11-741/11-641/11-441 – Machine Learning for Text Mining
Homework 2: k-means Clustering

TA in charge: Pengfei Wang

Due: February 4, 2016, 11:59 PM

This assignment will give you experience implementing the k-means clustering algorithm on news articles. The assignment consists of three major parts:

1. A clustering program that produces clusters of documents.
2. Several experiments with the program
3. A report describing your program, the experiments, and your conclusions

A word of advice: please do not leave this homework to the last minute. It may take some time to properly understand and implement the algorithm, and you will certainly need time to perform experimentation. **It is highly recommended that you begin early—ideally you will spend half of your time on the implementation and half of your time on the experimentation and analysis.** You may find it beneficial to construct a toy dataset and run the algorithm by hand before attempting to write your code.

1 The dataset

We will use a subset of the TDT4 data. It contains almost 2000 documents, split into a development set and a test set. As this data is provided by LDC, it is **subject to the following guidelines**:

- You MAY NOT distribute the TDT4 subset to anyone outside of the class.
- You MUST remove the dataset from your computer when the class is over.
- The files are accessible only within the CMU network. If you need to access them from your home, use the CMU library VPN service.

The data you will need is available as a .tgz file: HW2_data . The file contains a README describing the data and formatting, which you should read before beginning your implementation.

2 The Program

Your program is responsible for doing the following:

1. Reading in the vector data from the TDT4 subset
2. Producing clusters of documents using k-means
3. Writing the final clusters to log files and calculating performance on macro-average F1

In your implementation of k-means, please use the **cosine similarity** metric for comparisons between vectors in your baseline system. Your code must operate on the provided document vectors, not the raw text.

Your program should be designed to use two different strategies for choosing the initial cluster centroids:

1. Randomly selecting cluster centroids
2. Picking cluster centroids using k-means++

In your experiments, you will provide results under both settings.

3 Evaluating Results

For the evaluation of document clusters, we provide a script that will compare your output clusters to the gold standard development set clusters, reporting the macro-average F1.

4 Corpus Exploration

In addition to running your experiments, you must do some exploration of the dataset. Please analyze the **document vectors** rather than the **raw text** (we wish to analyze the corpus after stemming and stopword removal). In your report, include the following pieces of information:

1. Provide the following corpus statistics **for the development set**:
   - the total number of documents
   - the total number of words
   - the total number of unique words
   - the average number of unique words per document
2. Provide the following corpus statistics **for the test set**:
   - the total number of documents
   - the total number of words
   - the total number of unique words
   - the average number of unique words per document
3. For the first document in the **development set**, provide the following:
   - the total number of unique words
   - all of the word ids that occurred exactly twice in the document

5 The Experiments

Recall that k-means does not find the global optimum. Since the k-means step uses a different initialization each time, you will need to run your code several times in order to observe the average performance. For your report on the development set, we will be asking you to repeat each of your k-means experiments 10 times, and provide some statistics on your results. For the test set results, we will only be requiring you to submit a single result of clustering the documents.

5.1 Baseline experiments

Begin by running your full system on the provided document vectors, using **randomly selected cluster centroids**. Try adjusting **the total number of document clusters** and see how this affects performance. After finding reasonably good parameters for the number of clusters, try adjusting the **stopping criteria** for the algorithm.

5.2 k-means++

Now, we will try using a different initialization strategy for the cluster centroids. Run your document clustering using k-means++ for initialization.

5.3 Custom algorithm experiments

Next, you will develop a **custom algorithm** that expands on your approach. Try adjusting your system in some interesting manner to improve upon the baseline. For instance, you might adjust the term weighting scheme for the documents to use something other than just raw term frequency, or try using a similarity metric other than cosine similarity. Please document these changes in your written report.

Run this new system on the data, once again using **k-means++ for your centroid initialization**.

5.4 Test set experiment

Finally, you will submit a single result of the document clusters from running your best algorithm on the test set, named as "andrewID-test-clusters.txt".

6 What to Turn In

Please submit the required files packed in zip or tar.gz format (andrewID-hw2.zip or andrewID-hw2.tar.gz). The required files are described below, as well as the point distribution for each section.

6.1 Written report

Write your report in PDF, Word or plain text format (PDF is preferred). Please include your **name and Andrew ID** at the top of the first page of your report. Your report must contain the following sections, **each clearly labeled as an independent section**. We care about the quality of your report, so **please allocate a sufficient amount of time to your report**. An ideal report should be clearly written, well organized, and sufficiently detailed. In general, one sentence answers or analyses are not considered to be sufficiently detailed.

6.1.1 Corpus Exploration (5%)

The results of your corpus exploration as described above.

6.1.2 Experiments (50%)

For all your experiments, **please run your code 10 times and evaluate each run. In addition, please provide the values for the tuned parameters (number of document clusters, stopping criteria)**. Report the mean and variance for F1 over the 10 runs, as well as the best F1 achieved by the system.

1. **Baseline approach**: Run your baseline k-means system after tuning the parameters.
2. **k-means++ approach**: Run your k-means++ system after tuning the parameters.
3. **The custom algorithm**: Provide a detailed description of your custom algorithm. Include your motivation behind the algorithm: what problem were you trying to solve with this algorithm? How were you trying to improve upon the basic algorithm?

   Give results for your custom algorithm.

6.1.3 Analysis (20%)

Compare the results of these three systems on the reported metrics. Which approach saw the best results? Which approach performed the worst? For the custom algorithm, what worked (or didn't work) for this algorithm? Can you explain why you saw improvement (or not) over the basic algorithms?

6.1.4 The software implementation & data preprocessing (10%)

1. a description of your design decisions and high-level software architecture;
2. a description of major data structures (if any);
3. any programming tools or libraries that you used;
4. strengths and weaknesses of your design, and any problems that your system encountered

6.2 Source Code (10%)

Include your source code and all the scripts that you used to run your program, either as a zip or tar.gz file. The TA will look at your source code, so make sure that it is legible, has reasonable documentation, and can be understood by others. In particular, make sure there are clear instructions for how to run your code. This is a Computer Science class - we do actually care about your source code. **Note: although code documentation is worth only 10%, failing to submit source code will result in a zero grade**.

Pelase make sure you submission is self-contained especially when your implementation involves external linear algebra libraries. The TAs should be able to run your code without any manual modifications. Otherwise the grade for this part will be zero. It would be a good idea to test your implementation on the remote UNIX environment `https://www.cmu.edu/computing/clusters/software/timeshares.html` (which is similar to the environment where the TAs are going to run your code) before the submission.

6.3 Test Set Results (5%)

In addition to your written report, please submit a single result of the **document clusters** from running your best algorithm on the test set, named as "andrewID-test-clusters.txt".

Please make it easy for the TAs to see how you have addressed each of the requirements described for each section.

7 Restrictions

1. Your system must do this task entirely automatically. No manual intervention is allowed. The TA **will not** modify your source code in order to change the parameters or run a different experiment.
2. You must write all of the software yourself.

8 FAQ

- **Question**: How should we handle empty clusters?

  Answer: This is up to you. There are a variety of different strategies to do this – for instance, randomly picking a new point to move to that cluster. Regardless of what strategy you choose, please document your approach in your report.

- **Question**: Can we use an external library for linear algebra operations/sparse data representation?

  Answer: Yes, this is fine. Eigen (`http://eigen.tuxfamily.org/index.php?title=Main_Page`) and mahout-math (`http://mvnrepository.com/artifact/org.apache.mahout/mahout-math`, documentation at `http://archive.cloudera.com/cdh4/cdh/4/mahout/mahout-math/`) are several options – if you have another preferred library or wish to write your own, this is also acceptable. However, you are still expected to write your own code for the clustering –**using an existing clustering implementation will be considered as cheating**.

- **Question**: Why is my code so slow?

  Answer: Are you using a sparse representation of your data? Or if you are using a separate library for matrix computation, are you using the class for sparse matrices? Many such libraries contain classes for dense matrices as well as sparse matrices, so it is important to make sure you are using the correct one.

If your question is not answered there, please contact the TAs through Piazza.