Anonymization and Privacy Project

Vincent HENRIC and Lucas PLUVINAGE

March 2020

1 Introduction

In this project, we used the paper *Robust Deanonymization of large sparse datasets* [1] as a reference. We tried, first, to reproduce the main results of the paper; then to extend these results. We faced many challenges. One of paramount importance was the need to work with a very big dataset: our raw dataset is of size no less than 2.6 GB, and this does not take into account the joins we need to make. In this report, we describe the different models we put in place, as well as the results. We will also describe the main ideas about the way we overcome the computational challenges. For more details about the code, and how to reproduce the results, make sure to check the Readme of the corresponding Github repository: the code, the results and the images are freely accessible at the following repository: [2].

2 Reproduction of Results

In order to reproduce the main results of the article, we developed two implementations. The first mainly used the *Pandas* library but was not memory-efficient enough to handle the full experiments, was not continued with success for the extensions. Instead of optimizing the algorithm manually, we took advantage of *PySpark* to create a second, distributed, implementation. Even when executed on a single machine, this implementation has proven to be useful as Spark has various mechanisms to improve the overall execution of the algorithm. This includes caching (in-memory/in-disk) and multi-tasking.

The implementation reproduces what has been done to test the Netflix de-anonymisation algorithm. First it samples $N = 1000$ customers. Then it creates auxiliary data for each customer according to the auxiliary specification: number of correct/false ratings, and precision in the rating/date of rating. After that, the scoreboard-RH algorithm is applied, producing a matching score for each customer pair. If entropic de-anonymisation is requested, it derives a probability distribution from the scores and computes the resulting entropy. If best-guess de-anonymisation is requested, the eccentricity measure is returned with the best-guess. This allows to fine-tune the eccentricity threshold afterwards.

2.1 Best-guess de-anonymisation

Best-guess de-anonymisation has been performed to reproduce the **Figure 4.** of the article. For each experiment, 1000 customers are randomly selected, then auxiliary data is generated according to the experiment specification. We tested de-anonymisation for 2, 3-out-of-4 and 6-out-of-8 correct ratings auxiliary data, with exact ratings and 3/14 days margin on the date. On a 4-core i7 processor, loading the dataset takes 5 minutes, then each experiment takes 4 to 10 minutes and spills more than to 15GB of data on the disk. There's clearly room for optimization but at least it's runnable and has the benefit of being distributable. In particular the time-memory tradeoff could be fine-tuned a bit more.

With an eccentricity threshold of 1.5 as prescribed by the article, the following results are obtained:

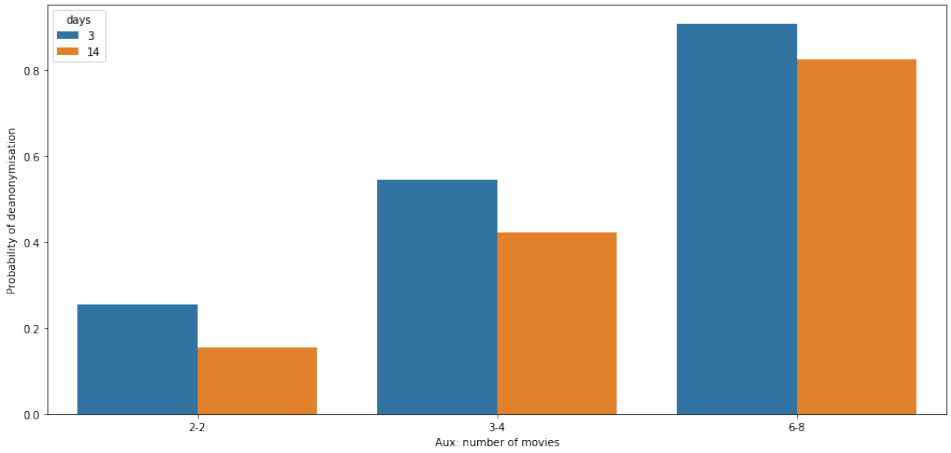

Figure 1: Best-guess de-anonymisation probabilities (N=1000, threshold=1.5)

These results are coherent but remain not as good as what the article claimed to achieve, especially when information is low (2 ratings auxiliary information). Something very critical in this algorithm which is the selection of the threshold. Looking at the ROC curve allows use to see the repartition of true positive rate and false positive rate when choosing the threshold. We can see that the 6-of-8 movies experiment performed very well.

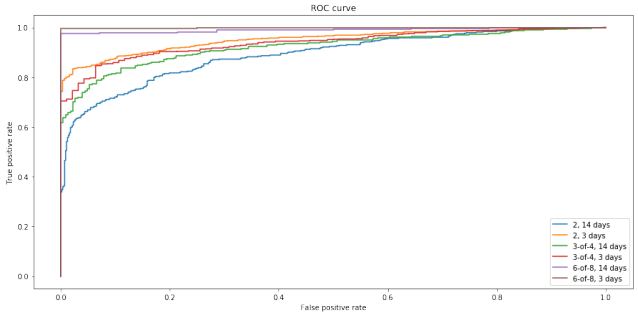

Figure 2: ROC curve of experiments

But something is off: the threshold should be chosen to make false positive rate and false negative rate equal. It's set to 1.5 in the article, but optimising it empirically to make the error rates equal yields a much lower value. For the 2 movies, 3 days precision experiment, the threshold value should be of *0.13*.

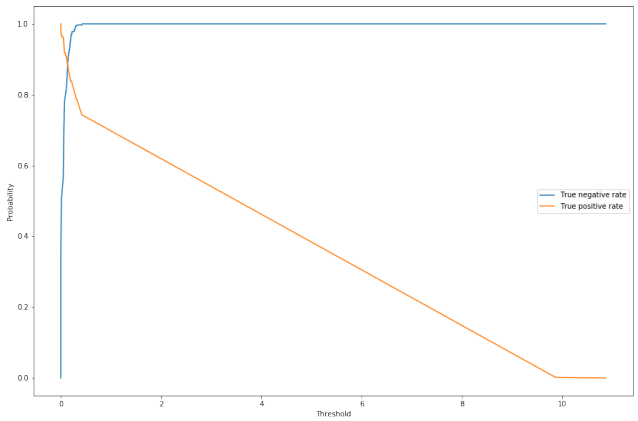

Figure 3: Threshold optimization. The two curves meet at the optimal threshold.

When applying the new threshold results are quite better and approach what has been claimed by the article. In particular, we obtain a **97.7%** de-anonymisation rate for 6-out-of-8 correct ratings with 14 days error margin.

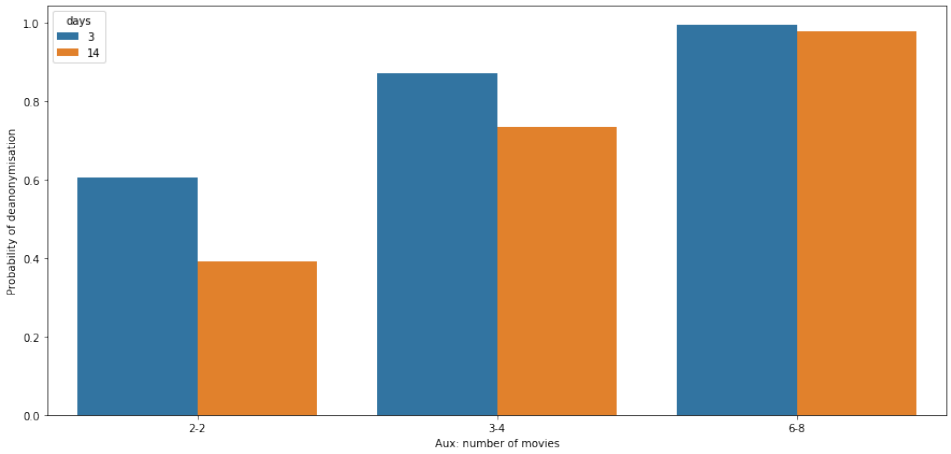

Figure 4: Best-guess de-anonymisation probability (N=1000, threshold=0.13)

2.2 Entropic de-anonymisation

Entropy results have also been measured in the same fashion. They are not as good as the article be remain somewhat coherent as entropy gain increases with the amount of auxiliary data. The maximum achieved entropy gain is of **9.3 bits**. It's quite low compared to what has been presented in the article, meaning that the confidence in the resulting probabilities is not that high.

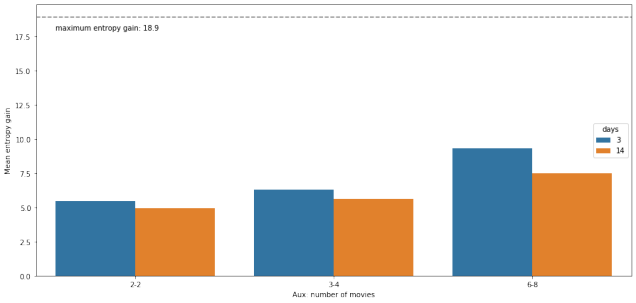

Figure 5: Entropy gain (N=1000)

3 Extensions

In the conclusion of the article, the authors mention a way Netflix could have made the de-anonymization more difficult: by simpling not releasing the movie titles. In this case, an auxiliary information, corresponding to ratings and approximate dates for a person and for a given movie, are more difficult to exploit. Indeed, we do not know anymore to which movie identifier the auxiliary information is related to. In this situation, we need to break anonymity along two dimensions: the customer dimension and the movie dimension. Note that it would impact the utility of the released dataset only if one aim for content-based filtering; for collaborative filtering, the movie title is not needed, and we have not utility loss.

This makes the problem much more challenging. One of the main arguments that explains why the algorithm of the paper works in the first place is the sparsity of the dataset. Now that we cannot attribute an auxiliary information to a movie identifier, to some extent, we

lose the sparsity argument, because many customers have seen a movie with a given rating at a given date; what is a rare event is a customer to have seen a specific movie (with a given identifier) with a given rating at a given date.

In the dataset, the dates are on a range of 2242 consecutive days, with five different ratings possible. That makes $2242 \times 5 = 11210$ possible values for the couple (rating, date). On a dataset that has more than 100 million reviews (100,480,507 to be more precise), we might have around 9,000 reviews on the same (rating, date) on average, whatever the movie. In the original task, as we had 17,770 available movies, it was much rarer to have a given (movie id, rating, date), because such combinations would exist on average 0.5 times. Even if we can conceive that the repartition of such tuples are not uniform, the order of magnitude is enlightening.

In this section, we will investigate several arguments, theoretical and computational, to understand better the challenges of this extra assumption of not having access to the movie titles in the Netflix dataset.

3.1 Probability calculations

We want to quantify precisely, for a given auxiliary information on s movies, what is the probability that a given customer has a movie review close enough to each auxiliary information. The term "close enough" is expressed as a number of accepted values for the tuple (rating, date), in other words a tolerance in the mistakes we can make on the rating and the date. For example, suppose the tolerance is of 0 for the rating, and of a range of 14 days around the real date. The probability that a uniform sample of (rating, date) falls within the union of the 29 single events(r, d-14),(r, d-13),..,(r, d+14) out of 11210 events (where (r,d) is the rating, date of an auxiliary information), will be $p = \frac{29}{11210} \approx 0.003$. We do not take into account the effect of the edges here, and we will continue with this hypothesis. In the following section, we will always refer to p as the probability of a given review to fall into the neighborhood/tolerance margin of a given auxiliary information.

Now that we have the probability for one movie review to be close to a given auxiliary information, can we quantify the probability that a given customer, who has n reviews, has at least one record which is close enough to a given auxiliary information ?

$p_n = \mathbb{P}$(at least one record out of n is in the neighborhood of a given auxiliary record)
$p_n = 1 - (1-p)^n$

Now, suppose we have s such auxiliary information. The neighborhoods of 2 auxiliary records can overlap or not. We will consider two bounding estimates of the probability:

1. the neighborhoods are identical (that is the s auxiliary records have the same ratings and dates
2. the neighborhoods have an empty intersection two by two (that is, they are completely distinct)

These two situations enable to calculate an interval for the probabilities. In the former scenario, we tend to underestimate the probability for a given customer to have different movie reviews in the neighborhoods of each of the auxiliary records; in the latter, we overestimate the probability. For a narrow neighborhood, one can consider the second hypothesis to be more realistic. From now on, we will denote X_n the random variable related to the event that each auxiliary record has at least one customer review in its neighborhood.

For the first hypothesis, it is clear that we need to use the cumulative distribution of the binomial distribution. The probability that exactly s out of n reviews of a customer are in the neighborhood is $\sim \mathcal{B}inom(s, n)$, so the probability that we have at least s reviews is

$$\mathbb{P}(X_n \geq s) = \sum_{k=s}^{n} \binom{n}{k} p^k (1-p)^{n-k}$$

For the second hypothesis, the calculation is much more complicated. We can show by induction on the number of auxiliary records s that the formula is

$$p_{s,n} = \sum_{k=0}^{s} \binom{s}{k} (1-kp)^n (-1)^k$$

proof

For $s = 0$, the probability is one, and the result is trivial.

To get a sense of the formula, let's demonstrate for $s = 1$. It suffices to observe that the result is the probability $p_{1,n} = p_n$ defined above.

Now, suppose that the formula holds for $s-1$, where $s \geq 1$. We examine each of the customer reviews in a given order (the number of reviews is finite). We can consider that the first review that is inside the neighborhood of any information record is the kth, for k in $0, .., n$. The event is satisfied if we can find at least one customer review in the neighborhood of each of the $s-1$ remaining auxiliary record, among the $n-k$ remaining reviews. The probability that the first review inside one of the neighborhood is the kth is:

$$(1-sp)^{k-1} sp$$

because the s neighborhoods are disjoints, so the probability to get inside one of the neighborhood is the sum, which yields a probability sp. That is the key difference with the first hypothesis: the auxiliary records' neighborhoods have overall a bigger coverage, which increases the probability.

The choice of k gives a set of disjoint events, so we sum on the different values for k, and the formula is:

$$p_{s,n} = \sum_{k=0}^{n} (1-sp)^{k-1} sp p_{s-1,n-k}$$

We simplify this expression, using the formula for $p_{s-1,n-k}$, which is part of the induction hypothesis.

$$p_{s,n} = \sum_{k=1}^{n} (1-sp)^{k-1} sp \sum_{l=0}^{s-1} \binom{s-1}{l} (1-lp)^{n-k} (-1)^l$$

$$p_{s,n} = \sum_{l=0}^{s-1} \binom{s-1}{l} (-1)^l sp \sum_{k=1}^{n} (1-sp)^{k-1} (1-lp)^{n-k}$$

The right factor can be calculated using the formula:

$$a^n - b^n = (a-b) \sum_{k=0}^{n-1} a^{n-1-k} b^k = (a-b) \sum_{k=1}^{n} a^{n-k} b^{k-1}$$

We obtain:

$$\sum_{k=1}^{n} (1-sp)^{k-1} (1-lp)^{n-k} = \sum_{k=0}^{n-1} (1-sp)^k (1-lp)^{n-1-k}$$

$$\sum_{k=1}^{n} (1-sp)^{k-1} (1-lp)^{n-k} = \frac{1}{p(s-l)} ((1-lp)^n - (1-sp)^n)$$

We plug in this simplification, we get:

$$p_{s,n} = \sum_{l=0}^{s-1} \binom{s-1}{l} (-1)^l sp \frac{1}{p(s-l)} ((1-lp)^n - (1-sp)^n)$$
$$p_{s,n} = \sum_{l=0}^{s-1} \binom{s-1}{l} \frac{s}{(s-l)} (-1)^l ((1-lp)^n - (1-sp)^n)$$
$$p_{s,n} = \sum_{l=0}^{s-1} \binom{s}{l} (-1)^l ((1-lp)^n - (1-sp)^n)$$

We then simplify the term on the right:

$$\sum_{l=0}^{s-1} \binom{s}{l} (-1)^l (1-sp)^n = \sum_{l=0}^{s} \binom{s}{l} (-1)^l (1-sp)^n - (-1)^s \binom{s}{s} (1-sp)^n$$
$$\sum_{l=0}^{s-1} \binom{s}{l} (-1)^l (1-sp)^n = 0 - (-1)^s \binom{s}{s} (1-sp)^n$$

Finally, we obtain that:

$$p_{s,n} = \sum_{l=0}^{s-1} (-1)^l \binom{s}{l} (1-lp)^n + (-1)^s \binom{s}{s} (1-sp)^n$$
$$p_{s,n} = \sum_{l=0}^{s} \binom{s}{l} (1-lp)^n (-1)^l$$

And the demonstration is finished.

3.2 Illustrations

Now that we have the formulae, what are the actual probabilities for the Netflix dataset ? To simplify, we still suppose that the distribution of ratings and dates is uniform in the dataset.

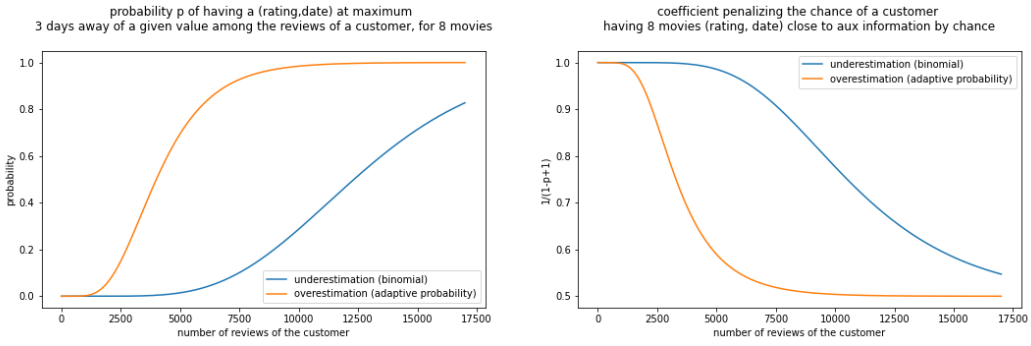

Figure 6: Probabilities and penalties put in place

We can see (on the left plot) that the probability, for a customer with a given number of reviews can be fairly high. We propose to put a penalty coefficient inversely proportional to this probability (on the right plot), in the same manner as the coefficient $wt(i)$ is used in the paper (more details in next sections).

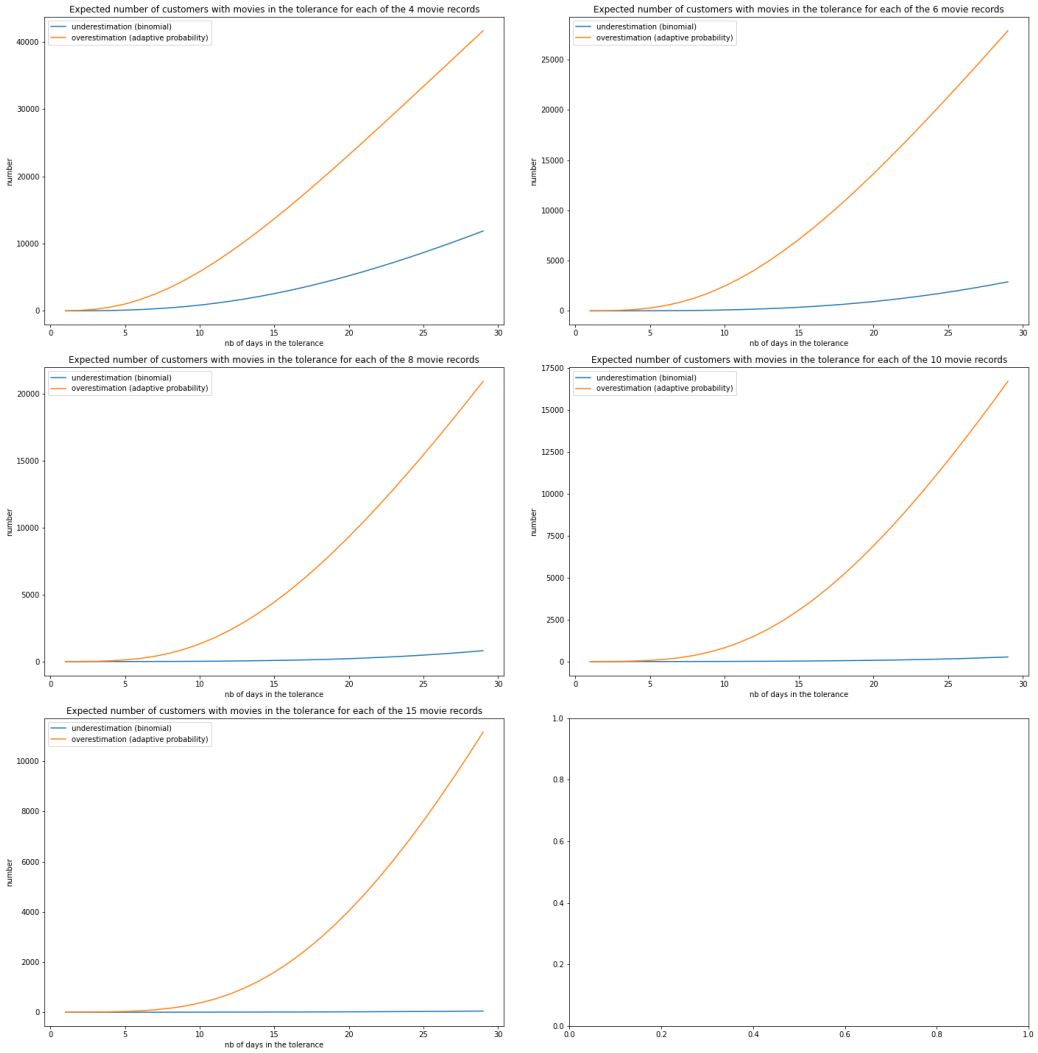

Figure 7: Expected number of customers with reviews in the neighborhood of each auxiliary records, for a number of respectively 4,6,8,10,15 auxiliary records, as a function of the number of days in the neighborhood

In this situation, it is easy to see that there is a high risk to find other customers with the exact same records, for different movies. In figure 7, we see the average number of customers that have reviews in the neighborhood of the auxiliary records. With 15 days of tolerance (7 on both sides of the real date), with 10 auxiliary records, we can still have around 2500 customers that might fulfill the conditions, thus being in competition with the customer we want to find. Clearly, we need to have enough auxiliary information on the one hand, and on the other hand small uncertainty on the dates and ratings, if we want to rightly guess the target of the de-anonymization task.

The following plot show the same results for the case where we have the movie titles in the Netflix released dataset. The probability of a single event (movie id, rating, date) this time is $p \approx 5e^{-9}$, compared to $p \approx 9e^{-5}$, a ratio of 17770, the number of movies. As we can see, finding a customer with reviews in the neighborhood of all the auxiliary records is much less likely, and the expected number of such customers is very small. It gives rooms for more robustness, including being able to de-anonymize the dataset while allowing mistakes in the auxiliary information. Note that the probabilities are so small that we encounter numerical instability from a number of three auxiliary records. For one auxiliary record, the lines are superposing. All the illustrations can be found in the Github repository, in the images folder, or the notebook on data exploration.

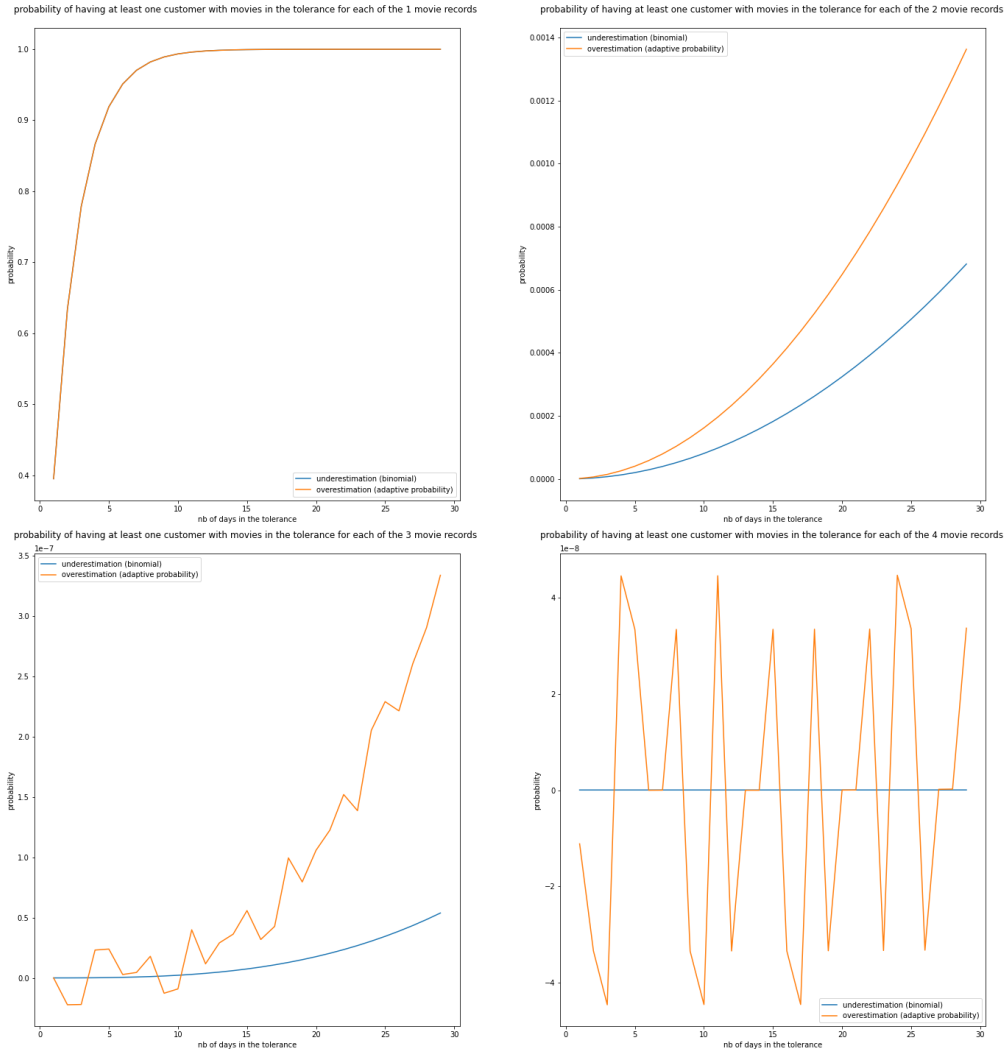

Figure 8: Probability for a customer to have reviews in the neighborhood of each of the auxiliary records, for a number of respectively 1,2,3,4 auxiliary records, as a function of the number of days in the neighborhood. From 3 records, numerical stability problems appear

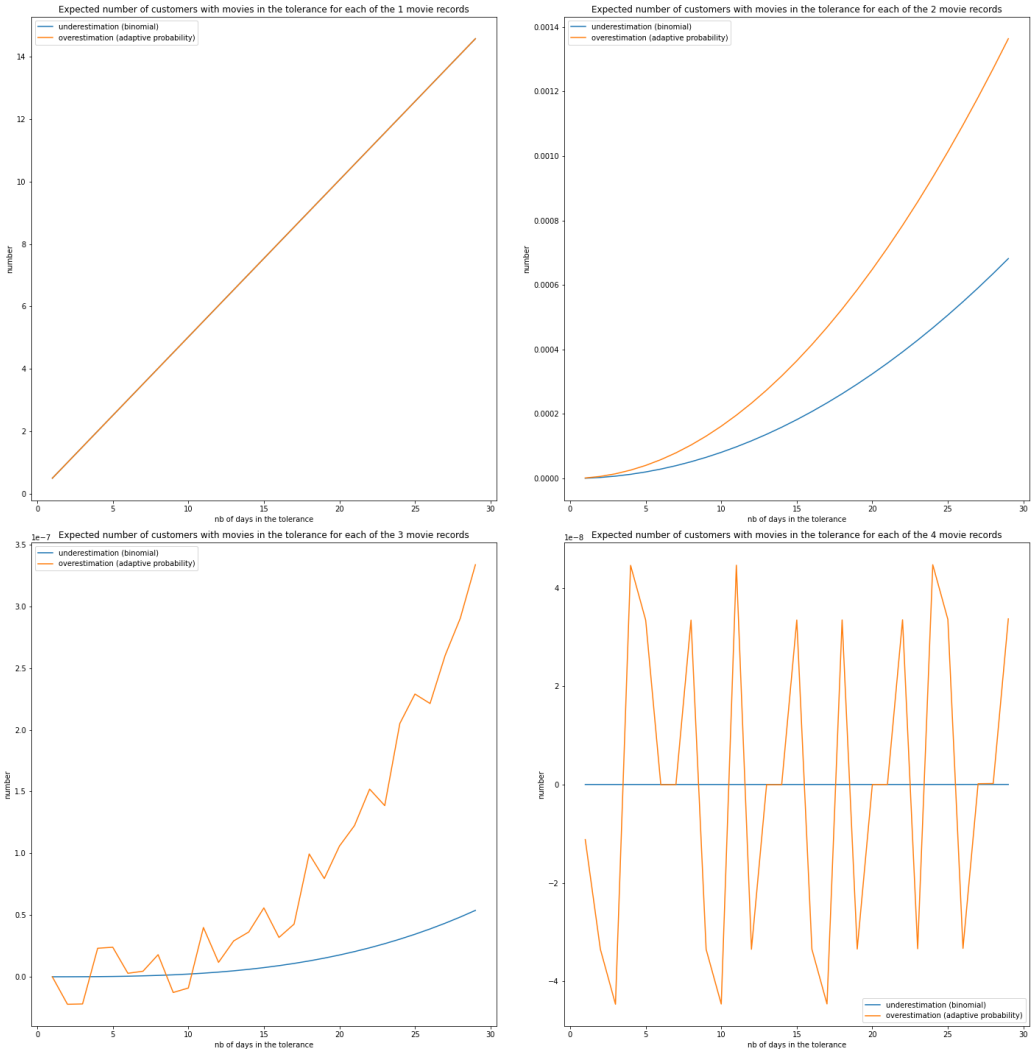

Figure 9: Expected number of customers with reviews in the neighborhood of each auxiliary records, for a number of respectively 1,2,3,4 auxiliary records, as a function of the number of days in the neighborhood. From 3 records, numerical stability problems appear

3.3 Unsuccessful alternatives

The first type of alternatives was the following: could we first de-anonymize the movie identifier ? If that is the case, then we would just need to repeat the results of the papers afterwards to proceed to the de-anonymization of the customers. Movies have a great number of public information, so it might be possible. The information we have at our disposal inside the dataset, for each movie identifier, are the number of reviews, the distribution of ratings and dates. Suppose we have complete information on the movie titles inside the dataset (we could have access to the movies accessible at a given time on Netflix platform, and we could suppose the not much movies have been discarded from the whole catalog by Netflix before releasing the dataset). The problem is equivalent to finding the appropriate permutation that maps the movie titles to the movie identifiers. As we have 17,770 movie identifiers, we would have 17,770! possibilities, clearly a huge number.

The idea would be to find exterior knowledge of the distribution of reviews and dates for each movie, calculate a similarity of the distribution for each movie identifier and movie title, and use some bipartite matching algorithm to find the correspondence.

To our knowledge, the official Netflix API is no longer available from 2014 [3]. So we cannot use what could have been the most reliable source of information. In this section, we

used IMDB API to get information about the movies. We could mainly get the number of reviews and the average rating. Basically, we wanted to have a feel of whether this method is promising, and if it is worth it to try to find more precise information about the distribution.

From the observation of the data, it is clear that there is a lot of noise in the comparison of the datasets. Even if there is a clear correlation in the average rating, the number of views on both platforms does not send a signal as encouraging. Of course, we are comparing the number of reviews on Netflix in early 2000s, to the number of reviews on IMDB in 2020, so it is understandable. We tried to use a simple similarity measure:

$$sim(x, y) = e^{|x-y|}$$

and the corresponding distance

$$d(x, y) = 1 - e^{|x-y|}$$

as well as the manhattan distance.

We then used a bipartite matching algorithm on only 30 movies, to see if the method was promising. Clearly, the results were not good, because only three movies out of thirty were matched. As we would have to extend this approach to 17,770 movies, we can conclude that the information we have is clearly not enough.

We could think of adding other aggregates, such as the number of reviews for each movies, but it will certainly not be enough, and the noise is too high.

Still, a valuable information from this attempt is that the difference of values of aggregates between Netflix and IMDB can follow some distribution, and as such we can control their differences. For example, it is clear that the difference in average rating is not so important, and follows a gaussian distribution, with a reasonable standard deviation

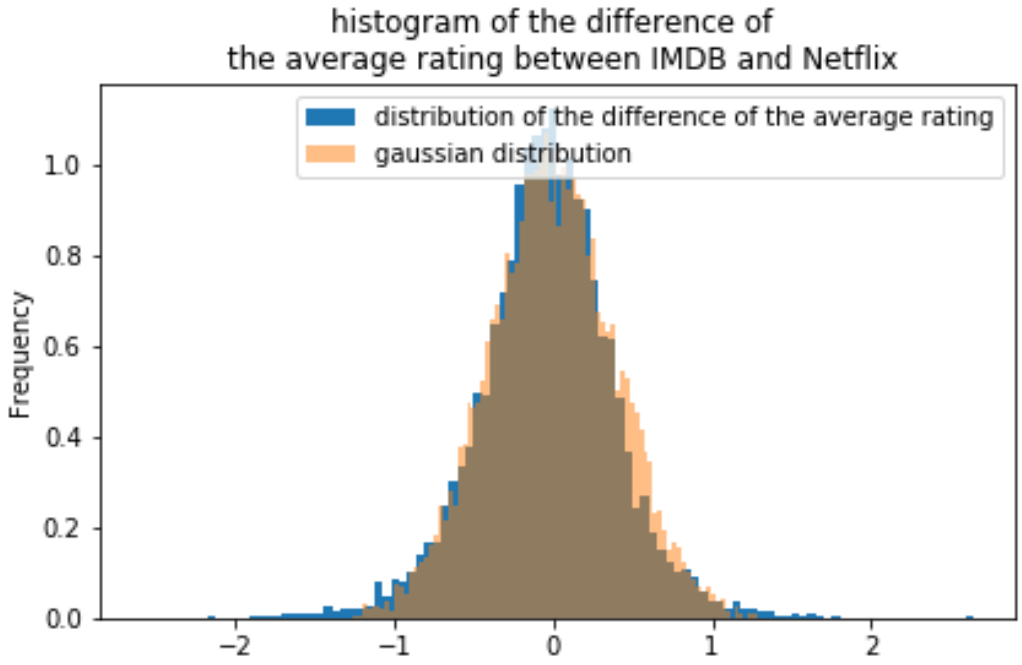

Figure 10: A gaussian distribution is fitted to the difference between the Netflix and IMDB ratings

3.4 Extension of the model

We also tried to extend the model presented in the paper; that is, trying to directly find the customer, and not solving the movie assignment problem. We take inspiration of the

Scoreboard-RH approach. The similarity, instead of being:

$$Sim(aux_i, r'_i) = e^{-\frac{|\rho_i - \rho'_i|}{\rho_0}} + e^{-\frac{|d_i - d'_i|}{d_0}}$$

is replaced by

$$Sim_{new}(aux_i, r'_j) = e^{-\frac{|\rho_i - \rho'_j|}{\rho_0}} + e^{-\frac{|d_i - d'_j|}{d_0}}$$
$$Sim_{new}(aux_i, r') = max_j Sim_{new}(aux_i, r'_j)$$

The similarity is expressed as between a movie i of aux and a customer record r', because we cannot directly attach the same movie identifier. Simple, we take the movie title j such that the similarity with the auxiliary movie is maximized. We hope that such movie title actually corresponds to the movie identifier from aux.

We also remark that some customers have reviewed close to all the movies, making them very likely to have one instance (rating, date) for all possible ratings, dates (recall that the number of possibilities is 11210).

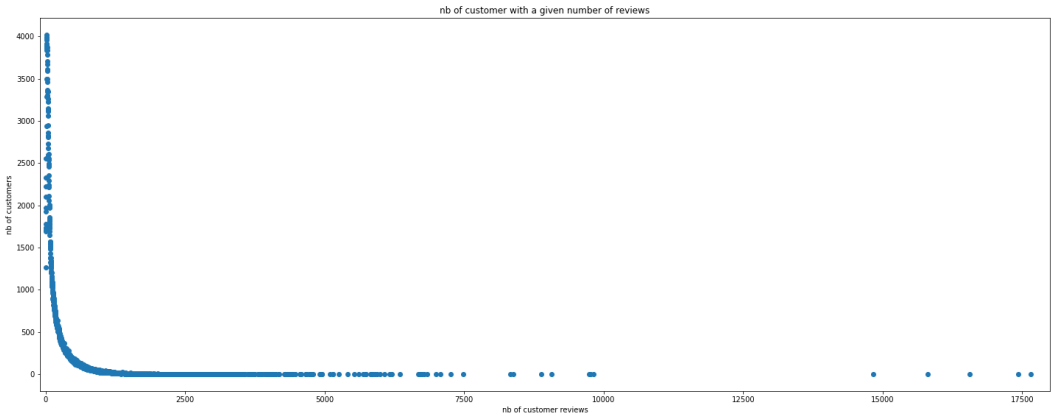

Figure 11: Number of customer with a given number of reviews

We propose to weight the similarity values with a coefficient, as is done in the paper with $wt(i)$. Here, we weight it inversely proportional to the probability that a given customer has a reviews in the neighborhood of all the auxiliary records, by chance. These weights will disadvantage the customers that have made a lot of reviews and thus have a high coverage.

In short, we replace the coefficient:

$$wt(i) = \frac{1}{\log |supp(i)|}$$

by a coefficient such as

$$wt(r) = \frac{1}{p_{s,n} + 1}$$

n being the number of reviews made by the customer attached to the record r.

Unfortunately, this computation uses a cartesian product between the whole dataset and the dataset made of auxiliary records. The auxiliary dataset is limited in size (for example, we might have no more than 10 auxiliary records on a given person we want to find in the dataset), so we can still control the size of this operation. Nevertheless, with our computers or even google colab, we had trouble to make it work. For the whole process to take place on 10 different experiments, each of which have 10 auxiliary records, the computation time was around 4 hours on colab. As the best guess result would not be significant, especially because the expected number of customers to have perfect match in ratings and dates with

auxiliary records is not neglectable, we rather use the metrics on the entropy reduction. The first results with the means we have are suspicious. Further investigations would be necessary to confirm the quality of the results.

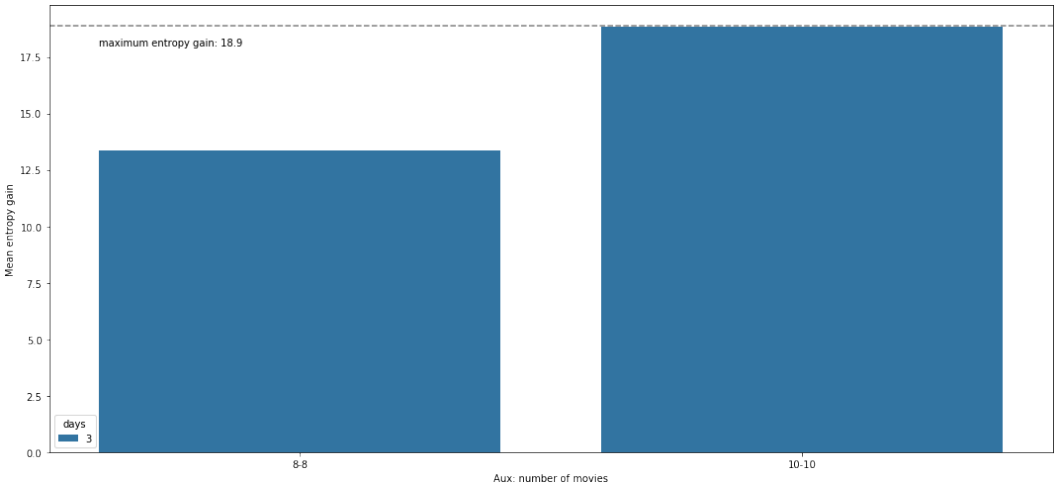

Figure 12: Entropy gain for experience without movies

Finally, we also propose another similarity measure, of the form:

$$Sim_{new}(aux_i, r'_j) = e^{-\frac{|\rho_i - \rho'_j|}{\rho_0}} + e^{-\frac{|d_i - d'_j|}{d_0}} + e^{-\frac{|\rho_i^{avg} - \rho_j^{avg\prime}|}{\rho_0^{avg}}}$$
$$Sim_{new}(aux_i, r') = max_j Sim_{new}(aux_i, r'_j)$$

We introduce here ρ_i^{avg} and $\rho_j^{avg\prime}$, which are respectively the average rating of the movie i on Netflix dataset and on IMDB dataset. We make sure this added term is not too big compared to the two other by fitting ρ_0^{avg} appropriately. The idea is to rule out matching between movies that do not have an average rating close enough. This is a way to reduce the number of candidate movies, and thus reduce the probability that a random customer has a record that by chance would be in the neighborhood (morally, we reduce the neighborhood to movies with an average rating close enough). In practice, we could use IMDB dataset, or as a proof of concept, use a gaussian noise from the Netflix average rating as the average rating used in the auxiliary information. We could show that a Gaussian noise with a reasonable standard deviation would be appropriate to do some tests. Of course, one of the shortcomings of this approach is that we used information we were not supposed to have: matching movies from IMDB and Netflix. But we could have compared data from two public apis, such as rotten tomatoes and IMDB to make our point that average rating are not far in different datasets.

We can formulate a critic for the way the similarity functions are designed. A customer review can be put in correspondence with several auxiliary records, if the latter are close enough. Indeed, for each auxiliary record, we take the customer review that maximizes a similarity measure, independently for each auxiliary record. We can propose a more robust, but more computationally intensive approach, that overcome this loophole. If we had s auxiliary record, we could first take the top s customer reviews for each auxiliary record. Then, do some bipartite matching to allocate one different customer review to each auxiliary record. We would take the set of customer reviews that match best the auxiliary records while making sure to have different customer review for each auxiliary record. At each time, the size and complexity of the bipartite matching can be controlled by the number of auxiliary records. Also, this would make the algorithm more in line with the probability calculations, and manage better the case of the de-anonymization of customers that have made several reviews in a short time.

4 Conclusion

In conclusion, we wanted in this project to investigate a little further and more in details the reasoning of the paper on de-anonymization of Netflix dataset [1]. We could reproduce the main results of the paper. We faced different kinds of difficulties, among which the computing resources available to us to do some calculations on such a dataset. We had also troubles to find some high quality auxiliary information, one of the reasons being the effect of time: while the Netflix dataset was released in the mid-2000s, many things have changed since. Even if we had access to a Netflix APIs, we would not have guarantees to access data from 15 years ago, nor do we have guarantees that Netflix did not do changes in their data releases, after the shockwave of the publications on the de-anonymization of their dataset. The extension that we proposed is difficult to put in place, because we lose the sparsity argument from the dataset. Still, new extensions could be proposed, such as keeping informations from different de-anonymization attacks, to be able to have higher and higher confidence in the matching of the movie titles with their identifiers in the Netflix dataset. To some extent, the first attacks should be more difficult, and once we have acquired enough confidence on the de-anonymization of movies, the de-anonymization of customers should be easier. Still, as we experiences, it is not an easy task, because there is a high probability that a random customer in the dataset has reviews that match the auxiliary records. We did not study the deviation of the actual review distribution compared to a uniform distribution, though. This might help in the task, and a deeper-thought probabilistic model could give better results. The main point was to show that the sparsity argument does not hold as much as in the paper.

We also feel that using average rating and popularity of movies could help in reducing the uncertainty of the de-anonymization of such dataset. Though, fitting correctly the parameters to take into account these information would require more time and computing power than the resources we had for this project.

References

[1] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," pp. 111–125, 06 2008.

[2] V. L. HENRIC PLUVINAGE, "Iasd privacy — Github," 2020. Available at https://github.com/VincentHenric/IASD_privacy.git.

[3] V. Khulbe, "Is there any public registered api available for netflix? — Quora," 2018. [Online; accessed 26-March-2020].