Programming Assignment 1 Instructions

Due by Jan 28, 2019 11:55 pm

Please note that written homework 1 is up.

Goal: In this assignment, we will apply the locality sensitive hashing technique learned in the lecture to a question dataset. The goal is: **for each** question X, find a set of questions Y in the data set such that Sim(X,Y) ⩾ 0.6, where the similarity is Jaccard.

Input Format: The datasets are given in tvs (tab-separated) format. The file contains two columns: `qid` and `question`. Four datasets provided in a single zip-compressed file are:

1. `question_4k.tsv`: This dataset contains 4,000 questions.
2. `question_50k.tsv`: This dataset contains 50,000 questions.
3. `question_150k.tsv`: This dataset contains 150,000 questions.
4. `question_290k.tsv`: This dataset contains 290,000 questions.

The dataset can be downloaded from here.

Output Format: output must be given in tsv forrmat, with two columns: `qid` and `similar-qids` where `qid` is the qid of the queried question and `similar-qids` is the set of similar questions given by their qids. The format of column `similar-qids` is comma-separated. If a question has no similar question, then this column is empty. Below is an example of the output format:

qid	similar-qids
11	
13	145970
15	229098,280602,6603,204128,164826,238609,65667,139632,265843,143673,217736,38330

The way to interpret the above sample output is: the question of qid 11 has no similar question, the question of qid 13 has 1 similar question of qid 145970 and the question of qid 15 has 12 similar questions. You can download a sample output tsv file here. The name of the output file must be `question_sim_[*].tsv` where [*] is replaced by the size of the dataset. For example, the output of the 4k question data set must be `question_sim_4k.tsv`.

There are two questions in this assigment. The first question is worth 15 points and the second question is worth 35 points, all of 50 points total.

Question 1 (15 points): Implement the native algorithm that, for each question, loops through the database, computes the Jaccard similarity and output questions of similarity at least 0.6. For full score, your algorithm must run in **less than 3 minutes** on the dataset `question_4k.tsv`.

Question 2 (35 points): Implement the locality sensitive hashing algorithm we learned in the class, with x = 0.6, s = 14 and r = 6, where s is the number of hash tables (we use b instead in the lecture slide) and r is the size of the minhash signature. For full score, your algorithm must run in **less than 10 minutes** on the dataset `question_150k.tsv`.

Note 1: As you may understand from the lecture, it could be that two non-similar questions are mapped to the same location in the locality sensitive data structure. This is called false positive. You must remove all false positives before writing to the output file.

Note 2: Submit your code and output data to the Connex

FAQ

Q1: Will 50k and 290k question datasets be graded?
Answer: No. They are provided for learning purposes.

Q2: How can we generate a random number in Python3?
Answer: Here is an example code that I use for generating a random 64-bit integer in my implementation.

Q3: What kind of hash function do you recommend for computing the minHash signature?
Answer: In my implementation, I use the linear hash function h(x) = (a*x +b) mod p, where a,b are two random 64-bits integers and p is a 64-bit prime integer. I set p = 15373875993579943603 for all hash functions.

Q4: How can I map a string (and a word specifically for this homework) to an integer so that I can feed it to the linear hash function in Q3.
Answer: I recommend the FNV hash function. You can download and install following the instruction in here. However, I use this library in a slightly different way. Here are steps: I download the libarary, look for the file name "__init__.py" in the downloaded package, rename it to "fnv.py", put to the source code folder and import to my code. Here is an example of how to import it. You may notice that there are three diffent hash functions in the example. I use this function `hash(data, bits=64)` in my implementation.

Q5: If I don't use python, where can I find a version of the FNV function implementation in other languages?
Answer: You can visit this site. It might have what you want.

Q6: Do you apply any advanced processing technique to nomarlize the datasets?
Answer: I don't. I want to keep the implementation as simple as possible for learning purpose. I do use `question.strip()` to remove possible white-space characters ended at each question. Then, I just use split function of Python3 `question.split()` to break a question into words. You may notice that in this implementation, "what" and "What" would be regarded as different words because I do not handle capitalization. You are welcome to use any technique that can help you improve the correctness of your algorithm, but keep in mind the running time constraint.

Q7: If the outputs of my implementation and another group's implementation are different, is this a problem?
Answer: No. Because the nature of randomness in locality sensitive hashing, I expect differences in the output. The assignment will mainly be graded based on: speed and your understanding of the algorithm reflected in your code. And don't forget the dicussion policy that I specified in class.

Want to go back to the course overview, click here.