Link Analysis on IMDb Dataset

Rita Folisi, Alessia Lombarda

July 16, 2021

We declare that this material, which we now submit for assessment, is entirely our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of our work. We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by us or any other person for assessment on this or any other course of study.

1 Introduction

In this project we have analyzed different link analysis algorithms applied to the data of the IMDb dataset. In Section 2 we describe the dataset used to perform the experiments and in Section 3 how we preprocessed the dataset to get useful informations needed by the algorithms. In Section 4 we illustrate the PageRank Algorithm and its Taxation Variant, the Topic Sensitive PageRank and the TrustRank variant together with the pseudocode of the presented algorithms. In Section 5 we then compare the performances in execution time and number of iterations of the different algorithms. The code is available on Github[1].

2 Dataset description

This study is based on the IMDb dataset, published on Kaggle, under IMDb non-commercial licensing[2]. IMDb includes informations about movies, tv shows, people involved and ratings. It is mostly employed in building recommender systems and in other machine learning tasks. The dataset contains five gzipped, tab-separated-values (TSV) formatted files in the UTF-8 character set. In particular, the files which are useful to our purpose are:

- `title.basics` containing informations about movies titles,
- `title.principals` containing informations about the principal cast for each title,
- `name.basics` containing informations about people involved in movies,
- `title.ratings` containing the IMDb ratings and votes information for titles.

For each dataset we have considered a subset of columns. In Figure 1 title.basics is presented and the columns considered are `tconst` and `genres`. The former expresses the alphanumeric unique identifier of the title; the latter includes up to three genres associated with the title.

	tconst	genres
0	tt0000001	Documentary,Short
1	tt0000002	Animation,Short
2	tt0000003	Animation,Comedy,Romance
3	tt0000004	Animation,Short
4	tt0000005	Comedy,Short

Figure 1: Columns extracted from title.basics

In Figure 2 title.principals is presented and the columns considered are `tconst`, `nconst` and `category`. The second one refers to the alphanumeric unique identifier of each member of the cast; the third one specifies the person's job and it is in string format.

	tconst	nconst	category
0	tt0000001	nm1588970	self
1	tt0000001	nm0005690	director
2	tt0000001	nm0374658	cinematographer
3	tt0000002	nm0721526	director
4	tt0000002	nm1335271	composer

Figure 2: Columns extracted from title.principals

[1] https://github.com/AlessiaLombarda/AMD_project

[2] https://www.kaggle.com/ashirwadsangwan/imdb-dataset

In Figure 3 name.basics is presented and the columns considered are `nconst` and `primaryName`. The latter expresses the name the person is most often credited with.

	nconst	primaryName
0	nm0000001	Fred Astaire
1	nm0000002	Lauren Bacall
2	nm0000003	Brigitte Bardot
3	nm0000004	John Belushi
4	nm0000005	Ingmar Bergman

Figure 3: Columns extracted from name.basics

In Figure 4 title.ratings is presented and the columns considered are `tconst` and `averageRating`. The latter refers to the weighted average of all the individual user ratings for each title.

	tconst	averageRating
0	tt0000001	5.6
1	tt0000002	6.1
2	tt0000003	6.5
3	tt0000004	6.2
4	tt0000005	6.1

Figure 4: Columns extracted from title.ratings

All the values inside the columns descripted are in string format, except for averageRating, which is in float format.

With regards to data size, title.basics has 6,321,302 rows, title.principals has 36,468,817 rows, name.basics has 9,706,922 rows and title.ratings has 993,153 rows.

3 Data organization and preprocessing

We used Kaggle API to import all the datasets described in the Section 2. The datasets were then processed as dataframes. In the datasets the missing or null values are expressed through `\N`, therefore we have specified to recognize them as NaN values as well. Subsequently, we have removed all NaN values from the datasets. For instance, in title.basics we have removed 501,323 rows with NaN values.

Since the task of this project focuses on the interactions between actors, we are only interested in members of the cast in which the column category is equal to "actor" or "actress". Therefore, we applied a filter in the dataset title.basics. On the other side, we extracted all the possible genres in which the movies can be classified, in order to compute the Topic Sensitive variant. Since the column genres can contain more than one genre and it is in string format, we needed to split the string in order to retrieve the single genre.

Afterwards, we merged all our datasets through an inner join, in order to keep only tuples with common key values. First of all, we merged title.principals and name.basics, so that we can have association between the movie and the actors and actresses' names. It will be useful in the visualization of the graph or in general to have a better comprehension of the interactions between actors and actresses. Secondly, we needed to merge title.basics and title.principals, in order to connect genres of the movie to the actors that took part in it. Lastly, we merged title.ratings and title.principals in order to associate each actor to the ratings of the movies in which he took part.

In order to perform further operations, we needed to sample the final dataset. Otherwise, it would be too time and space expensive. We sampled the datasets using an uniform distribution and we employed five sample rates: 0.00001, 0.0001, 0.001, 0.01, 0.1.

At last, we built some dictionaries to have a better comprehension of the association between actors and other features. In this way, we can easily manage data in order to perform the Page Rank algorithm and all its variants. In particular, we have the following dictionaries:

- `actors_dict` in which each actor is associated to all the movies he/she performed in,
- `movies_dict` in which each movie is associated to every actor who took part in it,
- `genres_dict` in which each genre is associated to every actor who is related to it (i.e. working on movie of this genre at least once),
- `ratings_dict` which associates each actor to the average rating of the movies he/she performed in,
- `links_dict` which associate each actor to the actors he/she performed with.

Links_dict describes the edges of the final graph, while the actors are the nodes. Therefore, the final graph performs an analysis of the association between actors who worked together in the

same movie. In Figure 5 we extracted a subgraph of the whole final network.

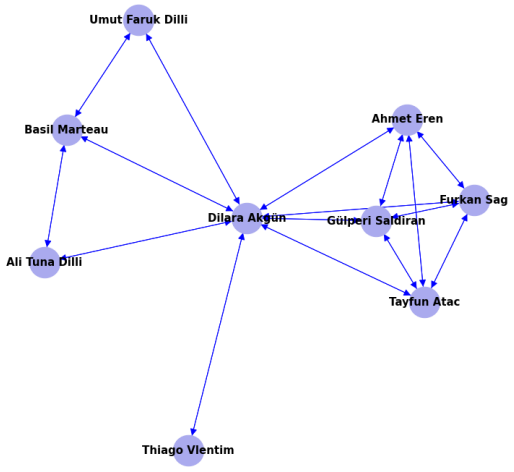

Figure 5: Subgraph extracted

4 Implemented algorithms

To address the problem of link analysis some algorithms have been developed. In this project some implementations have been analyzed: the *PageRank algorithm* and its variant with the *teleporting*, the *Topic Sensitive PageRank* and the *TrustRank* variant. In order to implement these algorithms and guarantee scalability we used the MapReduce Framework.

4.1 PageRank algorithm

PageRank is a function that assigns a real number to each node in the considered graph [LRU14]. This algorithm is based on the assumption we are working with a directed graph (in the case of the considered dataset, node indicate actors and links stand for actors that took part to the same project at least once). The process that leads to the computation of PageRank is the *Random Surfing Process*: at the beginning a population of a high number of individuals is assigned uniformly to the nodes of the graph; at a given time all individuals select uniformly at random one of the outgoing links of the node in which they are sitting. The process is repeated a certain number of times until a sort of stability is reached.
We say that the PageRank of a node is the probability of sitting in that node. Given the *transition matrix* M of n rows and n columns (n is the number of the nodes in the graph), its entries are in the form

$$m_{ij} = \begin{cases} 1/\mathrm{k}, & \text{if } j \text{ has } k \text{ arcs out,} \\ 0, & \text{otherwise,} \end{cases}$$

where i and j are nodes of the graph and k is the outer degree of the node j. A column vector extracted from the matrix M describes the probability distribution for the location of a random surfer: its j-th component is the probability that a surfer is at page j, that is the PageRank function. The PageRank algorithm proceeds as follows:

Algorithm 1 PageRank

$\underline{v}_0 \leftarrow \left[\frac{1}{n}\right]$
$t \leftarrow 1$
while $\|\underline{v}_{t+1} - \underline{v}_t\| > \epsilon$ **do**
  $\underline{v}_{t+1} \leftarrow M \cdot \underline{v}_t$
  $t \leftarrow t + 1$
end while
return $\underline{v}$

4.2 PageRank with taxation

The variant of the PageRank algorithm using taxation (or teleporting) is used to introduce some randomness and avoid problems due to spider traps and dead ends in the graph. This variant introduces the possibility for a random surfer to be teleported to another node of the graph; it is regulated by the parameter $\beta \in [0, 1]$. When β equals to 1, the algorithm corresponds to the basic version of PageRank explained in Section 4.1; for lower values of β, the algorithm introduces taxation. β is usually set to 0.85.

The PageRank algorithm with taxation proceeds as follows:

Algorithm 2 PageRank with taxation

$\underline{v}_0 \leftarrow \left[\frac{1}{n}\right]$
$t \leftarrow 1$
while $\|\underline{v}_{t+1} - \underline{v}_t\| > \epsilon$ **do**
  $\underline{v}_{t+1} \leftarrow M \cdot \underline{v}_t + (1 - \beta) \cdot \frac{1}{n}$
  $t \leftarrow t + 1$
end while
return $\underline{v}$

4.3 Topic Sensitive PageRank

One of the improvements that can be made to the PageRank Algorithm is that we can weight certain nodes more heavily because of their topics. This variant of the PageRank alters the way random surfers move, so that they would land on a node

of the chosen topic. In the case of IMDb dataset, the topics are the different genres a movie or TV series belongs to. It is also possible to extend this algorithm to the choice of many different topics. Chosen a topic t we define $\underline{s}$ a n-dimensional vector whose generic entry is

$$s_i = \begin{cases} 1, & \text{if } i \text{ is about topic } t \\ 0, & \text{otherwise.} \end{cases}$$

The Topic Sensitive PageRank proceeds as follows:

Algorithm 3 Topic Sensitive PageRank

$\underline{v}_0 \leftarrow \left[\frac{1}{n}\right]$
$t \leftarrow 1$
while $\|\underline{v}_{t+1} - \underline{v}_t\| > \epsilon$ **do**
  $\underline{v}_{t+1} \leftarrow M \cdot \underline{v}_t + (1-\beta) \cdot \frac{1}{\|\underline{s}\|}$
  $t \leftarrow t + 1$
end while
return $\underline{v}$

4.4 TrustRank variant

TrustRank Variant is a variant of topic sensitive PageRank in which the "topic" is a set of nodes believed to be trustworthy. In order to apply this variant to the algorithm it is necessary to define a trust vector $\underline{s}$, a n-dimensional vector whose generic entry is

$$s_i = \begin{cases} 1, & \text{if } i \text{ is trusted} \\ 0, & \text{otherwise.} \end{cases}$$

The algorithm to compute TrustRank is the same as the one shown in Section 4.3. Once both PageRank and TrustRank (based on a teleport set of trustworthy pages) have been computed it is useful to define the *Spam Mass Index*, the fraction of the PageRank coming from non trusted nodes. The Spam Mass Index is computed as

$$SMI = \frac{PR - TR}{PR};$$

a negative or small positive spam mass index for a node p means that p is trustworthy while a spam mass close to 1 suggests that p is probably spam. To implement this algorithm's variant, two different criteria have been chosen to define a trustful node: in the first version (*1st-TrustRank*) it's defined as an actor that took part in a number of projects that is greater than the overall mean, while in the second version (*2nd-TrustRank*) trustful actors are the ones whose mean rating over the movies in which they acted is greater than the overall mean.

5 Results and comparisons

We performed some experiments to test our code, measuring execution time and number of iterations for each algorithm employed. In order to have a better insight on how the algorithms scaled up with data size, we decided to use different sample rates. Thus, we performed for each algorithm five experiments, varying the sample rates. In Table 1 the correspondence between sample rates employed during experiments and the actual data sizes is presented. Furthermore, we iterated the whole process five times, in order to have more robust results. In the end, we performed a total of twenty-five experiments for each algorithm.

Sample Rate	Data Size
0.00001	34
0.0001	340
0.001	3,404
0.01	34,044
0.1	340,443

Table 1: Correspondence between Sample Rate and Data Size

In Figure 6 results concerning the execution time for each algorithm are illustrated. We computed for each sample rate the mean of the measurements obtained in each iteration. It is possible to notice that increasing the sample rate (therefore, enlarging the sampled dataset), we obtain in general an increase in execution time. However, this trend is not equal for all the algorithms considered. In general, this can depend on the random choice of the dataset but the variation should be minimum, as it happens in Topic Sensitive.

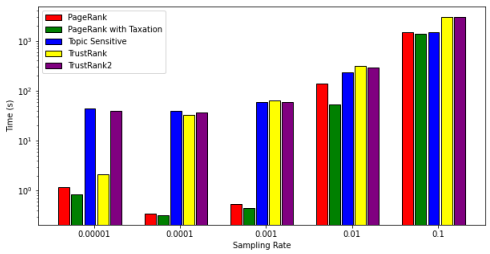

Figure 6: Execution time varying the sample rate: all algorithms

Nevertheless, using the lowest sampling rate (0.00001) we obtain far worst performances performing PageRank and PageRank with taxation. By inspecting logging files used during the experiments, we noticed that this situation happened

only at the first iteration of these two algorithms, since in further iterations with this sample rate the algorithms actually obtained better results. In particular, in the first iteration we obtained approximately three seconds, while in further iterations it took less than one second. In general, it is possible to notice that basic PageRank took more time than PageRank with taxation. Furthermore, Topic-Sensitive took more or equal time w.r.t 1st-TrustRank on small samples. We recall that in general TrustRank is based on the computation of both PageRank and a modified version of Topic-Sensitive. This result can be explained since in the Topic-Sensitive algorithm alone we perform more iterations than 1st-TrustRank. On the other side, 2nd-TrustRank took approximately the same time of Topic-Sensitive: as a matter of fact, their number of iterations are similar. In general, this measurement depends on the topics considered and on data.

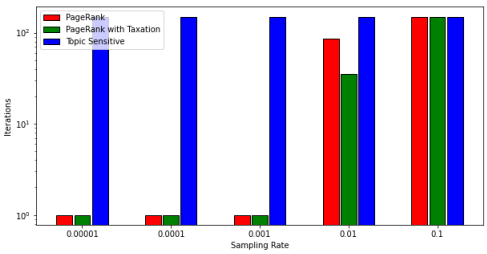

Figure 7: Number of iterations varying the sample rate: PageRank, PageRank with taxation, PageRank Topic Sensitive

TrustRank algorithms, as we can see in Figure 8 and in Figure 9, have to run first the PageRank algorithm and then a modified version of the Topic Sensitive Algorithm; we measured the number of iterations for each of them and plotted it in the graph.

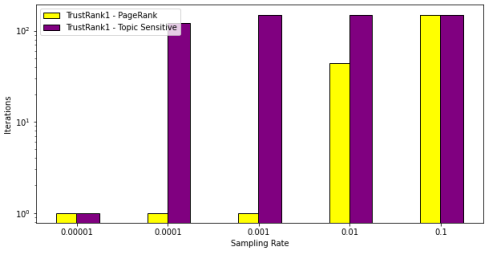

Figure 8: Number of iterations varying the sample rate: 1st-TrustRank

In particular we can see that the number of iterations of PageRank grows as the sampling rate increases (that is as the size of the sample increases) while the Topic Sensitive PageRank always uses the maximum number of iterations before stopping, let alone the case of the lowest sampling rate for the 1st-TrustRank; this can be due to the fact that we have set a maximum number of iterations so the Topic-Sensitive Algorithm is probably stopping before convergence. Increasing the number of maximum iteration could show us a similar dependence between sampling rate and number of iterations of the Topic-Sensitive PageRank in both TrustRank algorithms.

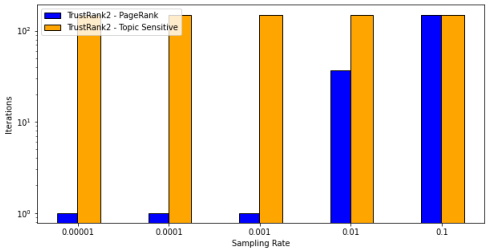

Figure 9: Number of iterations varying the sample rate: 2nd-TrustRank

6 Conclusion

This paper illustrates the PageRank algorithm and its variants applied to the problem of link analysis on a graph. In particular we performed some experiments on the IMDb dataset, comparing execution time and number of the iterations of the different algorithms as the sampling rate becomes smaller. Further improvements can be done to extend this work: it could be useful to compare these algorithms with other versions of PageRank to see if there is a significantly improvement in the performances. In order to have more robust results it could be interesting to execute these algorithms a higher number of times, to make the contribution of outliers negligible. Increasing the maximum number of iterations can also be helpful to see the behaviour of these algorithms as the amount of data processed grows. One last possible improvement is related to the Topic-Sensitive algorithm: in our experiments we considered a fixed set of three topics; it could be interesting to see how the performances vary as the number of topics (and so the cardinality of the interested set of nodes) grows.

References

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, USA, 2nd edition, 2014.