast to total about 8,000 tonnes against 48,000 in 1985/86.
- There, the producers expressed their political will to negotiate basic quotas, particularly in the face of the damaging drop in coffee prices after the council failed to agree quotas, Montes said.
- Coffee production was expected to drop slightly to about one mln bags of 60 kg each in the 1986/87 crop year ending June from 1.1 mln bags last year, he said.
- I definitely see 90 cents and would not rule out a brief drop to 85 cents," said Debra Tropp, a coffee analyst with Prudential Bache.
- COFFEE PRICE DROP NOT AFFECTING COLOMBIA'S DEBT the sharp fall in international coffee prices will not affect colombia's external credit situation, finance minister cesar gaviria told reuters.

Problems Faced:

Lack of proper documentation for PySolr and SOLR integration made us drive through various integration techniques and choosing PySolr parameters (such as timeout, debug query, scoring) was one of the essential tasks during project foundations.

Input Output operations with SOLR are heavy and time consuming so we came up with strategy to add documents in chunks rather then uploading the entire dictionary at same time.

Finding Noun Phrases took lot of processing time, hence we have to try out among the available techniques such as Core NLP and spacy and we decided to proceed ahead with spacy as it provided the best results in very nice format and in desired time.

Potential Improvements:

We can include the construction of a more varied sentence pair data set with human ratings and an improvement to the algorithm to disambiguate word sense using the surrounding words to give a little contextual information.

We can implement td-idf to calculate the cosine similarity between sentences that gives us better results than using Solr search.

Represent the information present in documents in RDF (Resource Description Framework) format where we save the relations among the entities in Subject Verb Object format.

Entity Extraction : to query on the information on the document knowledge we can save entities as features.

Conclusion:

Through this project, we have tried to device a method to find the most similar sentence from the corpus to the given sentence based on some NLP features. To evaluate and find the best NLP pipeline for sentence similarity, firstly we had a data set ready from Reuters corpus and a test sentence. Our aim was to find the most similar sentence from the corpus. In order to get the best accuracy, we gave weights to different features and checked which feature or combination of features gave the best accuracy in retrieving the most similar document.