Data Mining: Learning from Large Data Sets - Spring Semester 2014
Instructor: Prof. Andreas Krause

Course Project
Part I

Approximate near-duplicate search using Locality Sensitive Hashing

1 Introduction

In this project we are going to consider an application of Locality Sensitive Hashing to near-duplicate video detection. Suppose that there exists a company which offers a web service that allows users to upload various (cat) videos. If a user uploads a copyrighted video the company is sooner or later sued by the copyright owner. In order to solve this expensive problem, the company hired you.

Your task is to develop an efficient method to to detect whether a video is a near-duplicate of a copyrighted video, in which case you don't want to allow the user to publish it.

2 Dataset description

You are given two text files: "train.txt" and "duplicates.txt". Each line of the training dataset contains the features for one video file and is formatted as follows: string "VIDEO_XXXXXXXXX" followed by a list of space delimited integers in range $[0, 10000]$. You can consider them equivalent to shingles in the context of near-duplicate document retrieval. Furthermore, we will consider the Jaccard similarity based the video features.

Duplicates file contains the true near duplicates (documents that are at least 85% similar). Each line is a tab-delimited pair of integers where the first integer is always smaller. Note that similarity is a symmetric function.

3 Evaluation and Grading

You are asked to provide a Map function and a Reduce function written in Python.

The Map function receives files (formatted as "train.txt") as the standard input. The output of the Reduce function should be tab separated list of integers representing the duplicated documents your LSH based algorithm found. **The number of hash functions used has to be less than or equal to 256.**

We have provided some template code for both the Map and Reduce function as well as a function to check the F_1 score of your submission on the local training dataset.

Each submission on the project website will be run against a bigger dataset that contains many more instances than the training dataset but the sample given to you is representative, in a sense that you can tune the parameters of your algorithm locally (up to a certain degree). **The number of submissions per team is limited to 50. The submission with the highest F_1 score will be used for grading.**

The evaluation metric that we are going to use is F_1 score.

For each line of the output of your MapReduce program we will check whether two documents reported on that line are in fact at least 85% similar (we have the ground truth for the larger dataset). If they are this row will

count as one true positive. If they are not it will count as one false positive. In addition, each document pair that is similar but was not reported by your algorithm will count as one false negative. Given the number of true positives, false positives and false negatives, *TP*, *FP* and *FN* respectively, we can calculate:

- precision, also referred to as Positive predictive value (PPV), as $P = \frac{TP}{TP+FP}$, and
- recall, also referred to as the True Positive Rate or Sensitivity, as $R = \frac{TP}{TP+FN}$.

Given precision and recall we will calculate the F1[1] score defined as $F_1 = 2\frac{PR}{P+R}$. Your task is to maximize this score.

We will compare the F_1 score of your submission to two baseline solutions: a weak one (called "baseline easy") and a strong one (called "baseline hard"). These will have the F_1 of FBE and FBH respectively, calculated as described above. Both baselines will appear in the rankings together with the F_1 score of your submitted predictions.

Performing better than the weak baseline will give you 50% of the grade, and matching or exceeding the strong baseline on the **test set** will give you 100% of the grade. This allows you to check if you are getting at least 50% of the grade by looking at the ranking. If your F_1 score is in between the baselines, the grade is computed as:

$$\text{Grade} = \left(1 - \frac{\text{FBH} - F_1}{\text{FBH} - \text{FBE}}\right) \times 50\% + 50\%$$

3.1 Report

At the end of the course you will be requested to provide a team report which describes your solution. We include a template for LaTeX in the file "report.tex". If you do not want to use LaTeX, please use the same sections as shown in "report.pdf". You will be required to upload the reports after the last project of the course.

3.2 Deadline

The submission system will be open from **Monday, 03.03.2014, 17:00** until **Sunday, 23.03.2013, 23:59:59**.

4 Software

The following is available on each machine in the cluster:

- Python 2.7
- NumPy 1.8.0
- SciPy 0.13.1
- scikit-learn: 0.14.1

[1] http://en.wikipedia.org/wiki/F1_score