# Link Prediction for PIR

## DD2477 Search Engines and Information Retrieval Systems

## VT24

**Titouan Mazier** **Lorenzo Sibille** **Tristan Perrot** **Hani Anouar Bourrous**

### Abstract

The most popular search engines do not rely solely on TF-IDF or PageRank measurements to deliver search results for a given query but also incorporate the user information. We propose a search engine application that exploits search logs, which contain past searches and the corresponding "successful" search results, to provide personalized results to users. Based on the logs dataset AOL4PS, this project builds a query search system using Elasticsearch as the search engine and re-ranks the documents based on graph link-prediction techniques to provide personalized results. We designed an interface for users to input queries and select the desired document from the displayed ones. Experiments conducted on this search engine application using multiple queries show that the reranking method adopted allows a significant improvement in the accuracy of the results. The code for the implementation is available on GitHub at https://github.com/ttperr/Link-Prediction-for-PIR

## 1 Introduction

Historically, search engines retrieved documents based on measures computed with respect to the query alone. However, this "one-size-fits-all" approach shows intrinsic limitations in the results(1). The relevance of each document is in fact user-specific. For example, searching for "bicycle shops" must show different results to users in Stockholm or Hong Kong. Additionally, queries can have different information needs that can not be inferred by the search text only: the search "Giant" can aim at retrieving information about the bike maker Giant, the western film Giant, or the definition of the English word giant.

Personalization in Information retrieval (PIR) tailors search results based on information on the user querying the system. Google, the world's most popular search engine, introduced personalized search in 2005, even for users not having a Google account. Although the implementation adopted by Google is not publicly known, it is believed it is based at least on user language, localization, and web history. Today many digital service providers, such as Spotify, Amazon, Netflix, and Facebook, integrate personalized search engines and praise their recommendation systems as a key element for their popularity.

Beyond the different kinds of information about users that can be exploited, many techniques exist. For example, it is possible to combine information about multiple users, finding similarities between their behaviors and recommending documents based on it. This approach is typically referred to as collaborative filtering(2).

In this paper, we adopt a graph representation of user past behaviors to tackle PIR. In particular, the links in our graph represent the relevance of a document for a given user and query. It is thus possible to exploit link prediction, a network problem that aims at predicting which new links are most likely to emerge based on the graph topology and node features.

## 2 Related work

For unfamiliar readers, the work by J. Liu et al.(3) is a survey about Personalization in Information Retrieval which discusses the basic concepts of context and search behaviors and how they influence user-specific information needs. Furthermore, the main type of information and approaches used in recommendations are described, giving an exhaustive overview of the problem.

Many user information can be exploited, both explicit, which involves users in giving feedbacks (4; 5), and implicit, such as social bookmarks(6), eye tracking(7), and query history(8; 9; 10). Since our method starts from the latter, it is worth focusing on papers on the subject. M. Speretta et al. (8) create user profiles based on users' search history, classifying the searched text and results into hierarchical concepts. When performing new queries, the similarity between concepts contained in the retrieved documents and in the user profiles is exploited to rerank results. Similarly, N. Matthijs et al.(10) captures user information as a weighted list of terms, using bag of word similarities to reassign relevance weights. On the other hand, automatic query expansion based on contextual information deduced by previous queries is studied by X. Shen et al.(9).

Nowadays, state-of-the-art approaches recur to neural networks. For example, Y. Chen et al. (11) combine a Bayesian framework to estimate user preferences with a neural network that translates user implicit feedback into updates of the probabilistic model itself. In the same vein, Hema Yoganarasimhan (12) proposed a new approach using different machine learning tools to generate user features, ranking the results based on them. This method is scalable and optimizes the features for better accuracy and efficiency in computing.

Our method is based on a graph representation of the search history. Inspired by Google PageRank, a graph-based approach for PIR has been proposed by G. Jeh et al.(13). Another similar approach has been developed by H. Naderi et al.(14), exploiting a query-document graph and collaborative filtering; however, they required explicit feedback which is a major drawback.

After the creation of the graph representation, we exploit link prediction to tackle PIR, as it will be discussed more in detail in the following section. This task consists of predicting which new edges are likely to appear given the current graph topology. While most of the cutting-edge methods in graph prediction use Graph Neural Network representations(15; 16), we will focus on traditional methods. Before being approached as a supervised problem(17), link prediction has already been studied a lot as an unsupervised problem with several heuristics(18; 19; 20).

## 3 Method

For readers unfamiliar with graph theory, there is a glossary in annex A that details all the vocabulary needed to understand the methods used in this project.

We decided to use techniques that would require only the search logs. One can see the search log as a ternary/quaternary relation $\mathcal{L}(q, u, (s), d)$ where $q$ marks a query, $u$ a user, $s$ a user session and $d$ a successful result. In our case, a document is considered a successful match for a given pair query-user $(q, u)$ if the user $u$ clicked (opened) the document after performing the query $q$. This method is intrinsically biased, as a document can only be a match if it has been retrieved by the search engine first. It also assumes that all opened documents are relevant, which is usually not the case. However, this method also has the benefit of not requiring active feedback from the user, allowing for massive data collection on a large-scale application. We will therefore assume that the bias induced by our method is still a good enough approximation of the (unknown) ground truth.

To take advantage of the large but imprecise nature of our data, we want to use not only the information collected about the user but also to connect those to the information we have about "similar" users. Similar here is a quite vague definition that we have to define further.

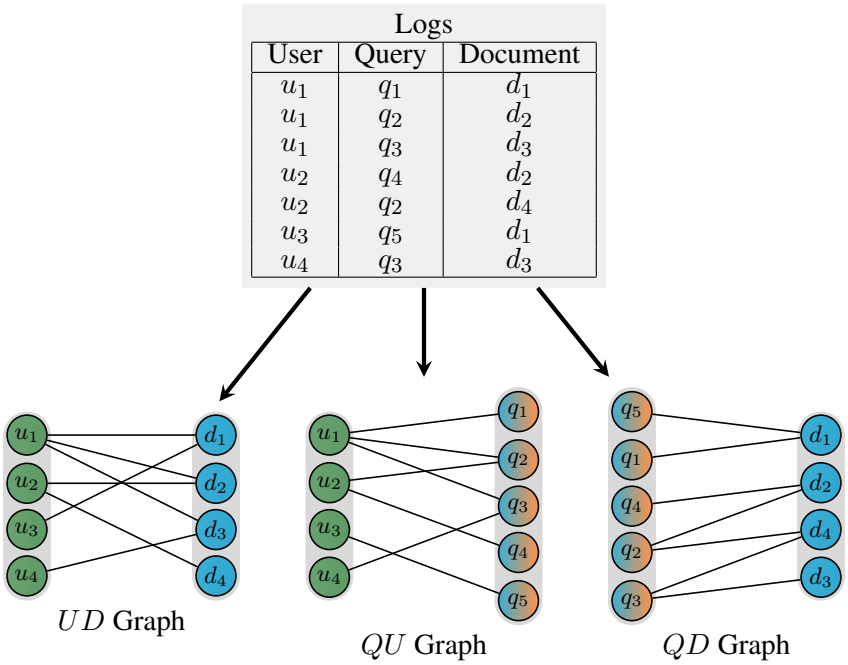


Figure 1: The solution proposed decomposes the search logs into navigable bipartite graphs.

### 3.1 Graphs of relationships

Our approach is first to simplify the exhaustive but hardly exploitable data contained in the logs toward graph representation, which offers a large variety of similarity measures. We identify three (five) easily exploitable bipartite graph representations:

1. $UD$ drawing the relation between the users and the documents, with links between a user $u$ and a document $d$ if $u$ clicked on $d$ in any query.
2. $QU$, linking users to the queries they performed.
3. $QD$, linking queries to their successful result documents.

4. $SD$ and $QS$ are defined the same way as $UD$ and $QU$ with users being replaced by user sessions.

These representations contain less information than the initial logs and can be seen as projections of the logs to somewhat 2-dimensional spaces.

We can note that no personalization can be performed based on $QD$ only, as it does not contain any information about the user. Similarly, $QU$ (and $QS$) while containing information about the user, can not be used to discriminate documents. Because of this, $UD$ constitutes an interesting focus to start with. The difference between $UD$ and $SD$ is about the time window of the search, $UD$ taking into account the whole user history while $SD$ only considers the recent actions (session) and should therefore be more fine-grained at the cost of data quality, a session being often no longer than a single search and not providing any exploitable information.

### 3.2 Graph topology measures

The ranking of the document is made by performing link prediction on some graphs and assigning the resulting score to the documents, before sorting them. The easiest way to assign link prediction score to a couple of nodes in a graph is to compute their similarity based on the graph topology:

- The Common neighbors method $CC(u,v) = |\mathcal{N}(u) \cap \mathcal{N}(v)|$ measure the number of neighbors the two nodes have in common. Due to the bipartite nature of our graphs, using this metric as it is commonly defined would always return 0. Because of that, we had to adapt it, instead of looking at the common neighbors, we count the number of edges between the two neighborhoods.
- The Adamic-Adar method (18) $A(u,v) = \sum_{z \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{\log(|\mathcal{N}(z)|)}$ adapt the common neighbors method by weighting each common neighbors relatively to its degree. The measure has been adapted just like common neighbors to fit the bipartite nature of the graph.
- The rooted page rank method(19) bases itself on Page rank but redirects all the jumps to a root node.
- The prop Flow method(17) is an unsupervised prediction method on networks, it represents the probability that a restricted random walk starting at $v_i$ ends at $v_j$ in $k$ steps or fewer using link weights as transition probabilities. The condition of breaking is that the walk ends upon reaching $v_j$ or upon revisiting any node including $v_i$. *PropFlow* is similar to *Rooted PageRank*, but in fact, it is a more localized measure of propagation, and it's insensitive to noise far from the source node.

Note that any graph can be exploited to alter the document rankings, even not obvious ones like $QU$, since in combination with $UD$ can partially reconstruct $UD$. However, we decided to limit our scope on $UD$ graph, on which a predicted link correspond to the prediction of the querying user selecting the potentially connected document. With this approach, Page Rank and Prop Flow were propagated by the querying user $u$, while the first two measures were computed between $u$ and all the retrieved documents $d$. The values associated to $d$ can be used as weights to re-rank them accordingly.

### 3.3 Machine Learning methods

Based on the work of Lichtenwalter et al.(17) we tried to use all the metrics discussed previously to feed a clustering/regression model. This approach allows us to combine the $UD$ graph information along with the $SD$ but also $QU$, $QS$, and $QD$ one. Unfortunately, although the method could have been powerful, the AOL4PS dataset is not suitable for machine learning methods. A feature in the dataset that would have been welcome is the rate of well-researched documents given by users, which is not the case here.

## 4 Experiments

### 4.1 Dataset

The AOL4PS dataset is made of over one million queries performed by users on the internet. In particular, each query contains information on the user performing it and its corresponding session, the query text, the top ten documents retrieved, and the web page opened by the user. The dataset is split into two disjoint sets: the training set, which constitutes the initial information to build the graph, and the evaluation set, which queries are used for evaluating performances and never inserted into the graph structure. The latter is made of 10000 randomly extracted query logs.

### 4.2 Metrics

In Information Retrieval evaluation, the crucial measure is document relevance, with respect to a specific query and possibly to a user. Unfortunately, in AOL4PS relevance is not directly assessed, but user clicks on documents can be exploited as proxy measures. In particular, we consider the clicked page as the only relevant document given user, query text, and session. This makes the evaluation task anomalous, as many traditional measures are meaningless in such a scenario and some slight modifications are needed.

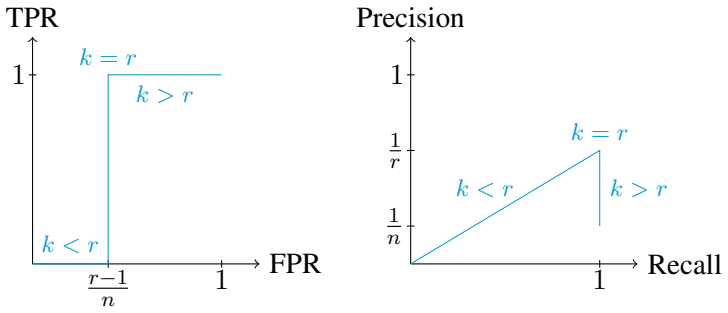


Figure 2: Elementary shapes of the ROC and Precision-Recall curves for a single measure point.

We call $r_{q,u,s,M}$ the rank of the clicked document by user $u$, performing the query $q$ in session $s$, with the ranking determined by system $M$. A user can possibly repeat the same query, making the combination of $u, q, s, M$ not necessarily unique. We drop $q, u, s$ for brevity, as all metrics are defined for a single data point.

Firstly, we define metrics that evaluate a single ranking by itself. These metrics can be computed for different systems and then compared, to measure the difference in performance.

- Reciprocal Rank, commonly used in Information Retrieval, which in our scenario is just the reciprocal rank of the only relevant document.

$$RR(r_M) = \frac{1}{r_M}.$$

- Top $k$ recall, which is binary in our case and corresponds to

$$R@k(r_M) = \begin{cases} 1 & \text{if } \; k \geq r_M \\ 0 & \text{if } \; k < r_M \end{cases}.$$

From these two metrics alone, it is possible to plot the ROC curve and the precision-recall curve. The ROC curve is simply a translated and rescaled plot of $R@k$ regarding $k$, while the precision-recall curve is uniquely identified by $RR$. In Figure 2, $n$ refers to the number of retrieved documents, $r$ is the rank of the relevant document and $k$ is the parameter of $R@k$. Secondly, we define metrics to directly compare two different rankings.

- Relative Discounted Gain, inspired from the Discounted Gain, is a direct measurement to compare the ranks of the relevant document in two systems $M_1$ and $M_2$. We define it as

$$rDG(r_{M_1}, r_{M_2}) = \begin{cases} \frac{r_{M_2} - r_{M_1}}{|r_{M_1} - r_{M_2}|} \frac{log_2(1+|r_{M_1} - r_{M_2}|)}{log_2(1+\min\{r_{M_1}, r_{M_2}\})} & \text{if } \; r_{M_1} \neq r_{M_2} \\ 0 & \text{if } \; r_{M_1} = r_{M_2} \end{cases}.$$

  This metric is symmetric, i.e. $rDG(a, b) = -rDG(b, a)$, and proportional to how well $M_1$ is performing wrt $M_2$. If it is positive, $M_1$ is outperforming $M_2$. The absolute magnitude is proportional to the rank difference, and inversely to the minimum rank. This implies for example that a difference of 2 in ranks will have a smaller impact than a difference of 4 in the ranks. At the same time, if that difference appears at high ranks the improvement is not relevant anymore (e.g. $rDG(2, 4) > rDG(8, 10)$). This last criterion is introduced since users tend not to go down the rankings, making only differences in the highest ones relevant.
- Kendall rank correlation coefficient measures how different two rankings are. It takes into account all the distinct couples of documents in both results, evaluating if they have the same relative order in the two rankings. It is computed as

$$\tau = \frac{n_c - n_d}{n(n-1)/2},$$

| | $R@1$ | $R@3$ | $R@5$ | $R@7$ | $RR$ | $rDG$ | $\tau$ |
|---|---|---|---|---|---|---|---|
| Reference | .1973 | .3282 | .3891 | .9603 | .3684 | - | - |
| Shortest Distance | .0858 | .4677 | .8174 | .9870 | .3504 | 0.1590 | .0125 |
| W. Shortest Distance | .1760 | .6209 | .9089 | .9933 | .4451 | 0.4680 | .0133 |
| Common Neighbors | .8600 | .9745 | .9972 | .9998 | .9173 | 1.6987 | .0888 |
| Adamic Adar | .8608 | .9744 | .9971 | .9998 | .9177 | 1.6995 | .0892 |
| Page Rank | .8480 | .9557 | .9930 | .9997 | .9050 | 1.6614 | .0699 |
| Prop Flow | .8459 | .9565 | .9932 | .9997 | .9042 | 1.6602 | .0715 |

Table 1: Evaluation metrics averaged over 10000 test queries. The documents considered and re-ranked are only the ten appearing in the log dataset. The reference is their corresponding initial ranking.

| | $R@1$ | $R@3$ | $R@5$ | $R@10$ | $RR$ | $rDG$ | $\tau$ |
|---|---|---|---|---|---|---|---|
| Reference | .0620 | .2028 | .2665 | .3543 | .1668 | - | - |
| Shortest Distance | .0021 | .0097 | .0138 | .0260 | .0261 | -1.1638 | .0738 |
| W. Shortest Distance | .0348 | .0517 | .0651 | .1024 | .0732 | -0.7528 | .0764 |
| Common Neighbors | .6550 | .7875 | .8211 | .8559 | .7331 | 3.1509 | .1070 |
| Adamic Adar | .6801 | .7920 | .8225 | .8570 | .7482 | 3.1880 | .1072 |
| Page Rank | .6816 | .7918 | .8182 | .8551 | .7472 | 3.1504 | .1027 |
| Prop Flow | .6484 | .7881 | .8250 | .8563 | .7299 | 3.1122 | .1011 |

Table 2: Evaluation metrics averaged over 4852 test queries in which ElasticSearch retrieved the relevant document. The documents considered and re-ranked are the 250 best matches retrieved by ElasticSearch. The reference is their corresponding initial ranking.

where $n_c$ is the number of document pairs with concordant relative ranking and $n_d$ discordant ones, and $n$ is the overall number of documents. It can be adjusted in case of ties. It is bounded between 1, which corresponds to the same exact order, and -1, representing the reversed one. Two uncorrelated rankings have $\tau = 0$.

### 4.3 Results

We evaluate our re-ranking system in two different settings. First, considering only the 10 documents appearing in the log samples. Here, we use as a reference the ranking of the ten documents in the logs themselves. The second approach is to perform the same test queries on our ElasticSearch system, retrieving 250 documents per search. The results are based solely on a short description, so the initial rankings can be quite noisy. Furthermore, the relevant document is not always retrieved: those cases are discarded, since we aim at evaluating the re-ranking approach rather than the initial retrieval. These documents are then re-ranked according to our graph measures, using as reference the initial ranking provided by ElasticSearch.

In both settings, we evaluate re-ranking based on multiple graph measures. The results are averaged over all queries. $rDG$ and $\tau$ are computed wrt the references of the corresponding setting. The results are reported in Table 1 and Table 2, respectively for the log documents and the ElasticSearch retrieved ones.

First, we notice a generally weak $\tau$ correlation, indicating that the graph measures are significantly altering the original rankings but without reversing them.

Although ElasticSearch was expected to achieve low performances, due to the retrieval based on short descriptions rather than full content, the AOL4PS log reference performs surprisingly badly. However, although its engine was positively biasing data since documents were clicked based on what it was displaying, results must be considered wrt the real case scenario: the system dates previously to 2006, when methods were not as developed as today, and the number of web pages that were ranked was massive. It is possible to see how the number of documents considered alters the performances by comparing our graph measures in the two different tasks when considering 10 documents (Table 1) or 250 (Table 2).

On the other hand, the proposed approach based on graph measures yields outstanding performances. Apart from Shortest Distances, which was unable to provide accurate personalization, Common Neighbors, Adamic Adar, Page Rank and Prop Flow achieved similar performances, showing the relevant document 86% of the time as the first result, and 97% in the top 3, considering the log documents only. Even on the harder task in which 250 documents were assigned a score, the relevant document was the first document displayed in 68% of the queries, and in 82% of them it was in the top 5. However, considering the computational cost, the best two algorithms are probably Common Neighbors and Adamic Adar, which were more than 10 times faster than the other two in our experience. Between them, Adamic Adar shows slightly better performances, and it is probably the algorithm to start from for further developments. However, it is still interesting to note that Prop Flow and Page rank are designed to explore the relation graph further, making them more relevant for situations with fewer connections.

## 5 Conclusions

In this work, we presented a search engine application based entirely on graph theory tools employing different link prediction methods.

Under the testing framework for the application's behavior, the search engine achieved notable results, particularly the Adamic-Adar algorithm 3.2, which demonstrated better performance when considering both speed and relevance as metrics. We also demonstrated that graph distance metrics are completely unsuitable for a PIR application, with drawbacks regarding both time efficiency and performance.

An interesting development to this project would be to change the framework, and eventually, the collected data to allow ML methods to improve the results of the link prediction. Such an improvement would also unlock the ability to exploit the information contained in the other relation graphs presented in this report.

## References


[1] S. T. Dumais, "Personalized search: Potential and pitfalls," in Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16, (New York, NY, USA), p. 689, Association for Computing Machinery, 2016.

[2] S. Shen, B. Hu, W. Chen, and Q. Yang, "Personalized click model through collaborative filtering," in Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM '12, (New York, NY, USA), p. 323–332, Association for Computing Machinery, 2012.

[3] J. Liu, C. Liu, and N. J. Belkin, "Personalization in text information retrieval: A survey," J. Assoc. Inf. Sci. Technol., vol. 71, no. 3, pp. 349–369, 2020.

[4] S. Jayarathna, A. Patra, and F. Shipman, "Mining user interest from search tasks and annotations," in 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013 (Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi, eds.), pp. 1849–1852, ACM, 2013.

[5] O. Vechtomova, E. Lam, and M. Karamuftuoglu, "Interactive search refinement techniques for hard tasks.," pp. 820–827, 01 2003.

[6] C. Biancalana, A. Micarelli, and C. Squarcella, "Nereau: a social approach to query expansion," in Proceedings of the 10th ACM Workshop on Web Information and Data Management, WIDM '08, (New York, NY, USA), p. 95–102, Association for Computing Machinery, 2008.

[7] G. Buscher, L. van Elst, and A. Dengel, "Segment-level display time as implicit feedback: a comparison to eye tracking," in Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '09, (New York, NY, USA), p. 67–74, Association for Computing Machinery, 2009.

[8] M. Speretta and S. Gauch, "Personalized search based on user search histories," vol. 2005, pp. 622– 628, 10 2005.

[9] X. Shen and C. X. Zhai, "Exploiting query history for document ranking in interactive information retrieval," in Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, SIGIR '03, (New York, NY, USA), p. 377–378, Association for Computing Machinery, 2003.

[10] N. Matthijs and F. Radlinski, "Personalizing web search using long term browsing history," in Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011 (I. King, W. Nejdl, and H. Li, eds.), pp. 25–34, ACM, 2011.

[11] Y. Chen, X. Sun, D. Gong, Y. Zhang, J. Choi, and S. Klasky, "Personalized search inspired fast interactive estimation of distribution algorithm and its application," IEEE Transactions on Evolutionary Computation, vol. 21, no. 4, pp. 588–600, 2017.

[12] H. Yoganarasimhan, "Search personalization using machine learning," INFORMS Analytics Collections Vol. 16: Advances in Integrating AI & O.R., 2020.

[13] G. Jeh and J. Widom, "Scaling personalized web search," in Proceedings of the 12th International Conference on World Wide Web, WWW '03, (New York, NY, USA), p. 271–279, Association for Computing Machinery, 2003.

[14] H. Naderi and B. Rumpler, "Percirs: a system to combine personalized and collaborative information retrieval," Journal of Documentation, vol. 66, pp. 532–562, 07 2010.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, (New York, NY, USA), p. 701–710, Association for Computing Machinery, 2014.

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, (New York, NY, USA), p. 855–864, Association for Computing Machinery, 2016.

[17] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla, "New perspectives and methods in link prediction," in Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '10, (New York, NY, USA), p. 243–252, Association for Computing Machinery, 2010.

[18] L. A. Adamic and E. Adar, "Friends and neighbors on the web," Social Networks, vol. 25, no. 3, pp. 211–230, 2003.

[19] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, (New York, NY, USA), p. 635–644, Association for Computing Machinery, 2011.

[20] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," Journal of the American Society for Information Science and Technology, vol. 58, no. 7, pp. 1019–1031, 2007.

## A Graph theory glossary

This section aims at providing the base knowledge in graph theory useful to understand this paper.

### Graph/Network

A **graph** is defined as $G = (N, E)$ where $N$ is a set of nodes and $E \in N \times N$ is a set of edges connecting two nodes. The word **network** can be used interchangeably with graph, but usually refers to a real-life instance of a graph.

### Nodes and edges

A **nodes** (or vertex) can contain any form of data. It is common to assign each node a name or a unique ID. In our case, each node also has a type (user, document, query, session), it is possible to have nodes with different types in a graph.

An **edge** (or link) is defined as $e = (u, v)$ where $u$ and $v$ are two nodes in the graph. It is also possible to add additional data to an edge, such as a **weight**. It is common to store a graph as a matrix $A$ of dimension $|N| \times |N|$ where $A_i, j = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise} \end{cases}$. This matrix is called the transition matrix.

### Bipartite graphs

The graphs used in this project are all **bipartite**, which means that the nodes are divided into two subsets $N = N_1 \sqcup N_2$ and all the edges connect the two subsets.
i.e. $\forall (u, v) \in E, u \in N_1, v \in N_2$.

### Neighborhoods

We defined the **neighborhood** of a node $v$ in a graph $G = (N, E)$ as the set of nodes $u_i \in N$ connected to $v$.
i.e. $\mathcal{N}(v) = \{u \in N | (v, u) \in E\}$

### Paths and walks

The notion of **paths** or **walk** in a graph is pretty intuitive, it designates a succession of edges, where each edge starts at the end of the last edge.

A random walk is a special kind of walk where each edge is picked randomly from the neighbors of the last node reached by the walk.