(1) Code: see the corresponding file q1.py

(2) The first map function makes tuples $((user1, user2), v)$, $v = 0$ if user1 and user2 are friends to each other, $v = 1$ if user1 and user2 have common friends. Then the following subtract by key function generates a set with tuples $((user1, user2), 1)$ that user1 and user2 are not friends. Then we use GroupByKey function and reduce function to get a set with $((user1, user2), n)$, where n is the number of common friends. The second map function make every tuple $((user1, user2), n)$ to $(user1, (user2, n))$. Then the following GroupByKey function groups tuples and sorts them by descending order of n. The following reduce function make old tuples with same key "$user1$" to the new form $(user1, (recommendation1, recommendation2, \ldots))$. In the end, print out the top 10 recommendations for each user.

(3)The recommendations for the selected users are:

User Tab	Recommendations
924	439,2409,6995,11860,15416,43748,45881
8941	8943,8944,8940
8942	8939,8940,8943,8944
9019	9022,317,9023
9020	9021,9016,9017,9022,317,9023
9021	9020,9016,9017,9022,317,9023
9022	9019,9020,9021,317,9016,9017,9023
9990	13134,13478,13877,34299,34485,34642,37941
9992	9987,9989,35667,9991
9993	9991,13134,13478,13877,34299,34485,34642,37941

Confidence is defined as the probability of occurrence of B in the basket if the basket already contains A:

$$conf(A \to B) = Pr(B|A) = \frac{Pr(A \cap B)}{Pr(A)}$$

Let's consider a scenario as follow:

Basket 1	A,B,E,F
Basket 2	A,B,E
Basket 3	A,B
Basket 4	A,D
Basket 5	B,C,E,F
Basket 6	B,C,E
Basket 7	B,C
Basket 8	B,C,D

In the table, we have

$$Pr(A) = \frac{4}{8} = 0.5$$

$$Pr(A \cap B) = \frac{3}{8} = 0.375$$

so

$$conf(A \to B) = Pr(B|A) = \frac{Pr(A \cap B)}{Pr(A)} = \frac{0.375}{0.5} = 0.75$$

The drawback of the confidence measure is that it might misrepresent the importance of an association. The confidence only accounts for how popular "A" are, but not "B". If "B" are also very popular in general, there will be a higher chance that a basket containing "A" also contains "B", thus inflating the confidence measure. In this table, we found that $Pr(B) = \frac{7}{8} = 0.875$, which means that "$B$" appears in baskets very often and inflats the confidence measure.

Lift measures how much more "A and B occur together" than "what would be expected if A and B were statistically independent". A lift value greater than 1 means that "B" is likely to be in a basket if "A" is in the basket, while the value less than 1 means that "B" is unlikely to be in a basket if "A" is in a basket. From the table, we have

$$lift(A \to B) = \frac{conf(A \to B)}{S(B)} = \frac{0.75}{0.875} < 1$$

Conviction compares the "probability that A appears without B if they were independent" with the "actual frequency of the appearance of A without B". A high conviction value means that the consequent is highly depending on the antecedent, and conviction is 1 if items are independent. From the table, we have

$$conv(A \to B) = \frac{1 - S(B)}{1 - conf(A \to B)} = \frac{1 - 0.875}{1 - 0.75} = 0.5$$

The conviction value means that "B" is not highly depending on the antecedent. By observations, lift and conviction take $Pr(B)$ into account. Hence, lift and conviction do not suffer from the drawback we mentioned before.

Let's consider a scenario as follow:

Basket 1	A,B,E,F
Basket 2	A,B,E
Basket 3	A,B
Basket 4	A,D
Basket 5	B,C,E,F
Basket 6	B,C,E
Basket 7	B,C
Basket 8	B,C,D

Confidence is not symmetric. There is a counterexample from the table:

$$conf(A \to B) = Pr(B|A) = \frac{Pr(A \cap B)}{Pr(A)} = \frac{0.375}{0.5} = 0.75$$

$$conf(B \to A) = \frac{Pr(A \cap B)}{Pr(B)} = \frac{0.375}{0.875} = 0.428$$

It is clear that $conf(A \to B) \neq conf(B \to A)$. Therefore, confidence is not symmetric.

Lift is symmetric by the following proof:

$$lift(A \to B) = \frac{conf(A \to B)}{S(B)} = \frac{Pr(B|A)}{Pr(B)} = \frac{\frac{Pr(A \cap B)}{Pr(A)}}{Pr(B)} = \frac{\frac{Pr(A \cap B)}{Pr(B)}}{Pr(A)} = \frac{Pr(A|B)}{Pr(A)} = lift(B \to A)$$

Conviction is not symmetric. There is a counterexample from the table:

$$conv(A \to B) = \frac{1 - S(B)}{1 - conf(A \to B)} = \frac{1 - 0.875}{1 - 0.75} = \frac{0.125}{0.25} = 0.5$$

$$conv(B \to A) = \frac{1 - S(A)}{1 - conf(B \to A)} = \frac{1 - 0.5}{1 - 0.428} = \frac{0.5}{0.571}$$

Therefore, conviction is not symmetric.

Confidence and conviction are desirable, but lift is not. If "B" occurs every time "A" occurs, then we have the following: $conf(A \to B) = 1$, $lift(A \to B)$ depends on the value of $Pr(B)$, $conv(A \to B) = \infty$.

Let's consider the table

Basket 1	A,B
Basket 2	A,B
Basket 3	C,D
Basket 4	E,F

$$conf(A \to B) = Pr(B|A) = \frac{Pr(A \cap B)}{Pr(A)} = 1$$

$$conv(A \to B) = \frac{1 - Pr(B)}{1 - conf(A \to B)} = \frac{1 - 0.5}{1 - 1} = \infty$$

$$lift(A \to B) = \frac{conf(A \to B)}{Pr(B)} = \frac{1}{0.5} = 2$$

$$lift(C \to D) = \frac{conf(C \to D)}{Pr(D)} = \frac{1}{0.25} = 4$$

We can find that though $A \to B$ and $C \to D$ are 100% of the time, they have different lift value.

Rule	Confidence
DAI93865 $\Longrightarrow$ FR040251	1.0
GR085051 $\Longrightarrow$ FR040251	0.999176276771005
GR038636 $\Longrightarrow$ FR040251	0.9906542056074766
ELE12951 $\Longrightarrow$ FR040251	0.9905660377358491
DAI88079 $\Longrightarrow$ FR040251	0.9867256637168141

Rule	Confidence
(DAI23334, ELE92920) $\Longrightarrow$ DAI62779	1.0
(DAI31081, GR085051) $\Longrightarrow$ FR040251	1.0
(DAI55911, GR085051) $\Longrightarrow$ FR040251	1.0
(DAI62779, DAI88079) $\Longrightarrow$ FR040251	1.0
(DAI75645, GR085051) $\Longrightarrow$ FR040251	1.0

We know that the numbers columns with m 1's out of n is $\binom{n}{m} = \frac{n!}{m!(n-m)!}$. The number of columns that have no 1's in any of the selected k rows is $\binom{n-k}{m} = \frac{(n-k)!}{m!(n-k-m)!}$. The probability of getting "don't know" can be written as Pr(don't know):

$$Pr(don't \quad know) = \frac{\binom{n-k}{m}}{\binom{n}{m}} = \frac{(n-k)!m!(n-m)!}{n!m!(n-k-m)!} = (\frac{n-k}{n}) \times (\frac{n-k-1}{n-1}) \times \cdots \times (\frac{n-k-m+1}{n-m+1})$$

Each of the m factors is at most $\frac{n-k}{n}$, so $Pr(don't \quad know) \leq (\frac{n-k}{n})^m$. Hence, the probability is at most $(\frac{n-k}{n})^m$.

We want $(\frac{n-k}{n})^m \leq e^{-10}$, the computation is as follow:

$$\begin{aligned}
(\frac{n-k}{n})^m \leq e^{-10} &\iff (1-\frac{k}{n})^m \leq e^{-10} \\
&\iff [(1-\frac{k}{n})^{\frac{n}{k}}]^{\frac{mk}{n}} \leq e^{-10} \\
&\iff e^{\frac{-mk}{n}} \leq e^{-10} \\
&\iff \frac{-mk}{n} \leq -10 \\
&\iff k \geq \frac{10n}{m}
\end{aligned} \tag{1}$$

Therefore, the smallest k is $\frac{10n}{m}$.

Consider the following input matrix, cyclic permutations, and signature matrix:

π_1	π_2	π_3
1	3	2
2	1	3
3	2	1

Table 1: cyclic permutations

S_1	S_2
0	0
1	1
0	1

Table 2: input matrix

S_1	S_2
2	2
1	1
3	1

Table 3: signature matrix

The Jaccard similarity between S_1 and S_2 is $\frac{1}{2}$, but the probability that the minhash values agrees is $\frac{2}{3}$.

By the definition of AND of Hash functions, we have $\mathcal{G} = \mathcal{H}^k$ is $(\lambda, c\lambda, p_1^k, p_2^k)$ sensitive. Therefore, for each $1 \leq j \leq L$ and $x \in T$, $Pr[x \in T \cap W_j] \leq p_2^k$. Since $k = \log_{\frac{1}{p_2}} n$, we have $p_2^k = \frac{1}{n}$. Hence, $Pr[x \in T \cap W_j] \leq \frac{1}{n}$, which implies that $\mathbb{E}[|x \in T \cap W_j|] \leq 1$. By the linearity of expectation, $\mathbb{E}[\sum_{j=1}^{L} |T \cap W_j|] \leq L$. By the definition of Markov's inequality, we know that $Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$. Therefore, $Pr[\sum_{j=1}^{L} |T \cap W_j|] \leq \frac{\mathbb{E}[\sum_{j=1}^{L} |T \cap W_j|]}{3L} \leq \frac{L}{3L} = \frac{1}{3}$.

G_j is $(\lambda, c\lambda, p_1^k, p_2^k)$ sensitive for $1 \leq j \leq L$. By definition, we have $Pr[g_j(x^*) = g_j(z)] \geq p_1^k$, so $Pr[g_j(x^*) \neq g_j(z)] \leq 1 - p_1^k$. And by definitions of:

$$L = n^\rho \tag{2}$$

$$\rho = \frac{\log \frac{1}{p_1}}{\log \frac{1}{p_2}} \tag{3}$$

$$k = \log_{\frac{1}{p_2}} n \tag{4}$$

By 4, we get

$$k = \log_{\frac{1}{p_2}} n \iff \log \frac{1}{p_2} = \frac{\log n}{k} \tag{5}$$

Substitute 5 to 3, we get

$$\rho = \frac{\log \frac{1}{p_1}}{\log \frac{1}{p_2}} = \frac{k \log \frac{1}{p_1}}{\log n} \iff \rho \log n = k \log \frac{1}{p_1} \iff n^\rho = (\frac{1}{p_1})^k \tag{6}$$

Substitute 6 to 2, we get

$$L = n^\rho = (\frac{1}{p_1})^k \iff p_1^k = \frac{1}{L} \tag{7}$$

By 7, we know that $Pr[g_j(x^*) \neq g_j(z)] \leq 1 - p_1^k = 1 - \frac{1}{L}$. By independence of g_j for $1 \leq j \leq L$, we have

$$Pr[\forall 1 \leq j \leq L, g_j(x^*) \neq g_j(z)] \leq (1 - \frac{1}{L})^L < \frac{1}{e}$$

Let U be the set of (c, λ)-ANN points, which means $U = \{x \in A : d(x, z) \leq c\lambda\}$. And there are two ways to report x^* is not a (c, λ)-ANN for $x^* \in U$.

(1): None of the (c, λ)-ANN points are hashed into the same bucket with z, which means $\forall j$, $1 \leq j \leq L$, $W_j \cap U = \phi$. Let's denote the event by E_1. Since $x^* \in U$, by Problem (b) we have:

$$Pr[E_1] \leq Pr[x^* \notin \bigcup_{j=1}^{L} W_j] = Pr[\forall 1 \leq j \leq L, g_j(x^*) \neq g_j(z)] \leq \frac{1}{e}$$

(2): There is at least one (c, λ)-ANN point x^* be hashed into one of buckets where z is in, but there are more than $3L$ points at distance greater than $c\lambda$ in the union of the buckets. Since if there are less than $3L$ points at distance greater than $c\lambda$ in $\bigcup_{j=1}^{L} W_j$, the algorithm will return a (c, λ)-ANN. Let's denote this every by E_2. By Problem (a), we know that the event E_2 appears with a probability less than $\frac{1}{3}$

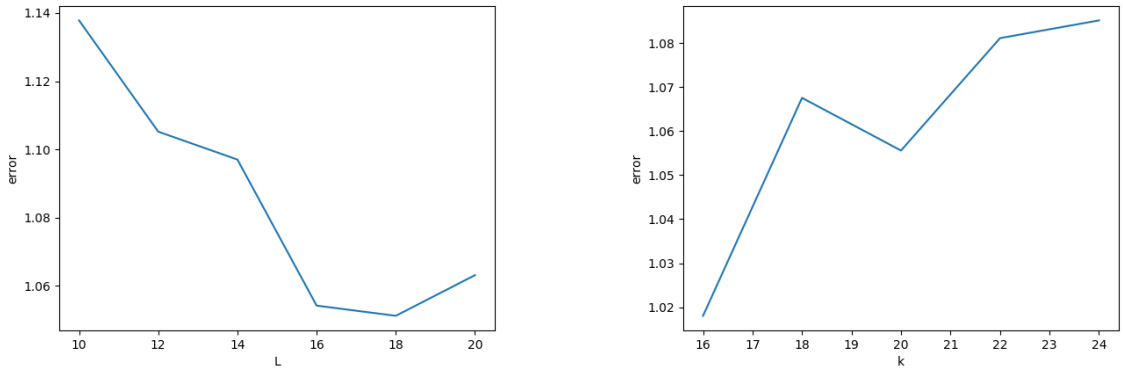
If we denote the probability of the point return by the algorithm is not a (c, λ)-ANN as p', we have $p' = Pr[E_1 \cup E_2] \leq Pr[E_1] + Pr[E_2] \leq \frac{1}{e} + \frac{1}{3}$. Hence, the probability p that the algorithm always return an actual (c, λ)-ANN point with a probability

$$p > 1 - \frac{1}{e} - \frac{1}{3}$$

.

(1) We run the experiments on Macbook Pro OS X Yosemite 10.10.5 2.6 Ghz Intel Core i5 8 GB 1600 MHz DDR3. For each of the image patches in columns 100, 200, 300, ..., 1000, the average search time for LSH is 0.199sec, and the average search time for linear search is 0.512sec.

(2) In figure 1a, we see that error decreases while the L growths. In figure 1b, we see that error increases while the K growths.



(a) error value vs. L

(b) error value vs. K

Figure 1: Error function of L and K

(3) The top 10 nearest neighbors by different methods are showed in the table below. There are 4 common neighbors between LSH(L = 10, k = 24) and linear search, and there are 7 common neighbors between LSH(L = 10, k = 10) and linear search. Like the curve in figure 1b, the error is smaller while k is small. We can briefly see that are a lot of white pixels in the middle of the original image from top left to right bottom. In figure 3, figure 4, and figure 5, we can observe that all the algorithms report images with lots of white pixels. The white pixels in most reported images are also from top left to right bottom or left to right.

Method	Top 10 nearest neighbors
LSH (L = 10, k = 24)	58690, 23633, 37252, 28054, 11947, 23843, 17800, 20468, 55955, 18350
LSH (L = 10, k = 10)	58690, 26168, 38169, 24692, 48596, 37742, 28054, 48783, 19091, 11947
Linear search	58690, 23633, 26168, 38169, 24692, 48596, 37742, 37252, 28054, 15852

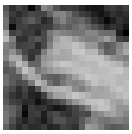


Figure 2: Original image

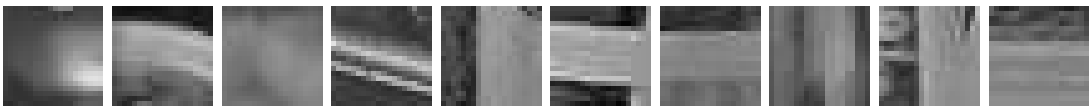


Figure 3: The 10 nearest neighbors by LSH method k = 24, L = 10

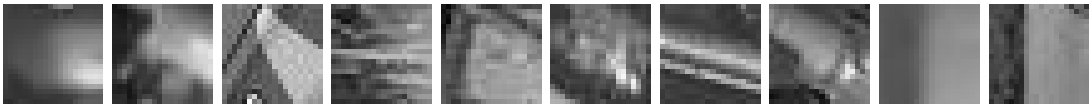


Figure 4: The 10 nearest neighbors by LSH method k = 10, L = 10

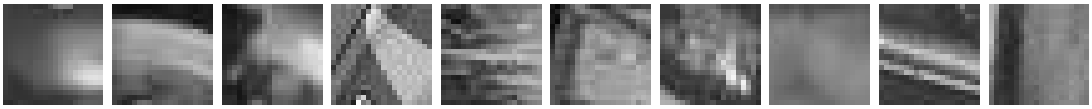


Figure 5: The 10 nearest neighbors by linear search method

Information sheet
CS246: Mining Massive Data Sets

Assignment Submission Fill in and include this information sheet with each of your assignments. This page should be the last page of your submission. Assignments are due at 11:59pm and are always due on a Thursday. All students (SCPD and non-SCPD) must submit their homework via Gradescope (http://www.gradescope.com). Students can typeset or scan their homework. Make sure that you answer each (sub-)question on a separate page. That is, one answer per page regardless of the answer length. Students also need to upload their code on Gradescope. Put all the code for a single question into a single file and upload it.

Late Homework Policy Each student will have a total of *two* late periods. *Homework are due on Thursdays at 11:59pm PT and one late period expires on the following Monday at 11:59pm PT*. Only one late period may be used for an assignment. Any homework received after 11:59pm PT on the Monday following the homework due date will receive no credit. Once these late periods are exhausted, any assignments turned in late will receive no credit.

Honor Code We strongly encourage students to form study groups. Students may discuss and work on homework problems in groups. However, each student must write down their solutions independently, i.e., each student must understand the solution well enough in order to reconstruct it by him/herself. Students should clearly mention the names of all the other students who were part of their discussion group. Using code or solutions obtained from the web (GitHub/Google/previous year's solutions etc.) is considered an honor code violation. We check all the submissions for plagiarism. We take the honor code very seriously and expect students to do the same.

Your name: Yen-Yu Chang

Email: yenyu@stanford.edu **SUID:** 006350488

Discussion Group: Cheng-Min Chiang, Fang-I Hsiao, Alvin Hou

I acknowledge and accept the Honor Code.

(Signed) Y.Y.Chang