POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Extending PageRank algorithm in new contexts

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Paolo Gerosa

Advisor: Prof. Alessandro Campi

Academic year: 2021-2022

1. Introduction

Every day billions of people use internet to search for information. Google Chrome is the most popular and used browser in the world and the reason why the Google product became so popular and used at the beginning of the 20^{th} century is PageRank; this algorithm successfully ranks the importance of the webpages to allow users to find proper information in the fastest way possible.
In order to rank the webpages available online PageRank computes a score for each webpage $j \in \{1, ..., |V|\}$ where $|V|$ is the cardinality of the set containing all the webpages available in internet; The score $x_j \geq 0$ of the j-th page P_j must meet two requirements:

- it must be high if it refers to a page cited by many other pages.
- it must be high when referring to a page cited by very significant pages.

The simple counting of links pointing to a page is not enough to represent the score since it does not require the second condition.

2. Objective

This work applies the PageRank algorithm in different contexts and explains diverse methods to enrich the available tools that can be used to rank a list of objects.
The algorithms can be applied whenever the objects analyzed are represented in a network. One of the initial complexities is how to define the edges of the graph; this is why PageRank cannot be developed in all the problems. This work also explains how to integrate the final rank using other information of the nodes in the graph: this is presented in the so-called PageRank with personalization.
In the end the described methods are powerful tools as they allow to get a final rank of the objects analyzed by combining different algorithms to create one output; this allows the system to use multiple types of knowledge to generate a unique result.

3. Algorithm and Development

A set of web pages $P_1, \ldots, P_n$ can be represented as a directed graph $G = (V, L)$ where V represents the vertices and coincides with all the pages P_i available online and $(P_i, P_j) \in L$ if and only if there exists a link from page P_i to page P_j.
We denote by $A = [a_{ij}]$ the adjacency matrix of $G = (V, L)$ such that:

$$\begin{cases} a_{ij} = 1 \text{ if } \exists \text{ link from } P_j \text{ to } P_i \\ a_{ij} = 0 \text{ else} \end{cases}$$

From the two conditions imposed to the score

it is more appropriate to define it such that the score x_i of page P_i is proportional, with c constant of proportionality equal for all pages, to the sum of the scores of the pages which refer to P_i:

$$x = cAx \;\Rightarrow\; Ax = \lambda x \tag{1}$$

If the solution exists then the problem reduces to the computation of an eigenvector associated to the eigenvalue $\lambda = 1/c$.
Let's now normalize A to $M = [m_{ij}]$:

$$\begin{cases} m_{ij} = \frac{1}{N_j} \text{ if } \exists \text{ link from } P_j \text{ to } P_i \\ m_{ij} = 0 \text{ else} \end{cases} \tag{2}$$

where N_j is the number of pages which can be directly reached from page P_j.
Pages with no forward links are called dangling pages. There are different ways to treat the dangling pages; the easiest way is to let all the dangling pages to point to each page of the web such that:

$$\widetilde{M} = M + \frac{1}{n} 1_n B^T \tag{3}$$

where $1_n = [1, ..., 1]'$ and B is a vector such that $B_i = 1$ if P_i is a dangling page and $B_i = 0$ otherwise.
$\widetilde{M}$ is a left stochastic matrix and it represents the transition matrix of a Markov chain describing a random walk of a "random surfer" between web pages assuming that surfing the web occurs according to the forward links of the last page visited; i.e. the "random surfer" is crawling the web by randoming clicking on successive links in the pages. If the "random surfer" is in a dangling page the probability transition is a uniform distribution of all the pages. However we will show how different ways to treat dangling pages will generate personalized ranks.
Even if $\widetilde{M}$ is a left stochastic matrix there is still one last issue: even if from one page of $\widetilde{M}$ it is possible to move at least to another page, not necessarily it is possible to go from any page to any other page i.e. the matrix $\widetilde{M}$ is not irreducible.
To face this issue it's sufficient to perturb M with the usage of a positive matrix $E >> 0$ to build the Google matrix:

$$G = (1 - \alpha)\widetilde{M} + \alpha E \tag{4}$$

where $\alpha \in (0, 1)$ is the perturbation factor.

To give a logical interpretation of the final Google matrix G we go back to the "random surfer" model: by adding the perturbation we consider that the surfer can move to other pages by directly entering the URL in the address bar. This is called teleportation and it allows to give a more sensible and realistic model to the "random surfer". E is the matrix distribution that the surfer uses when he chooses to jump to a new page by using teleportation, E is called personalization matrix.
Since G is a positive and irreducible matrix the invariant probability distribution exists: this will be the score of the nodes in the network.
If you have m different datasets containing information of the nodes in the network and $\alpha_1 = \alpha_2 = ... = \alpha_m = \alpha/m$ (uniform weight for the datasets) then it is possible to generalize PageRank algorithm using multiple personalization:

$$G = \left(1 - \sum_j \frac{\alpha}{m}\right) \widetilde{M} + \sum_j \frac{\alpha}{m} E_j = \tag{5}$$

$$= (1 - \alpha)\widetilde{M} + \sum_j \frac{\alpha}{m} E_j \tag{6}$$

4. PubMed

PubMed is one of the most popular search engine in the world accessing an enormous archive of biomedical and life sciences journal literature. Its database contains more than 33 million citations and abstracts of biomedical literature and on an average working day, there are about 2.5 million users conducting 3 million searches and 9 million page views.
PubMed uses Best Match sort to rank the articles that a user is searching for; this algorithm is applying machine learning using signals such as the number of matches between the search terms and the PubMed record, or the publication type and year.
In the PubMed application we would like to compute PageRank to rank the articles; to do so the network is built by:

- Considering each publication of PubMed associated to a query as a node of the graph.
- Considering as an edge of the graph each connection between two nodes i and j if article j is cited by article i.
- Considering the network as the directed graph composed by nodes and edges: this

network is a representation of the relations between the publications in PubMed.

The idea is to apply the PageRank algorithm in PubMed in order to compare its result with the one used by PubMed (Best Match sort algorithm) that uses only machine learning.

To understand both from a qualitative analysis and also from a quantitative point of view we decided to build a webapp using the Streamlit library in python; in this way users from Politecnico di Milano and other universities could work with the webapp and answer in a questionnaire to some simple questions regarding their opinions on the behaviour of the different methods proposed.

Regarding the three algorithms proposed on the webapp one is the basic match sort using machine learning defined by PubMed, one is PageRank using as personalization vector a distribution that considers information of the single publications, the last one uses multiple personalization vectors. The information used to build the personalization vectors is described in the following section. Observe that these algorithms are simply called *Algorithm 1*, *Algorithm 2* and *Algorithm 3* in the webapp so that users don't know which algorithm has been used behind the different ranks; this is done in order to avoid bias in the answers to the questionnaire.

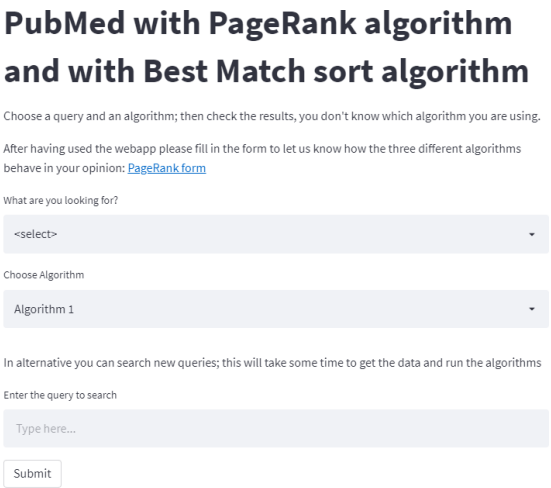


Figure 1: Webapp interface

4.1. Data

In order to build the personalization vector used in the PageRank algorithm we cooperated with Altmetric, a company that has the aim to track and analyze online activities around researches and publications from numerous diverse sources (public policy documents, social media, blogs, wikipedia...). Altmetric provided us with an API to collect many different data regarding each publication analyzed by using its DOI: the Digital Object Identifier of an article.

Data used in the analysis are the following:

- Citation by feeds count.
- Citation by posts count.
- Citation by tweeters count
- Citation by policies count
- Citation by patents count
- Citation by Wikipedia count
- Citation by accounts count
- Score
- Readers count
- Mendeley index
- Connotea index

We have an unsupervised ranking problem in which we would like to create a rank of all the citations based only on the above features; this rank will be fundamental to build a probability distribution on the publications: this will be used as personalization vector.

In figure 2 it is observable that taking into consideration only the first two principal components of the dataset is more than enough in order to explain more than 80% of the total variability: the new dataset has a reduced dimension of two and is simpler to build the personalization vector.

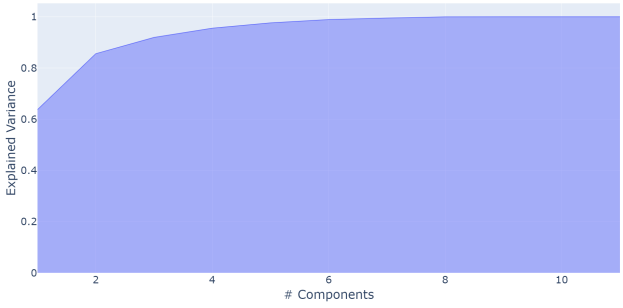


Figure 2: Explained variance wrt principal components

Using the loadings of the first 2 PCs and their meanings we can construct a personalization vector:

Algorithm 1 Order metric used to sort the elements

Consider two different publications i and j
Define $k > 0 \in R$
if $\mathrm{PC}_1(i) > \mathrm{PC}_1(j) + k$ **then**
    i is ranked before j
else if $\mathrm{PC}_1(j) > \mathrm{PC}_1(i) + k$ **then**
    j is ranked before i
else
    if $\mathrm{PC}_2(i) > PC_2(j)$ **then**
        i is ranked before j
    else if $PC_2(j) > PC_2(i)$ **then**
        j is ranked before i
    else
        i and j are in the same position
    end if
end if

4.2. Personalized distribution

After having built a ranking based on the Altmetric features it is finally possible to create a discete distribution over the publications. The distribution that has been chosen is the Geometric one since it is a flexible distribution for this problem.
In particular the distribution considered in the teleportation of the Markov chain random walk simulations is a Truncated Geometric (truncated since the number of publications are not unlimited); indeed let X the random variable "what is the next page chosen by the random surfer?", let i the publication in the i-th position of the rank and $F(x; p)$ the cumulative distribution function of a Geometric of parameter p evaluated in x then

$$X \sim \mathrm{TruncGeom}(p) \tag{7}$$

$$P(X = i) = \frac{p(1-p)^{i-1}}{F(|V|; p)} \tag{8}$$

where |V| is the cardinality of the publications set.
After having tuned the parameter for several queries the final value of p chosen is $p = 0.25$, this value is a good tradeoff to both avoid giving too much importance to the first pages and to avoid uniform weights.

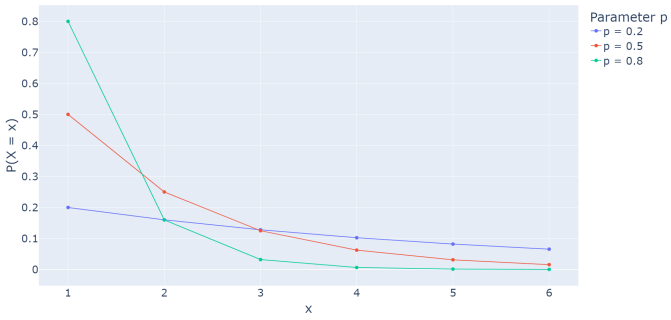


Figure 3: Geometric distribution for different parameters p

4.3. PageRank with two personalization

This algorithm is applicable in PubMed as for each publication we have two different types of information: one is provided by Altmetric; the other is referred to the ranking given by Best Match sort. It is known that Best Match sort uses machine learning to focus on the content of each publication and the historical information of the articles.
The new Google matrix is the following:

$$G = (1 - \sum_{j=1}^{2} \alpha_j)\widetilde{M} + \sum_{j=1}^{2} \alpha_j E_j \tag{9}$$

where E_j is the personalization matrix identifies by the j^{th} dataset and α_j is the perturbation factor associated to the j^{th} dataset.
With the ranking given by the Altmetric features and the ranking given by Best Match sort it is possible to compute two distribution as already discussed in 4.2.
The two distributions are used to compose matrices E_1 and E_2 to apply in the PageRank with multiple personalization. Uniform weights $\alpha_1 = \alpha_2 = \alpha$ have been chosen and since more information on the publications have been used it is understandable to increase the importance of the distributions with respect to the topology of the network: that's why α is increased from 0.15 used in the previous algorithms to 0.2.

4.4. Results

It is simple to evaluate the performance of the algorithm. Indeed PubMed is used by many users all over the world hence we can consider that its searching engine is a good rank to compare the result with.

This is done by analyzing the answers in the PageRank form of the webapp; the results are interesting: out of 20+ users of the app that filled the form it seems that the algorithm is performing well. *Algorithm 2* is referring to PageRank with personalization, *Algorithm 3* is Best Match sort, the PubMed one and finally *Algorithm 1* is Pagerank with two personalizations.

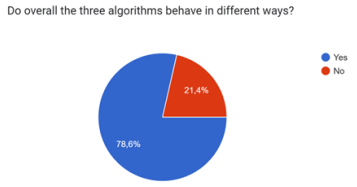


Figure 4: Behaviour of the three different algorithms

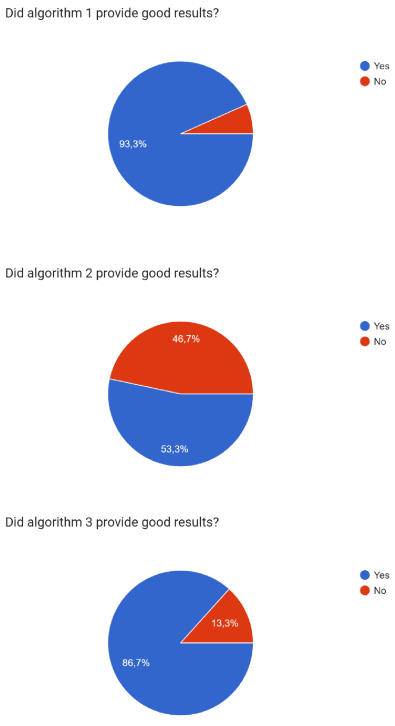


Figure 5: Good overall results for the three algorithms

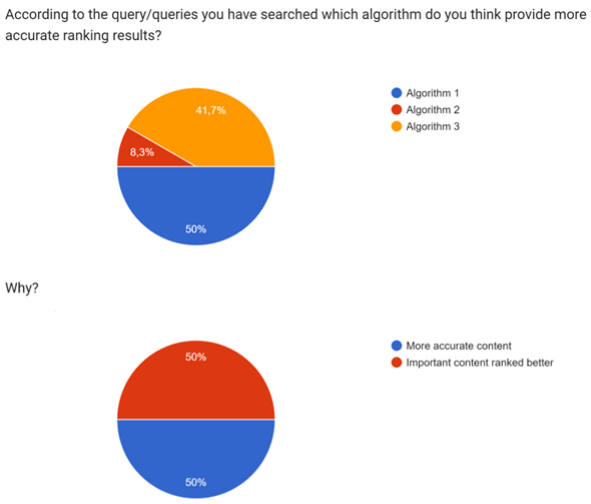


Figure 6: Best algorithm and why

Firstly the majority of the users have stated that the three different algorithms behave in different ways.
Overall the three algorithms have provided good results for the users.
Furthermore on average *Algorithm 2* is the less performing one to provide more accurate results to the queries; on the other hand *Algorithm 1* is the most efficient one, this corresponds to PageRank with multiple personalization.

5. Conclusions

Generalized PageRank allows PageRank to be used in combination with other rankings, this empowers to integrate PageRank with whichever other algorithms that rank a list of objects.
One downside is that the basic PageRank cannot be used in all the contexts but only when the lists of objects to be ranked can form a meaningful directed graph, meaning a network where the score given to the nodes (to rank the nodes in the output) must meet two requirements:

- it must be high if it refers to a node linked by many other nodes.
- it must be high when referring to a node linked by very significant nodes.

It is not always easy to create this network but this work analyzed three completely different environments where a network can be built.
The problem of ranking a list of objects is a very complex problem as there is no absolute solution. Indeed ranking the webpages for the users

can depend on the user that is using the algorithm, the best solution for the user is subjective. In the PubMed analysis users were slightly more satisfied by the results given by PageRank in comparison to the ones given by Best Match sort, this means that on average the algorithm is performing well to spot the best articles to show to the users.

References

[1] Nicolas Fiorini, Kathi Canese, Grisha Starchenko, Evgeny Kireev, Won Kim, Vadim Miller, Maxim Osipov, Michael Kholodov, Rafis Ismagilov, Sunil Mohan, James Ostell, and Zhiyong Lu. Best match: New relevance search for pubmed. *PLOS Biology*, 8 2018.

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, 11 1999. After publishing this work L. Page and S. Brin founded Google.

[3] Ernesto Salinelli and Franco Tomarelli. *Discrete Dynamical Models*. Springer, 2015.

[4] Jenny Tjan. How to optimize the personalization vector to combat link spamming. Master's thesis, 6 2016.

[5] Elliot Yates and Louise Dixon. Pagerank as a method to rank biomedical literature by importance. *Source Code for Biology and Medicine*, 12 2015.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Extending PageRank algorithm in new contexts

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Paolo Gerosa**

Student ID: 965323
Advisor: Prof. Alessandro Campi
Academic Year: 2021-22

Abstract

The PageRank algorithm or Google algorithm was introduced by Larry Page, one of the two founders of Google, in 1999; this algorithm is still partially used by Google in order to rank the webpages in the Google search engine. One of the interesting aspects of this algorithm is how to start from a really complex problem and end up with an effective but simple solution. Indeed the trademark Google reminds of a stratospheric number (googol=10^{100}): the reason is that the scale of the problem that the search engines have to face and solve is enormous.

Given a query of a random user the aim is to order the importance of all the pages related to that query. The algorithm uses a graph of connections between the nodes (webpages) based on the hyperlinks between the pages and it returns a score for each page; this score determines the position of the page in the final rank. The higher the score, the higher the page will be on the list given in output to the user.

This work has the scope to apply PageRank in different contexts to prove that this algorithm is still valuable to rank a list of objects connected one another. The algorithm is integrated by using statistical tools to make it more powerful for each context analyzed. The final aim is to enrich the available tools that can be used to solve the so-called ranking problems.

Keywords: PageRank, Network, Ranking, Search Engine

Abstract in lingua italiana

L'algoritmo PageRank o algorithmo di Google è stato introdotto da Larry Page, uno dei fondatori di Google, nel 1999. Questo algoritmo è ancora parzialmente utilizzato nel motore di ricerca sviluppato da Google. Uno degli aspetti interessanti e innovativi di questo algoritmo sta nel fatto di risolvere un problema alla base molto complicato applicando concetti basilari della matematica e statistica. Difatti il nome Google deriva dal numero stratosferico, parte del suo significato (Googol=10^{100}): la ragione sta nella complessità del problema che il motore di ricerca deve risolvere.

Data una query di un utente casuale l'obiettivo è di trovare un ordine di importanza delle pagine relative alla query. L'algoritmo usa una rete diretta dove le connessioni tra i nodi (pagine web) è basata sugli hyperlink tra le pagine web; la classifica finale delle pagine web è determinata da uno score calcolato tramite l'algoritmo. Più elevato è lo score di una pagina e più alta sarà la posizione della pagina nella classifica.

Questo lavoro ha l'obiettivo di applicare PageRank in diversi contesti per dimostrare che questo algoritmo ha ancora un valore per classificare una lista di oggetti. L'algoritmo viene integrato utilizzando strumenti di statistica applicata per renderlo adattabile ai diversi contesti analizzati. L'obiettivo finale è quello di arricchire con nuovi metodi i modelli già esistenti per classificare liste di oggetti.

Parole chiave: PageRank, Grafo, Classifica, Strumento di ricerca

Contents

Abstract **i**

Abstract in lingua italiana **iii**

Contents **v**

1 Introduction **1**
1.1 Objective . . . 2

2 Algorithm and Development **3**
2.1 PageRank algorithm . . . 3
2.1.1 Introduction . . . 3
2.1.2 PageRank with personalization vector . . . 5
2.2 Example . . . 6
2.3 Development . . . 9
2.3.1 Personalization vector and comparison metrics . . . 9
2.3.2 Diagonal traversal . . . 11
2.4 Generalized PageRank with personalization . . . 12
2.5 Metrics to evaluate results . . . 12

3 PubMed **15**
3.1 Data preparation . . . 15
3.2 Development . . . 16
3.3 Webapp . . . 17
3.4 Personalized distribution . . . 18
3.4.1 Data . . . 18
3.4.2 The problem . . . 19
3.4.3 PCA . . . 20
3.4.4 The distribution . . . 23

3.4.5 Distribution for the personalization . . . 24
3.5 PageRank with two personalization . . . 25
3.6 Results . . . 26

4 Twitter **31**
4.1 Data preparation . . . 31
4.2 Development . . . 32
4.3 Results . . . 34

5 Tennis **37**
5.1 The problem . . . 37
5.2 Data preparation . . . 37
5.3 Development . . . 38
5.4 Results . . . 41
5.4.1 Day 4 . . . 41
5.4.2 Day 5 . . . 42
5.4.3 Day 7 . . . 42
5.4.4 Day 8 . . . 43

6 Related works **45**

7 Conclusion **49**
7.1 Algorithm performance . . . 50

Bibliography **53**

A Appendix A **55**

List of Figures **59**

List of Tables **61**

Acknowledgements **63**

1 | Introduction

Every day billions of people use internet to search for information. Google Chrome is the most popular and used browser in the world; this is because of its search engine: the software that allows users to search for information on the web ranks the results. The reason why the Google product became so popular and used at the beginning of the 20^{th} century is PageRank; this algorithm successfully ranks the importance of the webpages to allow users to find proper information in the fastest way possible.

The exact implementation of Google search engine is not publicly known as it is protected by copyright; furthermore the algorithm has changed and drastically improved over the years. On the other hand it is known that PageRank is still internally used in the search engine a a small part of the huge system to rank the webpages.

In order to rank the webpages available online PageRank computes a score for each page $j \in \{1, ..., |V|\}$; notice that $|V|$ is the cardinality of the set containing all the webpages available in internet. This score vector allows then to rank the webpages in decreasing order from the most important page to the least important one.

The score $x_j \geq 0$ of the j-th page P_j must meet two requirements:

- it must be high if it refers to a page cited by many other pages.
- it must be high when referring to a page cited by very significant pages.

The simple counting of links pointing to a page is not enough to represent the score since it does not require the second condition.

Indeed, a major page as https://www.wikipedia.org/ has hundreds of thousands of pages pointing to it. This fact generally implies that the page is quite important and should be ranked high. On the other hand if a web page has a citation from the wikipedia home page, it may be only one link but it is a very important one. This implies that this page must be ranked higher than other pages with more links but from less important pages.

This project uses and integrates the PageRank algorithm in three different environments:

- Twitter social media: plain algorithm.

- Pubmed search engine: combination of algorithms and PageRank with personalization.
- Wimbledon tennis grand slam: multiple personalization.

1.1. Objective

This work applies the PageRank algorithm in different contexts and explains diverse methods to enrich the available tools that can be used to rank a list of objects.

The algorithms can be applied whenever the objects analyzed are represented in a network. One of the initial complexities is how to denote the edges of the graph; this is why PageRank cannot be developed in all the problems.

On the other hand PageRank is a useful tool as it uses the topology of the network. This work explains how to integrate the final rank using other information of the nodes in the graph. This is done by incorporating the result of the plain PageRank with other ranks.

In the end the described methods are powerful tools as they allow to get a final rank of the objects analyzed by combining different algorithms to create one output; this allows the system to use multiple independent information to generate a unique result.

The work considers only ways to make the final results as powerful as possible; the project does not take into consideration the time complexity of the algorithms. Indeed for example in PubMed the time taken to collect data and finalize the result is approximately 5 minutes, this is due to the fact that the code is developed in Python and the data is collected every time from scratch.

To make the algorithms time performant it is suggested to download and store the data server side and develop the code in C++.

2 | Algorithm and Development

2.1. PageRank algorithm

2.1.1. Introduction

A set of web pages $P_1, \ldots, P_n$ can be represented as a directed graph $G = (V, L)$ where V represents the vertices and coincides with all the pages P_i available online and $(P_i, P_j) \in L$ if and only if there exists a link from page P_i to page P_j. We denote by $\mathbb{A} = [a_{ij}]$ the adjacency matrix of $G = (V, L)$ such that:

$$\begin{cases} a_{ij} = 1 \text{ if } \exists \text{ link from } P_j \text{ to } P_i \\ a_{ij} = 0 \text{ if } \nexists \text{ link from } P_j \text{ to } P_i \end{cases} \tag{2.1}$$

Notice that the sum of the elements in the i-th column of $\mathbb{A}$ represents the number of outlinks (or forward links) of P_i, i.e. the number of pages which can be directly reached from P_i, while the sum of the elements in the i-th row of $\mathbb{A}$ represents the number of inlinks (or backward links) of P_i, i.e. the number of pages having a link which leads to P_i.

From the two conditions imposed to the score it is more appropriate to define it such that the score x_i of page P_i is proportional, with c constant of proportionality equal for all pages, to the sum of the scores of the pages which refer to P_i:

$$x = c\mathbb{A}x \;\Rightarrow\; \mathbb{A}x = \lambda x \tag{2.2}$$

If the solution exists then the problem reduces to the computation of an eigenvector associated to the eigenvalue $\lambda = 1/c$.

One of the main issues of this formulation is the risk to overestimate the score of the pages that have many incoming links from irrelevant pages. A possible solution is to normalize

$\mathbb{A}$ to $\mathbb{M} = [m_{ij}]$:

$$\begin{cases} m_{ij} = \frac{1}{N_j} \text{ if } \exists \text{ link from } P_j \text{ to } P_i \\ m_{ij} = 0 \text{ if } \nexists \text{ link from } P_j \text{ to } P_i \end{cases} \tag{2.3}$$

where N_j is the number of pages which can be directly reached from page P_j.

We can observe that $\mathbb{M}$ is nearly a left stochastic matrix i.e. a matrix such that each element is non negative and each column sums up to 1. In reality $\mathbb{M}$ is not a stochastic matrix since there could exist pages with no forward links; in this case there are columns with sum of elements equal to 0 and the matrix is not stochastic. Pages with no forward links are called dangling pages.

There are different ways to treat the dangling pages; the easiest way is to let all the dangling pages to point to each page of the web such that:

$$\widetilde{M} = M + \frac{1}{n} 1_n B^T \tag{2.4}$$

where $1_n = [1, ..., 1]'$ and B is a vector such that $B_i = 1$ if P_i is a dangling page and $B_i = 0$ otherwise.

Notice that $\widetilde{M}$ is a left stochastic matrix and it represents the transition matrix of a Markov chain describing a random walk of a "random surfer" between web pages assuming that surfing the web occurs according to the forward links of the last page visited; i.e. the "random surfer" is crawling the web by randoming clicking on successive links in the pages. If the "random surfer" is in a dangling page the probability transition is a uniform distribution of all the pages. However we will show how different ways to treat dangling pages will generate personalized ranks.

The dominant eigenvalue of the stochastic matrix $\widetilde{M}$ is $\lambda_{\widetilde{M}} = 1$ and the computation of the rank of the web pages is equivalent to the computation of the eigenvector R associated to $\lambda_{\widetilde{M}} = 1$ i.e. the invariant probability distribution for the associated Markov chain $G_{\widetilde{M}}$:

$$R = \widetilde{M}R, \; R_i \geq 0 \tag{2.5}$$

Even if $\widetilde{M}$ is a left stochastic matrix there is still one last issue: even if from one page of $\widetilde{M}$ it is possible to move at least to another page, not necessarily it is possible to go from any page to any other page i.e. the matrix $\widetilde{M}$ is not irreducible and the directed graph $G_{\widetilde{M}}$ associated to $\widetilde{M}$ is not strictly connected. This property is important because it guarantees, through the application of important theorems of Markov chain processes, the existence and unicity of the invariant distribution i.e. the solution of the problem.

2.1.2. PageRank with personalization vector

To face this issue it's sufficient to perturb $\widetilde{M}$ with the usage of a positive matrix $E >> \mathbb{0}$ to build the Google matrix:

$$G = (1 - \alpha)\widetilde{M} + \alpha E \tag{2.6}$$

where $\alpha \in (0, 1)$ is the perturbation factor.

To give a logical interpretation of the final Google matrix G we go back to the "random surfer" model: by adding the perturbation we consider that the surfer can move to other pages by directly entering the URL in the address bar. This is called teleportation and it allows to give a more sensible and realistic model to the "random surfer". E is the matrix distribution that the surfer uses when he chooses to jump to a new page by using teleportation, E is called personalization matrix and at first we will consider it as a random distribution for all the pages. However we will show how different choices of E will generate personalized ranks.

G is a strictly positive transition matrix hence G is irreducible and primitive; in addition G is stochastic because both $\widetilde{M}$ and E are stochastic. By Markov-Kakutani theorem $A.2$ G accepts an invariant probability distribution and by Perron-Frobenius theorem $A.3$ this invariant distribution is unique:

$$GR = R, \quad R \geq 0, \quad \sum_j R_j = 1 \tag{2.7}$$

R is the dominant eigenvector of G and its components R_j give the final ranks of the webpages; we are not much interested in the value of the R_js but in their decreasing order.

By notorius theorems ($A.4, A.5$ and $A.6$) it is known that:

$$\lim_{k \to +\infty} G^k P_0 = R \text{ for all } P_0 \text{ initial distribution} \tag{2.8}$$

Furthermore all the columns of G^k converge to R as k tends to inf:

$$\lim_{k \to +\infty} G^k = [R|R|...|R] \tag{2.9}$$

These theoretical results allow computers to solve the real problem and the difficulties related to the size of the involved matrices. Indeed the computation algorithm of the dominant eigenvector by the power method can be very fast: Theorem $A.6$ states that

the convergence velocity is linked to the second eigenvalue of G.

If we consider the following discrete dynamical system:

$$P_{k+1} = GP_k, \qquad P_0 \text{ probability distribution} \tag{2.10}$$

Relations 2.8 and 2.9 describe the asymptotic behaviour of the system and by exploiting the composition of G we reduce the system to the following one:

$$P_{k+1} = (1-\alpha)\widetilde{M}P_k + \alpha E, \qquad P_0 \text{ probability distribution} \tag{2.11}$$

Since $\widetilde{M}$ is a sparse matrix it is much more useful to use the power method on the last system.

Since G is a positive and irreducible matrix the invariant probability distribution exists and another method to compute it is to simulate a random walk on the Markov chain represented by the Google matrix G. By 2.8 the Montecarlo simulation of a Markov chain with a number of steps $k \to \infty$ identifies the invariant distribution with the average time that the surfer has spent on the single nodes of the network.

2.2. Example

Let's make an example to clarify the above explanation and let's consider the following network:

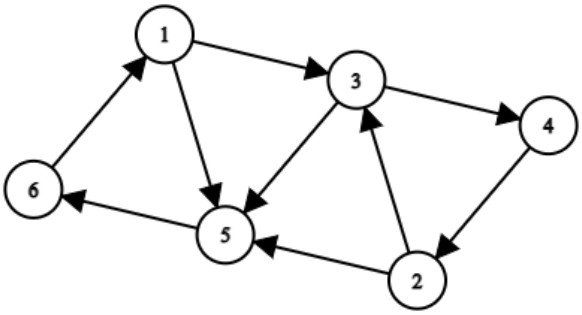


Figure 2.1: First basic example of a graph.

Let's consider 6 nodes, by simplicity analyzed as 6 web pages on internet, and let's say

that P1 is cited by P6, P2 is cited by P4, P3 is cited by P2 and P1, P4 is cited by P3, P5 is cited by P1, P2 and P3; finally P6 is cited only by P5.

First of all notice that in this network of 6 pages no page is a dangling page because each one has at least one forward link. Moreover observe that the graph is already strongly connected since from any couple of pages i and j it is possible to find a path that connects page i to page j. In this easy example the affinity matrix is the following:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \Longrightarrow \widetilde{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

To calculate the PageRank vector we consider $\alpha = 0.15$ and we use the Uniform personalization matrix that for each row has a uniform distribution $e = [\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}]$.

It follows that G is denoted by:

$$G = (1-\alpha)\widetilde{M} + \alpha E = \begin{bmatrix} 0.025 & 0.025 & 0.025 & 0.025 & 0.025 & 0.875 \\ 0.025 & 0.025 & 0.025 & 0.875 & 0.025 & 0.025 \\ 0.45 & 0.45 & 0.025 & 0.025 & 0.025 & 0.025 \\ 0.025 & 0.025 & 0.45 & 0.025 & 0.025 & 0.025 \\ 0.45 & 0.45 & 0.45 & 0.025 & 0.025 & 0.025 \\ 0.025 & 0.025 & 0.025 & 0.025 & 0.875 & 0.025 \end{bmatrix}$$

The PageRank vector calculated by the power method is with convergence tolerance 10^{-6} is

$$\pi = [0.208, 0.103, 0.157, 0.092, 0.224, 0.216]$$

and the final rank of this network given by the PageRank algorithm is:

Page 5
Page 6
Page 1
Page 3
Page 2
Page 4

Notice that Page 5 is the most important one for this algorithm as there are three nodes

that are directly linked to it. Even if page 6 has only one node directly linked to it, that node is page 5 hence page 6 is in the second position of the ranking. The same observation can be done for page 1. Even if pages 2 and 4 have the same number of inner links than page 5 they are in the last positions.

Let's now suppose that it is known from outside that page 2 refers to an important content in the network or that page 2 can be more useful for the "random surfer" that is browsing the 6 pages. Hence we can personalize the algorithm by changing the personalization matrix to take into consideration this hypothesis keeping always the perturbation factor $\alpha = 0.15$. If we consider for each row of the new personalization matrix the distribution $\widetilde{e} = [\frac{1}{10}, \frac{1}{2}, \frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \frac{1}{10}]$ we get the following result:

$$\widetilde{G} = \begin{bmatrix} 0.015 & 0.015 & 0.015 & 0.015 & 0.015 & 0.865 \\ 0.075 & 0.075 & 0.075 & 0.925 & 0.075 & 0.075 \\ 0.44 & 0.44 & 0.015 & 0.015 & 0.015 & 0.015 \\ 0.015 & 0.015 & 0.44 & 0.015 & 0.015 & 0.015 \\ 0.44 & 0.44 & 0.44 & 0.015 & 0.015 & 0.015 \\ 0.015 & 0.015 & 0.015 & 0.015 & 0.865 & 0.015 \end{bmatrix}$$

The new PageRank vector computed is:

$$\widetilde{\pi} = [0.161, 0.206, 0.156, 0.066, 0.222, 0.189]$$

and the final rank of this network given by the algorithm is:

Page 5
Page 2
Page 6
Page 1
Page 3
Page 4

It is interesting to notice that a simple change in the personalization matrix results in a change in the standing of the web pages. Page 5 is still the first page ranked but now page 2 becomes the second most important page since we put a priori more probability to page 2 in the teleportation part of the algorithm; however notice that page 2 and page 5 have very close scores hence this result is not robust since a small change in the personalization distribution will provide different result especially in the top three pages.

Page 1, 3 and 4 are in the same relative order, indeed page 1 is still the most important of the three as it is the only one linked directly by page 2.

2.3. Development

The objectives of this project is not only to explain the different ways in which PageRank can work but also to enrich the available tools in order to let the algorithm performs in the best way in the different environments in which it is used.

Two of the main issues of the algorithm are related to the dangling nodes and the teleportation distribution. The two matters in question are really important since they are strongly related to the final result. Indeed if a "random surfer" is considered it is fundamental to understand what is the behaviour when he/she lands in a dangling node and what happens when teleportation occurs.

If we want to model the behaviour of a "random surfer" browsing the nodes in a network we need to consider that the basic approach of a uniform distribution in case of dangling page or teleportation is not satisfactory enough. Indeed the real distribution will be based on the importance of the different nodes and also on the nature and attitude of the user who is accessing the network. This is why this distribution is related to personalization of the results; it can be used to find different ranks based on the user's characteristics and personality. Indeed one of the main reason to call it personalization is because Google wanted different distribution vectors to model different types of surfers in the most accurate way. Of course we don't have any information regarding a "random surfer" browsing the internet but it is still useful to try different models for the personalization matrix and understand how the rank results behave in the different situations.

2.3.1. Personalization vector and comparison metrics

It is satisfactory to use as personalization matrix a matrix in which the vectors columns are the same; in this case we consider that in case of teleportation the user will behave in the same way even if he/she is in different pages. That's why from now on we will refer to the personalization matrix as personalization vector.

Different distributions used are:

- Uniform distribution, method with no further knowledge about the nodes.
- Weighted distribution, based on features that measure the importance of the nodes isolated from the network.
- Weighted content distribution, based on the content verification of the information in the nodes i.e. if it matches properly with the query fixed at the beginning.

Notice that even if the Uniform distribution is not satisfactory enough to model the behaviour of a "random surfer" it is still useful since it allows to observe how much the PageRank vector depends on the difference in personalization vector; somehow the Uniform distribution is used as a standard metric of comparison.

In order to compare the different results the Kendall Tau distance is used in order to quantify the similarities between the different ranks. Kendall Tau distance is a metric function that measures the number of pairwise disagreement between two ranking lists. From the algorithmic side this distance is computed with the usage of the bubble sort algorithm since it is equivalent to the number of necessary swaps to do to place one list in the same order of the other list; indeed the metric is also called bubble sort distance. The Kendall tau ranking distance between two lists τ_1 and τ_2 is

$$K_d(\tau_1, \tau_2) = |\{(i,j) : i < j, [\tau_1(i) < \tau_1(j) \wedge \tau_2(i) > \tau_2(j)] \vee [\tau_1(i) > \tau_1(j) \wedge \tau_2(i) < \tau_2(j)]\}|.$$

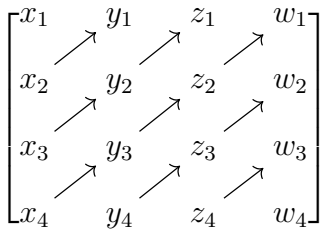
where $\tau_1(i)$ and $\tau_2(i)$ are the rankings of the element i in τ_1 and τ_2 respectively; $K_d(\tau_1, \tau_2)$ will be equal to 0 if the two lists are identical and $\frac{1}{2}n(n-1)$ if one list is the reverse of the other.

It is important to reckon that there is not a real solution to the question: what is the perfect rank for this query? The answer is subjective because it depends on the user that is asking the question. That's why it is also relevant to compare the different ranks qualitatively by asking different users to use the algorithms and leave a feedback on their behaviours; we don't expect a uniform answer but at least we will have an idea on what a sample of the users thinks about the different ranks.

When we ask the question about "comparing qualitatively different ranks" what we mean is to consider two ranks in parallel, examine the orders couple by couple and understand if it's more important the element of the first rank or the one of the second rank. More important signifies how significant and relevant an element is in terms of content and quality of the page (always considering the query searched).

Furthermore in closed environment a new rank can be obtained using a combination of the real rank given by the environment and the one given by PageRank algorithm. Indeed if the two ranks of the webpages are put respectively in the rows and the columns of a square matrix then a diagonal traversal can be computed; in this case a new result can be worked out where the final sort are the ordered elements found in the diagonal traversal where the element of the row and the element of the column matches. This is a new rank using the combination of the two ranks.

2.3.2. Diagonal traversal



In the above matrix a diagonal crossing is represented: starting from top left the traversal is done by running across each diagonal of the matrix as shown in the matrix above. In this example matrix the order of the elements given by the diagonal traversal is: x_1, x_2, y_1, x_3, y_2, z_1 and so on and so forth.

Let's now suppose to have 5 pages to be ranked: a, b, c, d and e. Let's consider two different ranks of the pages: one orders the page as a, b, e, d, c; while the other orders them as b, c, a, e, d.

We can then build a square matrix by taking the two ranks in rows and columns as shown below:

$$\begin{array}{c} \\ b \\ c \\ a \\ e \\ d \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & e & d & c \end{array} \\ \begin{bmatrix} ba & bb & be & bd & bc \\ ca & cb & ce & cd & cc \\ aa & ab & ae & ad & ac \\ ea & eb & ee & ed & ec \\ da & db & de & dd & dc \end{bmatrix} \end{array}$$

By computing the diagonal traversal of this matrix a new rank is built adding elements one by one whenever the element of the row is the same of the element of the column. In the diagonal traversal the first cell to have the same element is first row and second column: b. The second one is third row and first column: a; and so on and so forth.

The final rank given by the combination of the two is the following:

$$[b, a, e, c, d]$$

2.4. Generalized PageRank with personalization

It is not possible to use indipendent data to create a single personalization vector, in order to use PageRank the algorithm must be generalized to handle multiple personalization. Indeed if you consider internet and two different datasets: one is the dataset that provides the quality of the webpages and the other contains the preferences of the pages given by the users; to use PageRank based on both the datasets it is not possible to apply the personalization with only one vector. That's why it is useful to apply a generalization of the PageRank algorithm by using multiple personalization matrices:

$$G = (1 - \sum_j \alpha_j)\widetilde{M} + \sum_j \alpha_j E_j \tag{2.12}$$

where E_j is the personalization matrix identifies by the j^{th} dataset and α_j is the perturbation factor associated to the j^{th} dataset.

$1 - \sum_j \alpha_j$ is the weight importance given to the topology of the network with respect to the information of the nodes contained in the datasets.

If you have m different datasets and $\alpha_1 = \alpha_2 = ... = \alpha_m = \alpha/m$ (uniform weight for the datasets) then

$$G = \left(1 - \sum_j \frac{\alpha}{m}\right)\widetilde{M} + \sum_j \frac{\alpha}{m} E_j = (1 - \alpha)\widetilde{M} + \sum_j \frac{\alpha}{m} E_j \tag{2.13}$$

By searching the invariant distribution in the new Google matrix G it is possible to find the solution to the Generalized PageRank: a way to integrate PageRank with multiple different personalization.

2.5. Metrics to evaluate results

The most difficult aspect of ranking algorithms is how to evaluate their performance and efficacy. Indeed the problem of ranking is an unsupervised learning problem; there is no correct answer to the question: what is the best way to rank this list of objects? The answer is subjective and it depends on the characteristics of the objects that a user considers to make preferences.

This is why in the three different applicative cases diverse ways are considered to evaluate the results. In general the approach used is to examine the result in comparison with

another rank taken into consideration; if a second rank is present the result is firstly compared in a quantitative way using the Kendall tau distance between the two; this is done to check if the two ranks are in disagreement or if they provide similar results.

If they are dissimilar a qualitative approach is used to analyze which rank is better. This is different for each case examined as there is no absolute method to qualitatively evaluate the results; this strongly depends on what is the case that has been investigated. For each example before analyzing the results it is explained what criteria and metrics have been used to evaluate the performance of the algorithm.

3 | PubMed

PubMed is one of the most popular search engine in the world accessing an enormous archive of biomedical and life sciences journal literature. Its database contains more than 33 million citations and abstracts of biomedical literature and on an average working day, there are about 2.5 million users conducting 3 million searches and 9 million page views. Before 2018 the algorithm used by PubMed to rank the results of a specific query was the standard PubMed Best Match sort: it was based on a weighted term frequency algorithm that calculates the frequency with which terms appear in PubMed records. Those frequencies were then applied in a weighted fashion to return a ranked list of PubMed citations that match with the query terms. Recently PubMed added to the algorithm a more sophisticated part that includes machine learning to re-rank the top articles. This algorithm combines over 150 features that are helpful for finding best matching results. Most of these signals are computed from the number of matches between the search terms and the PubMed record, while others are either specific to a record (publication type, publication year...) or specific to a search (search length...). Best match sort remains a content driven algorithm that does not consider the geometry of the network built by using citations between publications: that's why it's interesting to compare Best Match sort with PageRank algorithm.

3.1. Data preparation

To build the Google G matrix for this application the first step is to create the network composed by the published articles in PubMed.

In analogy with the Google matrix built on the web it is possible to build it by:

- Considering each publication of PubMed associated to a query as a node of the graph.
- Considering as an edge of the graph each connection between two nodes i and j if article j is cited by article i.
- Considering the network as the directed graph composed by nodes and edges: this

network is a representation of the relations between the publications in PubMed.

To create this network the Python library BeautifulSoup is used in order to download the HTML of the PubMed webpages; then a process of scraping allows us to gather all the information regarding the different citations.

Now it is possible to transform the affinity matrix associated to the network into the stochastic matrix $\widetilde{M}$ by:

- Construct M using equation 2.3.
- Transform M into $\widetilde{M}$ using equation 2.4

The final matrix G is built by using equation 2.6 setting at first the perturbation factor $\alpha = 0.15$ as this is the constant chose by Google to apply the PageRank algorithm on the web.

Different personalization vectors will be used to compare different ranking results as explained in the below sections.

3.2. Development

The idea is to apply the PageRank algorithm in PubMed in order to compare its result with the one used by PubMed (Best Match sort algorithm) that uses machine learning and does not consider the topology of the network related to the system.

Another interesting aspect is to combine the two algorithms considering the rank given by Best Match sort in the personalization part of the PageRank algorithm; this implies merging the features of machine learning with the ones related to the geometry of the network of citations. Doing so we can understand the importance of the personalization vector used in the teleportation of the PageRank algorithm; indeed if the results do not change it means that the personalization is negligible, otherwise it means that different personalization vectors can alter the results leading to diverse ranks.

The first queries tested in the simulations of the algorithm by different users are related to distinct scientific topics; the topic of autism in children (Attention-Deficit/Hyperactivity Disorder (ADHD) and Applied Behavior Analysis (ABA)), the medical topic (asthma treatment guidelines) and pharmaceutical topic (ibuprofen).

In these cases the networks built are not enormous as the results given by PubMed for these queries are not more that 10000 citations. Moreover the dangling nodes are not a big issue because for the queries tested only a low number of publications have no citations;

indeed more than 98% of the publications found have at least one citation.

3.3. Webapp

To understand both from a qualitative analysis and also from a quantitative point of view we decided to build a webapp using the Streamlit library in python; in this way users from Politecnico di Milano and other universities could work with the webapp and answer in a questionnaire to some simple questions regarding their opinions on the behaviour of the different methods proposed.

Indeed the webapp shows the results of the rank of three different algorithms for different queries; moreover a search button has been added to let the users search for new queries inside PubMed. We decided to present at most the top 30 results for each query since usually this number is more than enough to let the surfers find what they are searching. The scraping considers only the results considered in the first 10 pages of PubMed results. This is to restrict the amount of time the user has to wait for getting the results. We considered the first 10 pages since it is not restricted to state that important results for the users are in the first 10 pages.

Regarding the three algorithms proposed one is the basic match sort using machine learning defined by PubMed, one is PageRank using as personalization vector a distribution that considers information of the single publications, the last one is the combination of the first and second algorithm using the diagonal traversal already introduced in the chapter *Objectives*. The three algorithms are described more in details in the following sections. It is interesting to understand if the third algorithm can spot the best peculiarities of best match sort matching them with the ones of PageRank which considers more the topology of the network. Observe that these algorithms are simply called *Algorithm 1*, *Algorithm 2* and *Algorithm 3* in the webapp so that users don't know which algorithm has been used behind the different ranks; this is done in order to avoid bias in the answers to the questionnaire.

It is important to mention that pure PageRank algorithm with no personalization or with uniform personalization does not provide good results since it occurs that for some queries top results of the rank are important publications in the network built but they don't match properly the query that the user has been searched. That's why it is reckoned that in order to find good results it is fundamental to use a personalization vector which verifies the content of the articles and considers only the once that matches with the query examined.

One method tried to take into account the content of the articles was to consider at the end of the PageRank score computations only the publications that are a good match with the query considered and remove all the ones that don't agree in terms of content with what it has been searched. This proposal has not been considered at the end because it is difficult to define a threshold to understand if an article can be a match or not with the query; therefore it is better to apply the information verification in the complete network in the teleportation of the PageRank algorithm.

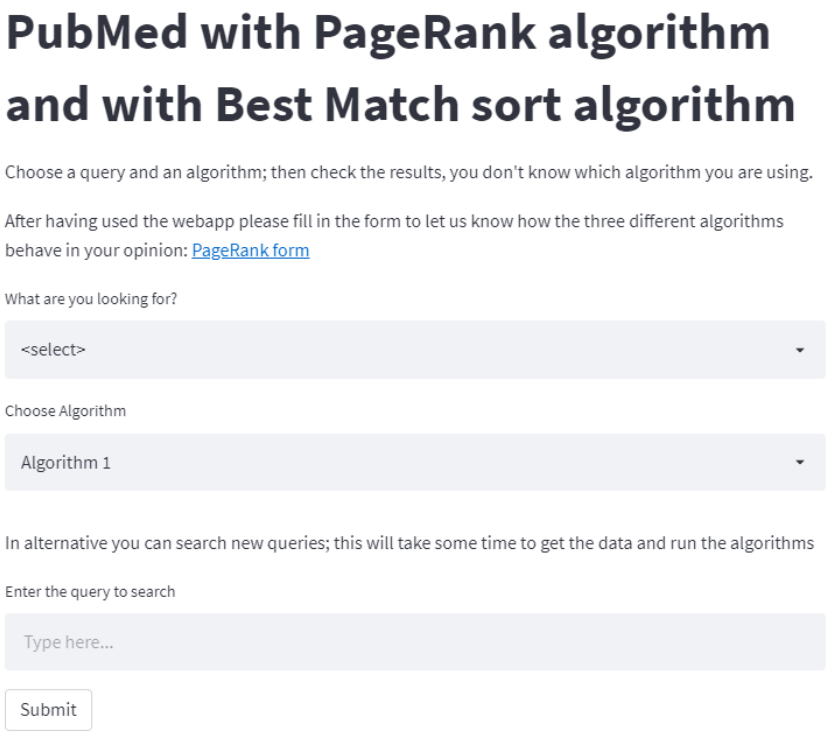


Figure 3.1: Webapp interface

3.4. Personalized distribution

3.4.1. Data

In order to build the personalization vector used in the PageRank algorithm we cooperated with Altmetric, a company that has the aim to track and analyze online activities around researches and publications from numerous diverse sources (public policy documents, social media, blogs, wikipedia...). Altmetric provided us with an API to collect many different data regarding each publication analyzed by using its DOI: the Digital Object Identifier of an article.

Data used in the analysis are the following:

- Cited by feeds count: Number of blogs that have mentioned the publication.
- Cited by posts count: A "post" is any online document that links to one or more research objects (i.e. a post is a mention or a group of mentions). This field contains the number of distinct posts that include one or more mentions of the research object in question.
- Cited by tweeters count: Number of distinct Twitter users that have mentioned the article.
- Cited by policies count: Number of policy documents that have mentioned the publication.
- Cited by patents count: Number of patents that have mentioned the publication.
- Cited by Wikipedia count: Number of citations on Wikipedia
- Cited by accounts count: The sum of all "cited by" entries in social media.
- Score: Weighted count of all of the attention a research output has received. It is based on 3 main factors: Volume, Sources and Authors.
- Readers count: Total number of readers.
- Mendeley: Number of unique Mendeley users that have added copies of a particular document to their personal library.
- Connotea: Number of readers in Connotea (until the website was available).

3.4.2. The problem

We have an unsupervised ranking problem in which we would like to create a rank of all the citations based only on the above features; this rank will be fundamental to build a probability distribution on the publications: this will be used as personalization vector. The problem can be solved in an easy way due to the fact that all the features satisfy the propriety of monotonic importance, indeed the higher the feature is and the higher the publication will be in the ranking.

Since there are too many covariates and it can be supposed that some of them are highly correlated it is convenient to perform a PCA (principal component analysis) in order to consider only the principal components that explain a high proportion of variability of the data.

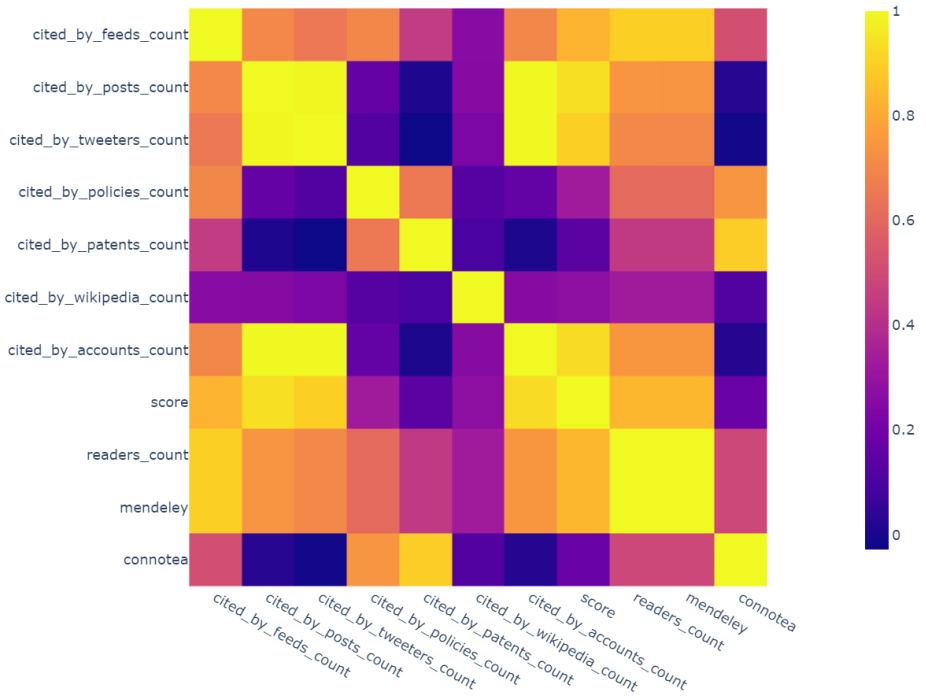


Figure 3.2: Heatmap of the covariates for the query "Denver autism"

In figure 3.2 it is observable that several features are positively correlated, hence it is useful to understand if it is possible to reduce the dimension of the dataset without losing significant information on the variability of the dataset.

3.4.3. PCA

All the results shown are related to the fixed query "Denver autism" but it is important to mention that many queries have been tested and all the results are really similar, that's why we can use a specific query to explain the general method of the analysis.

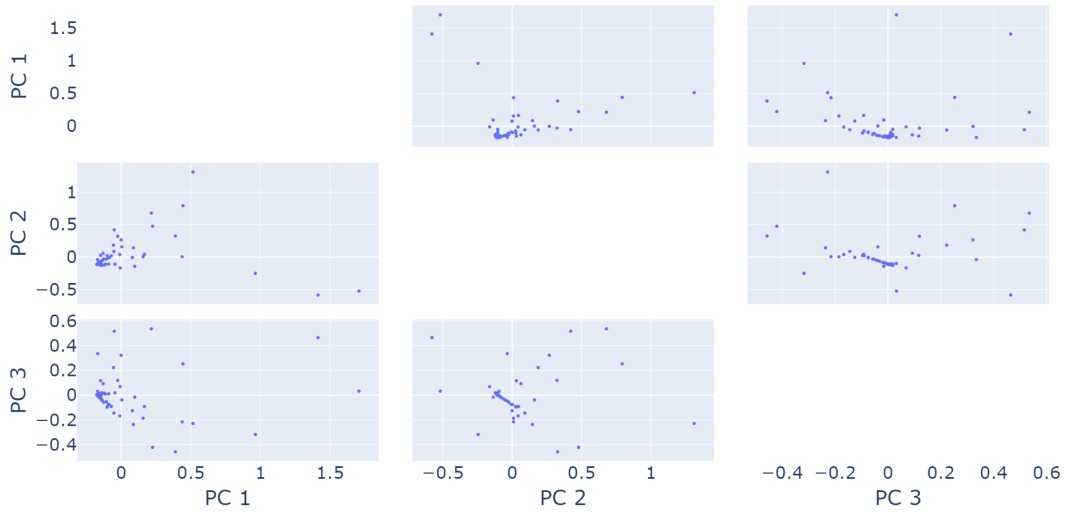


Figure 3.3: Plots of the first three PCs

In figure 3.4 it is observable that taking into consideration only the first two principal components is more than enough in order to explain more than 80% of the total variability, that's why it's satisfying to examine the first two principal components so that the new dataset has a reduced dimension of two.

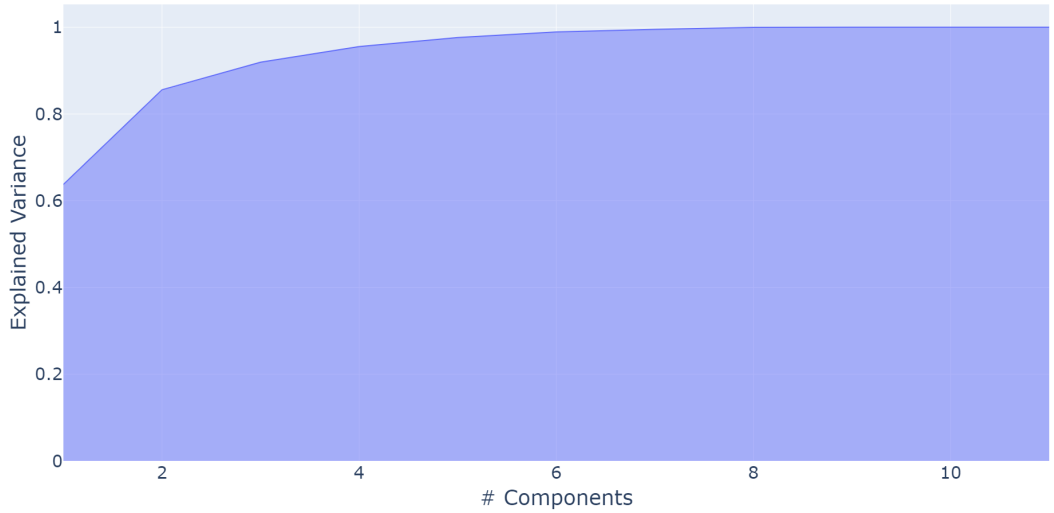


Figure 3.4: Explained variance wrt principal components

In the interest of creating a ranking based on the two principal components it is necessary to interpret the loadings to understand how the variables contribute to each one of the principal components considered. From figure 3.5 it is understandable that the first principal component is a weighted average of all the features, where the variables are all

equally balanced except for features 4, 5 and 10 (cited by patents count, cited by wikipedia count and Connotea respectively) which have lower weights. That's a great result as we can use this PC to build a rank easily since all the loadings are in agreement.

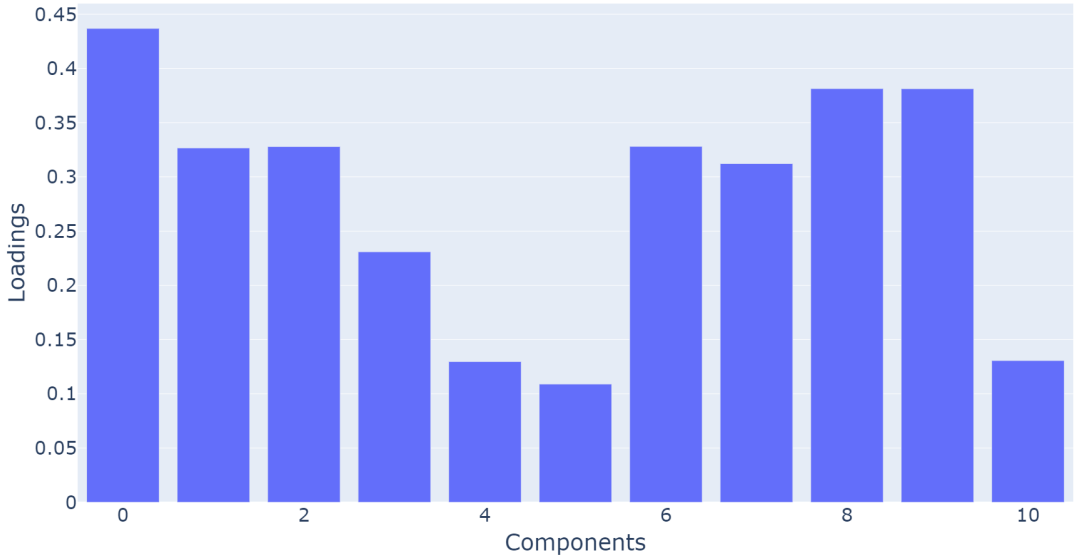


Figure 3.5: Loadings of the first PC

Regarding the second PC from figure 3.6 it is noticeable that the interpretation is more complex; this principal component represents a contrast between from one side the number of readers (components 8 and 9 are readers count and Mendeley count) and wikipedia count, on the other side the citations in the social media (components 1, 2 and 6 refer to posts, tweeters and accounts).

Many studies find out that Mendeley readers typically appear at least a year before citations due to delays between other researchers reading a paper and their new study being published. That's why this variable can help to predict the impact that a recently-published article will have in future. This means that also this second PC gathers useful information of the dataset in order to rank the publications; hence we would like to use it in the order algorithm.

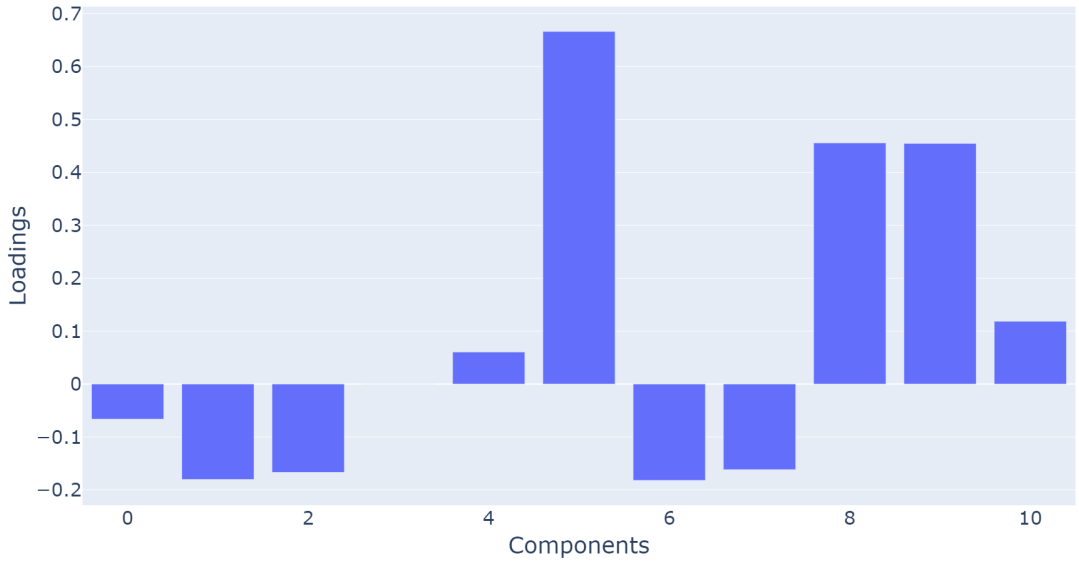


Figure 3.6: Loadings of the second PC

3.4.4. The distribution

As already discussed in the previous sections in order to build a distribution on the publications we need to use the two principal components to create a ranking. Based on the interpretation of the loadings the following ordering system has been chosen:

Algorithm 3.1 Order metric used to sort the elements

Consider two different publications i and j
Define $k > 0 \in \mathbb{R}$
if $\mathrm{PC}_1(i) > \mathrm{PC}_1(j) + k$ **then**
    i is ranked before j
else if $\mathrm{PC}_1(j) > \mathrm{PC}_1(i) + k$ **then**
    j is ranked before i
else
    if $\mathrm{PC}_2(i) > PC_2(j)$ **then**
        i is ranked before j
    else if $PC_2(j) > PC_2(i)$ **then**
        j is ranked before i
    else
        i and j are in the same position
    end if
end if

The idea behind the algorithm is simple: if the difference between the first PC of the two elements is high enough then the element corresponding to the higher PC is ranked higher, otherwise the second PC is considered and the element with the higher value of second PC is ranked higher. This algorithm is strongly related to the interpretation of the PCs; indeed when the first PCs of two elements are closed we consider the second PC that is a prediction of the impact of the articles in the future. The higher k is and the higher is the importance given to the second principal component.

Notice that even if the second PC has negative weights for some features we consider it in a positive way in the algorithm. This is due to the fact that we are examining it only when the first PCs (that is a weighted average of all the features) are very close.

To finally build the ranking of the publications the last point is to choose the parameter k. This is done from a qualitative analysis of several queries and by tuning the parameter k in the range $[0.1, 0.7]$. In the end $k = 0.35$ has been chosen.

3.4.5. Distribution for the personalization

After having built a ranking based on the Altmetric features it is finally possible to create a discete distribution over the publications. The distribution that has been chosen is the Geometric one since it is a flexible distribution for this problem.
In particular the distribution considered in the teleportation of the Markov chain random walk simulations is a Truncated Geometric (truncated since the number of publications are not unlimited); indeed let X the random variable "what is the next page chosen by the random surfer?", let i the publication in the i-th position of the rank and $F(x; p)$ the cumulative distribution function of a Geometric of parameter p evaluated in x then

$$X \sim \text{TruncGeom}(p) \Longrightarrow \mathbb{P}(X = i) = \frac{p(1-p)^{i-1}}{F(|V|; p)} \tag{3.1}$$

where |V| is the cardinality of the publications set.

The parameter p is the one that gives more or less importance to the ranking generated by the Altmetric features. Indeed if p is large it means that the probability to choose one of the first publications in the rank will be very high with respect to the others, on the other hand if p is small it means that the Geometric distribution is converging to a Uniform distribution. This is shown in figure 3.7.

As already done with parameter k in the Order algorithm also with the parameter p we proceeded by analyzing the different results qualitatively for diverse values of p in the range $[0.1, 0.8]$; too small and too high values of p have been discarded a priori in order

to avoid extreme results in the final ranking.

After having tuned the parameter for several queries the final value of p chosen is $p = 0.25$, this value is a good tradeoff to both avoid giving too much importance to the first pages and to avoid uniform weights.

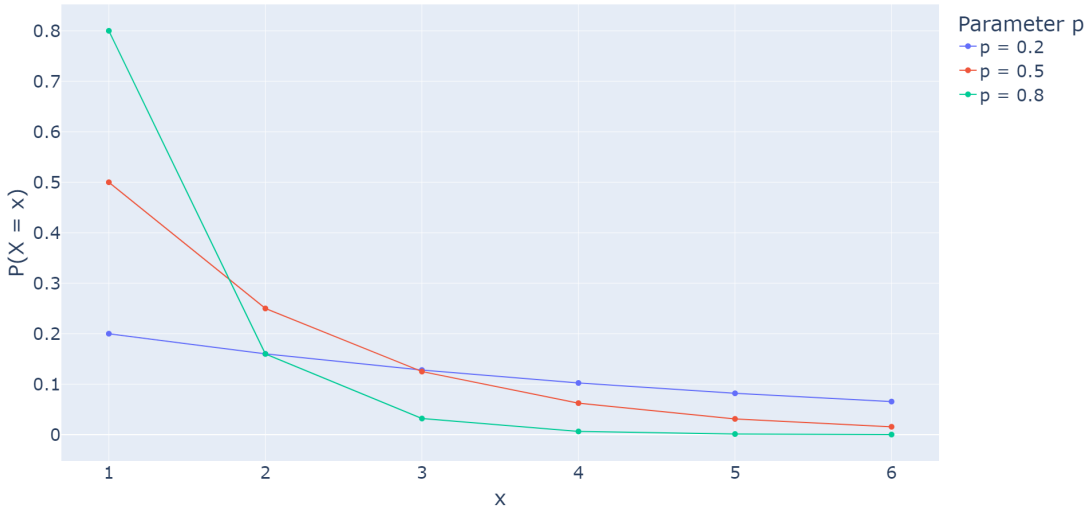


Figure 3.7: Geometric distribution for different parameters p

3.5. PageRank with two personalization

This algorithm is applicable in PubMed as for each publication we have two different types of information: one is provided by Altmetric; the other is referred to the ranking given by Best Match sort. It is known that Best Match sort uses machine learning to focus on the content of each publication and the historical information of the articles; indeed some of the features used by Best Match sort are publication year or type, number of query term matches in title and click information (one year of logs for each document). We do not have a dataset of the data used by Best Match sort but we have the final ranking given by PubMed, this contains all the information used by the machine learning algorithm.

The new Google matrix is the following:

$$G = (1 - \sum_{j=1}^{2} \alpha_j)\widetilde{M} + \sum_{j=1}^{2} \alpha_j E_j \tag{3.2}$$

where E_j is the personalization matrix identifies by the j^{th} dataset and α_j is the perturbation factor associated to the j^{th} dataset.

With the ranking given by the Altmetric features and the ranking given by Best Match sort it is possible to compute two distribution as already discussed in 3.4.5.

The two distributions are used to compose matrices E_1 and E_2 to apply in the PageRank with multiple personalization. Uniform weights $\alpha_1 = \alpha_2 = \alpha$ have been chosen and since more information on the publications have been used it is understandable to increase the importance of the distributions with respect to the topology of the network: that's why α is increased from 0.15 used in the previous algorithms to 0.2.

The new rankings computed in this way are similar to the ones operating a diagonal traversal, indeed the Kendall tau ranking correlation function proves that the two rankings are in agreement as the measure is positive and close to 1 for mostly all the queries analyzed.

3.6. Results

In this case it is simpler to evaluate the performance of the algorithm. Indeed PubMed is used by many users all over the world hence we can consider that its searching engine is a good rank to compare the result with.

This is done by analyzing the answers in the PageRank form of the webapp; the results are interesting: out of 20+ users of the app that filled the form it seems that the algorithm is performing well. *Algorithm 2* is referring to PageRank with personalization, *Algorithm 3* is Best Match sort, the PubMed one and finally *Algorithm 1* is the combination of the two rankings via diagonal traversal.

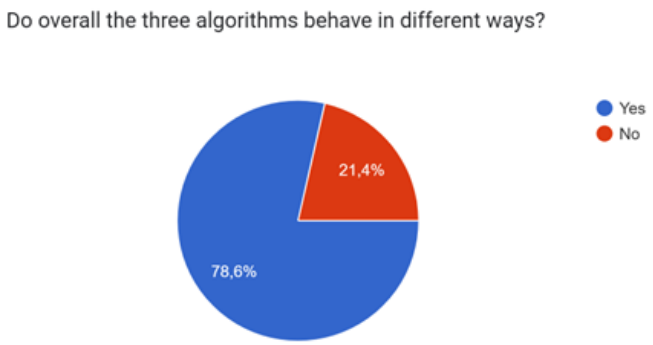


Figure 3.8: Behaviour of the three different algorithms

Firstly the majority of the users have stated that the three different algorithms behave in different ways. This is also proven by the fact that mostly all the queries searched have a Kendall tau correlation measure between the rankings of the three algorithms that is negative, indicating strong disagreement between the different results.

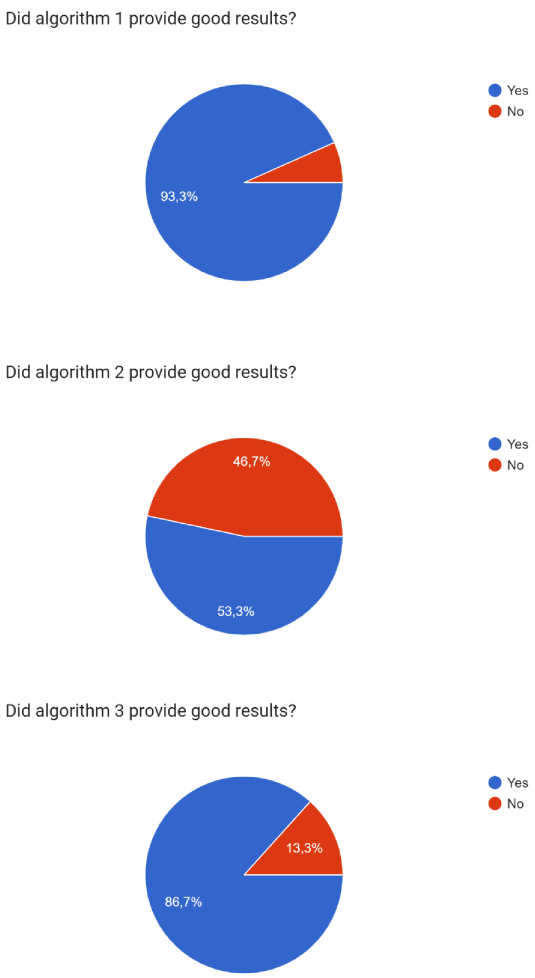


Figure 3.9: Good overall results for the three algorithms

Overall the three algorithms have provided good results for the users, more specifically *Algorithm 1* and *Algorithm 3* perform really well to spot articles that the users were looking for. As for *Algorithm 2* the analysis proves that several users are not satisfied by the rankings of the algorithm.

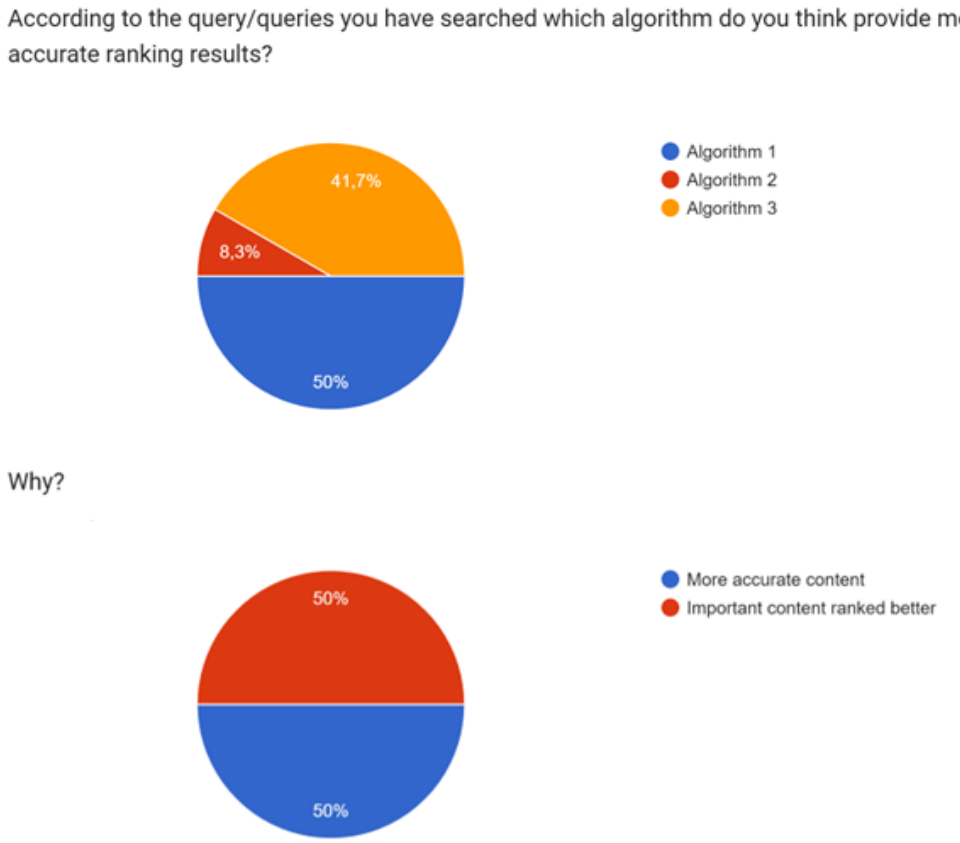


Figure 3.10: Best algorithm and why

Furthermore on average *Algorithm 2* is the less performing one to provide more accurate results to the queries; on the other hand *Algorithm 1* is the most efficient one.

The reason why PageRank with personalization is not effective is the accuracy of the content of the articles ranked, indeed the personalization does not consider how well the articles match with the query searched. This means that important content articles are ranked better, but not all of them match properly with the query in input. *Algorithm 1* seems to be able to spot the best peculiarities of best match sort with the ones of PageRank. In this way the result considers from one side the matching content between articles

and queries given by best match sort and from the other the topology of the network analyze in PageRank with personalization.

Moreover users that vote for *Algorithm 3* used very popular queries while less popular queries performed better in the ranking of *Algorithm 1*. This is expected as best match sort uses machine learning algorithms and these work better with more data in input, that's why new queries do not have enough data to allow artificial intelligence to perform well. On the other hand as PageRank does not need to train any machine learning models also not popular queries perform well.

4 | Twitter

The second application chosen to simulate the algorithm (instead of the world wide web) is Twitter; this decision has been made since Twitter is a close environment and the network of interest can be built more efficiently through the usage of a public API provided by Twitter which allows Python developers to interact with the application in an easy and fast way.

Twitter is the widest blogging social media that proves a billion tweets from many users. Each tweet refers to a specific topic, and the tweets can be retweeted or quoted by another user. A quote tweet is a retweet with an added comment which allows you to add your thoughts on the retweet while still giving the original post an exposure.
In the proposed methodology, a network graph is built from Twitter where the tweets act as nodes and retweets or quotes relations is the directed edge between the nodes i.e. two nodes i and j are connected if and only if tweet i is quoting tweet j.

Moreover it is important to consider that in Twitter the ranks of the tweets are most likely also related to what the users like based on their past activities in Twitter and on the pages that the users follow. In this case we are taking into consideration a user new to the platform that follows no one and is just searching a query; this is due to the fact that we don't have any information regarding the different users hence we cannot apply any personalization.

4.1. Data preparation

First step is to create the Google G matrix for the Twitter application.

To rank the tweets the idea is that tweets with many retweets or quotes are ranked higher; moreover tweets that have been retweeted or quoted by important tweets are ranked higher. In this situation there is still an analogy between this application and the Google problem to rank webpages.

In analogy with the Google matrix built on the web it is possible to build it by:

- Consider each tweet in Twitter that mentions a query (using hashtag) as a node of the graph.
- Consider as an edge of the graph each connection between two nodes i and j if tweet j is retweeted or quoted by tweet i.
- Consider the network as the directed graph composed by nodes and edges.

The software Gephi and its Twitter streaming importer plugin to build a graph representing the system of tweets and quotes. Now as in the previous application it is possible to transform the affinity matrix associated to the network into the stochastic matrix $\widetilde{M}$ and then into the final matrix G setting the perturbation factor $\alpha = 0.15$ to use the same Google constant.

In this case as already introduced a user new to the platform is considered: this means that the personalization vector is substituted by a uniform distribution just to make sure that the final matrix is strongly connected.

4.2. Development

After fixing a query of interest the algorithm proceeds firstly to scrape all the tweets related to the query in a certain period of time and secondly through the help of the software Gephi and its Twitter streaming importer plugin to build a graph representing the system of tweets and quotes.

The queries used in the simulations of the algorithm are related to Covid and to the war between Russia and Ukraine because these are the two main topics of 2022 and there are many data related to these queries in Twitter. The two query used are #Covid or #Covid_19 and #Ukraine or #UkraineRussiaWar.

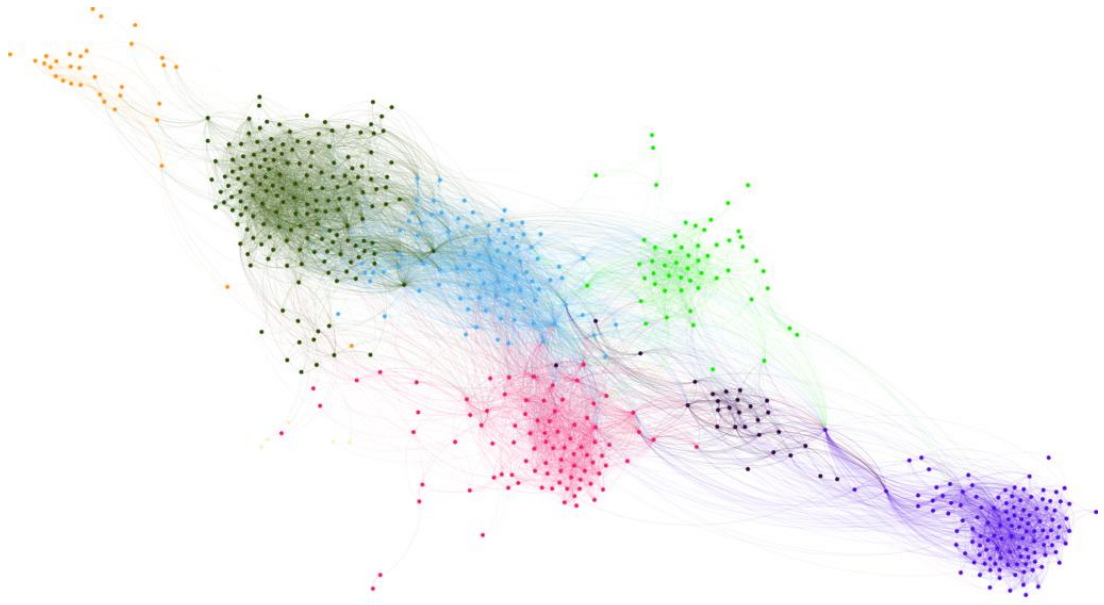


Figure 4.1: Example of Gephi network visualization.

Due to the performance of the algorithm that decreases exponentially with the growth of the netword we considered single days of tweets using networks of at most 10000 between nodes and edges in order to avoid time and memory inefficiency. This is also due to the fact that Twitter gave us capacity to analyze up to 10 millions tweets per month using their API. Moreover each time that the API is requested it works for 10000 tweets.

In this case in the Twitter environment we also compare the results of the complete network using also dangling nodes i.e. tweet with no retweets or quotes with the network built by removing tweets with really low number of relations. Indeed fixing a threshold a new network can be built considering only the nodes with total interactions that are less than the threshold. This is done because in particular environment such as Twitter the tweets with really low number of retweets or quotes can be considered negligible, indeed in the results those tweets will be the last ones in the total rank. In this case the PageRank issue related to dangling pages is not a problem since tweets with zero interaction are rarely linked to important nodes of the network. However since we don't want to lose important information the threshold is low and is tuned in order to get similar results to the one given by the complete network.

The best threshold found is 1 i.e. in order to not lose much information with respect to the complete network it is sufficient to remove from the network only the dangling nodes, nodes that don't have any retweets.

4.3. Results

In this case there is no comparison rank to take into consideration to analyze the results in relation with another rank; indeed firstly Twitter uses personalization as the ranks depend on the users and their preferences. Even considering a user that has just created an account on Twitter to minimize the preference algorithm of Twitter searching engine there is still an issue related to the fact that Twitter considers all the tweets associated to a query. This algorithm uses at maximum 10000 live tweets imported with Gephi and this makes the comparison useless.

The qualitative way to measure the performance of the algorithm is to analyze the content of the tweets ranked as top 10: in particular the content of the tweet is examined to understand how it is related to the query searched and if it contains relevant or useful information.

Due to the privacy policy of Twitter single accounts and posts can't be shown, results have to be exposed in an aggregated way after using its API.

For the queries related to Covid the algorithm seems to perform well even if no personalization has been used; indeed in the top 10 of the tweets ranked by the algorithm 5 tweets were published by doctors, 2 by professional journalists and 1 by a politician, moreover the top three doctors and their tweets were related to the analysis of the covid situations with a link of number of cases in different locations.

One issue related to the PageRank algorithm with no personalization is that it does not consider any information associated to the content of the tweet; this is proven by the result of the "covid_19" query: in the top 10 tweets one is a tweet of a famous european football club that informs its supporters that the coach of the club has been tested positive to the covid. It's clear that one limit of the algorithm is the fact that it examines only the topology of the network without taking into consideration the text and the subject of the tweets. In this case the results can be simply modified by using a personalization vector; for example a uniform distribution over the scientific and political tweets and giving null probability to the others. In this case the tweet from the football club is not in the top 10 anymore while the other tweets remain the same. Also the Kendall tau ranking correlation function confirms that the two rankings are in disagreement as the measure is slightly negative.

In this case we can describe a general user with no specific preferences satisfied by the result as the tweets contain related and useful information about the covid.

As for the queries related to the war in Ukraine the algorithm performs really well as the top results contain tweets from journalists or newspapers without any outliers; indeed other significant tweets in the rank are from important politicians in Europe.

In general from a qualitative analysis it can be stated that the algorithm performs well in Twitter; this is due to the fact that a generic user with no preferences can be satisfied by the results. This was expected since tweets with many retweets and quotes are created from popular scientists and doctors in the queries associated to covid and from journalists and politicians in the queries related to the war in Ukraine. One controversy of PageRank in this case is the fact that it considers only the networks without further analyzing the content of the tweets. This can lead to outliers in the results as discussed in the example of the "covid_19" query.

5 | Tennis

The ATP rankings are the merit-based method used by the Association of Tennis Professionals (ATP) for determining the qualification for entry as well as the seeding of players in all singles and doubles tournaments. Ranking points are awarded according to the stage of tournament reached, and the prestige of the tournament, with the four Grand Slam tournaments awarding the most points (the winner gets 2000 points). The rankings are updated every Monday, and points are dropped 52 weeks after being awarded (with the exception of the ATP Finals, from which points are dropped on the Monday following the last ATP Tour event of the following year).

5.1. The problem

In every tennis tournament there is always a problem during the matches of the tennis players; in the different tournaments there are always good courts in which players want to play and bad courts that players want to avoid. Every year this issue arises since the best tennis players want to be in the centre court that is the most prestigious one with the highest capacity.

ATP ranking is a good overall ranking for the players but it does not consider many important factors; indeed tennis players are really different based on the surfaces in which they play, it is well known that Rafael Nadal is really good in the red-clay courts (Roland Garros), while Roger Federer is considered the best in the grass court (Wimbledon). Moreover tennis players have different number of supporters independently on their ATP ranking, Gael Monfils is a French player loved by many supporters since he is a showman and his matches can be really interesting to follow.

5.2. Data preparation

First step is to create the Google G matrix for the Tennis Wimbledon application.

To rank the players using the network the idea is that player with many wins in the last

year are ranked higher. In this situation there is still an analogy between this application and the Google problem to rank webpages.

In analogy with the Google matrix built on the web it is possible to build it by:

- Considering each tennis player as a node of the graph.
- Considering as an edge of the graph each connection between two nodes i and j if and only if player j has won against player i in a match
- Considering the network as the directed graph composed by nodes and edges.

The matches examined are the ones of the most important tournaments (Grand Slams and Master 1000) in 2022 (the dataset is available in the website "http://www.tennis-data.co.uk/data.php").

Now as in the previous applications it is possible to transform the affinity matrix associated to the network into the stochastic matrix $\widetilde{M}$ and then into the final matrix G setting the perturbation factor $\alpha = 0.15$ to use the same Google constant.

Multiple personalization vectors will be used to compare different ranking results as explained in the below sections. PageRank with multiple personalization will be used.

5.3. Development

This experiment has the objective to compute the PageRank of the tennis players considering all the factors mentioned above in the teleportation/personalization part of the algorithm, in this way tennis players are ranked in different tournaments in order to choose which one deserves more the best courts available in the tournament; in particular the algorithm is applied to the next important tournament which is Wimbledon.

Regarding the personalization vectors to use in the PageRank algorithm three distributions are built on all the nodes of the network:

- The first type of information used is the ATP ranking since it's an overall ranking of the tennis players active in 2022.
- The second one is the number of followers of the tennis players on instagram. This number is extremely correlated with the number of supporters that the players have.
- The third one is based on the tennis court in which the analysis is done, in this case the algorithm is applied to Wimbledon 2022 which is the next important tournament available. Wimbledon is a grass court hence a ranking of the players in grass is

computed (using the information of 2021 tournaments in grass), then a distribution of all the players is built.

The Geometric distribution has been chosen since it's a distribution on discrete values and it is flexible with the change of the parameter p, as already discussed the parameter p is set equal to 0.2.

The three distributions have been used in the teleportation part of the PageRank algorithm in order to consider not only the topology of the network but also the information of the single nodes; hence PageRank with multiple personalization has been chosen.

The final ranking found with this method is the following:

PageRank with personalization

	Tennis player
1	Nadal R.
2	Alcaraz C.
3	Djokovic N.
4	Zverev A.
5	Tsitsipas S.
6	Medvedev D.
7	Fritz T.
8	Berrettini M.
9	Ruud C.
10	Kecmanovic M.
11	Davidovich Fokina A.
12	Hurkacz H.
13	Korda S.
14	Norrie C.
15	Auger-Aliassime F.
16	Monfils G.
17	Dimitrov G.
18	De Minaur A.
19	Rublev A.
20	Kyrgios N.
21	Sinner J.
22	Cilic M.
23	Schwartzman D.
24	Murray A.
25	Opelka R.

Table 5.1: Ranking of tennis players for Wimbledon tournament given by PageRank.

5.4. Results

Firstly it has to be considered that there is substantial difference between the first two applicative cases and this one. Indeed in this case the PageRank algorithm is used to automate the process of choosing the courts in which the players will play their games; this procedure is something done manually by the ATP tennis association.

If the results are similar it means that the algorithm can be used to automate this process of selections of players and courts. Indeed in this case to evaluate the results of the algorithm it is sufficient to compare the choices made by the Wimbledon organizers with the ones that the algorithm would choose. Every day of the tournament it is possible to check in which courts the players played and compare this with what the algorithm would have suggested.

Not all the days will be shown in the tables below but only relevant days, for the others the analysis is similar.

5.4.1. Day 4

Court	Winner	Loser
Centre Court	Rafael Nadal	Ričardas Berankis
Court 1	Stefanos Tsitsipas	Jordan Thompson
Court 1	Alex De Minaur	Jack Draper
Court 2	Nick Kyrgios	Filip Krajinović
Court 3	Liam Broady	Diego Schwartzman
Court 3	Daniel Elahi Galán	Roberto Bautista Agut

Table 5.2: Games during the fourth day.

These decisions are all in agreement with the ranking given by the algorithm, indeed Nadal deserved Centre court, Tsitsipas and De Minaur are over Kyrgios hence they merits court 1 while Kyrgios has a better positions than tennis players in court 3 and this justifies why he played in court 2.

5.4.2. Day 5

Court	Winner	Loser
Centre Court	Novak Djokovic	Miomir Kecmanović
Centre Court	Cameron Norrie	Steve Johnsonn
Court 1	Carlos Alcaraz	Oscar Otte
Court 2	Jannik Sinner	John Isner
Court 2	Frances Tiafoe	Alexander Bublik
Court 3	Jack Sock	Maxime Cressy
Court 3	Tommy Paul	Jiří Veselý

Table 5.3: Games during the fifth day.

During the fifth day there is one difference since for the algorithm Carlos Alcaraz would have deserved to play in the centre court instead of Cameron Norrie. Apart from this important decision the others are in agreement with PageRank ranking.

5.4.3. Day 7

Court	Winner	Loser
Centre Court	Novak Djokovic	Tim van Rijthoven
Centre Court	Jannik Sinner	Carlos Alcaraz
Court 1	Cameron Norrie	Tommy Paul
Court 2	David Goffin	Frances Tiafoe

Table 5.4: Games during the fifth day.

In this case the organizers change positions between Alcaraz and Norrie as the first is playing in the centre court while the second is playing in court 1, the opposite of the decision in day 5. This is in agreement with the algorithm as both Alcaraz and Djokovic are playing in the centre court.

5.4.4. Day 8

Court	Winner	Loser
Centre Court	Nick Kyrgios	Brandon Nakashiman
Centre Court	Rafael Nadal	Botic van de Zandschulp
Court 1	Taylor Fritz	Jason Kubler
Court 2	Cristian Garín	Alex De Minaur

Table 5.5: Games during the fifth day.

During the eighth day the only controvertial decision is related to Kyrgios' match as for the algorithm he probably would have deserved to play in court 1.

In general apart from singular outliers decisions it seems that the algorithm is in agreement with the choices of the organizers of Wimbledon tournament.

It is interesting how in this case PageRank with multiple personalization vectors could consider different metrics such as how many followers the players have or what the performance of the players in Wimbledon (grass court) is; the final result is a combination of the topology of the network with information regarding players and tournament.

6 | Related works

In the original work [11] of L. Page and S. Brin they proposed the solution to the problem of a not strictly connected graph: the teleportation vector. They didn't mention any details to what vector to use in order to both have a strictly connected network and make the solution better for the users. Many attempts have been made to enrich or modify the algorithm in order to obtain better results. For example [16] tackle the link spamming problem proposing an algorithm for computing a complete set of independent eigenvectors for the second eigenvalue helping to detect link spamming. In our work we focus on trying to obtain better results exploiting some personalization vectors; in general personalization vectors can be used to personalize the ranking using the information and preferences of the users or, as in these applications, knowledge of the nodes in the network. Indeed in this work we did not consider the preferences of the users using the algorithm, still we inspected different personalization vectors to make the algorithm more powerful. As applied both in PubMed and in the Tennis Wimbledon tournament, it is possible to use any information associated to the nodes of the graph to combine the topology of the network and the information of the nodes:

- In the PubMed application we have used data collected by Altmetric, a company that tracks and analyzes online activitites around researches and publications.
- In the Tennis application we have used multiple personalization vectors considering ATP ranking of the players, number of followers of the tennis players on Instagram and the ability of the players in tournaments played on grass.

Another important aspect of the algorithm is that PageRank can only be used when a list of objects can be represented by a meaningful directed graph; this is not always simple to apply. In the three applications three different types of network are built; these graphs are constructed in analogy with the ones used in articles already published:

– In PubMed, according to article [18], "High citation rates are proposed to be predictive of article quality, thus in turn, scientific importance". On the other hand citation count can be not accurate and fragile, hence article [18] proposes PageRank since "Much as web pages are interconnected through hyperlinks, scientific articles

are themselves linked via their citations". But one of the issue related to PageRank with no personalization, as a result of the article [18], was that "A statistically significant correlation between PageRank and citation count was observed with a high correlation coefficient ($R = 0.905$)".

That's why in this work PageRank with two personalization vectors is proposed; this new algorithm considers both the topology of the graph but also analyses two types of knowledge: the first one is described by the features from Altmetric detailed in 3.4.1, the second one considers the results given by PubMed with Best Match sort. Indeed PubMed recently changed its ranking algorithm with Best Match sort that uses only machine learning; according to article [4] "the Best Match algorithm is trained with past user searches with dozens of relevance-ranking signals (factors), the most important being the past usage of an article, publication date, relevance score, and type of article". In the webapp we have seen that PageRank with multiple personalization performs slightly better than best match sort according to a sample of 20+ users. Of course this algorithm is only a prototype as it is not been tested by a enough populated sample of users; still this is an important result to consider when comparing pure machine learning algorithms and the ones considering only the network.

– In the Tennis application we have built the graph in analogy with article [3]: as described in the article "We represent the data set as a network of contacts between tennis players. This is a very natural representation of the system since a single match can be viewed as an elementary contact between two opponents. Each time the player i plays and wins against player j, we draw a directed connection from j to i".

  Even if the graph is built in the same way the applications were different: in article [3] the author uses all the matches played by professional players between 1968 and 2010 to overall rank all the professional players. On the other hand in this work we only examined matches in 2022 and considered different information of the players (inside the personalization vectors) to rank them specifically for the Wimbledon tournament. Moreover the final purpose of this work was not just to rank the players but to use this rank in order to choose in which courts the players will have to play in Wimbledon in an automatic way.

– In twitter we have built the graph in analogy with articles [12] and [7] with the difference that in the two articles the nodes of the graph were the Twitter accounts; as explained in [12] "We represent Twitter sharing diffusion networks as directed,

unweighted graphs following this: for each unique URL we process all tweets containing that hyperlink and build a graph where each node represents a unique user and a directed edge is built between two nodes whenever a user re-tweets/quotes, mentions or replies to another user". This basically means that in the new user graph, an edge from user x to another user y indicates that at least one tweet from user x has retweeted one or more tweets from user y. In our application the nodes of the graph represent the tweets and an edge from tweet x to tweet y indicates that tweet x retweets or quote tweet y. A similar approach is described in [13] where the same network graph is used to calculate the Page Rank score and to compare it with other node properties like centrality. Moreover [8] exploits a similar technique to rank topical experts and [6] applies it to a very narrow twitter category. Surveys of other attempts to measure user influence on Twitter are in [14] and in [2]. This type of approach is not the only possible one, a long research stream started from studies like [1] and investigating intriguing ideas such as mixing page rank and clustering as in [9] oe exploiting Markov chains as in [10].

Moreover the application in our work has the purpose to rank the tweets in order to show them to the users while the two articles taken in comparison focus more on the credibility of Twitter users and classification of malicious information.

Other useful applications can be found in article [5] such as bookrank to rank and suggest books to the users or TrustRank and BadRank to provide information on the "spaminess" of particular pages. For example TrustRank has the purpose to combat spam by filtering the web based upon reliability. The method is a semi-automated process that firstly selects a small set of seed pages to be evaluated by an expert. Once the reputable seed pages are manually identified, a crawl extending outward from the seed set looks for similarly reliable and trustworthy pages.

Another way to use PageRank to filter spam pages is also presented in the work [17]; in this thesis the author applies particular personalization vector optimizing it by suppressing the effect of link spamming. Indeed firstly the algorithm finds the spamming pages since one of the effective methods of link spamming is creating irreducible subsets in the original matrix. Creating irreducible subsets is very effective, that is because if a surfer gets there, it cannot leave by following outlinks and can only exits the subset by using teleportation.

All these considerations can conclude that this work differs from the other applications mostly because of the usage of the personalization vectors, firstly because different types of knowledge is considered to rank the objects and secondly because of PageRank with multiple personalization: the new algorithm to extend PageRank with personalization by

using multiple vectors.

7 | Conclusion

In conclusion this work has analyzed different ways to use and integrate PageRank with other algorithms.

Generalized PageRank allows PageRank to be used in combination with other rankings, this empowers to integrate PageRank with whichever other algorithms that rank a list of objects.

One downside is that the basic PageRank cannot be used in all the contexts but only when the lists of objects to be ranked can form a meaningful directed graph, meaning a network where the score given to the nodes (to rank the nodes in the output) must meet two requirements:

- it must be high if it refers to a node linked by many other nodes.
- it must be high when referring to a node linked by very significant nodes.

It is not always easy to create this network but this work analyzed three completely different environments where a network can be built.

This work has the scope to apply PageRank in different contexts to prove that this algorithm is still valuable to rank a list of objects connected one another. The algorithm is integrated by using statistical tools to make it more powerful for each environment analyzed. The final aim is to enrich the available tools that can be used to solve the so-called ranking problems.

By using a truncated geometric distribution or another discrete distribution, as examined in PubMed analysis, it is possible to create a distribution on the set of nodes starting from a rank. This distribution can be used in PageRank as the personalization vector. When multiple ranks are available it is possible to generalize it by applying the generalized PageRank that uses multiple personalization vectors.

The problem of ranking a list of objects is a very complex problem as there is no absolute solution. Indeed ranking the webpages for the users can depend on the user that is using the algorithm, the best solution for the user is subjective. In the PubMed analysis users

were more satisfied by the results given by PageRank in comparison to the ones given by Best Match sort, this means that on average the algorithm is performing well to spot the best articles to show to the users.

In the Tennis analysis there is also no best solution but for the Wimbledon tournament the algorithm has very similar results to the choices made by the ATP organizations. This means that in this situation PageRank is helpful to automate the process of choosing which player is playing in which court.

As already stated and described in general it is not always easy to build the graph as this has to represent a meaningful relation between the objects to rank, otherwise the final rank cannot be powerful.

From these considerations we can state that PageRank can still have an impact on several applications. Moreover the generalization of PageRank with multiple personalization allows developers to use it in combinations with other ranking algorithms, making the tool even more powerful.

7.1. Algorithm performance

As already stated in the introduction this work considers only ways to build a powerful tool result oriented; this project does not take into consideration the time complexity of the algorithms. Indeed this algorithm is developed only using Python.

For all the applications data is not stored locally but it is downloaded every time that the algorithm runs. PubMed webapp is the only exception as each query searched online is stored server side for 10 days.

For Twitter there was a big restriction given by the API that gives a maximum capacity of tweets scraped every day and every month. For Tennis only professional players competing in tournaments masters 1000 and grand slams are considered hence there was no issue related to time performance as there were not more than 50 players in the network.

PubMed is the only application where we decided to consider only the first 10 pages of results in order to avoid the users to wait for too much time to get the results. We now do not consider the time needed to scrape the results in PubMed and just consider the time needed to compute the invariant distribution.

Here is the graph that represents the time used by the algorithm to compute the invariant distribution against the number of pages considered in the scraping data preprocessing, this is applied by using a convergence tolerance of 10^{-6}:

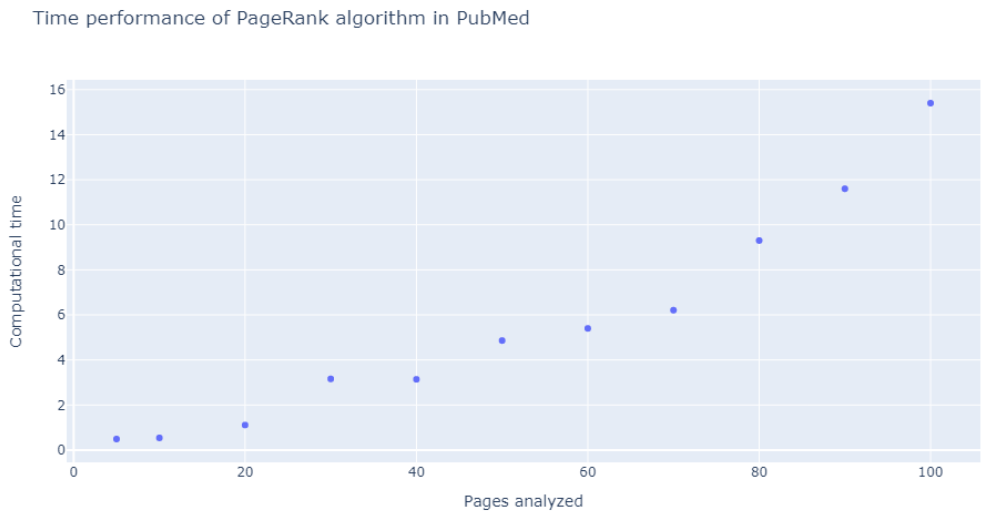


Figure 7.1: Time performance of the algorithm in PubMed

The expected result is an exponential function but of course since we are not using too many pages here it seems more a quadratic function, probably representing the first branch of the exponential function.

Even if in this work the time performance of the algorithm is not considered we can state that for datasets up to 1000 data the algorithm performs well as no more than 15 seconds are needed to compute the invariant and generate the ranking of the articles. Of course Google has to dial with enormous datasets and this algorithm could not work in such an environment. Still this algorithm can be used in many other situations where the first dataset is not enormous or when a preprocessing algorithm is performed to filter only the top 1000 nodes in the network.

Bibliography

[1] M. Cha, H. Haddadi, F. Benevenuto, and K. Gummadi. Measuring user influence in twitter: The million follower fallacy. *Proceedings of the International AAAI Conference on Web and Social Media*, 4(1):10–17, May 2010. doi: 10.1609/icwsm.v4i1.14033. URL `https://ojs.aaai.org/index.php/ICWSM/article/view/14033`.

[2] O. K. Chien, P. K. Hoong, and C. C. Ho. A comparative study of hits vs pagerank algorithms for twitter users analysis. In *2014 International Conference on Computational Science and Technology (ICCST)*, pages 1–6, 2014. doi: 10.1109/ICCST.2014.7045007.

[3] Filippo Radicchi. Who Is the Best Player Ever? A Complex Network Analysis of the History of Professional Tennis. *PLOS One*, 2011.

[4] N. Fiorini, K. Canese, G. Starchenko, E. Kireev, W. Kim, V. Miller, M. Osipov, M. Kholodov, R. Ismagilov, S. Mohan, J. Ostell, and Z. Lu. Best match: New relevance search for pubmed. *PLOS Biology*, 8 2018.

[5] D. F. Gleich. Pagerank beyond the web. *SIAM Review*, 1 2015.

[6] A. A. Hamed and A. Zia. Mining climate change awareness on twitter: A pagerank network analysis method. In *ICCSA*, 2015.

[7] A. Heavey and E. Karagoz. Analysing credibility of twitter users using the pagerank algorithm. Master's thesis, KTH, 2017.

[8] P. Lahoti, G. De Francisci Morales, and A. Gionis. Finding topical experts in twitter via query-dependent personalized pagerank. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, page 155–162, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349932. doi: 10.1145/3110025.3110044. URL `https://doi.org/10.1145/3110025.3110044`.

[9] A. Naik, H. Maeda, V. Kanojia, and S. Fujita. Scalable twitter user clustering

approach boosted by personalized pagerank. *International Journal of Data Science and Analytics*, 6:297–309, 2017.

[10] A. N. Ngaffo, W. El Ayeb, and Z. Choukair. Mining user opinion influences on twitter social network: Find that friend who leads your opinion using bayesian method and a new emotional pagerank algorithm. In *2019 15th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 680–685, 2019. doi: 10.1109/IWCMC.2019.8766571.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 11 1999. After publishing this work L. Page and S. Brin founded Google.

[12] F. Pierri, C. Piccardi, and S. Ceri. Topology comparison of twitter diffusion networks effectively reveals misleading information. *Scientific Reports*, 1 2020.

[13] S. Priyanta and I. N. P. Trisna. Social network analysis of twitter to identify issuer of topic using pagerank. *International Journal of Advanced Computer Science and Applications*, 10(1), 2019. doi: 10.14569/IJACSA.2019.0100113. URL `http://dx.doi.org/10.14569/IJACSA.2019.0100113`.

[14] F. Riquelme and P. González-Cantergiani. Measuring user influence on twitter: A survey. *Information Processing and Management*, 52(5):949–975, 2016. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2016.04.003. URL `https://www.sciencedirect.com/science/article/pii/S0306457316300589`.

[15] E. Salinelli and F. Tomarelli. *Discrete Dynamical Models*. Springer, 2015.

[16] A. Sangers and M. B. van Gijzen. The eigenvectors corresponding to the second eigenvalue of the google matrix and their relation to link spamming. *Journal of Computational and Applied Mathematics*, 277:192–201, 2015. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2014.09.014. URL `https://www.sciencedirect.com/science/article/pii/S0377042714004105`.

[17] J. Tjan. How to optimize the personalization vector to combat link spamming. Master's thesis, Delft, 6 2016.

[18] E. Yates and L. Dixon. Pagerank as a method to rank biomedical literature by importance. *Source Code for Biology and Medicine*, 12 2015.

A | Appendix A

Theorem A.1. *Suppose M is stochastic and P is a sequence that satisfies:*

$$\boldsymbol{P}_{k+1} = M\boldsymbol{P}_k \tag{A.1}$$

such that $M = [m_{ij}]$ and $\boldsymbol{P}_k = [\boldsymbol{P}_k^1, \boldsymbol{P}_k^2, ..., \boldsymbol{P}_k^n]'$. If the following condition

$$\begin{cases} 0 \leq \boldsymbol{P}_k^j \leq 1, \ \ j = 1, ..., n \\ \sum_{j=1}^{n} \boldsymbol{P}_k^j = 1 \ \forall k \in \mathbb{N} \end{cases} \tag{A.2}$$

holds for a fixed value $\widetilde{k}$, then it holds for all $k > \widetilde{k}$.

Proof. Relation A.1 says that $\mathbf{P}_{k+1}$ is a linear combination of the columns of M whose coefficients are the components of $\mathbf{P}_k$. Since $m_{ij} \geq 0$ and $\mathbf{P}_k^j \geq 0 \forall i, j$ then $\mathbf{P}_{k+1}^j \geq 0 \ \forall j$. Furthermore

$$\sum_{j=1}^{n} \mathbf{P}_{k+1}^j = \sum_{j=1}^{n} \left(\sum_{h=1}^{n} m_{jh} \mathbf{P}_k^h \right) = \sum_{h=1}^{n} \left(\sum_{j=1}^{n} m_{jh} \mathbf{P}_k^h \right) = \sum_{h=1}^{n} \left(\sum_{j=1}^{n} m_{jh} \right) \mathbf{P}_k^h = \sum_{h=1}^{n} \mathbf{P}_k^h = 1 \tag{A.3}$$

It follows $\mathbf{P}_{k+1} \leq 1 \ \forall j$.

The validity of A.1 has been proven for k $= \widetilde{k} + 1$. The general case (k $> \widetilde{k}$) follows by induction. □

Theorem A.2 (Markov-Kakutani)**.** *A transition matrix on a finite set of states Ω has always at least an invariant probability distribution.*

Proof. Let M be a Markov matrix on a finite set of states. The existence of an invariant probability distribution for M coincides with the existence of a fixed point for the

continuous function $M : S \to S$. Given $\mathbf{V} \in S$, $k \in \mathbf{N}$ 0 we define

$$\mathbf{V}_k = \frac{1}{k} \sum_{h=0}^{k-1} M^h \mathbf{V} \tag{A.4}$$

Then $\mathbf{V}_k \in S \; \forall k$, thanks to $A.1$.

By the Bolzano-Weierstrass Theorem, there exist a subsequence $\mathbf{V}_{k_l}$ and $\mathbf{W} \in S$ such that

$$\lim_{l \to +\infty} \mathbf{V}_{k_l} = \mathbf{W} \tag{A.5}$$

$\mathbf{W}$ is an invariant probability distribution because, after cancelling equal terms in the two summations,

$$\mathbf{V}_{k_l} - \mathbf{W} = \frac{1}{k_l} \left(\sum_{h=0}^{k_l-1} M^h \mathbf{V} - \sum_{h=0}^{k_l-1} M^{h+1} \mathbf{V} \right) = \frac{1}{k_l} \left(\mathbf{V} - M^{k_l} \mathbf{V} \right) \tag{A.6}$$

and then, passing to the limit $l \to +\infty$ in both sides and taking into account that $k_l \to +\infty$, $\|\mathbf{V}\|_{\mathbb{R}^n} \leq 1$ and $\|M^{k_l}\mathbf{V}\|_{\mathbb{R}^n} \leq 1$, it follows that $(\mathbf{V} - M^{k_l}\mathbf{V})/k_l \to 0 \Rightarrow$

$$\lim_{l \to +\infty} \mathbf{V}_{k_l} = \mathbf{W} \tag{A.7}$$

□

Theorem A.3 (Perron–Frobenius). . *If $M >> 0$, meaning $M = [m_{ij}]$ where $m_{ij} > 0$, then its eigenvalue of maximum modulus (called dominant eigenvalue), denoted by $\lambda_M > 0$, is unique, real, greater than zero and simple (algebraically and, therefore, geometrically). Furthermore, there exists a strictly positive eigenvector $\boldsymbol{V}_M$ (called dominant eigenvector) associated to $\lambda_M > 0$.*

Theorem A.4. *Given a Markov chain with n states and transition matrix $M = [m_{ij}]$, if there exists the limit of a column of M_k as $k \to +\infty$, namely $\exists j \in 1, 2, ..., n : \forall i \in 1, 2, ..., n$ $\exists \lim_{k \to +\infty} m_{ij}^{(k)} = \beta_i$ (where $m_{ij}^{(k)} = [M^k]_{ij}$ then $\beta = [\beta_1, ..., \beta_n]$ is an invariant probability distribution.*

Proof. $\beta_i \geq 0$ since M is a Markov matrix. Moreover

$$\sum_{i=1}^{n} \beta_i = \sum_{i=1}^{n} \lim_k m_{ij}^{(k)} = \lim_k \sum_{i=1}^{n} m_{ij}^{(k)} = 1. \tag{A.8}$$

Furthermore

$$\beta_i = \lim_k m_{ij}^{(k)} = \lim_k m_{ij}^{(k+1)} = \lim_k \sum_{s=1}^{n} m_{is} m_{sj}^{(k)} = \sum_{s=1}^{n} \lim_k m_{is} m_{sj}^{(k)} = \sum_{s=1}^{n} m_{is}\beta_s. \tag{A.9}$$

Therefore $M\beta = \beta$. □

Theorem A.5. *Each regular Markov chain with n states has a unique invariant probability distribution* $\boldsymbol{P}$*: the positive eigenvector (probability distribution) associated to the dominant and simple eigenvalue 1. If* M *is the stochastic matrix of a regular Markov chain and* $\boldsymbol{P}$ *is the corresponding invariant probability distribution, then all the eigenvectors and generalized eigenvectors of* M *different from (multiplies of)* $\boldsymbol{P}$ *are orthogonal to the uniform probability distribution* $[1/n, 1/n, ..., 1/n]'$ *which is the (strictly positive) dominant eigenvector of* M'*. Moreover, for every initial probability distribution* $\boldsymbol{P}_0$*:*

$$\lim_k M^k \boldsymbol{P}_0 = P \tag{A.10}$$

Moreover all the columns of M^k *converge to* $\boldsymbol{P}$ *as* $k \to \infty$*:*

$$\lim_k M^k = [\boldsymbol{P}, \boldsymbol{P}, ..., \boldsymbol{P}] \tag{A.11}$$

Theorem A.6. *if* A *is a primitive matrix of order n,* $\boldsymbol{V^A}$ *is the positive eigenvector of norm 1 associated to the dominant eigenvalue* $\lambda_A = \lambda_1 > |lambda_2| \geq ...|\lambda_n|$*, then:*

- *if* $\lambda_2 \neq 0$ *then as* $k \to +\infty$

$$A^k = (\lambda_A)^k \boldsymbol{V}^A (\boldsymbol{V}^A)' + \mathcal{O}\left(k^{m_2-1}|\lambda_2|^k\right) \tag{A.12}$$

  where m_2 *is the algebraic multeplicity of* λ_2*.*

- *if* $\lambda_2 = 0$ *then for* $k \geq n-1$

$$A^k = (\lambda_A)^k \boldsymbol{V}^A (\boldsymbol{V}^A)' \tag{A.13}$$

List of Figures

2.1 First basic example of a graph. 6

3.1 Webapp interface 18
3.2 Heatmap of the covariates for the query "Denver autism" 20
3.3 Plots of the first three PCs 21
3.4 Explained variance wrt principal components 21
3.5 Loadings of the first PC 22
3.6 Loadings of the second PC 23
3.7 Geometric distribution for different parameters p 25
3.8 Behaviour of the three different algorithms 26
3.9 Good overall results for the three algorithms 27
3.10 Best algorithm and why 28

4.1 Example of Gephi network visualization. 33

7.1 Time performance of the algorithm in PubMed 51

List of Tables

5.1 Ranking of tennis players for Wimbledon tournament given by PageRank. . 40
5.2 Games during the fourth day. 41
5.3 Games during the fifth day. 42
5.4 Games during the fifth day. 42
5.5 Games during the fifth day. 43

Acknowledgements

As with many researches that result in writing a thesis on the cover there should not be only the name of the researcher but also the names of all the heroes that make this work happen providing, at different levels and degrees, assistance, encouragement and guidance, and without whom I would not have succeeded.

I am very grateful to all the people who dedicated me time, effort and love during the last years; without them it could not have been possible for me to achieve my final academic challenge: gain the Master degree.

Firsly I would like to thank my professor and supervisor Alessandro Campi, for his help and support during the last year. I owe him a lot not only for the thesis but also for the passion that he gave me during the teaching, his passion enticed me to become a software engineer and also to be a better person.

Secondly my family who always accompanied me and supported me during the last 5 years. I always tried to give my best because you were an example to follow for me and I wanted to make you proud of me. A special thanks to my mum who always tried to drive me on the right path, even if our relation was not always easy we both know that we love and care each other, and that's what eventually matters.

I would like to extend my gratitude to all the people who took time to use the webapp, this was really useful and your suggestions made the app better and this work concrete.

A huge thanks to my friends and fellow students, without you all this trip could not be possible. We faced many challenges together and we overcame many obstacles together. Most importantly we always worked together as we were part of a team. I learnt a lot from you and you were really an example for me, with you I felt I could push myself beyond my limits and I learnt that nothing is impossible if you do it with your friends.

Last, but most importantly, I would like to thank my girlfriend Laura. You always were there to support, love and motivate me. These were 5 long years and we also had to face many obstacles together as a couple; we always did it, no matter how hard it was. In the end we arrived here together. Thank you, my love.