CS 562: Problem Set 2

March 2, 2014

This collection of problems will test your understanding of similarity metrics, and LSH-based algorithms for determining similarity. The problem set is due back on **Saturday, Mar. 15th by midnight** on Sakai. Please read and understand the following general instructions:

- Collect all the answers to written questions in a single **PDF** document called **set2.pdf** - **please make sure that you give this exact name to your document.**
- For the questions that require coding, create a **zip file** called **set2.zip** from this directory. **All code must run on the clamshell server** - so please test your code on that machine in advance of submitting it. A README file that describes *how* I should test your code **from the command line** on **clamshell** is mandatory, as is appropriate documentation in code.

Problems

1. Suppose that a certain distance metric computes document distances on a continuous scale of [0,1]. Suppose that you are given a family of (0.2, 0.5, 0.6, 0.25)-sensitive hash functions from which an amplified LSH scheme is to be constructed using n =80 randomly chosen hash functions divided into b bands of r rows each with the AND-construction within each band and the OR-construction across the bands. You are also given an upper bound on the probability of announcing both false positives and false negatives, say 10 percent, from the LSH scheme (i.e. *neither* kind of misclassification can occur with probability more than 0.1). Can this be achieved? Explain your answer with by constructing an appropriately annotated plot (implemented in program **p1.py**).
2. Exercises 3.7.1 and 3.7.2 from the MMDS textbook that involve computing *sketches* using vectors of length 4.
3. Exercise 9.2.1 from the MMDS textbook on scaled cosine distance.
4. You will find two document sets on Sakai (under Resources →Data). These sets have been archived and compressed. Start with the set **med_doc_set.tar.gz**, and once you have fig-

ured out the steps using sklearn, repeat this for the other set, **big_doc_set.tar.gz**. In particular, you should write a Python program named **p4.py** that find similarities in the document set (whether medium or small). For the submission, you can assume that the big document set will be available to your code in the subdirectory **big_doc_set**.

sklearn has a module called **feature_extraction** that allows you to extract features from raw data using Vectorizer objects (see the documentation). In particular, the sub-module **text** has classes to create document feature matrices. Study the API carefully, experiment with it, and create matrices for features that are **3-word shingles** for the document set (these are also called n-grams). You can experiment with the options (omitting stop words etc.) and report the ones that you used.

You should create two matrices for the data set: a count matrix to be used with the bag-of-words Jaccard similarity distance metric and a TF.IDF matrix to be used with the cosine similarity distance metric. See the definition of TF.IDF in the MMDS text: scikit-learn has APIs for extracting these scores. Note that the Vectorizers will produce document-feature with documents as rows and features as columns: you will be essentially working with the transpose of this matrix if you want to compute document similarity.

Now, construct LSH signature schemes for finding similar documents with both the Jaccard and cosine distance metrics (as described in the textbook) using a banding scheme with six random hash functions per band ($r = 6$) and four bands ($b = 4$). Recall that each signature requires a random hash function.

(a) For the Jaccard distance, use a minhash signature obtained from a universal family of hash functions: e.g. suppose your documents have 2003 features. Choose a large prime, say p =32452867. Then a hash function from the family will be characterized by a random $1 \leq a < p$ and a random $0 \leq b < p$. The minhash signature for a document will be the minimum value of $((ax + b)\%p)\%m$ over all features x in the document.

(b) For the cosine distance, form the dot product of every feature vector (corresponding to a document) with a random unit vector. The sign (+1 or -1) of the product will be the signature of the document.

Note that you would identify two documents as being a candidate pair if their hashed signatures are identical for all the functions in at least one of the four bands. Create appropriate reports and visualizations (you choose the formats) for describing document similarity (note that you may want to eventually compute exact distances for candidate pairs to minimize false positives). Try to experiment: for example, does it affect your similarity prediction much if you

only keep the features that appear in union of the top 10 features (ranked by TF.IDF values) for each document?

5. Form Chapter 3 of the LFD book, Problem 3.3 parts (a), (b), (c) and (d). Your program should be named **prob3.3.py** with clear documentation indicating the code corresponding to the last three parts of the problem.