I decided to approach this problem using Spark RDD, because of its efficient implementations of the map and filter functions, which are very useful for this problem. The lazy evaluation is also very useful to get the results we need immediately, without waiting for the whole RDD to be processed.

The Spark pipeline I used has the following step, each one leading to a new RDD:

- Load the .txt file on a RDD, so that we can use all the Spark features.
- Split the .txt file in 2 columns, 1 for the user id (int) and 1 for all the friends he has (set).
- For each user, compute the cardinality of the intersection between his set of friends and the set of friends of all the other users.
- Sort those cardinality in descending order, for each user in the RDD.
- Return only the top 10 matches, for each user, those are the recommended friends.

In the code there's also a `sort` *operation on the RDD. That is done in order to get immediately the recommendations for the requested users, in fact if we only need those we can instruct Spark to fetch only the first N elements in a RDD. In this way, thanks to the lazy evaluation, we are able to get the recommendations we need in a couple of seconds, without performing all the computations on the whole RDD*

```
For user 11; I recommend the users [27552, 7785, 27573, 27574, 27589, 27590, 27600, 27617, 27620, 27667]
For user 8997; I recommend the users [8998, 8987, 8992, 9001, 9003, 9009, 4849, 7174, 7279, 7364]
For user 2791; I recommend the users [21185, 8783, 13280, 18359, 18363, 23667, 35740, 2204, 2786, 5996]
For user 4985; I recommend the users [79, 577, 4839, 4984, 4986, 4987, 4988, 4989, 4990, 4991]
For user 8961; I recommend the users [12241, 8973, 8965, 8963, 8966, 8967, 7174, 8969, 12243, 7177]
For user 4049; I recommend the users [4871, 4875, 4889, 8492, 8685, 439, 660, 1100, 1137, 1156]
For user 5060; I recommend the users [5052, 5057, 5086, 14271, 98, 364, 575, 596, 611, 622]
For user 739; I recommend the users [732, 367, 381, 336, 21526, 28064, 677, 704, 728, 736]
For user 1724; I recommend the users [1711, 1663, 1712, 1718, 1662, 1697, 1700, 1715, 1716, 1658]
For user 9550; I recommend the users [9554, 9533, 9544, 9558, 153, 1220, 1421, 1436, 1951, 2413]
For user 3151; I recommend the users [3161, 43162, 3230, 3450, 8692, 161, 2036, 3136, 3137, 3162]
```

Figure 1: Recommendations for the requested users

I have also run the full calculation of all the recommendations for all the users, and obviously, it required a bit more of computation time since we have 50.000 users. **It was 11 minutes**.

Confidence $\text{conf}(A \to B) = \Pr(B|A)$ does not consider the prior probability $\Pr(B)$. This is a drawback because a high confidence value can be misleading if B is a frequent item, regardless of the association between A and B.

- **Lift**:
$$\text{lift}(A \to B) = \frac{\text{conf}(A \to B)}{S(B)},$$
where $S(B) = \frac{\text{Support}(B)}{N}$. Lift takes $\Pr(B)$ into account by comparing the observed co-occurrence of A and B to what would be expected if they were independent, using the value of S(B), to account for high frequent items.

- **Conviction**:
$$\text{conv}(A \to B) = \frac{1 - S(B)}{1 - \text{conf}(A \to B)}.$$
Conviction compares the "expected probability of A occurring without B if A and B were independent" with the "observed frequency of A occurring without B." To do that it uses the S(B) value to scale the metric accordingly to the frequency of B.

In both cases, looking at the formulas, we can see that the measure for **Lift** and **Conviction** decreases when S(B) increases, that makes sense because it accounts for the fact that B might be a very frequent item, regardless of the rest.

A measure is symmetrical if measure$(A \to B) =$ measure$(B \to A)$.

- **Confidence**: Not symmetrical.

$$\text{conf}(A \to B) = \Pr(B|A) \neq \Pr(A|B) = \text{conf}(B \to A).$$

- **Lift**: Symmetrical.

$$\text{lift}(A \to B) = \frac{\text{conf}(A \to B)}{S(B)} = \frac{S(A \cap B)}{S(A) \cdot S(B)} = \frac{\text{conf}(B \to A)}{S(A)} = \text{lift}(B \to A).$$

- **Conviction**: Not symmetrical.

$$\text{conv}(A \to B) = \frac{1 - S(B)}{1 - \text{conf}(A \to B)} \neq \frac{1 - S(A)}{1 - \text{conf}(B \to A)} = \text{conv}(B \to A).$$

Here's an example that shows how Confidence and Conviction are not symmetric Consider the following two sets of transactions:

- $A = \{milk\}$, $B = \{bread\}$
- Let there be 100 transactions in total.
- Suppose 80 transactions contain A (milk), and 40 transactions contain both A and B (milk and bread).
- Then, $\text{conf}(A \to B) = \frac{40}{80} = 0.5$ and $S(B) = \frac{60}{100} = 0.6$.
- Therefore, $\text{conv}(A \to B) = \frac{1-S(B)}{1-\text{conf}(A \to B)} = \frac{1-0.6}{1-0.5} = \frac{0.4}{0.5} = 0.8$.

Now consider the reverse:

- Suppose 60 transactions contain B (bread), and 40 transactions contain both A and B (milk and bread).
- Then, $\text{conf}(B \to A) = \frac{40}{60} = 0.67$ and $S(A) = \frac{80}{100} = 0.8$.
- Therefore, $\text{conv}(B \to A) = \frac{1-S(A)}{1-\text{conf}(B \to A)} = \frac{1-0.8}{1-0.67} = \frac{0.2}{0.33} \approx 0.61$.

Since conv$(A \to B) \neq$ conv$(B \to A)$, conviction is not symmetric.

Since conf$(A \to B) \neq$ conf$(B \to A)$, confidence is not symmetric.

A measure is desirable if it reaches its maximum value for perfect implications ($\text{conf}(A \to B) = 1$).

- **Confidence, desirable**: Reaches its maximum value of 1 for perfect implications. That is because the Confidence is, by definition, the conditional probability and a probability cannot be > 1.
- **Lift, not desirable**: has its maximum value at $+\infty$, but in the case where we have a perfect implication of only 1 transaction $\{A, B\}$ the Lift is only equal to 1. Moreover, we can make the Lift arbitrarily large by making the denominator smaller. That can be obtained by adding a big sequence of random transactions.
- **Conviction, desirable**: has its maximum value at $+\infty$, which is reached when the denominator is equal to 0. That is exactly what happens when $\text{conf}(A \to B) = 1$. That means, when the Confidence approaches 0, the Conviction approaches $+\infty$. At the same time, we could also say that the Conviction is not properly defined when the Confidence is 1, and this is clearly a negative side of it (mathematically it tends to infinity).

  We discard the trivial case in which the numerator is 0 and the supports are null.

```
Rule: DAI93865 --> FRO40251 with Confidence: 1.0
Rule: GRO85051 --> FRO40251 with Confidence: 0.999176276771005
Rule: DAI88079 --> FRO40251 with Confidence: 0.9867256637168141
Rule: FRO92469 --> FRO40251 with Confidence: 0.983510011778563
Rule: DAI43868 --> SNA82528 with Confidence: 0.972972972972973
```

Figure 2: Top 5 rules, 2 items

```
Rule: DAI23334, ELE92920 --> DAI62779 with Confidence: 1.0
Rule: DAI55911, GRO85051 --> FRO40251 with Confidence: 1.0
Rule: DAI75645, GRO85051 --> FRO40251 with Confidence: 1.0
Rule: ELE17451, GRO85051 --> FRO40251 with Confidence: 1.0
Rule: ELE20847, FRO92469 --> FRO40251 with Confidence: 1.0
```

Figure 3: Top 5 rules, 3 items

$$P(\text{ don't know }) = \frac{\binom{N-M}{K}}{\binom{N}{K}} = \frac{\frac{(N-M)!}{K!(N-M-K)!}}{\frac{N!}{K!(N-K)!}} = \frac{(N-M)!(N-K)!}{N!(N-K-M)!}$$

$$= \prod_{i=1}^{M-1} \frac{1}{N-i} \cdot \prod_{i=1}^{M-1} N-K-i = \prod_{i=1}^{M-1} \frac{N-K-i}{N-i} \leq \prod_{i=1}^{M-1} \frac{N-K}{N} = \left(\frac{N-K}{N}\right)^M$$

Q.E.D.

$$\left(\frac{N-K}{N}\right)^M \leq e^{-10} \iff \left(1-\frac{K}{N}\right)^M \leq e^{-10} \iff$$

$$\iff \log\left(\left(1-\frac{K}{N}\right)^M\right) \leq -10 \iff M\log\left(1-\frac{K}{N}\right) \leq -10 \sim$$

$$\sim -M\frac{K}{N} \leq -10 \iff K \geq 10\frac{N}{M}$$

$$Q.E.D.$$

$$S_1 = \{c\}, \quad S_2 = \{a, c\}$$

$Item$	S_1	S_2	
a	0	**1**	
b	0	0	
c	**1**	1	

$Item$	S_1	S_2	
b	0	0	
c	**1**	**1**	
a	0	1	

$Item$	S_1	S_2	
c	**1**	**1**	
a	0	1	
b	0	0	

$$P(\text{Agree}) = \frac{2}{3}, \quad J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{1}{2}$$

$$P(\text{Agree}) \neq J(S_1, S_2)$$

$$Q.E.D.$$

$$P_{2,k} = P_2^{\log_{(1/P_2)}(N)} = \frac{1}{N}$$

$$P\left(\sum_{i=1}^{L} |T \cap W_i| \geq 3L\right) \leq \frac{1}{3} \iff$$

$$\frac{1}{3L} E\left[\sum_{i=1}^{L} |T \cap W_i|\right] \leq \frac{1}{3} \iff$$

$$\sum_{i=1}^{L} E\left[|T \cap W_i|\right] \leq L \iff$$

$$\sum_{i=1}^{L} \frac{1}{N} \cdot |T| \leq L \iff$$

$$\sum_{i=1}^{L} \frac{1}{N} \cdot N \leq L \iff$$

$$L \leq L$$

Q.E.D.

$\mathrm{P}_{1,k} = P_1^{\log_{\frac{1}{P_2}}(N)}, \quad L = N^{\rho}, \quad P = \frac{\log P_1}{\log P_2} \implies P_1 = P_2^{\rho}, \quad | \quad x^{\frac{1}{\log x}} = e$

$\implies \frac{1}{e} \geq \mathbb{P}\left(\forall\, 1 \leq i \leq L,\, g(x_i) \neq g(z_i)\right) = (1 - P_{1,k})^L = \left(1 - P_1^{\log_{\frac{1}{P_2}}(N)}\right)^L = \left(1 - P_1^{\frac{-\log(N)}{\log P_2}}\right)^L =$

$\left(1 - P_2^{\frac{-\rho \log N}{\log P_2}}\right)^L = \left(1 - e^{-\rho \log N}\right)^N = (1 - N^{-\rho})^L = \left(1 - \frac{1}{L}\right)^L$

$$\frac{\partial}{\partial L}\left(1 - \frac{1}{L}\right)^L \geq 0 \quad \forall\, L \geq 1 \quad \text{and} \quad \lim_{L \to +\infty}\left(1 - \frac{1}{L}\right)^L = \frac{1}{e}$$

$$\left(1 - \frac{1}{L}\right)^L \leq \frac{1}{e}$$

Q.E.D.

We position ourselves in the worst possible situation and we try to find a probabilistic bound from there. If we find a theoretical guarantee (fixed constant) in the worst case scenario, we will have a constant guarantee which holds in general.

z is the reference point, for which we want to find a (c, λ)-ANN. From the text, we are assuming to always have one point that is closer to z than λ, which is a valid (c, λ)-ANN.

- Assume we have only one point closer than $c \cdot \lambda$ to z. This point is also closer than λ, since it is the one we must always have (as mentioned in the text).
- Assume that if we can sample the $3L$ points such that all the points are wrong (no valid (c, λ)-ANN), we always do that.

From the previous assumption, we can see that the only way to pick the only good point we have is the following: We must have a scenario where the number of wrong points in the hash buckets matching with z is smaller than $3L$ (probability $\geq \frac{2}{3}$). In this way we are forced to also choose the only good point. Then, we also need that the only good point we have must match at least once with z, so that it's in the same hash bucket (probability $\geq \left(1 - \frac{1}{e}\right)$). Finally, if a good point is the only one in the sampled 3L points (from the buckets) it will be always chosen, since its distance will be the smallest.

Given these assumptions, we can see that even in the worst possible case, we still have a probability larger than $\frac{2}{3} \cdot \left(1 - \frac{1}{e}\right) \sim 0.42$ of extracting an actual (c, λ)-ANN.

Q.E.D.

```
Average LSH search time: 0.157146 s
Average linear search time: 0.425582 s
```

Figure 4: The average time for LSH and Linear Search

Obviously, the LSH is much faster (almost 3 times faster) since it does not need to iterate the whole dataset to check the distance with every point, and finally retrieve the N best match.

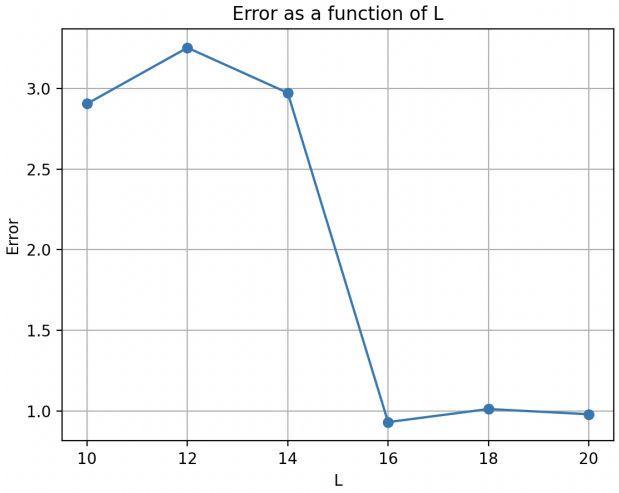


Figure 5: Average error increasing L

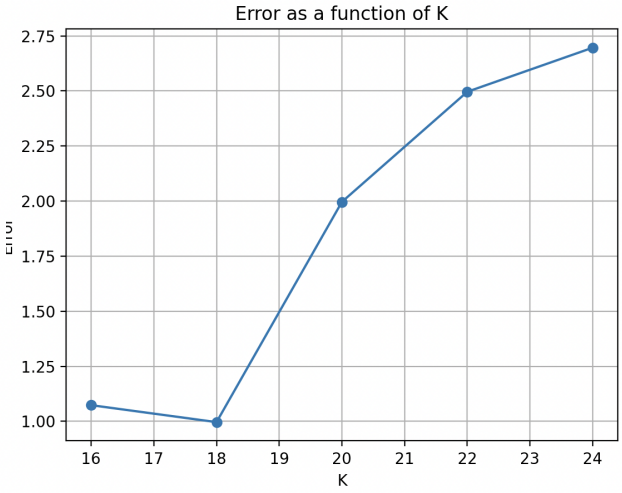


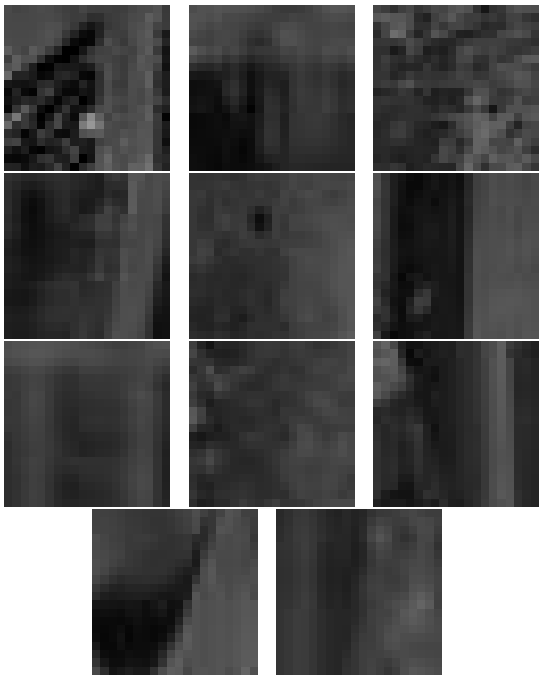
Figure 6: Average error increasing K

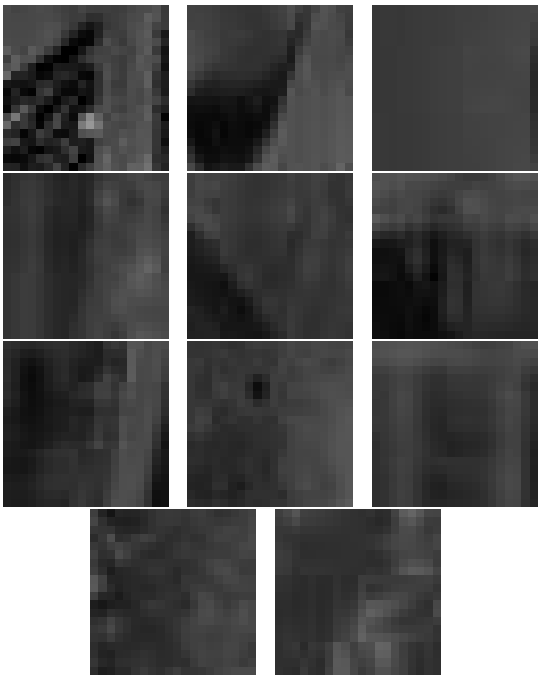
Increasing the L seems to reduce the error. This makes sense, since the probability of correctly identifying the true nearest neighbors increases because there are more opportunities for similar points to be hashed to the same bucket, and then chosen afterwards.

Increasing the K seems to increase the error. This makes sense, since the probability that truly similar points end up in the same bucket decreases. If 2 truly similar points fail to match in all the K hash functions they won't be considered similar, even though they actually are.

In each block of images, the first one (top-left) is the base reference, image 100. The following 10 are the best match found by the 2 algorithms. On the right we have the results of LSH algorithm and on the left the results of the linear search (ground truth).

As expected a lot of them matches, in this case 70% of the images are exactly the same.





As we can see, a lot of images are not similar to the original one (first image of both blocks) and those can be classified as false positives. For example, *image 5* in the left block (linear search) and *image 3* in the right block (LSH) seems to not be related at all with the reference image. Finally, we can observe that LSH was able to retrieve the best images, accordingly to linear search, such as *image 2* and *image 6* in the right block. This means that, as expected, LSH is a very good method for searching similar items, even in significant high dimensional spaces with a lot of noisy images.

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: Davide Ettori _____

Email: detto3@uic.edu _____ **SUID: 655693998**

Discussion Group: _____

I acknowledge and accept the Honor Code.

Davide Ettori _____