CS6200 Information Retrieval

Homework8: Clustering and Topic Models

Objective

In this assignment, you will cluster documents, detect topics, and represent documents in topic space.

Data : We are using the AP89 collection.

A) Topic Models per Query

For any given query, select a set of documents that is union of top 1000 BM25 docs (for that query) and the qrel docs (for that query). The set will be a little over 1000 docs since there will be considerable overlap.
Then use one of the packages listed below to perform LDA. Your output must contain

- the topics detected, with each topic represented as distribution over words (list the top 20-30 words)
- the documents represented as topic distribution. Each document should have listed the most relevant 10 topics or so

Repeat the process for each one of the 25 queries.

LDA option 1 : vowpal_wabbit(Bishwajeet) . Harder to use, but the best tool in practice for large data

1. On Ubuntu, vowpal_wabbit may be installed using the following set of commands
sudo apt-get install libboost-program-options-dev zlib1g-dev
sudo apt-get install libboost-python-dev
git clone git://github.com/JohnLangford/vowpal_wabbit.git
cd vowpal_wabbit
make
make install #may require root previleges

2. Input Data format for LDA
1. The expected input format is
| <num-1> <num-2> <num-3>
| <num-5> <num-2>
Each line corresponds to a document.

Use your own hashing function to convert from word to a number and make sure the
word representation is consistent across the different tuples of the input. In
the above example, num-2 appears in both the documents ie. the same word appears
in both the documents. These words should be consistently encoded into numbers

3. Running the LDA
The LDA may be invoked with the following commandline args
vw -d <input-file-name> --lda <num-topics> --lda_D <total-num-docs> --passes 10 -c -k --readable_model <model-file>

This will read from <input-file-name> and output the readable model(ie. not binary) file to <model-file>.
The other command line options may be found here - https://github.com/JohnLangford/vowpal_wabbit/wiki/Command-line-arguments

4. Making sense of the readable model file
The output format of the file is:
1. The first 10 lines contain metadata. They need to be ignored.
2. From the 11th line onwards, all lines will contain 10(assuming you wanted 10 topics) floating point numbers. Those are the topics.
3. The line number tells you the number the word represents(ie <num-1>, <num-3> etc)
4. Sort each column and then print the word in the following way.
4.1 Get the top 10 column values in column 0. Note their indices.
4.2 Print them by spreading aross all the topics. So
Topic-0 Topic-1 Topic-2 ..
col0-0 col0-1 col0-2 ..
4.3 Do the above 2 steps for the required number of topics.

5. General Reference
https://github.com/JohnLangford/vowpal_wabbit

LDA option 2 : python sklearn (Nikhil)

1) Install scikit i.e. sklearn package for python

2)Convert the Top set of documents for each query into document-term matrix. You can use CountVectorizer from sklearn.feature_extraction.text package
It is good idea to restrict stop words, words with very high document frequency and also words not occouring more than say 1 or 2 time. These all can be passed in CountVectorizer

3)Train by fitting the sparse matrix obtained from above to LatentDirichletAllocation from sklearn.decomposition. This will fetch N topics with M features as word. Sort

features probability to get top K words.

4)Transform the model to get distribution of topic over documents. Sort them again and list top 3 topics per document.
Also fetch top 3 topics for each query by performing by rank measure of topics or by checking for feature occurrences in each document and sorting top 3 topic for max score

LDA option 3 : Python LDA (Rashmi) - easiest to use

Python LDA : https://pypi.python.org/pypi/lda
The above link has the documentation for using the library assuming the dataset is already available as lda.datasets.
For custom datasets, we need to follow below steps:

1. Import numpy
2. Import lda
3. Convert the Top set of documents for each query into document-term matrix. This can be carried out using various python packages. For ex: textmining or nltk
4. Fit the document-term matrix to LDA
5. Model provides 2 probability distribution after computation
   a. Topic – word probability distribution. Lookup with document-term matrix, the top 10-20 words will form the topic.
   b. Document – topic probability distribution

LDA option 4 : Mallet (java)

http://mallet.cs.umass.edu/topics.php (http://mallet.cs.umass.edu/topics.php)
http://programminghistorian.org/lessons/topic-modeling-and-mallet (http://programminghistorian.org/lessons/topic-modeling-and-mallet)
https://piazza.com/class/io1w9cnmpk23i6?cid=195 (https://piazza.com/class/io1w9cnmpk23i6?cid=195)

B) LDA-topics and clustering

Run LDA on the entire AP89 collection, with about T=200 topics. Obtain a representation of all documents in these topics.
Then run a clustering-partition algorithm on documents in this topic representation. partition means every document gets assigned to exactly one cluster, like with K-means. You are free to use any clustering library/package. Target K=25 clusters. List each cluster with its documents IDs. You should have an ES index set up so we can look up documents by ID.

How to evaluate: There are about 1831 relevant documents in total. Consider all the pairs of two relevant documents, that is (1831 choose 2). For each pair, count a 1 in the appropriate cell of the following confusion table:

	same cluster	different clusters
same query		
different queries		

Rubric

15 points
Set up the data per query
20 points
Correctly set up and run LDA
20 points
Get the output, both docs->topics and topics->words probabilities/scores
15 points
Analyze the result by manual inspection. Does the docs->topics representation makes sense for the relevant documents. Only need to check 5 docs/query for a few queries

30 points
part B