NLP Final Project Report

CS:6320.501

Team: TJU

Hxw161630 Hui Wang

Hxh161030 Hongyao Huang

Contents

Task1

Problem description

Create a corpus of News articles. Your corpus should contain at least: 1,000 articles and 100,000 words

Proposed solution

Download “Australian Broadcasting Commission 2006” corpus from nltk, which contains over 20000 sentences and 700000 words.

Full implementation details

1. programming tools
   Language: Python 2.7.13
   Tools: nltk.corpus
   Using command*: from nltk.corpus import abc*

Task2

Problem description

Implement a shallow NLP pipeline to perform the following:

- Keyword search index creation
  - Segment the News articles into sentences
  - Tokenize the sentences into words
  - Use SOLR to build search index of tokenized sentences.
- Natural language query parsing and search
  - Segment user’s natural language query into sentences
  - Tokenize the sentences into words
  - Run a search/match against the search index created from the

corpus

- Evaluate the results of at least 10 search queries for the top-10 returned sentences

Full implementation details

- Programming tools
  - Programming language: python 2.7.13
  - Tools
    - SOLR
    - Python packages
      - NLTK
      - JSON
      - urllib2
- Keyword search index creation
  - Use NLTK tools to segment the articles into sentence, and tokenize the sentences into words.

```
# import abc corpora from nltk
from nltk.corpus import abc
import string

# Segment the corpus into sentences and words
sentences_list = abc.sents()
words = abc.words()
```

  - Write the processed data into a xml file in the following format:

    <doc>

    <field name = “id”>sentence id<\field>

    <field name = “name”> sentence content< \field >

    <field name = “word”> $word_1$< \field >

    ……

    <field name = “word”> $word_n$< \field >

    </doc>

  - Create a new core in SOLR, then post the xml file to our core using

command ‘bin/post’. Here, we have already index the keywords into search index.

- Natural language query parsing and search
  - Use nltk tools to segment query into sentence, then tokenize it into words.
  - Getting the search result in JSON format with a URL link. For example, getting search result of query ‘PM denies knowledge’ we generate a URL link as following:
    - http://localhost:8983/solr/task3/select?q=PM+denies+knowledge&wt=json&df=word&fl=id,name
    - Code for generating query URL link:

```
# Query input to SOLR
q_solr = 'http://localhost:8983/solr/task3/select?q='
q_solr = q_solr + q_words[0]
for i in xrange(1,len(q_words)):
    q_solr = q_solr + '+' + q_words[i]
q_solr = q_solr + '&wt=json&df=word&fl=id,name'
print(q_solr)

connection = urlopen(q_solr)
response = json.load(connection)

print response['response']['numFound'], "documents found."
```

- Evaluate the results of at least 10 search queries for the top-10 returned articles.
  - Example search query: ‘PM denies knowledge of AWB kickbacks’
    - Here is the top ten returned sentence:

```
13928 documents found.
 Id, Name = 0    [u'PM denies knowledge of AWB kickbacks The Prime Minister has denied he knew AWB was paying kick
 Id, Name = 7859    [u'But AWB International chairman Ian Donges denies damages could run to tens of millions of
 Id, Name = 7417    [u"AWB finds more kickbacks documents Thousands of documents previously withheld by AWB ' s i
 Id, Name = 4647    [u"Meanwhile , the Western Australian Farmers Federation denies it is withdrawing support for
 Id, Name = 4986    [u'If the action is successful , it could see AWB forced to make a payout triple the value of
 Id, Name = 7418    [u"The hearing into AWB ' s $ 290 million in kickbacks to the former Iraqi Government has reo
 Id, Name = 1167    [u'Attacking the plan , AWB chairman Brendan Stewart told Grains Week delegates , the industr
 Id, Name = 6521    [u'AWB estimates the royal commission into alleged kickbacks to Iraq will cost the company be
 Id, Name = 3287    [u"Senator Harkin ' s spokesman Dave Townsend says what happened to the $ 200 million in alle
 Id, Name = 7795    [u'AWB plays down Cole inquiry to growers Wheat exporter AWB has told Victorian grain growers
```

- Analysis:

  As we can see from the result, the first sentence is the best matching sentence from SOLR. Because we are doing exact matching in this task, the first sentence is the exact match of our query. For the rest sentences in the result, we can see those sentences are somewhat matching with the query.

- Problems encountered and solution
  - Non-ASCII Character

    In our corpus, initially the text are encoded in Unicode, but we have some characters are not encoded in ASCII. To solve this problem, we use following code:

```
str_temp = " ".join(sentences_list[i])
str_temp = str_temp.replace('&','')
str_temp = filter(lambda x: x in printable, str_temp)
sentences_str.append(str_temp)
```

  - Reserved character and keywords in XML

    When we generating the XML code, we encountered some characters are reserved keywords in XML like the character '&'. We solved it by replacing it with empty characters.

  - No default search field in SOLR

    When entering search query in SOLR, we first got nothing. After reading through SOLR's documentation, we found that we need to specify search field by adding '&df=our search field' in the URL we are used to search.

Task3

Problem description

- Semantic search index creation
  - Segment the News articles into sentences
  - Tokenize the sentences into 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
- Syntactically parse the sentence and extract phrases, head words, and dependency parse relations as features
- Using WordNet, extract hypernymns, hyponyms, meronyms, and holonyms as features
- Index the various NLP features as separate search fields in a search index with SOLR

- Natural language query parsing and search
  - Extract features listed above of user's natural language search query.
  - Run a search/match against the separate or combination of search index fields created from the corpus
- Evaluate the results of at least 10 search queries for the top-10 returned articles

Full implementation details

- Programming tools
  - Programming language: python 2.7.13
  - Tools
    - SOLR
    - Python packages
      - NLTK
      - JSON
      - urllib2
- Semantic search index creation
  - Segmentation and tokenization processes are similar to the previous.
  - Lemmatize:

    We use the WordNetLemmatizer provided by NLTK to do lemmatization. First, we run a POS tag to sentences, and then for each word in each sentences, we run the lemmatizer. If a word has

a POS tag of verb, for example VBZ, we set the parameter 'pos=v' when doing lemmatize.

- Stem:

  We use the PorterStemmer in NLTK to do stemming. After tokenization, we run the stemmer on each word of each sentences.

- Part-of-speech tag:

  We use the NLTK built-in POS tagger*, nltk.pos_tag()*, to do POS tagging on each sentences.

- Using WordNet, extract hypernyms, hyponyms, meronyms, and holonyms as features

  For each tokenized sentences, we first extract the '*synset*' of each word. We assume the best sense of a word is the first sense in its synset. Based on the synset, we extract hypernyms, hyponyms, holonyms and meronyms of each word separately.

- Index the various NLP features as separate search fields in a search index with SOLR

  Similarly, we use bin/post command to upload the xml file

- Natural language query parsing and search
  - Extract features listed above of user's natural language search query.
  - Run a search/match against the separate of search index fields created from the corpus
- Evaluate the result

  We run separately searching for each feature we extracted. For example, for POS tag, we first run the POS tag on the query sentence, and use POS tags as search field in SOLR.

```
Input query: PM denies knowledge
Input search field: POS
['PM', 'denies', 'knowledge']
[('PM', 'NNP'), ('denies', 'VBZ'), ('knowledge', 'NN')]
['PM', u'deny', 'knowledge']
['PM', u'deni', u'knowledg']
[u'psychological_feature', u'psychological_feature', u'psychological_feature']
[u'ability', u'power', u'attitude', u'mental_attitude', u'cognitive_factor', u'conte
[]
[]
http://localhost:8983/solr/task3/select?q=NNP+VBZ+NN&wt=json&df=POS&fl=id,name
```

Task4

Problem description

- Improve the shallow NLP pipeline in task 2 by using a combination of NLP features extracted in task 3.

Idea

First, we think the process of “lemmatize and stem” is necessary as our normalization.
Then we also, need to query the field hypernym or hyponym or holonym or meronym. Because they have related sense. For example, if we want query about a ‘dog’, then the sentence contains ‘puppy’ should also be the goal sentence.

In conclusion, we combined lemmatization, stem and hypernym as combined features into SOLR.

Full implementation details

- As our previous tasks implementation, we generate a URL link containing our seach fields and query words after pre-processing of query. In this task, we combined all these into a URL link.
  - Input query: “PM denies knowledge”
  - URL: http://localhost:8983/solr/task3/select?q=stem:PM+deni+knowledg%20lemma:PM+deny+knowledge%20hypernym:psychological_feature+psychological_feature+psychological_feature&wt=json&fl=id,name