SET EXPANSION

FINAL PROJECT REPORT

INFORMATION RETRIEVAL AND EXTRACTION

CSE474, SPRING '16

PROFESSOR: VASUDEV VARMA

MENTOR: MRUGANI KURTADIKAR

ROMIL PUNETHA (201505568)

DEEP JAHAN GREWAL (201364124)

SANDEEP KASA (201301145)

TEAM 25

April 15, 2016

SET EXPANSION

FINAL PROJECT REPORT

Table of Contents

INTRODUCTION

General Idea:

- given a small list of instances of some (unknown) class of entities
- Consult a corpus in order identify similar entities to add to the class

Seed expansion refers to completing a given set with terms relevant to the set. Some terms known as seed terms are provided and we try to fill in the set with words falling in the same category as the seed terms. The major task is to find the relation between the seed terms, else the set will be filled with inappropriate words.

Problem Statement and Goals

The dataset used to train the model is Wikipedia dataset.

Let:

- S = set of seed entities of a class C
- E = candidate entities
- f(C,e) = x, where $0 \leq x \leq 1$ is the degree of C-membership of $e \in E$

Goal: Learn f()

- approximate f(C,e) by f(S,e), and then compute f(S,e) for all $e \in E$
- Consider f(S,e) as similarity function between all seed elements and a single e.
- compute f(S,e) by using a vector space model, i.e., by computing the similarity of feature vectors

Vector Space Model for Set Expansion

- Each element (i.e., each s ∈ S or e ∈ E) wj is represented by a vector of numerical features vec(wj).
- Given such a representation vec(), compare elements by standard distance functions, e.g. cosine.
- Choice of features defines the information that is captured and transferred to vec().
- For seed expansion: mainly type similarity

How to determine type similarity?

- Observation: humans easily group and list type similar entities
- "American Airlines' general rule is you can only bring one personal item such as a briefcase, purse or laptop bag on-board and one small piece of luggage."
- The early British painters like Tilly Kettle, John Zoffany, John Smart, George Chinnary, William Hodges and others painted in oil.
- Goal: Identify such enumerations in text, and gather information about class similarity.
- Assumption: if two elements consistently co-occur in lists, they are likely to be of a similar semantic class.

Identification of lists in texts

- Simple approximation: identify pairs of elements that belong to lists
- Assumption: lists are composed by sequences of pairs of coordinated words by
    - explicit coordination elements (and, or, …)
    - commas
- Look for text fragments likes:
    - "… nea, neb and nec …", „… nea, neb or nec …"
    - "I lived in Paris, Berlin and London.", „Experience with Java, C++ or Lisp is required."
- Conclude: when instances of such patterns are found in text, then pairs (nea, neb) and (neb, nec) co-occur in coordination (for simplicity, no assumption for pair (nea, nec)).

Vector Representation

- Main idea: represent candidates and seed elements as vectors encoding their co-occurrence frequency.
- Let NE::= {ne1, .., neN} be all named entities that co-occur in a text.
  - Define j^{th} component of vec(nei) = |(nei,nej)|
  - Similar: for seed set S, S(j) is defined as |(s,nej)|, s ∈ S
  - This means: a vector of a nei collects all nej, which co-occur with nei in an enumeration
- Two vector spaces
  - VS^x considers only pairs from explicit coordination's (more restricted, less noise, lower recall)
  - VS'considers pairs from explicit and comma coordination's (more noise, e.g., „… X, Y …")

Member Function F ()

ne_j \ ne_i	ne_1	ne_2	…	ne_N	s_1	…	s_m
ne_1	0	12		4	5		12
ne_2	3	0		11	3		6
ne_3	1	0		10	0		3
…							
ne_N	0	3		0	1		0

Consider the similarity of ne_j with **all** seed elements s_i as similarity between corresponding vectors.

use f^x or f^t depending on vector space

$$f(S,e) = \cos(vec(S),vec(e)) = \sum_{i=1...m} \cos(vec(s_i), vec(e))$$

$$\cos(vec(x),vec(y)) = (vec(x) * vec(y)) / (norm(x) * norm(y))$$

Code Files

The project contains the following code files :

1. search.py - calls searchAPIs
2. getpage.py - downloads the pages and parses it
3. word2vec/compare.py - generates ranking according to word2vec
4. patterRecognition.py - expands sets by pattern recognition
5. htmloutput.html - downloaded htmls
6. indexfile – index
7. main.py- calls all the other files, takes the input and produces the output

Process

- Configuring the searchAPIs of google, bing, duckduckgo, farooo, Wikipedia, twitter and webhose.
- Passing seed terms to these APIs and getting the top 10 search results in the form of URLs (twitter and webhose generate data directly.).
- Crawling the links obtained from above and downloading the web pages into a single file named 'htmloutput.html'.
- Parsed the web pages using an html parser written by us in python to extract data from the html files. This includes omitting the scripts in the webpages ad removal of stopwords.
- For pattern recognition, only the wrappers that contained more than 2 seed terms were considered for pattern extraction. Wrappers other than table, ol, li, and ul were also considered.
- The Wikipedia corpus that was available in xml format was preprocessed into text format before it could be used to train the word2vec model
- The word2vec model was then trained using the Wikipedia corpus using the gensim.model library in python.
- A model.bin file is generated after the above step that contains the words as a feature vector with a dimensionality of 300 each.
- Now we have the tokens from the webpages. We pass the seed terms and the tokens to the word2vec model and we get a score for each token.
- Sort the token based on the score.
- Search for the words obtained through pattern matching in the set of tokens (here we consider top 40 results for faster comparisons). Increase the score of the words that are common to the pattern matching results and the top 40 results from word2vec model.
- Recalculate the rank of each word and display the top 'n' results.

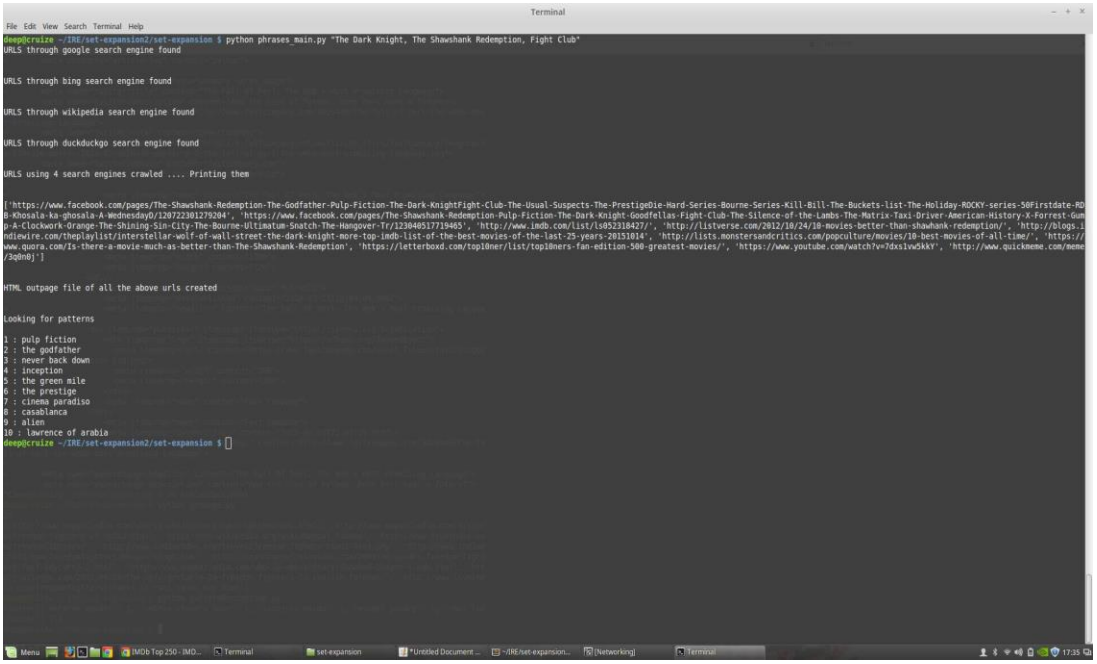
  Note: For phrase queries, the word2vec model could not be trained. So phrase query results are based entirely on the pattern recognition technique.

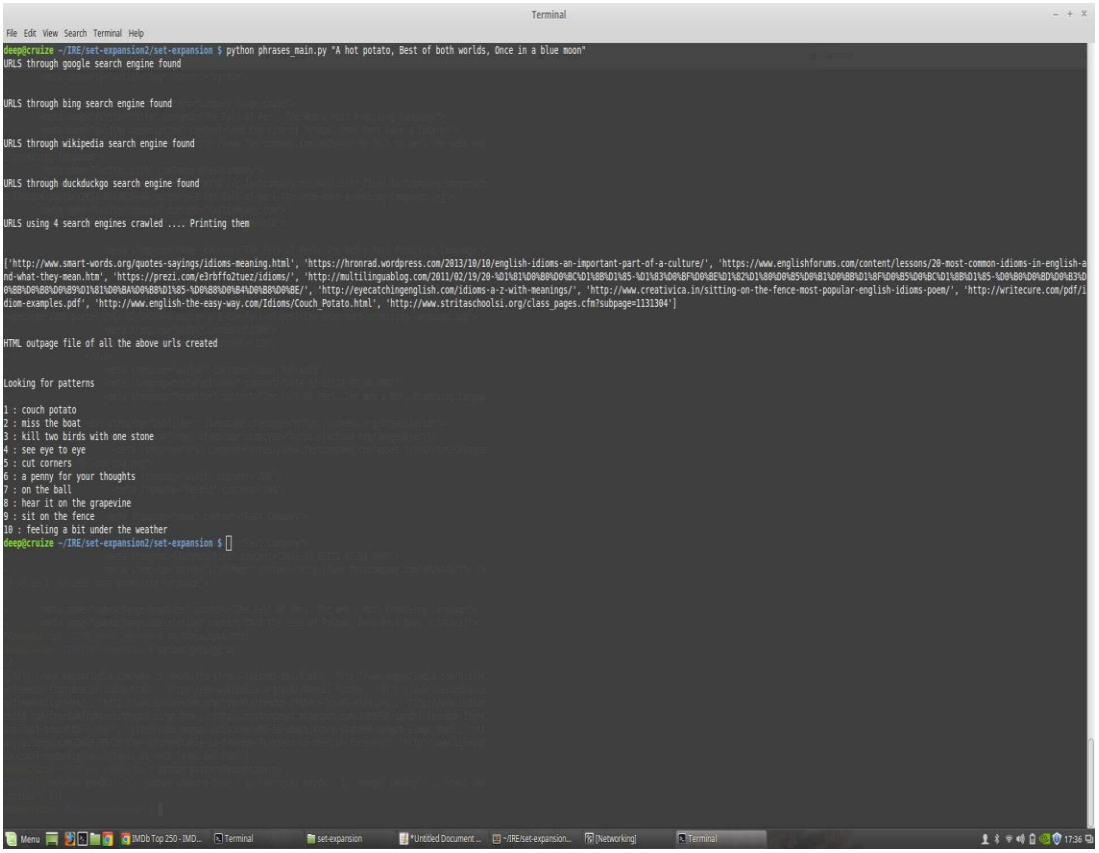
Input and Output(for seed terms)

```
sandy@SandyPC:~/SEM-6/IRE/Project/Mine$ python main.py 'python,java,perl,php'
URLS through google search engine found


URLS through bing search engine found


URLS through wikipedia search engine found


URLS through duckduckgo search engine found


URLS using 4 search engines crawled .... Printing them


HTML outpage file of all the above urls created


Parsing and getting all the tokens from the html output file

Writing all the tokens to a indexfile
Model loaded


Printing Results


1 : javascript
2 : scripting
3 : mongodb
4 : linux
5 : tcl
6 : lisp
7 : cpan
8 : numpy
9 : doctest
10 : gnu
sandy@SandyPC:~/SEM-6/IRE/Project/Mine$
```

```
sandy@SandyPC:~/SEM-6/IRE/Project/Mine$ python main.py 'cricket,football,volleyball'
URLS through google search engine found


URLS through bing search engine found


URLS through wikipedia search engine found


URLS through duckduckgo search engine found


URLS using 4 search engines crawled .... Printing them


HTML outpage file of all the above urls created


Parsing and getting all the tokens from the html output file

Writing all the tokens to a indexfile
Model loaded


Printing Results


1 : rugby
2 : hockey
3 : squash
4 : bowling
5 : tennis
6 : sport
7 : soccer
8 : boxing
9 : baseball
10 : darts
sandy@SandyPC:~/SEM-6/IRE/Project/Mine$
```

Input and Output (for phrase queries)





External Links

1. Dropbox Link to the project (Code, PPT, Report)
2. Presentation Link to Slide share
3. Github Link
4. Website Link of the project
5. Video Link