MACHINE LEARNING

Assignment 3: Singular Value Decomposition

Due date: Monday, 3 October 2016, 11:55 PM
Maximum number of files: 1
Type of work: Individual work
Objective - In this assignment you will implement a basic search engine using a technique called Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI): According to Wikipedia, **LSI** is a technique in natural language processing of analyzing relationships between a set of documents and the terms they contain by projecting terms and document in a common space such that terms and documents that are closely associated are placed near each other. LSI assumes that words that are close in meaning will occur in similar pieces of text. A term-document matrix containing word counts per document (rows represent unique words and columns represent each document) is constructed and then SVD is used to project it into a lower dimensional subspace (i.e. reduce its rank). Words and documents can now be compared by taking the cosine of the angle between the two low dimensional vectors. For more details please refer to this paper.

Problem Statement: You are given a bunch of documents (numbered 1 to 5000 with title of document in the first line) and your task is three fold:
1) Given a document return similar documents
2) Given a word/term return similar terms
3) Given a query return the most relevant set of documents corresponding to that query

The assignment has to be implemented in Python. You need to write a function lsi.py that takes three files as input(corresponding to tasks listed above) and outputs three files. lsi.py should be run as follows:
python lsi.py -z 200 -k 10 --dir Directory --doc_in <name of input document file> --doc_out <name of output document file to be generated by code> --term_in <name of input term file> --term_out <name of output term file to be generated by code> --query_in <name of input query file> --query_out <name of output query file to be generated by code>
where
-z: Dimensionality of lower dimensional space
-k: # of similar terms/documents to be returned
--dir: Directory containing input documents
--doc_in: Input file containing list of document titles (one per line) corresponding to whom k similar documents are to be returned.
--doc_out: Each line of this file should have titles of k documents (separated by ';<tab>' i.e semicolon followed by tab) that are similar to the document in corresponding line of doc_in
--term_in: Input file containing list of words (one per line) corresponding to whom k similar words/terms are to be returned.
--term_out: Each line of this file should have k words (separated by ';<tab>' i.e semicolon followed by tab) that are similar to the word in corresponding line of term_in
--query_in: Input file containing list of queries (one per line) corresponding to whom k relevant documents are to be returned.

--query_out: Each line of this file should have titles of k documents (separated by ';<tab>' i.e semicolon followed by tab) that are relevant to the query in corresponding line of query_in
Sample I/O files are attached.

Helpful Commands:
1) To separate words from a sentence in python:
import re
pattern = re.compile(r'\W+')
words = pattern.split(sentence)

2) For creating sparse matrix
import scipy.sparse as sp
<sparse matrix> = sp.csc_matrix((data, (row_ind, col_ind)), [shape=(M, N)])

3) To do SVD
from scipy.sparse.linalg import svds
u, s, vt = svds(tfidf, z, which = 'LM')

4) To find similarity between two vectors use the cosine similarity metric.

In case of any query please post on piazza or you can mail:
Himanshu Jain: himanshu.j689@gmail.com

VPL 3.1.3

My courses

ADMINISTRATION

Course administration

My profile settings

You are logged in as Sarisht Wadhwa (Log out)
1601-COL341