AI506: Data Mining and Search (Spring 2023)

Homework 1: Weighted Locality Sensitive Hashing

Release: March 10, 2023,
Due: March 24, 2023, 11:59pm

With the advent of the age of e-commerce, it has become convenient to buy and sell an abundant amount of products. Finding similar items is an important task for both customers and retailers. Recommending similar items can help customers find complementary items that might not have been considered otherwise. Also, retailers can manage the products by categorizing products based on their similarities. However, it is not a trivial task to exactly compute the similarities among a vast amount of items. Locality Sensitive Hashing (LSH) is an algorithm that is well-suited to handle this problem.

In this assignment, you will find similar pairs of items in the MovieLens 10M dataset (`https://grouplens.org/datasets/movielens/10m/`). The dataset contains 10 million ratings applied to 10,677 movies by 69,878 users. You will employ the weighted Jaccard similarity between the items to identify similar pairs. While the naive Jaccard similarity only measures the similarity between sets of users who consumed the items, the weighted Jaccard similarity takes how similarly the users rated the items into account.

To begin, you will review the definition of the weighted Jaccard similarity. Then, you will familiarize yourself with the metric by solving some examples manually. Finally, you will implement LSH utilizing the provided Python skeleton code."

1 Weighted Jaccard Similarity [40 Points]

In this section, you will take a time to understand the weighted Jaccard similarity. Given two *sets* A and B, the Jaaccard similarity $\mathcal{J}(A, B)$ between them is defined as:

$$\mathcal{J}(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

The *weighted* Jaccard similarity $\mathcal{J}_w(X, Y)$ between two *vectors* X and Y of equal length and with integer weights is defined as:

$$\mathcal{J}_w(X,Y) = \frac{\sum_i \min(X[i], Y[i])}{\sum_i \max(X[i], Y[i])}.$$

For more details, refer to `https://en.wikipedia.org/wiki/Jaccard_index`.

1.1 (Written) Computing the weighted Jaccard similarity [10 Points]

Assume we have integer ratings of items from users ranging from 1 to 5. This can be represented as an item-user rating matrix where rows represent the items and columns represent the users. We consider the missing values of the item-user rating as zeros. First, consider the following example item-user rating matrix M with three items and five users:

$$M = \begin{bmatrix} 3 & 0 & 1 & 5 & 0 \\ 2 & 1 & 0 & 5 & 1 \\ 5 & 0 & 4 & 1 & 0 \end{bmatrix}$$

- (10 points) Calculate by hand the weighted Jaccard similarity between the three pairs of rows in M. Which of the second and third items is more similar to the first item? What if we do not consider the weights (i.e., ratings)? Is the second (or third) item still more similar to the first item?

1.2 (Written) Connection to the Jaccard similarity [10 Points]

Below, you should show that the weighted Jaccard similarity between items can be computed by the naive Jaccard similarity if we properly transform an item-user rating matrix into a binary matrix.

- (10 points) Consider the weighted matrix M above. Transform this matrix into a binary matrix $M_B \in \{0,1\}^{3\times s}$ where $s = 25$ is the number of *shingles*. The Jaccard similarity between items i and j computed from M_B should be equivalent to the weighted Jaccard similarity computed from M, i.e., $\mathcal{J}(M_B[i], M_B[j]) = \mathcal{J}_w(M[i], M[j])$.

Hint: Each column in M can be transformed into five binary columns in M_B since there are five different ratings (i.e., $\{1, 2, \cdots, 5\}$).

1.3 (Written) Computing MinHash Signatures [20 Points]

Next, solve the following problems, which will help you understand the mechanism of LSH.

- (10 points) Calculate the minhash signatures of each row of M_B using the following 3 hash functions: $h_1(x) = 7x + 1 \mod 25$; $h_2(x) = 5x + 2 \mod 25$; $h_3(x) = 3x + 5 \mod 25$ where x is the index of each column (from 1 to 25).
- (10 points) Calculate the estimated Jaccard similarities of the three pairs of the rows and compare them with the true Jaccard similarities.

2 Locality Sensitive Hashing (LSH) [60 Points]

The starter code is provided for you to implement the LSH algorithm and analyze it. Fill in the blanks in the code marked as Programming assignments.

2.1 Dataset: MOVIELENS 10M

In this assignment, you will find similar items in the MOVIELENS 10M dataset, given a query item. You are given two files: `Rating.txt`, and `MovieInfo.txt`. In `Rating.txt`, each line consists of user, movie, and rating, separated by a comma. User and movie indexes are integer values starting from 0, and the rating is an integer value between 1 to 5. `MovieInfo.txt` contains the movie index and corresponding title in each line, separated by a comma.

Note that the considered dataset contains ratings given by users to the items, and thus the weighted Jaccard similarity will be used to measure the similarities between items. Think carefully about how LSH can be used to find pairs of items similar in terms of weighted Jaccard similarity, recalling the exercises you did in the previous section.

2.2 Preliminaries [20 Points]

Before constructing the main algorithm, let's implement the preliminaries. You are allowed to use `NumPy`. Any other external libraries are not allowed. Here, you are given `M.txt`, which contains the item-user rating information that is equivalent to the ratings given in M above.

1. (5 points) Implement the `weighted_jaccard_similarity` to compute the weighted Jaccard similarity of two given weighted vectors. Compute the weighted Jaccard similarity between the three pairs of items given in M.
2. (10 Points) Implement the `build_items_to_shingle_dictionary` to create the shingle for the items. The Jaccard similarity between items i and j computed from the shingles should be equivalent to the weighted Jaccard similarity computed from the rating matrix. Recall how we constructed M_B from M in the previous section.
3. (5 points) Implement the `jaccard_similarity` to compute the Jaccard similarity of two given sets. Compute the Jaccard similarity between the three pairs of items using the shingles of each item.

2.3 Locality Sensitive Hashing [15 Points]

In this section, you will implement the LSH algorithm. It receives signature matrix and `b` (i.e., the number of bands) and `r` (i.e., the number of rows in each band) as input parameters, and return the candidate pairs of the similar items.

1. (15 Points) Implement the `lsh` function to find all candidate pairs of the similar items using LSH algorithm.

Notes: Use python's dictionary to make your hash table, where each column is hashed into a bucket. Convert each column vector (within a band) into the tuple and use it as a key of the dictionary.

2.4 Analysis [15 Points]

In this section, you will analyze the LSH algorithm in terms of query speed and the various measures of relevance. In this assignment, we fixed `M` (i.e. the number of hash functions) into 100, `b` to be the divisor of `M`: $b \in \{1, 2, 4, 5, 10, 20, 25, 50\}$, and $s = 0.4$ (i.e. the similarity threshold for checking condition positives).

1. (5 points) Implement the `query_analysis` function to compute the query time, precision, recall, and F1 score as `b` and `r` change.
2. (5 Points) Attach your query time graph using TAs' pre-implemented code. How does the query speed change as `b` changes? Explain the results in terms of the computational complexity.
3. (5 Points) Attach your precision, recall, and f1-score graphs using TAs' pre-implemented code. How does each measure change as `b` changes? Which `b` gives the best result in terms of each measure? Explain the results in detail.

2.5 Case Study [10 Points]

Now that you have implemented the entire process of LSH, it's time to utilize it. You are given the following 4 movies with corresponding indices in the given dataset:

- **Star Wars: Episode VI - Return of the Jedi (1983)** - index 37
- **Wallace & Gromit: A Grand Day Out (1989)** - index 1,459
- **Toy Story (1995)** - index 96
- **Harry Potter and the Goblet of Fire (2005)** - index 1,239

1. (10 Points) For each movie above, search for the 10 movies that are estimated to be the most similar to it. Consider the weighted Jaccard similarity as the similarity metric and use the LSH function you have implemented. Provide the titles of the retrieved movies and report the Precision@10 by comparing them with the ground-truth movies that are obtained by computing the exact weighted Jaccard similarity with the query movies.

3 How to submit your assignment

1. Download the attached file. It contains the skeleton codes.
2. Fill in the skeleton codes.
3. Modify the name of the file and zip it into hw1-[your student id].tar.gz, which should contain the following files:
   - **hw1.ipynb**: this file should contain your source code.
   - **report.pdf**: this file should contain your answers to the questions. Also, please attach your implementations and if you have a test message, please attach the results as well. For the written assignments, we prefer you to write down the equations in word or latex.
   - **readme.txt**: this file should contain the names of any individuals from whom you received help, and the nature of the help that you received. That includes help from friends, classmates, lab TAs, course staff members, etc. In this file, you are also welcome to write any comments that can help us grade your assignment better, your evaluation of this assignment, and your ideas.
4. Make sure that no other files are included in the tar.gz file.
5. Submit the tar.gz file at KLMS (`http://klms.kaist.ac.kr`).

You may encounter some subtleties when it comes to implementation, please come up with your own design and/or contact Geon Lee (geonlee0325@kaist.ac.kr) or Kyuhan Lee (kyuhan.lee@kaist.ac.kr) for discussion. Any ideas can be taken into consideration when grading if they are written in the *readme* file.