CS449 - Project Milestone 2 (Report)

Saoud Akram - SCIPER : 273661

Remark : In this report, the term "movie" and "item" are used interchangeably.

Technical specifications:

- **Model :** Surface Laptop 3
- **Processor :** Intel(R) Core(TM) i7-1065G7U CPU @ 1.30Ghz (8 CPUs), ~1.5GHz
- **RAM :** 16384 MB
- **OS :** Windows 10
- **System type :** x64 based system (64 bit Operating System)
- **Scala Version :** 2.12.13

Part 2.3

1. Compute the prediction accuracy. Report the result. Compute the difference between the Adjusted Cosine similarity and the baseline (cosine-baseline). Is the prediction accuracy better or worse than the baseline ?

The MAE we get using the cosine similarities is approximately 0.7478, which is lower than the baseline MAE. The difference between the cosine MAE and the baseline MAE is around -0.01913 (i.e. the difference is negative), which means that the prediction accuracy using the adjusted cosine similarities is better than the baseline.

2. Implement the Jaccard Coefficient. Provide the mathematical formulation of your similarity metric in your report. Compute the prediction accuracy and report the result. Compute the difference between the Jaccard Coefficient and the Adjusted Cosine similarity Is the Jaccard Coefficient better or worse than Adjusted Cosine similarity?

The Jaccard Similarity coefficient between two sets is given by : $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ (where $J(A,B) = 0$ if $|A| = 0$ or $|B| = 0$ (or both))

In our case, to compute the similarity of user u and v, we take $A = I(u)$ and $B = I(v)$.

The MAE we get using the Jaccard similarity is 0.7625, which is slightly higher than the Adjusted Cosine MAE (we observe a difference of approximately 0.0148), which means that the Adjusted Cosine Similarity has a better prediction accuracy. However, it is still has a slightly better prediction accuracy when compared to the baseline.

3. In the worst case and for any dataset, how many $s_{u,v}$ computations, as a function of the size of U (the set of users), have to be performed? How many would that represent for the 'ml-100k' dataset?

In the worst case, $s_{u,v} \neq 0 \; \forall u, v \in U$. As such, the number of computations is the number of distinct (u, v) pairs.

An important observation to make is that $s_{u,v} = s_{v,u}$, by definition. As such, when counting the number of distinct (u, v) pairs, we consider a pair (u, v) to be the same as a pair (v, u) (informally, we can say that the order does not matter).

There is also no need to compute $s_{u,u}$, as we will never use it (i.e. we will omit the pairs of the form (u, u) when counting all distinct pairs). Why ? Consider the following. Let a user-item pair be denoted as (u^T, i^T) if it is in the training set and (u^t, i^t) if it is in the test set. As we know, there is no user-item pair which is both in the training set and in the test set at the same time. To compute the prediction for a given pair (u^t, i^t) in the test set, we first need to compute the user-specific weighted-sum deviation for item i^t, which is given by : $\hat{\bar{r}}_{\bullet,i^t}(u^t) = \frac{\sum_{v^T \in U(i^t)} s_{u^t,v^T} * \hat{r}_{v^T,i^t}}{\sum_{v^T \in U(i^t)} |s_{u^t,v^T}|}$. If we assume that we need s_{u^t,u^t} to compute the sum in the numerator (or in the denominator), then we assume that there exists $v^T \in U(i^t)$ such that $v^T = u^t$. In plain English, this means that, that we assume that u^t is in the set of users in the training set that have seen movie i^t. However, we previously stated that for all user-item pair in the test set, the same pair does not exist in the training set (informally, this means that, if we need s_{u^t,u^t}, then (u^t, i^t) must be in the training set, which isn't possible so we have a contradiction). Therefore, we can conclude that we don't need $s_{u,u}$ as we will never use it to compute the user-specific weighted-sum deviation for an item.

Both these observations point to a combination without repetition. As such, the number of $s_{u,v}$ computations required is given by : $C\binom{n}{2} = \frac{n(n-1)}{2}$ where $n = |U|$

For the ml-100k dataset, the number of users is 943 and as such, "in the worst case", the number of similarities to be computed is $C\binom{943}{2}$ which gives us 444153 .

Remark: Given that, given our training set, there exists zero similarities (i.e. pair of users such that $s_{u,v} = 0$). We don't compute zero similarities and as such, the total number of $s_{u,v}$ computations for our execution is 411546, which is lower than in "the worst case".

4. Compute the minimum number of multiplications required for each possible $s_{u,v}$ on the ml-100k/u1.base. What are the min, max, average, and standard deviation of the number of multiplications? Report those in a table.

min	max	average	stddev
1.0	332.0	13.089	18.172

Remark: Note that we say here that the minimal number of multiplication required to compute each $s_{u,v}$ is 1. However, here, we only took into account non-zero $s_{u,v}$. For a zero $s_{u,v}$, the number of multiplications required is, of course, 0.

5. How much memory, as a function of the size of U, is required to store all possible $s_{u,v}$ (both zero and non-zero values), assuming both require the same amount of memory? How many bytes are needed to store only the non-zero $s_{u,v}$ on the 'ml-100k' dataset, assuming each non-zero $s_{u,v}$ is stored as a double (64-bit floating point value)?

Note that 64 bits is 8 bytes.

For all possible $s_{u,v}$ (i.e. both zero and non-zero values), the total amount of memory required in bytes is given by $C\binom{n}{2} * 8$, where $n = |U|$. If we were to compute all $s_{u,v}$ in the ml-100k dataset, then we would require $C\binom{943}{2} * 8 = 3553224$ bytes of memory, which is around 3.4 Megabytes.

For only non-zero $s_{u,v}$ on the ml-100k dataset, we require 3292368 bytes, which is around 3.15 Megabytes

We gain around 0.25 Megabytes by not computing zero $s_{u,v}$

6. Measure the time required for computing predictions including computing the similarities $s_{u,v}$. Provide the min, max, average, and standard-deviation over five measurements. Discuss in your report whether the average is higher than the previous methods you measured in Milestone 1 (Q.3.1.5)? If this is so, discuss why.

min [s]	max [s]	average [s]	stddev [s]
42.0469	49.2056	46.2495	2.6098

(Please note that the timings were converted into seconds and rounded to the 4th decimal, as it allows for better readability. For more precise answers, please consult the appropriate .json file which contains the exact timings in μs)

We can observe that, on average, it takes around 46 seconds to compute the predictions using the adjusted cosine similarities.

Comparing to the baseline method, which takes around 4 seconds to compute the predictions, this adjusted cosine method is 12x slower.

This is due to several facts.

First of all, we need to compute all similarities between each pair of users. As we have seen previously, there are 411546 non-zero $s_{u,v}$ to compute. We will see this in details question 2.3.7, but this already takes a few seconds to execute.

After computing the $s_{u,v}$, we do `leftOuterJoin` operation in order to compute the user-specific weighted-sum deviation for each item in the test set. This operation is **extremely** slow (which makes sense since there are 411546 distinct $s_{u,v}$). One way to get around this problem would be to compute each $s_{u,v}$ on a *by-need* basis (i.e. instead of computing all non-zero $s_{u,v}$ first and then perform a `leftOuterJoin` operation in order to compute the the user-specific weighted-sum deviation for each item in the test set, we would only compute the $s_{u,v}$ as we need them). More exploration is required.

Remark: Our attempt at making this `leftOuterJoin` operation faster was to make sure that we do not compute "unnecessary" similarities. This is the reason why, when computing all similarities, we didn't compute those of the form $s_{u,u}$ (because, as seen previously, we never use them). We also made sure that if we compute $s_{u,v}$ we do not compute $s_{v,u}$ (as $s_{u,v} = s_{v,u}$).

7. Measure only the time for computing similarities. Provide the min, max, average and standard deviation over ten five measurements. What is the average time per $s_{u,v}$ in microseconds? On average, what is the ratio between the computation of similarities and the total time required to make predictions? Are the computation of similarities significant for predictions?

min [s]	max [s]	average [s]	stddev [s]	time per $s_{u,v}$ [μs]	ratio
8.8246	10.7429	9.6011	0.632	23.3293	0.2076

(Please note that the timings were converted into seconds (except for the "time per $s_{u,v}$", which we kept in μs) and rounded to the 4th decimal, as it allows for better readability. For more precise answers, please consult the appropriate .json file which contains the exact timings in μs)

It takes approximately 9.6 seconds to compute all similarities between users. Just the act of computing the similarities takes twice as much time as is required to compute the entire baseline MAE (which, as mentioned earlier, takes slightly less than 4 seconds to obtain on average). We can therefore see that the similarity method, although slightly more accurate than the baseline MAE, comes at a high computational cost.

We can see that, on average, it takes 23 μs for each $s_{u,v}$, which is a lot considering that we have to compute exactly 411546 non-zero $s_{u,v}$.

We can also observe that computing the similarities represents around 20% of the total computational time required to make predictions. Whilst we expected a higher ratio, this means that actually computing the $s_{u,v}$ is not the most time consuming task when computing the predictions using the adjusted cosine similarities. It is the `leftOuterJoin` operation which we perform right after that takes up most of the time. Again, as mentioned earlier, there are ways to get around this `leftOuterJoin` operation, however, we didn't explore them as we were sufficiently contempt with out current implementation.

Part 3.1

1. What is the impact of varying k on the prediction accuracy? Provide the MAE (on ml-100k/u1.test) for k = 10, 30, 50, 100, 200, 300, 400, 800, 943. What is the lowest k such that the MAE is lower than for the baseline method ? How much lower?

k =10	k=30	k=50	k=100	k=200	k=300	k=400	k=800	k=943	Lowest k	MAE Diff
0.8449	0.7976	0.7778	0.7582	0.7499	0.7479	0.7474	0.7477	0.7478	100	-0.008650

Recall that the baseline MAE is approximately 0.7669.

Up until $k = 400$, we can observe that the higher k is, the lower the MAE (and thus, the higher the prediction accuracy becomes). However, we can see that the MAEs for $k = 800$ and $k = 943$ are each slightly higher than the MAE for $k = 400$. Does that mean the $k = 400$ must be the "go-to" k which we have to use when computing predictions on all datasets ? No. Since we didn't do any cross-validation before, all we observe here is that, for this particular split of the dataset, we get the lowest MAE (and the highest prediction accuracy) when $k = 400$. In other words, this means that $k = 400$ might not be the most optimal choice for another test set.

We can that the lowest k for which the MAE is lower than the baseline MAE is $k = 100$, with a difference of -0.0087 approximately ($lowestk - basline$) . This proves to us that we do not need to keep all similarities to get a better MAE than the baseline. In our case, only keeping the 100 most significant similarities for each user was sufficient to get a better result.

Remark: Notice that the MAE we get for $k = 943$ is exactly the same as the MAE we computed for question 2.3.1. This makes sense since $k = |U| = 943$ (in other words, we take into account all similarities, just like we did in part 2.3.1).

Remark: In order to get those results, we considered that a user **cannot** be its own neighbour. Why ? Consider the following. Let's assume that a user can be its own neighbour. We know that $\forall u \in U$, $s_{u,u} = 1$ (which is the maximum possible similarity level). As such, when picking the top k similarities, $s_{u,u}$ will ALWAYS be picked (as a user can be its own neighbour). However, as we mentioned in question 2.3.3, $s_{u,u}$ is never used to make a prediction. As such, we end up picking (and storing) $k - 1$ "useful" similarities and 1 non-useful similarity per user, which, from our point of view makes no sense (it's better to pick and store k useful similarities as, for the same amount of memory, we end up having better results).

2. What is the minimum number of bytes required, as a function of the size of U, to store only the k nearest similarity values for all possible users u, i.e. top k $s_{u,v}$ for every u, for all previous values of k (with the ml-100k dataset)? Assume an ideal implementation that stored only similarity values with a double (64-bit floating point value) and did not use extra memory for the containing data structures (this represents a lower bound on memory usage). Provide the formula in the report. Compute the number of bytes for each value of k in your code.

Let $|U| = n$

Recall that 64 bits is 8 bytes.

Assuming each user u has at least k neighbours (i.e. other users v such that $s_{u,v} \neq 0$), then the minimal number of bytes required to store all similarities is given by: $n * k * 8$

This makes sense there are k similarities per user and each similarity takes 8 bytes to be stored.

Using this formula, the theoretical results we should get are the following :

k=10	k=30	k=50	k=100	k=200	k=300	k=400	k=800	k=943
75440	226320	377200	754400	1508800	2263200	3017600	6035200	7113992

The real results we get for each k are the following:

k=10	k=30	k=50	k=100	k=200	k=300	k=400	k=800	k=943
75440	226320	377200	754400	1508800	2263200	3016888	5950040	6584736

Our formula works $k \in \{10, 30, 50, 100, 200, 300\}$, however it doesn't work for $k \in \{400, 800, 943\}$. This is due to the fact that our assumption (i.e. the fact that each user has at least k neighbours) doesn't hold for such large k's. As a matter of fact, when taking the top k similarities for each user, we used the `take` method. If a user has p neighbours such that $p < k$, then the `take` method will return p similarities, which explains why the real results for these k's are lower.

Remark: It is important here to notice that, for this question, we didn't use our previous reasoning regarding the number of similarities. Typically, we said in question 2.3.3 that it was not necessary to compute both $s_{u,v}$ and $s_{v,u}$ as they are equal. However, to find the top k $s_{u,v}$ for each user u, it was too confusing for us to not compute both $s_{u,v}$ and $s_{v,u}$, so we ended up computing both (even though they are the same). This also allowed us to check our previous result (i.e. the adjusted cosine similarity MAE in question 2.3.1) when $k = 943$ (since the MAE for both implementation are the same, we're good). Since we compute both $s_{u,v}$ and $s_{v,u}$ we need to store each of them individually (so we take twice as much space). As such, the result we get for question 2.3.5 is also different compared to the result we get for question 3.2.2. In fact, the amount of space we need for non-zero similarities if we were to compute both $s_{u,v}$ and $s_{v,u}$ for all $u, v \in U$ is 6584736 Bytes.

3. Provide the RAM available in your laptop. Given the lowest k you have provided in Q.3.1.1, what is the maximum number of users you could store in RAM? Only count the similarity values, and assume you were storing values in a simple sparse matrix implementation that used 3x the memory

As mentioned in the first page, our machine has 16384 MB of RAM, which is 17179869184 Bytes.

As such, to find the maximum number of users that can fit in our RAM, we compute the following :

$n_{max} * k_{lowest} * 8 * 3 \approx 17179869184$

$\Leftrightarrow n_{max} = \text{floor}(\frac{17179869184}{k_{lowest} * 8 * 3})$ = 7,158,278

Remark: Here, we are assuming that we store exactly k_{lowest} similarities per user.

4. Does varying k has an impact on the number of similarity values $s_{u,v}$ to compute, to obtain the exact k nearest neighbours? If so, which? Provide the answer in your report.

No. Regardless of the value of k, we need to compute all possible similarities for a user u and then select the top k similarities. As such, we need to compute all $s_{u,v}$ before obtaining the exact k nearest neighbours.

Note that we could also follow a greedy approach in which we take similarities that are "close enough", but that doesn't always yield the exact k nearest neighbours.

5. Report your personal top 5 recommendations with the neighbourhood predictor with k = 30 and k = 300. How much do they differ between the two different values of k? How much do they differ from those of the previous Milestone?

Let's look at the results

For k = 30:

Movie id	Title	Prediction score
59	Three Colors: Red (1994)	5.0
169	Wrong Trousers	5.0
198	Nikita (La Femme Nikita) (1990)	5.0
206	Akira (1988)	5.0
272	Good Will Hunting (1997)	5.0

For k = 300:

Movie id	Title	Prediction score
814	Great Day in Harlem	5.0
848	Murder	5.0
868	Hearts and Minds (1996)	5.0
909	Dangerous Beauty (1998)	5.0
1144	Quiet Room	5.0

When it comes to $k = 30$, I have already seen the first and last movie, which I liked. However, I've heard about the 2nd movie and it doesn't seem like the kind of movie I'd watch. I know "Nikita" is a movie from a director I like (i.e. Luc Besson), but I haven't watched it yet. When it comes to Akira, it's been in my *to-watch list* for a long time as I know this movie has been a huge inspiration for Star Wars, which I am a great fan of.

For $k = 300$, the predictions aren't the same at all. The first movie seems to be a documentary, so there's little chance that I will watch it. I never heard of the 2nd movie, but I however it definitely seems like my type of movie, so I added it to my *to-watch list*. I've also never heard of "Hearts and Minds" nor "Quiet Room" and it doesn't seem like the kind of movie I would watch. Finally, I have seen dangerous beauty, but I didn't like it a lot.

Comparing both results, it seems that both methods yielded very different results, with $k = 30$ yielding more accurate results for me.

Let's now look at what we got using the baseline method in Milestone 1.

Movie id	Title	Prediction score
814	Great Day in Harlem	5.0
1122	They Made Me a Criminal (1939)	5.0
1189	Prefontaine (1997)	5.0
1201	Marlene Dietrich: Shadow and Light (1996)	5.0
1293	Star Kid (1997)	5.0

The only movie that is recommended by both the baseline and knn method (i.e. when k = 300) is "Great Day in Harlem". As mentioned earlier, this is a documentary, so there is little to no chance that I will take the time watching it.

Remark : Although it would seem that k = 30 yields better results for me, this does not mean that for all other users k = 30 will recommend better movies. Besides, it is important to keep in mind that we had to pick the 5 best movies according to the highest prediction score and the lowest index for reproducibility (this infers on which movies are recommended in the end).

Remark: In Milestone 1, for the bonus, we tried giving a higher weight to movies that have a higher number of ratings using IMDB's weighted rating formula. Let's look at the results if we try this with the knn method.

For $k = 30$:

Movie id	Title	Prediction score
302	L.A. Confidential (1997)	4.869300034
272	Good Will Hunting (1997)	4.812548733
327	Cop Land (1997)	4.791517615
310	Rainmaker	4.755777778
482	Some Like It Hot (1959)	4.729500703

For $k = 300$:

Movie id	Title	Prediction score
174	Raiders of the Lost Ark (1981)	4.376610481
496	It's a Wonderful Life (1946)	4.222087704
22	Braveheart (1995)	4.195365169
173	Princess Bride	4.130947956
963	Some Folks Call It a Sling Blade (1993)	4.110059651

That seems a lot more accurate.

For $k = 30$, I've watched and liked all movies, except for "Some Like it Hot"

For $k = 300$, I've watched and liked all movies, expect for "It's a Wonderful Life" and "Princess Bride"

Note that I will not be including these results in the recommendations.json as to not interfere with the automatic grading system, however, I added a `println()` *in my code to display those results (so if you run my code, you should see them).*