COL764 Assignment 1: Vector-space Retrieval Readme

Yash Malviya 2016CS50403

1 How to run?

The assignment has been coded in python 3 with file names arguments same as that mentioned in the assignment pdf.

1.1 Requirements

Pip install the following packages:

1. NLTK, with stopwords for english downloaded
2. BeautifulSoup
3. XML parser lxml

English stopwords were downloaded like so - *python -m nltk.downloader stopwords.* *'pip install -f requirements.txt'* automates installation of these dependencies.

1.2 Inverted Index

python invidx.py coll-path indexfile
indexfile.idx and indexfile.dict are generated.

1.3 Print Dictionary

python printdict.py indexfile.dict

1.4 Vector Search

python vecsearch.py –query queryfile –cutoff k –output resultfile –index indexfile –dict dictfile
Double hyphen (-) is used before the argument name

2 Implementation Details

2.1 Inverted Index

Document ID and text of the document is extracted from the TaggedTrainingAP files using xml parser of BeautifulSoup. Words are separated into tokens using regex a token can either be a word or tagged word. I iterate over the these token to count the tokens as well combined named entities, this is how index list for each doc is generated. After iterating over all the tokens in a doc index list is merged in to globally defined inverted index posting lists of the IR system. This positing list is stored as a dictionary of dictionaries. Outer dictionary is mapping from terms to posting list. Posting list is mapping from document ID to number of occurences of this term in the document.

Stopwords from nltk libraries are skipped while iterating over tokens. Terms in iteration over token are lowered (i.e. no capital characters left) and each named entity is saved as *N:named-entity*,

specific type such as *P:named-entity* (or *O:* or *L:*) along with treating each token of named entity as normal word too.

Python's pickle is used to get byte stream to store serialized data structures to files. Dictionary file stores dictionary containing mapping of term to tuple of document frequency and offset in index file. Index file stores maximum frequency of term in doc also called 1-norm. Index file stores the pickled global inverted index posting lists while calculating offset of each term in the file.

2.2 Print Dictionary

Simply unpickles/loads the dictionary from the file argument and prints it in specified format.

2.3 Vector Search

Regex the required query number and query from *Number* and *Topic* entry respectively. Only k-cutoff along with vector search was implemented. Prefix search and named entity retrieval were not implemented. Dictionary of term and terms and their 1-norm dictionary are loaded in one go. Words in query are split using space character. Posting list for each term is loaded when required by a query, its loaded only once next time term is RAM.

Doubly normalized TF and Smoothed IDF is used. Dot product of document and and query is calculated by retrieving elements from posting list. for k cutoff top k documents are retrieved using a variation heap sort.

Output file is stored in qrels format mentioned here i.e. *TOPIC ITERATION DOCUMENT RELEVANCY*