Homework 2

Large-Scale Data Analysis

Ninh Pham, Christian Igel

2 May 2017

Submission Deadline:
9 May 2017, 10:00

Note: There will be four or five regular homework assignments on the LSDA course. They must be passed in order to be eligible for the final exam. Here, "passing an assignment" means that you have to make a serious attempt to solve the corresponding exercises, which can, e.g., be measured in terms of points for the individual subtasks. For this second assignment, you will need again, in general, at least 50 out of the 100+10 points.

1 Finding Similarity Items (50 + 10* pts)

The assignment aims at investigating MinHash and Locality-sensitive Hashing (LSH) framework on real-world data sets. In class, we have seen how to construct signature matrices using random permutations. However, in practice, random permutation on very large matrix is prohibitive. Section 3.3.5 of your textbook (J. Leskovec, A. Rajaraman, J. Ullman, *Mining of Massive Datasets*, `http://www.mmds.org/`, in Chapter 3) introduces a simple but fast method to "simulate" this randomness using different hash functions. We encourage you to read through that section before attempting the assignment.

In the assignment, you write a Python program to compute *all pair similarity* on the "bag of words" data set from the UCI Repository[1] using the *Jaccard* similarity. The bag of words data set contains 5 text datasets which share the same pre-processing procedure. That is, after tokenization and removal of stopwords, the vocabulary of unique words was truncated by only keeping words that occurred more than 10 times. Therefore, it has the format: $docID$ $wordID$ $count$, where $docID$ is the document ID, $wordID$ is the word ID in the vocabulary, and $count$ is the word frequency. Note that $count > 10$. Since the Jaccard similarity does not take into account the word frequency, we simply ignore this information in the assignment. This means that we consider $count = 1$ for each pair $(docID, wordID)$.

We make use the ***KOS blog entries*** data set since we still need to run bruteforce algorithm to measure the accuracy of MinHash and LSH. However, students are encouraged to try with larger data sets, such as *NYTimes news article* and *PubMed abstracts*. Note that the data set is very sparse, so using the sparse matrix packages, e.g. scipy.sparse[2] in Python is highly recommended.

The assignment tasks and its point are as follows.

1. **Execute bruteforce computation (10 pts):** Compute all pair similarity with Jaccard and save the result into file (note that you might have to use the bruteforce result for the next task). You need to report the running time of bruteforce algorithm and the average of Jaccard similarity.

[1] https://archive.ics.uci.edu/ml/datasets/Bag+of+Words
[2] https://docs.scipy.org/doc/scipy-0.18.1/reference/sparse.html

2. **Compute the MinHash signatures for all documents (10 pts):** Compute the MinHash signatures (number of hash functions $K = 100$) for all documents. You need to report the running time of computing MinHash signatures.

3. **Measure the accuracy of MinHash estimator (10 pts):** Compute all pair similarity estimators based on MinHash. Repeat the procedure 2) and 3) with number of hash functions K ranging from $\{10, 20, \ldots, 100\}$ and plot the mean absolute error (MAE) of MinHash estimators on different K values. You need to report the running time of computing all pair based on MinHash with different K values, and the figure of MAE with K values on x-axis, and MAE values on y-axis.

4. **Exploit LSH (20 pts):** Implement LSH framework to efficiently solve the subproblem: "Finding *all* similar pairs of documents with Jaccard ≥ 0.6". In particular, using $K = 100$ hash functions, explain how to tune the parameter b (number of bands) and r (number of rows in one band) so that we achieve the false negatives of 60%-similar pairs at most 10%. You need to explain the cost of space usage influenced by these parameters. Given that setting, report the false positives (number of pairs with Jaccard < 0.6 / number of candidate pairs), and the probability that a far away pair with Jaccard ≤ 0.3 is in the candidate pair.

5. **Analysis of LSH (10* pts):** In class, we claimed that "Finding all pairs of documents can be done in $O(N)$ where N is the number of documents". Is the claim generally correct? Explain your answer.

What to submit?

1. A Python code with clear comments.

2. An `answers.pdf` reports the request values and your explanations on each task.

2 Getting Started with TensorFlow: Logistic Regression (50 pts)

This part of the assignment should provide some basic experience with TensorFlow.

The first step is to get TensorFlow working (see instructions on Absalon). Then download `LogisticRegressionWeedTensorFlowTensorBoardAssignment.py` and the data files `LSDA2017WeedCropTrain.csv` and `LSDA2017WeedCropTest.csv`.

Have a close look at `LogisticRegressionWeedTensorFlowTensorBoardAssignment.py` and execute it. You have to tell the program where to find the data using the `--data_dir=` command line option.

1. **TensorBoard (10 pts):** Start TensorBoard and inspect the learning curves of the 0-1 loss and the cross-entropy (e.g., see `LSDA TF Introduction.ipynb`).

   Deliverabes: Screenshot(s) of the TensorFlow plots showing the 0-1 loss and the cross-entropy

2. **Command line options (10 pts):** Inspect the code. How does `datadir=` work? Add a new command line option for setting the learning rate of the gradient-descent optimizer.

   Deliverables: Report the lines in the code you added and changed in the report

3. **Visualize compute graph (10 pts):** Extend the program such that the compute graph is stored and can be visualized using TensorBoard (e.g., see `LSDA TF Introduction.ipynb`). Note how `tf.name_scope` is used to group compute nodes.

   Deliverabes: Include the plot of the compute graph in your report

4. **Adjust learning rate (20 pts):** Play around with the learning rate. Which learning rate gives good results? When does a positive learning rate become clearly too large and clearly too small? What happens?

   Deliverabes: Report three positive learning rates, one that works well, one which is too high, and one which is too low; add screenshots of the 0-1 loss and the cross-entropy for the three learning rates; short discussion of observations

Note: When you restart the program, you may need to delete the directory with the log data (e.g., `rm -r /tmp/logLog`) beforehand or set a different directory using the `--summary_dir=` command line option.