Eötvös Loránd University
Faculty of Science

Centrality on dynamic graphs

MSc thesis

Béres Ferenc

Applied mathematics MSc.

Supervisor:

Benczúr A. András Ph.D.

Department of Operations research

Budapest, 2015

Acknowledgements

I would like to thank Benczúr András, my supervisor, who made it possible for me to work on this interesting topic. I enjoyed every time of this research.

I would like to express my great appreciation to my colleague Pálovics Róbert for the discussions we had on this thesis. His observations and ideas helped me a lot in writing my first English thesis.

I owe special thank to my colleagues at SZTAKI who let me to use their recommender framework for the link prediction tests.

I also want to thank my family for being understanding and supportive throughout my education.

Finally, I would like to thank my girlfriend for always listening so carefully when I was speaking about the experiments enthusiastically.

Contents

Introduction

There are several node centrality measures that try to determine important vertices of a directed graph. These measures are constrained to use only the structural properties of the network. While these centrality indices are widely investigated, yet there is less focus on their performance in temporal graphs. In this thesis, we intend to give a method for predicting the central nodes of dynamic graphs in advance.

In Chapter 1, we present most of the well known centrality measures, by following the thorough categorization of Paolo Boldi and Sebastiano Vigna [4]. Later, we conduct experiments on a subset of indices, discussed in the first chapter. Particularly, we examine how indegree (Section 1.1.1), outdegree, negative β-measure (Section 1.1.2), PageRank (Section 1.2.4) and SALSA authority and hub scores (Section 1.2.6) can be applied for temporal networks.

Our goal is to solve the centrality prediction problem explained in Section 7.1. The task is to infer the top k central nodes of a dynamic graph for the next time interval, using only graph information from previous intervals. In order to predict central vertices, we solve the link prediction problem introduced by Liben-Nowell and Kleinberg [19]. From the predicted edges we construct a graph for the next time interval, on which centrality measures can be computed. In Section 5.3, we show that the link prediction problem for temporal graphs can be viewed as a recommendation task with implicit ratings. The instances of the training data are the formerly seen links of the directed network. In our model, we solve the link prediction problem with matrix factorization. The optimization method used in the experiments is online stochastic gradient descent (SGD), which is detailed in Section 5.2.

After computing centrality scores of a particular measure, ranked lists of length k can be constructed. These lists contain the top k central node according to each measure. For the evaluation of our model, we used precision (P@k) and normalized discounted cumulative gain (NDCG@k). These are both popular metrics for comparing graded lists [4]. Furthermore in Section 2, we mention other ideas for comparisons proposed by information retrieval.

In this thesis, we examined the Twitter movement datasets described in Chapter 6. The corresponding world-wide events are the *Occupy Wall Street*, *15-October* and *20-N* global protests from 2011 [28–30]. There is an additional dataset from 2012 as well, which is related to the *Yo Soy 132* movement [31]. Twitter is a highly temporal social system with dynamically evolving communities. Several different dynamic graphs can be constructed from the user interactions of this social network. These graph types are explained in Section 3.2. Myers and Leskovec [21] showed that the Twitter network is highly dynamic with about 9% of all connections changing in a month. Thus, in order to infer central nodes, the factors driving the dynamics of this social network must be considered. In Chapter 4, we present some recent results about information diffusion on dynamic graphs [1, 9, 21, 24].

We evaluated our model on the mention graphs (Section 3.2.3) that were extracted from our movement datasets. Here, a link $x \to y$ corresponds to the event when user x mentioned user y in a Twitter message. These graphs are indeed temporal as the mention time is available for all edges. The experiments showed that these networks are higly dynamic as well. In any time interval, the fraction of the new incoming users is high. Therefore, new edges dominate the links of each time frame. However, we also realized that new edges with previously present endnodes can be the key to effective centrality predictions. These are called homophily edges (Section 7.3.1).

For the proper evaluation of our model we introduced a baseline model in Section 7.2. The proposed baseline gives the top k central nodes of the previous time interval graph as a prediction for the next. After solving the link prediction problem for the given mention graphs, we computed the formerly mentioned centrality measures on the inferred networks. Finally in Section 7.4, we compare our results with the baseline model. We find that for some measures our model performs better than the baseline. Thus, it can be useful for predicting key users in a global movement.

Chapter 1

Centrality measures

A graph is an ordered pair $G = (V, E)$. Set V contains the vertices or nodes of G, and E consists of two-element subsets of V. The elements of E are the edges or links of G. A graph is directed, when the end-points of its edges are not interchangeable. These graphs are denoted by $D = (V, A)$, where A is the set of directed edges or arcs. Every directed edge $x \to y$ has a source x, and a target y. Graphs are widely used because they can represent various real world networks like electrical, social and transportation networks. When using graphs for modelling these networks, many natural processes can be generalized into a graph theoretic problem.

For example, from a social network one can define a graph, whose nodes represent the people, and the edges every relations between them. In this social graph, the most crucial information we would like to extract is its relevant nodes. This task is far from obvious, as the number of measures that we can use for such purposes is quite abundant. In [4] there is a thoroughly detailed enumeration based on the defining property of these centrality measures. In this section, we will follow the same categorization and notations.

1.1 Geometric measures

In case of geometric measures the defining property is distance. Let $d(x, y)$ denote the distance between x and y vertices of the directed graph. These measures mostly depend only on the number of nodes existing at given distances.

1.1.1 Indegree

The indegree of node x is the number of its incoming arcs and it is denoted by $d^-(x)$. It is one of the most elementary and oldest measures, as it represents majority voting in case of directed edges. For example, $x \to y$ directed edge could mean, that node x voted for node y. One of its advantages is that it can be efficiently computed with one linear scan. On the contrary it is easily manipulated, for example with spam hyperlinks for a website. As a result it may not be the best choice for centrality measure but surely it can be used as a good baseline.

1.1.2 Negative β-measure

Equivalently to indegree only the 1-distance in-neighbourhood of node x is relevant. Let $d^+(v)$ denote the outdegree of vertex v. Negative β-measure is defined by

$$\sum_{y \to x} \frac{1}{d^+(y)}.$$

and it can be considered as *"Markovian indegree"*. This measure is inspired by the same l_1 normalization that is used for Seeley's index, PageRank, and SALSA described in Sections 1.2.2, 1.2.4, and 1.2.6.

1.1.3 Closeness

The closeness of node x is defined by

$$\frac{1}{\sum_y d(y,x)},$$

where $d(y,x)$ denotes the distance of x from y in the directed network. While indegree and negative β-measure were relying on the local graph structure, closeness is defined by the global graph structure. Thus, it is more costly to compute.

It is important to remark that the graph must be strongly connected. Without this condition the result will be a null score for all x node, that cannot coreach the whole graph. Nevertheless there have been many propositions on how to mend this troublesome quality of closeness. But the most straightforward idea is to exclude infinite distances

$$\frac{1}{\sum_{d(y,x)<\infty} d(y,x)}.$$

1.1.4 Lin's index

One of the ideas that tried to repair the definition of closeness for graphs with infinite distances was Nan Lin's. Lin's index defines the score of node x with a nonempty coreachable set as

$$\frac{|\{y|d(y,x)<\infty\}|^2}{\sum_{d(y,x)<\infty} d(y,x)}.$$

Nodes with an empty coreachable set have centrality 1 by definition.

This change in the definition means that closeness is not the inverse of a sum of distances, but rather the inverse of the average distance. One of the results of this modification is that closeness is normalized across the graph.

1.1.5 Harmonic centrality

Paolo Boldi and Sebastiano Vigna in [4] gave another solution on how to eliminate the problem of non-finite distances between nodes. The main idea is to use harmonic mean instead of arithmetic averaging. The reason why harmonic mean is involved is that it conveniently deal with ∞ distances, as $\frac{1}{\infty} = 0$.

The precise definition for the harmonic centrality of node x is

$$\sum_{x \neq y} \frac{1}{d(y,x)} = \sum_{d(y,x)<\infty, x \neq y} \frac{1}{d(y,x)}, \tag{1.1}$$

which is the reciprocal of the denormalized harmonic mean of distances. In [4] the authors found that harmonic centrality is strongly correlated to closeness in simple networks. Moreover, this definition also accounts for nodes y that cannot reach x. Thus, this measure can also be used in cases when the given graph is not strongly connected.

1.2 Spectral measures

In case of spectral measures the defining property is the left dominant eigenvector of some matrix derived from the graph. Naturally, the left dominant eigenvector is computed from the adjacency matrix A. Although, in many cases some kind of normalization is applied to A before computing the left dominant eigenvector. Depending on the chosen normalization we can get numerous spectral measures.

1.2.1 The left dominant eigenvector

Regarding spectral measures the first measure to remark is obviously the left dominant eigenvector of the adjacency matrix. The dominant eigenvector is the fixed point of an iterated process. Initially, every node have the same score. In each iteration, these values are replaced with the summed scores of in-neighbours for all vertices. Before the next iteration, the score vector is normalized. The process is repeated until convergence. However for non-strongly connected graphs, the dominant eigenvalue of the strongly connected components determines, whether the dominant eigenvector might or might not be nonzero on non-terminal components [2].

1.2.2 Seeley's index

Formerly in Section 1.2.1, it was emphasized that the dominant eigenvector was the fixed point of the iterated process in which every node was giving its score to its out-neighbours for summation in each iteration. At the beginning of the process all nodes had the same initial score. Seeley proposed one major modification regarding this process. He suggested that a node should equally divide its score among its successors rather that passing its entire reputation to its out-neighbours.

From a linear-algebra point of view, this corresponds to normalizing each row of the adjacency matrix using the l_1 norm. The matrix derived from the l_1-normalization process is stochastic, so the score to which the iterated computation converges can be interpreted as the stationary state of a Markov chain.

1.2.3 Katz's index

Katz defined his index through summation of all paths coming into a node x. In order to get a finite score he introduced an *attenuation factor* β with which a weight could be calculated for the paths. Katz's index can be expressed as

$$k = \mathbf{1} \cdot \sum_{i=0}^{\infty} \beta^i A^i \tag{1.2}$$

due to the interplay between the powers of the adjacency matrix and the number of paths connecting two nodes. To get a final value from the above summation, β must be smaller than $\frac{1}{\lambda}$, where λ is the dominant eigenvalue of A.

Katz also recognised that the index can be expressed using linear algebra operations

$$k = \mathbf{1} \cdot (1 - \beta A)^{-1}, \tag{1.3}$$

where $\mathbf{1}$ is the vector with uniformly 1 coordinates. Furthermore, due to Brauer's theorem on the displacement of eigenvalues, Katz's index is the left dominant eigenvector of a perturbed matrix

$$\beta\lambda \cdot A + (1 - \beta\lambda) \cdot e^T \cdot \mathbf{1},$$

where e is a right dominant eigenvector of A such that $\mathbf{1}e^T = \lambda$. Hubell [11] recommended a generalization for Katz's index in which some *preference vector* v is used instead of $\mathbf{1}$. For (1.2) this generalization means, that paths can be weighted individually depending on their starting node. The normalized limit of Katz's index is he dominant eigenvector when $\beta \to \frac{1}{\lambda}$. Nevertheless, the limit depends on v, if the dominant eigenvector is not unique [26].

1.2.4 PageRank

Recently, PageRank is one of the most frequently discussed and cited spectral measure in use, mainly because of its alleged use in Google's ranking algorithm. PageRank [5] is defined by the unique vector p which satisfies equation

$$p = \alpha \cdot p\bar{A} + (1 - \alpha)v, \tag{1.4}$$

where $\bar{A}$ is derived from the adjacency matrix A with the same l_1-normalization, that was used in the formulation of Seeley's index and the negative β-measure. PageRank has two additional parameter. A *damping factor* $\alpha \in [0, 1)$, and a *preference vector* v. The only constraint for v is that it must be a distribution.

However, it is important to note that p is not necessarily a probability distribution if A has null rows. There has been several propositions on how to make $\bar{A}$ stochastic. A common solution is to replace every null row with the preference vector v. Another popular idea is to add loop arcs to all nodes with zero outdegree (dangling nodes).

Equation 1.4 is solvable even without any patching, as after reorganizing the formula we get

$$p = (1 - \alpha)v(1 - \alpha\bar{A})^{-1}. \tag{1.5}$$

Moreover, another equation can be formulated for PageRank

$$p = (1 - \alpha)v \sum_{i=0}^{\infty} \alpha^i \bar{A}^i, \tag{1.6}$$

which shows that Katz's index and PageRank differ only by a constant factor and by the l_1 normalization applied to the adjacency matrix. This is similar to the difference between the dominant eigenvector and Seeley's index. If A has no null rows, or $\bar{A}$ has been patched to be stochastic, PageRank can be equivalently defined as the stationary distribution of the Markov chain whose transition matrix is

$$\alpha\bar{A} + (1 - \alpha)\mathbf{1}^T v.$$

1.2.5 HITS

HITS algorithm was introduced by Kleinberg in a web-based context [15]. The main difference from the former measures is that this algorithm examines two different qualities, "authoritativeness" and "hubbiness", for each node at once. The important nodes will be those where these two qualities mutually reinforce each other. Thus, a page is a good authority if many good hubs link to it. Likewise, a page is a good hub if it links to many good authorities.

This recursive definition suggests an iterative process that computes the a_i authority and h_i "hubbiness" vectors for each iteration i at the same time. Initially let $a_0 = \mathbf{1}$, and then applying the following update rule

$$h_{i+1} = a_i \cdot A^T, \tag{1.7}$$

$$a_{i+1} = h_{i+1} \cdot A. \tag{1.8}$$

Although, the limit of the process can be concluded easier if one uses the following formulas for the recursions

$$a_{i+1} = a_i \cdot A^T A, \tag{1.9}$$

$$h_{i+1} = h_i \cdot AA^T. \tag{1.10}$$

This method converges to a^*, the left dominant eigenvector of the matrix $A^T A$, which contains the final authority scores. Shifting the process with one step it follows, that h^*, the left dominant eigenvector of AA^T, will contain the final hubbiness scores.

Let σ_i denote the i^{th} singular value of A. Suppose, that l is the index of the last non-zero singular value ($\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_l > 0 = \sigma_{l+1}$). A theorem from linear algebra states, that the best at most k-rank approximation of A is

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T (k \leq l),$$

where v_i is the i^{th} dominant eigenvector of $A^T A$, and u_i is the i^{th} dominant eigenvector of AA^T. Thus, the limit authority and hub vectors yield the best rank one approximation of the adjacency matrix

$$A_1 = \sigma_1 h^* a^{*^T}.$$

1.2.6 SALSA

Formerly, in 1.1.2 and 1.2.2 it was shown that with l_1-normalization new measures can be discovered from already existing ones.

SALSA measure was introduced by Lempel and Moran [17] and similarly to HITS it also computes authority and hubbiness scores for each node, but there is an additional l_1-normalization on A and A^T matrices. The process starts with $a_0 = \mathbf{1}$ and proceed with

$$h_{i+1} = a_i \cdot \bar{A}^T, \tag{1.11}$$

$$a_{i+1} = h_{i+1} \cdot \bar{A} \tag{1.12}$$

However, SALSA does not necessarily need such an iterative process to be computed. An alternative way is to compute the connected components of the symmetric graph induced by the matrix $A^T A$. In this graph, x and y are adjacent if x and y have some common predecessor in the original graph. In [17] it was shown that the SALSA score of a node is the ratio between its indegree and the sum of indegrees in the same component, multiplied by the ratio between the component size and the number of vertices.

1.3 Path-based measures

In addition to distance, all shortest paths coming into a node are taken into account in case of path-based measures. Some of the aforementioned centrality measures, like indegree and Katz's index, could also belong to this category. Thus, only betweenness will be detailed in this section, which relies on an entirely different aspect.

1.3.1 Betweenness

Let σ_{yz} denote the number of shortest paths going from y to z. A subset of these paths also passes through node x, and suppose their number is $\sigma_{yz}(x)$. The betweenness measure of node x is defined by

$$\sum_{y,z \neq x, \sigma_{yz} \neq 0} \frac{\sigma_{yz}(x)}{\sigma_{yz}}$$

The definition tries to capture the intuition that if a significantly large fraction of shortest paths passes through x, then x is an important junction point of the graph. Moreover, Boldi et al. in [3] showed that removing nodes with high betweenness score results in an instant network disruption.

Chapter 2

Comparing centrality measures

2.1 Metrics for comparing rankings

In Chapter 1 it was shown that several centrality measures could be used to extract the most important nodes from a graph. But, which should be applied for temporal network analysis remains a question. After computing centrality scores of a particular measure, ranked lists can be constructed. In this section, we present several metrics that are used for evaluating models by comparing ranked lists. Naturally, common metrics such as precision and recall can be used for this task. Nevertheless, information retrieval (IR) has developed ranking metrics, that capture the user behaviour more effectively.

In pattern recognition and information retrieval with binary classification, precision and recall is a common method to evaluate the performance of a model. A binary classificator assigns positive or negative labels to each data point of the test dataset. For example, a data item is true positive, if both of its original and the classifier-assigned label are positive (Figure 2.1). The model performs well, when the assigned values correlate with the labels in the original dataset.

To access the goodness of information filtering methods sometimes it is enough to examine only a subset of the result. In these models every document, or item has a relevance judgement. Our goal is to find a model that can extract the most relevant items from the data. Especially for recommender systems and search engines, the list of the first

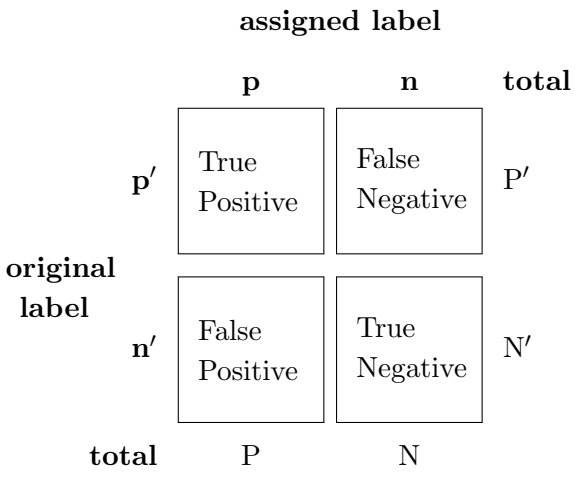


Figure 2.1: Item sets of a binary classificator

k highly rated elements, returned by the model, is compared to the graded list of items with the biggest relevance judgements. Precision and recall do not consider the relevance of the items. Thus to exploit the performance of models through graded list, other metrics should be used.

In general for metrics, that can evaluate graded lists, there are two main types of user models, position models and cascade models. In position models, such as discounted cumulative gain in Section 2.1.2 and rank biased precision in Section 2.1.4, independence is assumed between documents in distinct positions. It means that the probability of examination for a given position does not depend on whether the user was content with the documents on the former positions. In the cascade model it is presumed that there is some dependency among the documents on the result list. This model assumes that users view the elements of the graded lists from top to bottom. Moreover, the user has a certain probability of being satisfied at each position. In other words, a session ends when the user finds the document that he was looking for. Expected reciprocal rank, detailed in Section 2.1.5, is an example for cascade model.

2.1.1 Precision and recall

Precision tries to capture the amount of relevant items by the ratio of the true positive and positively rated data points.

$$Precision = \frac{|\{\text{true positive items}\} \cap \{\text{positively labelled items}\}|}{|\{\text{positively labelled items}\}|} \tag{2.1}$$

In many real world application the size of the dataset in enormous. Therefore, we are only interested in the model performance evaluated on a subset of the data. Indeed, there is a generalization of precision on the first k most important data points, which if often denoted by $P@k$.

$$\text{P@}k = \frac{|\{\text{first } k \text{ true positive items}\} \cap \{\text{positively labelled items}\}|}{k} \tag{2.2}$$

Although we prefer classifiers with good precision, sometimes this metric can be misleading. Suppose that the dataset is dominated by negative items. In this case, we cannot expect a good precision. Hence, models that can identify most of the true positive items are considered better. Recall is defined by

$$Recall = \frac{|\{\text{true positive items}\} \cap \{\text{positively labelled items}\}|}{|\{\text{true positive items}\}|}. \tag{2.3}$$

The main disadvantage of precision and recall is that these metrics does not support graded relevance. Thus, in a model with distinct item relevancies, other metrics should be considered.

2.1.2 Discounted cumulative gain

Examination of ranked lists is a possibility to evaluate the performance of web search engines. The proposed methods and metrics are usually tested on the same fixed set of documents and queries. The relevance judgements of the documents related to a given query are collected by asking human editors. The judgement for document i is denoted by rel_i. There are two main assumptions regarding the relevance scores [13].

1. Highly relevant documents are more valuable than marginally relevant documents,

2. and the greater the ranked position of a relevant document, the less valuable it is for the user. In other words it is less likely that the user will ever examine the document.

In particular, metrics for graded relevance generally evaluated on the first k items returned by an information filtering method. First, there is cumulative (CG) gain that still does not integrate the position of an item, but it takes into account the relevance of a document. Cumulative gain based on the first k item is defined by

$$\mathrm{CG@}k = \sum_{i=1}^{k} rel_i. \tag{2.4}$$

Järvelin and Kekäläine [13] proposed a modification of cumulative gain, which considers the position of the documents too. In discounted cumulative gain (DCG) the graded relevance values are reduced logarithmically proportional to their position in the result.

$$\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)}. \tag{2.5}$$

With this logarithmic reduction a relevant item is indeed penalized if it gets ranked lower. Moreover, if we want to focus on the top of the ranked list, there is an alternative formula for discounted cumulative gain. Burges et al. [6] defined this metric as

$$\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}. \tag{2.6}$$

With this formulation there is an extra emphasis on retrieving relevant documents.

2.1.3 Normalized discounted cumulative gain

Discounted cumulative gain is a proper metric to evaluate graded lists. However, the length of these lists can vary for different queries. A normalization must be introduced in order to compare search methods over multiple queries consistently. Assume that the length of a particular query's result is k. Then the ideal discounted cumulative gain (IDCG) for k is computed to normalize the DCG@k of the given method. Let L be the ranked list containing the k document with the biggest relevance judgements. Ideal discounted cumulative gain for k is the DCG@k of list L

$$\mathrm{IDCG@}k = \mathrm{DCG@}k(L). \tag{2.7}$$

The normalized discounted cumulative gain is defined by

$$\mathrm{NDCG@}k = \frac{\mathrm{DCG@}k}{\mathrm{IDCG@}k}. \tag{2.8}$$

From the definition it is obvious that NDCG takes values from interval $[0, 1]$. Thus, it is well comparable across multiple queries. Indeed, a perfect ranking algorithm finds the ideal graded list L. Thus the NDCG for this algorithm is 1.

2.1.4 Rank biased precision

This metric was proposed by Moffat et al. in [20]. It is also based on the assumptions stated in Section 2.1.2. Thus, relevant documents with higher positions in the graded list are penalized. In contrast to discounted cumulative gain it is done in a more realistic way.

Rank Biased Precision (RBP) tries to capture user behaviour by introducing a persistence factor p. This is the probability that the user will go on looking at the next element in the list. Thus, in RBP the relevance of a document at a given position is reduced by the probability of the user discovering it.

$$\text{RBP@}k = (1-p) \cdot \sum_{i=1}^{k} rel_i \cdot p^{i-1} \tag{2.9}$$

An ideal rank biased precision (IRBM) can be defined similarly as the ideal discounted cumulative gain. By introducing normalized RBP this measure is also cross-queryable.

$$\text{nRBM@}k = \frac{\text{RBM@}k}{\text{IRBM@}k} \tag{2.10}$$

Furthermore, with persistence factor p this measure can be personalized for each user. Although in case of search engines it is not intended.

2.1.5 Expected reciprocal rank

Expected reciprocal rank (ERR) proposed by Chapelle et. al [7] is a cascade model. In this subsection, the same notations will be used as in [7].

Let R_i denote the probability for a given user, that he is satisfied with the document returned by the filtering method at position i. These values are estimated from the click logs by maximum likelihood. Especially, in [7] it is assumed that a user finds a document relevant as an $\mathcal{R}$ function of the relevance judgement.

$$R_i = \mathcal{R}(rel_i) \tag{2.11}$$

In this model, $\mathcal{R}$ can be chosen arbitrarily. The authors of [7] also gave a generalized definition for cascade model using Equation (2.11) and a utility function φ. With a fixed φ they defined expected reciprocal rank by

$$\text{ERR@}k = \sum_{i=1}^{k} \frac{1}{i} \cdot Pr(\text{user stops at position } i) = \sum_{i=1}^{k} \frac{1}{i} \cdot \prod_{j=1}^{i-1} (1-R_j)R_i. \tag{2.12}$$

As for a given set of R_i, the likelihood of a session when the user stops at position r is

$$\prod_{j=1}^{r-1} (1-R_j)R_r.$$

2.1.6 Kendall Tau correlation coefficient

Let L_1 and L_2 be two ranked lists with the same length k. The (x, y) document pair is concordant if their order is the same in both list. If X denotes the number of concordant and Y the number of unconcordant pairs, then the Kendall tau correlation coefficient is defined by

$$\tau = \frac{X-Y}{X+Y}. \tag{2.13}$$

From the definition it follows that if L_1 and L_2 are perfectly the same then $\tau = 1$. Contrarily, if they perfectly disagree then $\tau = -1$.

2.2 Using centrality measures in information retrieval

In information retrieval mostly text based algorithms are used (e.g. BM25). But Paolo Boldi and Sebastiano Vigna [4] researched how centrality measures can perform in this problem. They conducted experiments on the classical TREC GOV2 collection [10] (about 25 million web documents) and the 149 associated queries. It was showed that harmonic centrality achieved the best overall scores among centrality measures. It was surprising as geometric measures have not been used in IR so far.

In [4] the authors have solved the corresponding Boolean conjunction of terms for each query, that resulted in a subset of matching web pages. From each subset an induced graph was defined. Then the nodes were ranked separately according to the centrality measures detailed in Chapter 1. Finally the results were compared for every measure to the state-of-the-art BM25 function.

In the performance evaluation of these measures they used P@10 and NDCG@10 metrics. The results were reported twice. With and without *nepotistic links* (inter-host links). With all induced links harmonic centrality, indegree and HITS performed the best. Although the NDCG@10 = 0.1438 of harmonic centrality is still small compared to NDCG@10 = 0.5842 of BM25. Without nepotistic links β-measure, SALSA and PageRank achieved the best precision and normalized cumulative gain scores, which were less than in the former case.

However, the ranking based on centrality measures could not outperform text-based state-of-the-art methods, it has still given an idea about which measures should we use for temporal network analysis.

Chapter 3

Centrality on Twitter

Over the last years social network analysis came into the spotlight as there is a ever-growing demand from commercial enterprises to extract some ground truths about how information diffuses on these networks. With these information, the efficiency of recommender systems could be improved significantly which would lead to the general satisfaction of the users. Although there are numerous popular social networks in use (e.g. Facebook, LinkedIn, MySpace), recently the most researched one is Twitter. In this chapter, we will present the basic building blocks of Twitter along with its most favourable properties.

3.1 Twitter microblogging framework

Lately, Twitter has become a popular microblogging tool. In this case, microblogging means that registered users are enabled to post only 140-character messages. The messages posted by a user are called tweets and the user who posted them is the tweeter. By default all tweets a user posts is accessible to the public. Therefore, a general user must direct his attention towards specific users. Indeed, in Twitter every user can follow other users, and the tweets of users followed by him will be present on his personal timeline. However, it is important to note that beside this dense follower network there is a sparse and hidden underlying "friendship" network which drives the usage of Twitter [12].

Retweeting is an intensely analyzed feature of Twitter. When a user posts a tweet, that is seen by his followers immediately, they can decide whether the tweet is worth spreading within the network. The followers who decided to re-share the content of the tweet are called retweeters. Usually, the retweeted message remains unchanged, but the retweeter can append arbitrary message to it. One constraint is that the retweeted message always contains *"RT: @originaluser"* followed by the original message. By this feature, information can spread through all over the world within minutes, as the immediate followers of the retweeter will also be aware of the original tweet, thus they can also retweet.

Besides retweeting, there is another way how information can propagate on Twitter. Users are given the opportunity to mention each other. A mention can be any Twitter update that contains *"@username"* anywhere in the body of the tweet. In this case, the users appointed in the mention will be notified about this tweet, independently from the fact whether they are followers of the tweeter or not. Another popular use case for mentions is when they mark some reference, for example the news media or the blogger the information originated from.

The basic motivation of users to join Twitter is to keep track of friends and keep friends updated. Moreover, it is also perfect as a mass media source. Besides normal users, several news agencies are presented in Twitter, such as CNN Breaking News (@cnnbrk), BBC News (World) (@BBCWorld) and Reuters Top News (@Reuters). Twitter is very relevant as a social network as well as a news media.

The number of Twitter users, as well as the tweet and retweet traffic is increasing in a terrific pace. This huge amount of message must be categorized in order to keep it accessible. Therefore, the microblogging community started to use so-called hashtags with which the tweets are categorized manually. Hashtags are indeed practical as they can be used for either searching for certain topics or to be able to follow certain conversations about a certain topic on Twitter. The only constraint for hashtags is to start with a hash symbol #. On the other hand, there is no other regulation about hashtag usage, which leads to some incoherence. In Twitter the main problem of hashtags is that for a given topic many different hashtags can exist. The name or the abbreviation of an event often differ geographically. For example, for Tour de France the *#tdf*, *#tourdefrance*, *#cycling* or *#procycling* occurred as well [32]. The lack of structure and uniformity in hashtags can lead to information leakage as the users cannot know about all existing hashtags of an ongoing event. There have been some attempt to recommend commonly used hashtags for users, before they post their tweet, in order to avoid messages without hashtags and discrepancy [32].

As we have shown, Twitter has many useful features. The tweet length limit encourages users to create posts frequently as long status reports cannot fit into a message. Moreover, they should use mentions and hashtags in order to create understandable and informal tweets. Furthermore, the tweets and retweets of users are accessible through JSON logs, which is a very compact and convenient file format to work with. Nevertheless, Twitter is popular among researchers due to the tremendous amount of up-to-date information.

3.2 Twitter-induced temporal graphs

Twitter is an ever-changing network, thus every graph mentioned in this section mutates over time. In structure these graphs can significantly differ as separate features of Twitter determine the edges and their direction. In every subsection we will give a small example of the actual Twitter-induced graph. These examples are based on the same user activity example displayed in Figure 3.1. The data is in JSON format is indeed the file format of Twitter logs. Retweet messages include *"RT @user"* in their text attribute, while there is no such prefix for tweet messages. The marked user in this prefix is the original tweeter of the related retweet cascade. In this example the messages are sorted according to their time of creation.

3.2.1 User-follower graph

In the follower graph a directed edge from node x to node y represents that x follows y. This is a very viral graph. On the one hand new vertices are added to the graph when new users join to Twitter. On the other hand, edges can be added and deleted according to whether the given user decided to follow or unfollow an other user.

Here, we will present the user-follower graph based on Figure 3.1. Initially, $user\_A$ and

```
{
    "created_at": "2015-03-20 10:59",
    "text": "I hate London. It's always rainy.",
    "place": "London",
    "user": {
            "id": 01,
            "name": "user_A"
    }
}
{
    "created_at": "2015-03-20 11:09",
    "text": "The wheather is nice today",
    "place": "Budapest",
    "user": {
            "id": 02,
            "name": "user_B"
    }
}
{
    "created_at": "2015-03-20 11:15",
    "text": "RT @user_B: @user_A says in London it's rainy",
    "user": {
            "id": 03,
            "name": "user_C"
    }
}
{
    "created_at": "2015-03-20 11:16",
    "text": "RT @user_B: @user_A is always complaning. Rather check out @bbcweather.",
    "user": {
            "id": 04,
            "name": "user_D"
    }
}
{
    "created_at": "2015-03-20 11:18",
    "text": "RT @user_A: Me too",
    "user": {
            "id": 05,
            "name": "user_E"
    }
}
{
    "created_at": "2015-03-20 11:20",
    "text": "RT @user_B: Yes, indeed",
    "place": "Budapest",
    "user": {
            "id": 06,
            "name": "user_F"
    }
}
{
    "created_at": "2015-03-20 11:22",
    "text": "RT @user_B: Yeah, @bbcweather is usually reliable.",
    "user": {
            "id": 07,
            "name": "user_G"
    }
}
```


Figure 3.1: User activity example for Twitter users. The data is in JSON format. Retweet messages include *"RT @user"* in their text attribute. The marked user in this prefix is the original tweeter of the related retweet cascade.

user_B tweeted 2 different messages. Thus, *user_C* follows *user_B* and *user_E* follows *user_A* in order to be able to retweet the former tweets. Similarly, if we suppose that there was no intermediary user between *user_B* and *user_F*, than *user_F* follows *user_B*. The same can be said about *user_D* and *user_C*, or *user_G* and *user_D*. Furthermore, *user_C* and *user_D* follows user *user_A*, because they are both aware of *user_A*'s tweet and his tweeting habits. Finally, *user_G* follows @*bbcwheather* as it seems he has some experience about it. Although *user_D*'s retweet include @*bbcwheather*, maybe he does not follow that medium.

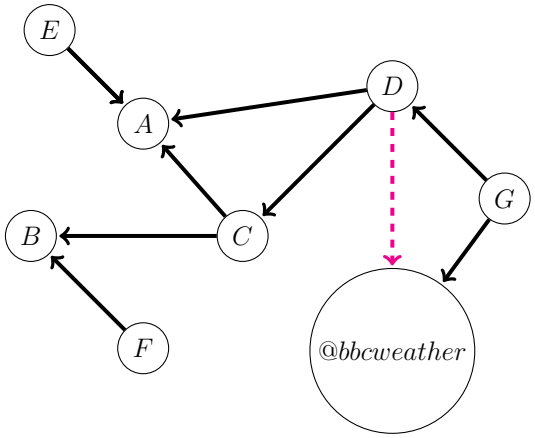


Figure 3.2: User-follower graph example.

This is one of the most extensively studied Twitter-induced graph. Through empirical experiments Myers and Leskovec [21] gave a model for follower count burst prediction, which I will detail in Section 4.1.2. However, Huberman et al. [12] expressed that a link between any two people does not necessarily imply an interaction between them. The reason for this is a general social network phenomenon. Most of the users are intent on having large follower or friend count, but only with a few of them do they really keep in touch on a daily manner.

3.2.2 Retweet graph

In Twitter a tweet can diffuse through the network by persistent retweeting. As a specific tweet is retweeted from user to user, large cascades can form. These cascades contain edges that represent information spread between the corresponding nodes of retweeters. There are two different retweet graphs. In both cases the source of the directed edge is the retweeter, but the target of the edges do differ.

Non-rooted retweet graph

From the two version of retweet graph, non-rooted retweet graphs represent better the cascade of the original tweet spreading through the network. Suppose the original tweeter was z and y was the first through whom user x saw z's tweet. In this case, there is a link from x to y in the retweet graph.

Now, we will present the non-rooted retweet graph based on Figure 3.1. The diffusion of 2 cascades will be examined, as originally there was 2 tweets. The edges corresponding two separate cascades have different colors.

1. The cascade originating from *user_A*'s tweet is very simple. It only consists of one edge, which represents that *user_E* retweeted *user_A*'s tweet.

2. The second cluster contains five nodes. The original tweeter *user_B*. Then from the timestamps it can be concluded that *user_D* retweeted *user_C*. Instantly, after he saw that *user_C* retweeted *user_B*'s tweet. Finally, *user_G* retweeted after he saw *user_D* writing about *@bbcweather*.

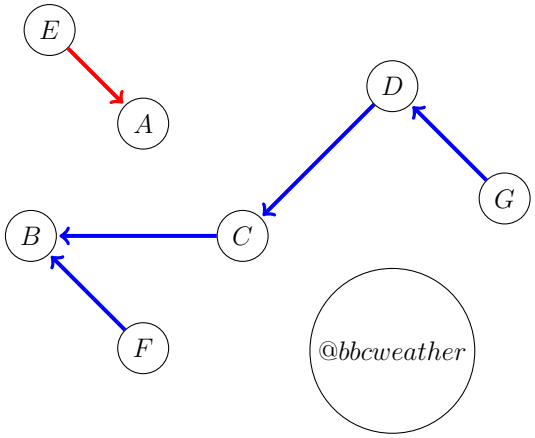


Figure 3.3: Non-rooted retweet graph example.

The growth of similar cascades is a well researched topic in social network related context. Cheng et al. found a robust model for relative cascade growth prediction. They showed that a cascade becomes more predictable over time as more of its reshares can be observed [8].

Nevertheless, it is important to note that non-rooted retweet graphs cannot be constructed directly. In Twitter logs only the original tweeter is listed, but not the user who the retweeter first saw in the actual cascade. However, there are some results on how to infer the non-rooted retweet graph from cascade time-series on the follower graph [24]. The proposed method will be detailed in Section 4.3

Rooted retweet graph

Rooted retweet graphs represent retweet cascades as we can directly extract it from the data. Suppose the original tweeter was z and y was the first through whom user x saw z's tweet. In this case, the link's target will be z instead of y. Information diffusion cannot be represented in this retweet graph.

The rooted retweet graph of Figure 3.1 also contains 2 cascades. Each cascade is a star whose center node is the original tweeter, with all retweeters linking to it.

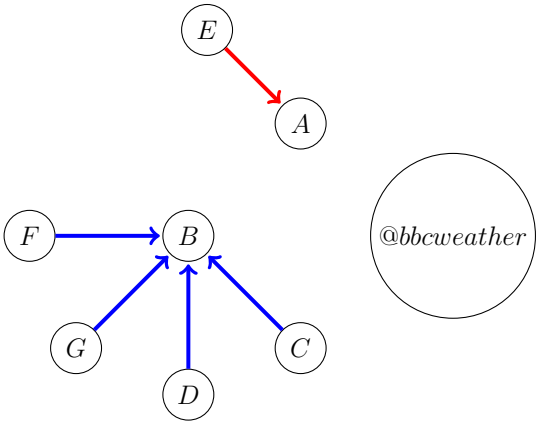


Figure 3.4: Rooted retweet graph example.

3.2.3 Mention graph

In the former sections it was showed how links can be defined by follower relationships and retweeting. The main disadvantage of the former models is that typically the information needed to construct the graphs is not directly accessible. However, for mention graphs all edges can be constructed conveniently, as *@-mentions* are easily extractable from the text. Suppose that x mentioned y, then there is a directed edge from x to y in this Twitter-induced graph. There is no difference between mentions occurring in tweets or retweets.

Now, the mention graph based on Figure 3.1 is sparser than the user-follower graph and retweet graphs induced by the same example. It consists of 4 edges as only *user_A* and @*bbcwheather* were mentioned.

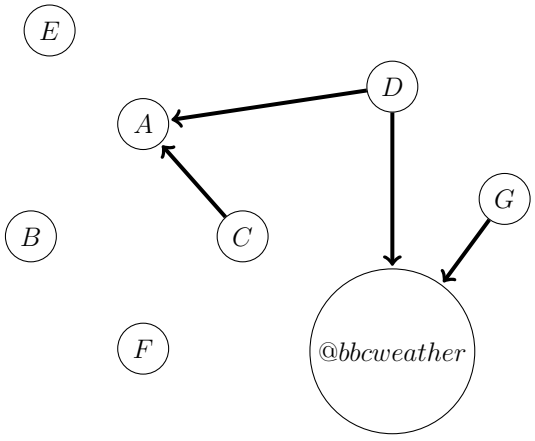


Figure 3.5: Mention graph example.

Particularly, in our experiments we focus on mention graphs induced by the datasets described in Chapter 6.

3.2.4 Event induced subgraphs

The amount of daily messages occurring in Twitter is quite immense. Most of the tweets and retweets are public, thus they must be filtered by hastags if we are interested in a special events. If hastags corresponding to this topic are selected, then the messages that contain any of these tokens induce the subgraph of this event. The main problem about

event filtering is how to determine every related hashtag. Zangerle et al. [32] developed a hashtag recommendation method that can help users finding these tokens. Now we will give a short outline of their recommender system.

First the user inputs the tweet message that he intends to post. Then all hastags contained in the most similar previously posted messages are returned from a database. Similarity was evaluated using *term frequency–inverse document frequency* (TF-IDF) statistic [25], that was also used in [21] for tweet content based user comparisons. Finally, all hastags extracted from queried messages are ranked. The system will recommend hastags for the user's given message according to this ranked list. In [32] the authors evaluated their recommender system using three different strategies.

- Overall popularity: hastags with the most occurrences are preferred.
- Recommender popularity: hashtags with the most occurrences within the set of recommendation candidates are preferred.
- Similarity ranking: hashtags contained in the most similar message are preferred.

Among the former cases similarity ranking performed the best.

Chapter 4

Information diffusion on dynamic networks

Myers and Leskovec [21] showed that the Twitter network is highly dynamic with about 9% of all connections changing in a month. Thus, in order to infer central nodes, the factors driving the dynamics of this social network must be considered. In this chapter, we will present some novel results about information diffusion on dynamic graphs. Usually, these experiments were tested on Twitter related data.

In Section 4.1 the results of Myers et. al [21], and Chierichetti et al. [9] will be discussed. They both designed models on how to identify key events or bursts in the information flow. While Chierichetti et al. [9] predicted global events from the amount of tweets and retweets, in [21] the authors focused on local bursts in the user-follower network.

The results of Bakshy et al. [1] will be detailed in Section 4.2. In this article, the authors tried to predict the influential users of a Twitter user-follower graph by generating diffusion cascades, according to several propagation rules. At first, they tried to extract influential vertices with regression merely based on network features. The main problem was the minority of cascades with significant size. They tried to introduce cascade content information into the model, but it did not improve the results.

Finally, in Section 4.3 the algorithm of Rodriguez et. al [24] is discussed, which can be used for inferring the structure and the edges of a latent temporal diffusion networks from node activation time series. Moreover, they examined an interesting aspect of centrality for many real-time events. However, their work is not Twitter related, but it has a direct application for non-rooted retweet graphs. As we mentioned in Section 3.2.2, the main problem for these graphs is that the edges, that belong to a given tweet cascade, cannot be recovered from Twitter data. Only the activation times of the retweeters are known for each cascade. Nevertheless, once the edges of the non-rooted retweet graph are inferred, centrality measures can be applied to extract important nodes.

4.1 Predicting bursts in network dynamics

Myers et. al [21] found that the dynamics of Twitter network structure can be characterized by steady rates of change, interrupted by sudden bursts. Significant events can generate a sudden increase in the amount of information flowing through the system. Otherwise, the number of follower relationships in the network is slowly increasing.

4.1.1 Event detection

Chierichetti et al. [9] proposed a robust model for the real-time identification of key events. They examined tweet and retweet production/consumption patterns around these incidents. The experiments showed that there is a "heartbeat" phenomenon in the balance of primary and secondary information spreading. When and important event unfolds, the users are busy with tweeting about it, as they try to report everything. Therefore, nobody has time to retweet these messages. Whereas after the event, there is a huge amount of tweets to be retweeted. Thus in this case, the secondary information spreading dominates the network. The authors used this phenomenon to obtain a simple classifier which, by only evaluating the tweet/retweet volume could detect these events.

4.1.2 Changes in the number of followers

Besides event detection, the bursts in tweet or retweet volume can be used for predicting local changes in the user-follower network (Section 3.2.1). Myers and Leskovec [21] also studied when would a burst in tweet count result in an unfollow burst for a particular user. Similarly, it was also considered whether a change in retweet count would cause a follower burst.

- *Unfollow burst:* Users in a coordinated way can drop their connections to the information source, in case of offensive tweets or high tweet frequency.
- *Follow burst:* Simultaneously, they can also create new connections to the information source. It is more likely when a specified user retweets more often. In this case, he can gain new followers, as new users are exposed to his tweets.

Naturally in [21], there were periodic fluctuations in the arrival rate across the hours of the day. To identify these bursts, periodicity had to be removed from the data. Let t_i denote the i^{th} hour of the month and $x = \{x_1, x_2, \cdots, x_n\}$ be the number of new follows a user receives for each hour of the month. The authors examined the difference between actual new follows and expected follows during t_i:

$$f(t_i) = x_i - E[x|h(t_i)] = x_i - \frac{\sum_{j;|t_i - t_j| \leq 48, h(t_i)=h(t_j)} x_j \cdot w(t_i - t_j)}{\sum_{j;|t_i - t_j| \leq 48, h(t_i)=h(t_j)} w(t_i - t_j)}, \tag{4.1}$$

where $h(t)$ is the hour of day for t, and $w(t)$ is an exponentially decaying weight function. The parameters of this function are set using maximum likelihood.

They defined the *ego-network* of a user. For a given person, it is the subgraph containing its followers and all induced follow relationships. The result showed that during a follower burst the tweet similarity increases between the user and his followers. Moreover, this relation also escalates among members of the ego-network. For the comparisons Myers et al. used the cosine similarity of the *term frequency–inverse document frequency* (TF-IDF) weighted word vectors between the two users' aggregated tweet documents. Nevertheless, the number of weakly connected components, and the follow edge density of the ego-network also increased after such a burst.

In addition to the former observations, they also developed a model for predicting retweet-follow bursts. They realized that the majority of new follows are the result of *triadic closure*. For example, suppose there is an increase in the retweet count of user A.

Then the probability that a given follower B will retweet is also higher. So the followers of B can discover A for the first time. The model is based on the idea, that the set of similar users in A's *2-hop neighbourhood* are the most likely to connect after a retweet burst.

4.2 Influence prediction

In word of mouth marketing there are two strategies for maximizing the diffusion [1].

1. Seeding the information, or new products with a few particular individuals, called "influentials". It is possible that the factors causing their significance are unknown.

2. Or targeting a broader spectrum of users ("influencers"), whose user parameters are desirable.

Bakshy et al. [1] measured influence in terms of the size of the entire diffusion tree associated with the events. Their Twitter related data consisted of several diffusion events. The seed users of these incidents were active in each month of the two month long observation window. The goal of this research was to predict the most influential users of the second month from the first with regression trees. The independent seeders were the root nodes. Then according to three different assumptions about the influence process, cascades were forming from each seeder along the follower relationships:

1. *first influence:* Suppose that a particular user u is exposed by multiple followee. Let v be the first among them who got acquainted with a given topic T. Then, u will belong to the same cascade, in which v is present.

2. *last influence:* In this case, u will be linked to the cascade of the followee, who got last infected with T.

3. *split influence:* Finally, in this assumption there is equal probability of all followees infecting u. Only those followees matter, who were exposed to T.

After the cascades were generated for a chosen influence rule, they computed individual-level influence as the logarithm of the average size of all cascades for which that user was a seed. Bakshy et al. [1] fit regression tree for two models, with cross validation. In the first only network properties, while in the second additional content informations were also used as features.

4.2.1 Prediction without content

In this model they tried to classify influencial users based on these features:

- Seed user attributes: #followers, #friends, #tweets, date of joining

- Past influence of seed users: average, minimum, and maximum past total/local influence

where past local influence indicates the average number of reposts by that user's immediate followers in the first month. On the other hand, past total influence refers to average total cascade size over the same time period [1].

The authors found that only the past local influence and follower count were the two informative attributes. For these features, they got almost perfect predictions for the

average value at each cut of the regression tree. However, the prediction for leaf nodes are very far from the actual values. This ambiguity is the result of the fact, that only a few cascades gets to be enormous and succesful. Most of them remain insignificantly small. So personalized predictions cannot be made on the average size of cascades.

4.2.2 Prediction with content

It was also examined whether information about the content of the seeded message can improve the model. They used Amazon's Mechanical Turk (AMT) to recruit human classifiers. With this popular service for survey and experimental research, Bakshy et. al. [1] could extract content related features from the data, such as rated interestingness, rated positive feeling, willingness to share via social networks etc. They found that none of the formerly introduced content features could improve the model's performance.

4.3 Inferring network structure from time series

4.3.1 Modelling diffusion networks

Rodriguez et. al [24] researched this problem in general for both static and dynamic diffusion networks. They modelled information diffusion as a continuous process over the edges of an unobserved network. The transmission rate for edge $i \to j$ is denoted by α_{ij} in the static setting. For every i, j nodes $\alpha_{ij} \to 0$ means that the transmission from i to j tends to be arbitrary long. In the dynamic case, the rate of information diffusion can change over time for each edge. Thus, in this setting the transmission rate at time t is denoted by $\alpha_{ij}(t)$. The authors developed an online inference algorithm (NetRate) which can determine the transmission rates from cascade time series. Their method computes the unknown $\alpha_{ij}(t)$ variables for all i,j pairs. At a given time t the edges with positive $\alpha_{ij}(t)$ are the predicted links for the latent diffusion network.

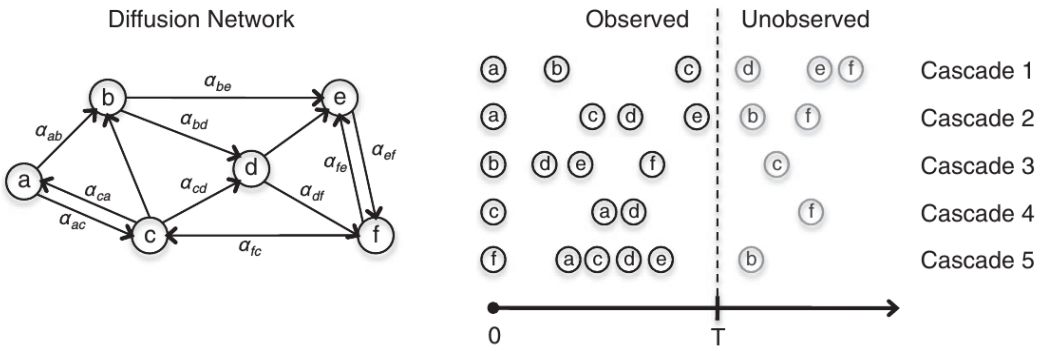


Figure 4.1: In [24] the α_{ij} transmission rates are computed from the cascade activation time series.

In both settings, the number of nodes N is fixed. For each cascade the information always spread from activated nodes to non-activated ones. This way cascades are induced on the fixed population (Figure 4.1). Rodriguez et. al [24] observed multiple cascades spreading at the same time. However, they assumed that these cascades propagate independently from each other. For a given cascade c the activation times for each node form a vector $t^c = (t_1^c, \cdots, t_N^c)$. They denoted the conditional likelihood of transmission between nodes j and i by $f(t_i|t_j;\alpha_{j,i})$. Here, t_i and t_j are the activation times corresponding to

$j \to i$ transmission ($t_j < t_i$). To properly analyze the transmission process several other functions were examined.

1. Cumulative density function, denoted as $F(t_i|t_j;\alpha_{j,i})$ for t_i and t_j activations.
2. The *survival function* for $j \to i$ edge express the probability, that node j does not cause node i to activate by time t_i

$$S(t_i|t_j;\alpha_{j,i}) = 1 - F(t_i|t_j;\alpha_{j,i}).$$

3. The *hazard function* incorporates the instantaneous activation rate, of edge $j \to i$

$$H(t_i|t_j;\alpha_{j,i}) = \frac{-S'(t_i|t_j;\alpha_{j,i})}{S(t_i|t_j;\alpha_{j,i})} = \frac{f(t_i|t_j;\alpha_{j,i})}{S(t_i|t_j;\alpha_{j,i})}.$$

Let A denote the set of all transmission rates for the static case

$$A := \Big\{\alpha_{j,i}|i,j \in \{1,\cdots,n\}, i \neq j\Big\}. \tag{4.2}$$

Similarly, $A(t)$ is the set of transmission rates for the dynamic case. Consider a given cascade t with activation vector $(t_1,\cdots,t_N)$. For cascade t, the survival of node i until time t_i can be computed by taking the product of the survival functions of the formerly activated nodes.

$$S(t_i|t_1,\cdots,t_N \setminus t_i;A) = \prod_{t_k<t_i} S(t_i|t_k;\alpha_{k,i}) \tag{4.3}$$

Suppose, there is a T time observation window. The subset of cascade t activated until T is denoted by $t^{\leq T}$. In the diffusion model of Rodriguez et. al [24], it is assumed that activations are conditionally independent given the parents of the activated nodes. In other words, the likelihood factorizes over the vertices

$$f(t^{\leq T};A) = \prod_{t_i \leq T} f(t_i|(t_1,\cdots,t_N) \setminus t_i;A) \tag{4.4}$$

Each factor of the former product can be computed by summing over disjoint events. These events correspond to a node j that first infected node i. Otherwise, i survived from other active nodes.

$$f(t_i|(t_1,\cdots,t_N) \setminus t_i;A) = \sum_{t_j<t_i} f(t_i|t_j;\alpha_{j,i}) \times \prod_{k:k\neq j,t_k<t_i} S(t_i|t_k;\alpha_{k,i}) \tag{4.5}$$

The product in (4.5) can be expanded with a factor that causes independence from the summation.

$$f(t_i|(t_1,\cdots,t_N) \setminus t_i;A) = \prod_{k:t_k<t_i} S(t_i|t_k;\alpha_{k,i}) \sum_{t_j<t_i} \frac{f(t_i|t_j;\alpha_{j,i})}{S(t_i|t_j;\alpha_{j,i})} \tag{4.6}$$

If we use the definition of the hazard function and the former simplification (4.6), the likelihood (4.4) can be expressed as

$$f(t^{\leq T};A) = \prod_{t_i \leq T} \prod_{k:t_k<t_i} S(t_i|t_k;\alpha_{k,i}) \sum_{j:t_j<t_i} H(t_i|tj;\alpha_{j,i}). \tag{4.7}$$

However, the information about not activated notes must be also considered.

$$f(t;A) = \prod_{t_i \leq T} \prod_{t_m > T} S(T|t_i;\alpha_{i,m}) \times \prod_{k:t_k<t_i} S(t_i|t_k;\alpha_{k,i}) \sum_{j:t_j<t_i} H(t_i|tj;\alpha_{j,i}) \tag{4.8}$$

Let $C = \{t^1, \cdots, t^{|C|}\}$ denote the set of cascades. Using the assumed independency, the likelihood over all cascades is the following

$$f(\{t^1, \cdots, t^{|C|}\};A) = \prod_{t^c \in C} f(t^c;A) \tag{4.9}$$

The value of the transmission rates is computed with maximum likelihood estimation on $f(\{t^1, \cdots, t^{|C|}\};A)$.

For the experiments, Rodriguez et. al [24] considered the exponential (Exp), power-law (Pow), and Rayleigh parametric models (Ray) (Figure 4.2). Nevertheless, for these models the likelihood of transmission tends to zero, as $\alpha_{ij} \to 0$. Furthermore, with Rayleigh model infection spreading can be simulated as it was formerly used in some epidemiological researches [14], [27].

Model	Transmission likelihood $f(t_i\|t_j;\alpha_{j,i})$		Log survival $\log S(t_i\|t_j;\alpha_{j,i})$	Hazard $H(t_i\|t_j;\alpha_{j,i})$
Exp	$\begin{cases} \alpha_{j,i} \cdot e^{-\alpha_{j,i}(t_i-t_j)} \\ 0 \end{cases}$	if $t_j < t_i$ otherwise	$-\alpha_{j,i}(t_i - t_j)$	$\alpha_{j,i}$
Pow	$\begin{cases} \frac{\alpha_{j,i}}{\delta}\left(\frac{t_i-t_j}{\delta}\right)^{-1-\alpha_{j,i}} \\ 0 \end{cases}$	if $t_j + \delta < t_i$ otherwise	$-\alpha_{j,i}\log\left(\frac{t_i-t_j}{\delta}\right)$	$\alpha_{j,i} \cdot \frac{1}{t_i-t_j}$
Ray	$\begin{cases} \alpha_{j,i}(t_i-t_j)e^{-\frac{1}{2}\alpha_{j,i}(t_i-t_j)^2} \\ 0 \end{cases}$	if $t_j < t_i$ otherwise	$-\alpha_{j,i}\frac{(t_i-t_j)^2}{2}$	$\alpha_{j,i} \cdot (t_i - t_j)$

Figure 4.2: Pairwise transmission models examined in [24].

4.3.2 The NetRate algorithm

The likelihood over the examined cascade set C was formulated in (4.9) for the static case. A similar equation can be derived for the dynamic case with $A(t)$. Rodriguez et. al [24] defined the corresponding maximum likelihood optimization problem for each network inference problems. Although in both cases, the overall log-likelihood function is minimized.

1. Static network inference problem:

$$\begin{aligned} &\text{minimize}_A - \sum_{c \in C} \log(f(t^c;A)) \\ &\text{subject to } \alpha_{ij} > 0; i,j = 1 \cdots N; i \neq j \end{aligned} \tag{4.10}$$

   The inferred links of the network will be $i \to j$ pairs with positive α_{ij} transmission rate.

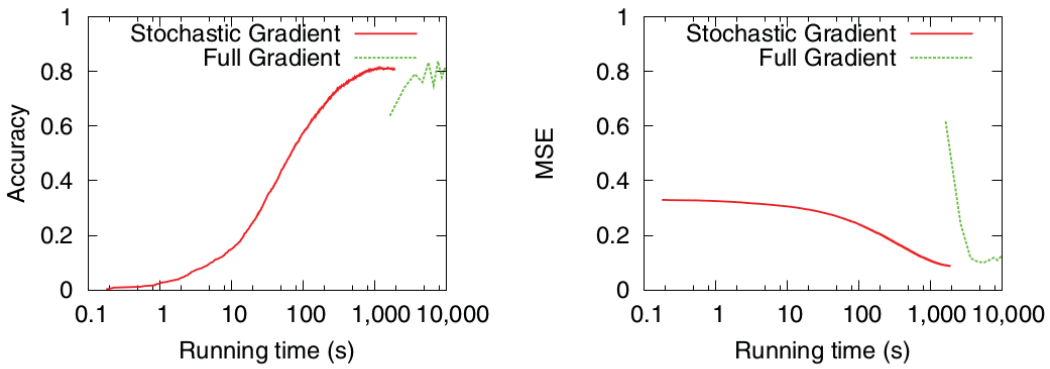
2. Dynamic network inference problem:

$$\text{minimize}_{A(t)} - \sum_{c \in C} w_c(t) \log(f(t^c; A(t)))$$
$$\text{subject to } \alpha_{ij}(t) > 0; i, j = 1 \cdots N; i \neq j, \tag{4.11}$$

where $w_c(t) > 0$ are weights that penalize old cascades. With these weights, the authors tried to simulate the priority of recent cascades over older ones. Indeed, new cascades have higher importance in inferring current network structure. The dynamic setting can be reduced to the static case (4.10), if the weights in (4.11) are equal for all $c \in C$ and constant over time.

In [24] it was proved that for log-concave survival functions and concave hazard functions both (4.10) and (4.11) problems are convex in A or $A(t)$ respectively. These conditions hold for the examined parametric models (Exp,Pow,Ray), detailed in Figure 4.2.

NetRate originally uses a full gradient method to solve the convex optimization problems (4.10) and (4.11). Moreover, Rodriguez et. al [24] also implemented this method with stochastic gradient descent, which they called InfoPath algorithm. They found that InfoPath was approximately one order of magnitude faster than NetRate, that uses full gradient method 4.3.



(a) Accuracy (b) MSE

Figure 4.3: Similar accuracy (a) and Mean Square Error (MSE) (b) could be achieved approximately one order of magnitude faster with InfoPath for the same input [24].

In Section 5.2, we also discuss stochastic gradient descent because later we will use this method to solve the link prediction problem for dynamic graphs.

4.3.3 Results of NetRate related to node centrality

In [24], there is a meticulous evaluation about how NetRate performs on synthetic and real-world data. Due to space constraints only centrality related results will be emphasized.

Rodriguez et. al [24] run their experiments on several 2011 world events-induced dynamic graph. They investigated the ratio of mainstream media and blogs among the top-100 most central nodes over time on these diffusion networks. For extracting central nodes they used the inverse definition of harmonic centrality detailed in Section 1.1.5. In their work the centrality of node x was defined by

$$c(x) := \sum_y \frac{1}{d(x, y)}, \tag{4.12}$$

where $d(x, y) = \infty$ if node y is not reachable from x. Harmonic centrality prefers nodes that are reachable from most of the vertices by a short path. While for the measure defined in (4.12) the vertices, from whom most of the other nodes are close, are more central.

After inferring the edges with their algorithm, the authors could compute the centrality score for each node. Rodriguez et. al [24] found that the percentage of mainstream media and blogs among central nodes do vary for different topics. In Figures 4.4(a) and (b) mainstream media is always more central. On the other hand, for other topics like Gaddafi in Figure 4.4(c), blogs have higher percentage among central nodes during the entire event. The balance between these two source of information can also change over time. While for UK Royal Weeding and NBA in Figures 4.4(d) and (e) the ratio remained approximately constant, for other topics like Occupy in Figure 4.4(f) a few bursts occurred. The aspect of separating central users, similar to the decomposition of nodes to mainstream media and blogs, is very important. In real time events sometimes a very small subset of supposedly insignificant users can dominate the network. Thus, to extract true central nodes, a filtering must be applied.

In Section 4.3 we gave a short description about the work of Rodriguez et al. They developed an algorithm for predicting the structure of an unknown diffusion network. Although the authors examined centrality on the inferred network, their method cannot be used for predicting central nodes in advance, as the forthcoming activation times of nodes are unknown. In Chapter 5 we will present a method that can be used for link prediction.

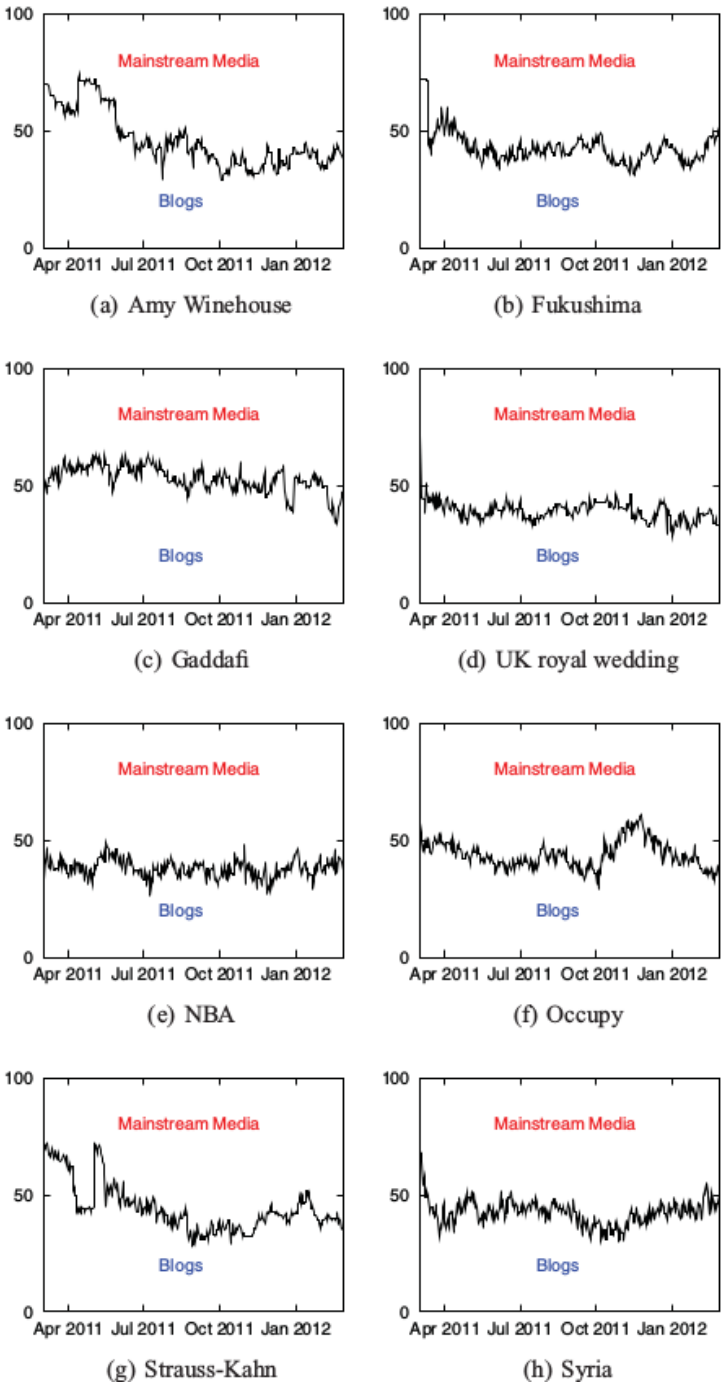


Figure 4.4: Percentage of blogs and mainstream media in the top-100 sites with highest centrality score. Mainstream media are represented in red, and blogs are represented in blue [24].

Chapter 5

Stochastic gradient descent and link prediction

In many real-life problems, the appearance of a link in a dynamic network can be associated with an event. The related task of inferring these directed edges is the link prediction problem introduced by Liben-Nowell and Kleinberg [19]. Particularly, in our work each directed edge corresponds to a mention event between Twitter users. In this thesis, we examine the link prediction problem as a matrix factorization problem for recommender systems. In Section 5.1, we give a short preview on how matrix factorization is used in recommender systems. Then, we solve the corresponding matrix factorization problem, explained in Section 5.3, with online stochastic gradient descent (SGD), which is described in Section 5.2.

5.1 Matrix factorization in recommender systems

Over the last few years, due to the Netflix prize several novel methods were proposed for collaborative filtering [16]. When recommendations are based on latent factors, matrix factorization methods proved to be effective. Suppose, we have n users and m items. The training set T consists of real user-item interactions, that are stored in a sparse *rating matrix* $R \in \mathbb{R}^{n \times m}$. These ratings can be explicit or implicit. In case of *explicit rantings*, there is available information about how the given user u felt about a particular item i. For example, a popular way is to rate items on a 1 to 5 scale. On the contrary, for an *implicit rating* there is no feedback on whether the user was satisfied with his choice. Only the fact that u saw item i is known. Therefore, $r_{ui} = 1$ for the elements of the training set.

Recently, several matrix factorization algorithms were proposed for recommender systems. These methods compute $P \in \mathbb{R}^{n \times k}$ and $Q \in \mathbb{R}^{m \times k}$ that minimize the objective function

$$F = \sum_{(u,i) \in T} (r_{ui} - p_u q_i^T)^2, \tag{5.1}$$

where $k \ll \min\{n, m\}$ in practice. Parameter k is the number of assumed latent factors that characterize user-item interactions. It can affect the quality of the approximation, but for bigger k values the runtime also increases significantly. After decomposing R into P and Q, the prediction for an arbitrary $(u, i) \notin T$ user-item pair is

$$\hat{r}_{ui} := p_u q_i^T, \tag{5.2}$$

where p_u is the u^{th} row of the *user matrix* P and q_i is the i^{th} row of the *item matrix* Q. In other words, we approximate the rating matrix R with

$$R_k = PQ^T.$$

This is similar to the dimensionality reduction of full matrices, which can be solved with *singular value decomposition* (SVD) [18]. Here, the difference is minimized only for the elements of the training set T (5.1).

There are several techniques to improve the quality of the user-item recommendations, which can be used independently in most cases for factor models.

1. For factor models, we can avoid over-training with *regularization* [16]. Here, the original objective function (5.1) is shifted

$$F = \sum_{(u,i)\in T} (r_{ui} - p_u q_i^T)^2 + \lambda \cdot (\|p_u\|^2 + \|q_i\|^2), \tag{5.3}$$

   where λ parameter is the *regularization rate*. If regularization is not considered ($\lambda = 0$), then the factor model may place too much focus on the known sparse elements of R. As a result, the model may give poor predictions for the unknown $(u, i) \notin T$ user-item pairs.

2. Sometimes it can be useful if the model incorporates some additional information about the users and the items which is independent from the factors. One possibility is to introduce *biases* into the model [16]. Let b_u denote the bias of user u. Similarly, a $\bar{b}_i$ bias is set for each item i. These variables should be taken into account for each (u, i) user-item prediction.

$$\hat{r}_{ui} := p_u q_i^T + b_u + \bar{b}_i \tag{5.4}$$

   Moreover, the objective function F (5.3) has to be adjusted according to

$$F = \sum_{(u,i)\in T} (r_{ui} - p_u q_i^T - b_u - \bar{b}_i)^2 + \lambda \cdot (\|p_u\|^2 + \|q_i\|^2) + \lambda_b \cdot b_u + \bar{\lambda}_b \cdot \bar{b}_i, \tag{5.5}$$

   where λ_b is the regularization rate for the user biases, and $\bar{\lambda}_b$ is the regularization rate for the item biases.

3. Another technique is to use an additional g transformation on the predicted $\hat{r}_{ui}$ value. For example, the formula is

$$\hat{r}_{ui} := g(p_u q_i^T + b_u + \bar{b}_i), \tag{5.6}$$

   if biases are also introduced. Generally, this transformation is the *sigmoid function*

$$sigmoid(t) = \frac{1}{1 + e^{-t}}. \tag{5.7}$$

5.2 Stochastic gradient descent (SGD)

5.2.1 SGD in general

Stochastic gradient descent is widely used in machine learning problems. It is used for minimizing objective functions that has the form of a sum. Compared to the standard gradient approach it is more cost-effective as in each iteration only an approximation of the gradient is needed. Let F denote the given objective function, which takes its minimum value at ω. While for the standard gradient method the gradient $\bigtriangledown F$ must be computed, SGD uses $\bigtriangledown F_i$ for approximation in the i^{th} iteration. From the former notations the update rule follows for both iterative processes

$$\omega = \omega - \alpha \bigtriangledown F(\omega) = \omega - \alpha \bigtriangledown F_i(\omega), \tag{5.8}$$

where α is a parameter, called the learning rate.

5.2.2 SGD for matrix factorization

The main difference between factor models is the order of how the p_u and q_i vectors are updated. Stochastic gradient descent is a method, which can learn the model in two different settings.

1. **Offline (Batch) SGD:** In each iteration, it loops through on all $(u, i) \in T$ ratings in an arbitrary order. For a given $(u, i) \in T$, first the user vector p_u, then the item vector q_i is updated.

2. **Online SGD:** in this setting the model is updated for each (u, i) user-item pair of the training set only once. Therefore, each iteration corresponds to a given $(u, i) \in T$. Similarly to offline SGD, for a given $(u, i) \in T$, first p_u, then q_i is updated.

The objective function, that is used in this factor model is mean square error (MSE). For a particular r_{ij} rating, it is defined by

$$F_{ij} = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - p_u q_i^T)^2. \tag{5.9}$$

First, the p_i user vector is updated according to (5.8).

$$p_i = p_i - \alpha \frac{\partial F_{ij}}{\partial p_i}, \tag{5.10}$$

After p_i has been updated, the item vector q_j needs to be refreshed as well.

$$q_j = q_j - \alpha \frac{\partial F_{ij}}{\partial q_j}, \tag{5.11}$$

From the definition of F_{ij} (5.9) the final update formula follows

$$p_i = p_i + \gamma(r_{ij} - \hat{r}_{ij})q_j = p_i + \gamma \cdot err \cdot q_j \tag{5.12}$$

$$q_j = q_j + \gamma(r_{ij} - \hat{r}_{ij})p_i = q_j + \gamma \cdot err \cdot p_i, \tag{5.13}$$

where the *learning rate* of the factor model is denoted by γ. It is important to note, that regularization can be applied for SGD as well. Although, for regularization the objective function is modified

$$F_{ij} := (r_{ij} - \hat{r}_{ij})^2 + \lambda \cdot (\|p_i\|^2 + \lambda \|q_j\|^2). \tag{5.14}$$

In this case, the following update formula can be derived [16]

$$p_i = p_i + \gamma(err \cdot q_j - \lambda p_u) \tag{5.15}$$
$$q_j = q_j + \gamma(err \cdot p_i - \lambda q_i). \tag{5.16}$$

Stochastic gradient descend can be also modified to incorporate biases or sigmoid transformation, that were mentioned in Section 5.1. Independently from these techniques, there is a way to improve SGD, when the given ratings are implicit. When a given (u, i) user-item pair is updated in the model, τ negative samples are generated to adjust the majority of positive samples for implicit ratings. The parameter τ is called the *negative sample rate*. For example, when p_u vector is updated an L_u list of items are sampled, where $(u, j) \notin T$ for all $j \in L_u$. Moreover, a new objective function will be used, which contains the negative samples as well.

$$F_{ui} := (r_{ui} - p_u q_i^T)^2 + (p_u q_{j_1}^T)^2 + \cdots + (p_u q_{j_\tau}^T)^2 \tag{5.17}$$

Similarly, for item i an L_i list of users are chosen, where $(w, i) \notin T$ for all $w \in L_i$. The formerly generated negative list of users, will be used in the following objective function.

$$F_{ui} := (r_{ui} - p_u q_i^T)^2 + (p_{w_1} q_i^T)^2 + \cdots + (p_{w_\tau} q_i^T)^2 \tag{5.18}$$

5.3 Link prediction with online SGD

For the link prediction problem, $R \in \{0, 1\}^{n \times n}$ is a squared matrix. Here, the elements of the T training set are the directed edges of the graph. When there is a link from node i to j, then $r_{ij} := 1$. The i^{th} row of the user matrix p_i, that is a k dimensional vector, tries to capture the quality of node i as a link source. Similarly, the j^{th} row of the item matrix q_j, represents node j being a link's target. Therefore, link prediction can be reduced to a recommendation problem with implicit ratings. The model is trained with online *SGD*. Each iteration corresponds to the occurrence of a directed edge ij. However, one should generate negative samples to train the model properly. In our case, it means that in the corresponding iteration of ij edge, we generate a list of nodes that node i did not link to. In our experiments, we generate these sample nodes uniformly.

Chapter 6

Movement datasets

We conducted our experiments on four different Twitter datasets. Each dataset is related to a civil protest or movement. First, we will give a summary on the size of the full datasets. Especially, the number of users and several other Twitter-related metric will be presented in Table 6.1.

	15-o	*occupy*	*yosoy*	*20-n*
Number of users	96,935	371,401	395,988	366,155
Number of tweets	410,482	1,947,234	2,439,109	3,535,155
Number of retweets	280,387	1,272,443	1,716,994	1,406,399
Tweets with mention(s)	281,830	1,396,051	1,844,134	2,340,840

Table 6.1: Twitter movement datasets

We extracted all tweets, that contained any mentions. Our dynamic mention graphs were constructed, from these messages. Figures 6.1, 6.2, 6.3, and 6.4 show, that these graphs are indeed dynamic, with several bursts in the number of edges and nodes. The graph and temporal parameters are shown in Table 6.2. It is important to note, that several directed edges in these graphs can correspond to a single message, as the number of mentions in a tweet is not fixed. For each experiment, we divided these events into time intervals. In this chapter, we will give a short description of each event, on which our aim is to predict central users in advance for the next time interval.

	15-o	*occupy*	*yosoy*	*20-n*
Number of vertices	67,153	335,167	336,081	282,944
Number of edges	247,329	1,837,946	2,252,070	1,503,633
Start date	2011-10-10	2011-10-10	2012-05-18	2011-10-29
End date	2011-10-31	2011-10-31	2012-12-27	2011-11-25
Number of days	22	22	223	27

Table 6.2: Dynamic mention graphs generated from the datasets

OccupyWallStreet dataset (occupy)

The original Occupy Wall Street protest began on September 17, 2011 in New York [30]. The participants demonstrated against the unequal distribution of wealth in the United Stated. On October 15, several local movements took place in many major cities in the

US. It is the reason, why we can observe a huge burst in the graph size on that day in Figure 6.1. Finally, the protesters were forced out of Zuccotti Park on November 15, 2011.

Our dynamic graphs were constructed only from messages that contained mentions. Thus, the directed edges of the mention graphs are from 2011 October 10 to 2011 October 31. In Figure 6.1, the number of nodes and edges are shown for each day.

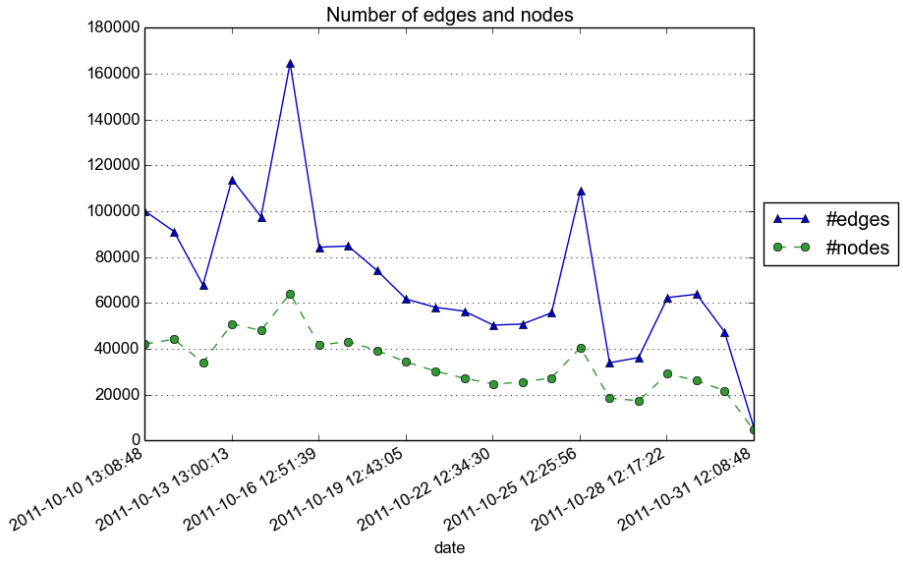


Figure 6.1: Number of users and edges in the *occupy* mention graph.

20-N dataset (20-n)

November 20, was the death date for both José Antonio Primo de Rivera and Generalissimo Franco. They are the two best known and controversial Spanish persons in the 20th-century. The 20-N symbolic abbreviation refers to their date of death, that is commemorated every year [29]. In 2011, the Spanish general election coincided with the 75th anniversary of de Rivera's death.

The mention graph, extracted from 20-N Twitter messages, contains directed edges from 2011 October 31 to 2011 November 24. The second burst in the number of edges and vertices in Figure 6.2 corresponds to November 20, that was the day of the election. With the exception of November 19, there is a steady increase in the size of the dynamic graph in the preceding 6 days of the anniversary.

15 October global protests dataset (15-o)

On October 15, 2011 several protests were held in the major cities of the United States and Europe [28]. These movements were mainly the results of the Arab Spring and the Occupy Movement.

On the formerly mentioned date of the demonstrations, an immense burst occurs in the number of edges and vertices of the mention graph. Nevertheless, Figure 6.3 shows that beside the main event our temporal graphs is stable.

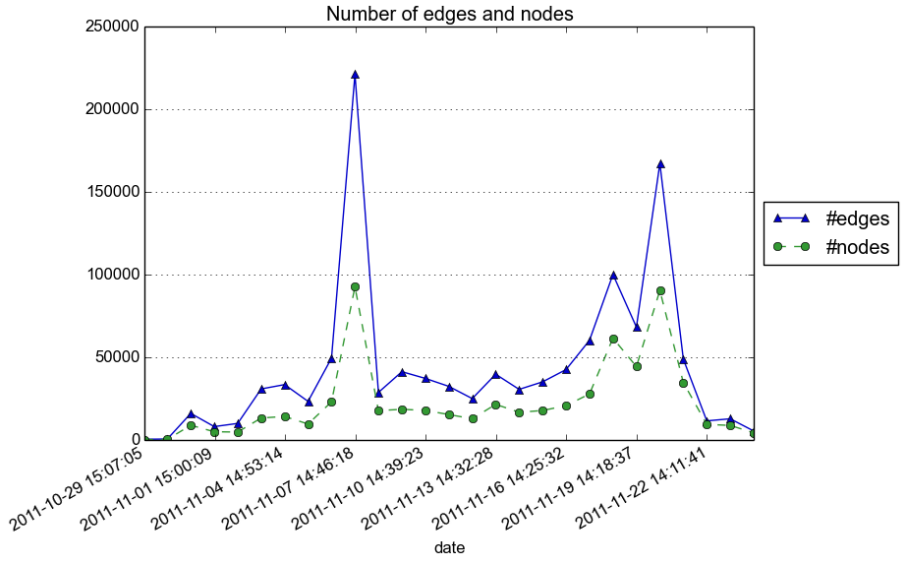


Figure 6.2: Number of users and edges in *20-n* mention graph.

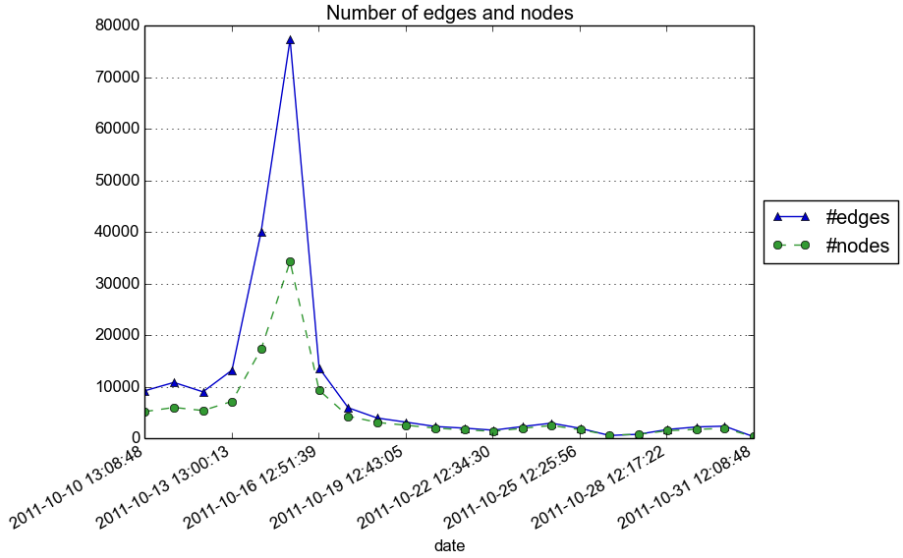


Figure 6.3: Number of users and edges in *15-o* mention graph.

Yo Soy 132 dataset (yosoy)

This social movement started on May 11, 2012 in Mexico [31]. Originally, 131 university students began this protest. Soon, students from several campuses started to support the founders. They were stating that "I am 132", that is "Yo soy 132" in Spanish. The protest was self-proclaimed as the "Mexican spring". On the other hand, the international media named it as the "Mexican occupy movement".

The directed edges are from 2012 May 18 to 2014 December 27. Figure 6.4 show that this data is highly dynamic, with several bursts in the number of edges and vertices.

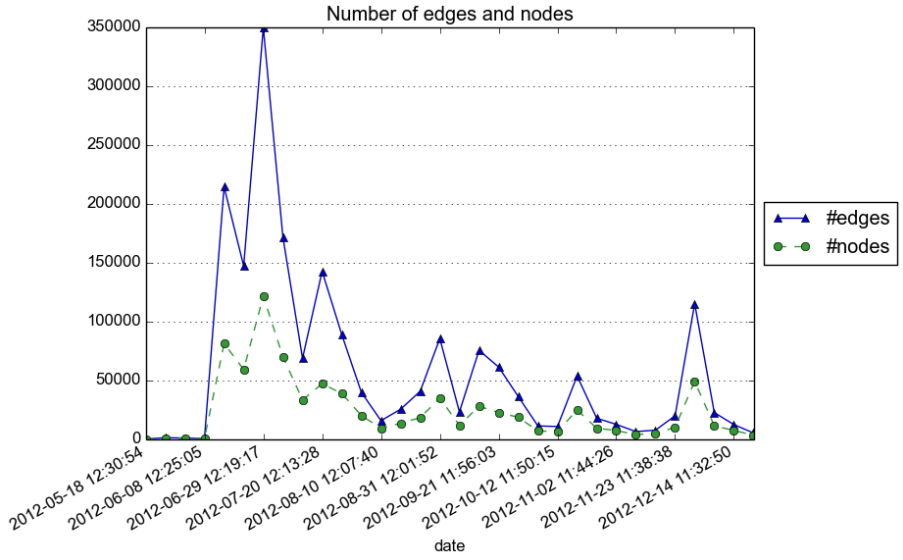


Figure 6.4: Number of users and edges in *yosoy* mention graph.

The main similarity of the formerly mentioned movements is their leaderless structure. It means that there was no leader appointed. The participants posted their point of view and demands on several social networking systems. Thus, over a short period of time these events got huge publicity and won the support of the masses. In these kind of leaderless movements, the key problem is to identify central, influential persons in advance. In the next chapter, we will present our factor-model based method for predicting centrality on dynamic graphs corresponding to such real-world events.

Chapter 7

Centrality prediction for dynamic graphs

In order to give centrality predictions, we solve the link prediction problem explained in Chapter 5. We proposed online SGD for this task. We compare the effectiveness of our model with the baseline method explained in Section 7.2. In the experiments, we compute the indegree (Section 1.1.1), outdegree, negative-β measure (Section 1.1.2), PageRank (Section 1.2.4) and SALSA (Section 1.2.6) centrality measures on the dynamic mention graphs, that correspond to the Twitter datasets described in Section 6. For PageRank and SALSA computation, we use iterative algorithms, that run for 10 iteration for both measures. In case of PageRank, the damping factor α was set for 0.85. All centrality score is computed with our C++-based framework, that is detailed in Appendix A.

7.1 Centrality prediction problem

Let D be a dynamic mention graph from a finite time window. Then a static directed G graph with a corresponding w weight function can be constructed from D. G will contain all ever existed vertices and edges of the dynamic graph. For an arbitrary uv edge of G, $w(uv)$ is the time when u mentioned v. Of course, multiple edges can occur. After G was constructed from the original dynamic graph, we decomposed it into subgraphs that correspond to consecutive disjoint time intervals with equal length. G_i is the induced subgraph of G, that contains all edges from the i^{th} time interval. Let Δt denote the length of each time interval. Usually, in the experiments Δt was set for one day.

In this thesis, our goal was to infer the top k central nodes of G_i, from the previous subgraphs G_j $(j < i)$. In our experiments, central nodes were extracted according to various centrality measures shown in Figure 7.1. However, it is important to note that we did not enable multiple edges in the G_i subgraphs. Therefore in each interval, only the directed edge related to the first $u \to v$ mention is present in the graph for all u,v pair of nodes. In the i^{th} time interval, let $L_i(k)$ denote the original ordered list of top k central vertices for a given measure. Suppose, that our model M also predicted these nodes for the same centrality measure and stored them in $\hat{L}_i^M(k)$. Naturally, the more match exists between these list, the better our model performs. For precise evaluations we used precision (2.1.1) and NDCG (2.1.3), that are both appropriate metrics for comparing $L_i(k)$ and $\hat{L}_i^M(k)$. The results for our online SGD-based model can be found in Section 7.4. Although, to find out whether our model is effective it should be compared to baseline

methods.

7.2 Baseline prediction from the previous interval

For centrality predictions, the first idea that comes to mind is probably to give the same top k nodes, that were central in the previous interval. Formally, it means $\hat{L}_i^B(k) := L_{i-1}(k)$. The quality of this baseline depends on Δt. Indeed, a highly dynamic graph can change significantly in structure over a longer Δt time frame. In Figure 7.1 the changes in the top 10 central users are shown for the *occupy* dataset from the first day to the second.

in_degree		out_degree		beta		pagerank		salsa_auth		salsa_hub	
id	delta	id	delta	id	delta	id	delta	id	delta	id	delta
300894	new	211977	new	300894	new	300894	new	31742	new	294068	new
31742	new	303621	new	332874	new	31742	new	4529	new	211977	new
4529	new	301061	new	31742	new	4529	new	300894	new	303621	new
237942	new	294068	new	4529	new	289462	new	237942	new	301061	new
332874	new	207670	new	17970	new	309436	new	93602	new	207670	new
93602	new	185921	new	237942	new	34463	new	332874	new	213061	new
34463	new	213061	new	93602	new	315751	new	46226	new	185921	new
46226	new	236714	new	289462	new	237942	new	17970	new	236714	new
315751	new	329614	new	315751	new	332874	new	34463	new	329614	new
17970	new	6312	new	34463	new	226244	new	210680	new	97483	new

in_degree		out_degree		beta		pagerank		salsa_auth		salsa_hub	
id	delta	id	delta	id	delta	id	delta	id	delta	id	delta
4529	2	68146	new	93602	6	4529	2	4529	1	68146	new
93602	4	236714	6	300894	-1	93602	new	93602	3	24020	new
237942	1	303621	-1	4529	1	301447	new	237942	1	236714	5
31742	-2	211977	-3	237942	2	126301	new	300894	-1	303621	-1
300894	-4	97483	new	96563	new	245951	new	31742	-4	211977	-3
46226	2	213061	1	276015	new	300894	-5	46226	1	97483	4
259165	new	207670	-2	226244	new	15218	new	259165	new	213061	-1
62816	new	192994	new	31742	-5	237942	0	62816	new	207670	-3
276015	new	24020	new	62816	new	281203	new	276015	new	84088	new
210680	new	84088	new	75153	new	31742	-8	210680	0	294068	-9

Figure 7.1: Changes in the top $k = 10$ central users in *occupy* from the first day to the second. The numbers and colors mark the number of position changes. In the first day every user is new.

Moreover, according to Figure 7.1, there are significant similarities between the top 10 central nodes of indegree, negative β-measure, PageRank and SALSA authority score. Several users occur in both of the formerly mentioned measure toplists. The same can be observed for outdegree and SALSA hub score.

The quality of this baseline prediction can be also evaluated with metrics detailed in Chapter 2. In Figures 7.2 and 7.3, for each interval i we computed NDCG for $\hat{L}_i^B(10)$ and $L_i(10)$. If the top 10 central nodes in interval i were the same as in the $(i-1)^{th}$ time frame, then NDCG score would be 1.0 for interval i. For both *occupy* and *20-n* datasets, the baseline prediction performed the best for SALSA authority score among the other centrality measures. For *20-n* dataset the previous interval based prediction performed better in average, than for *occupy*. Finally, while for *occupy* there were intervals were SALSA hubbiness and outdegree could perform better than the others, for *20-n* these two measures usually achieved significantly smaller NDCG than indegree, negative β-measure, PageRank and SALSA authority.

In Section 7.4 the experiments prove, that our online SGD-based model performs better

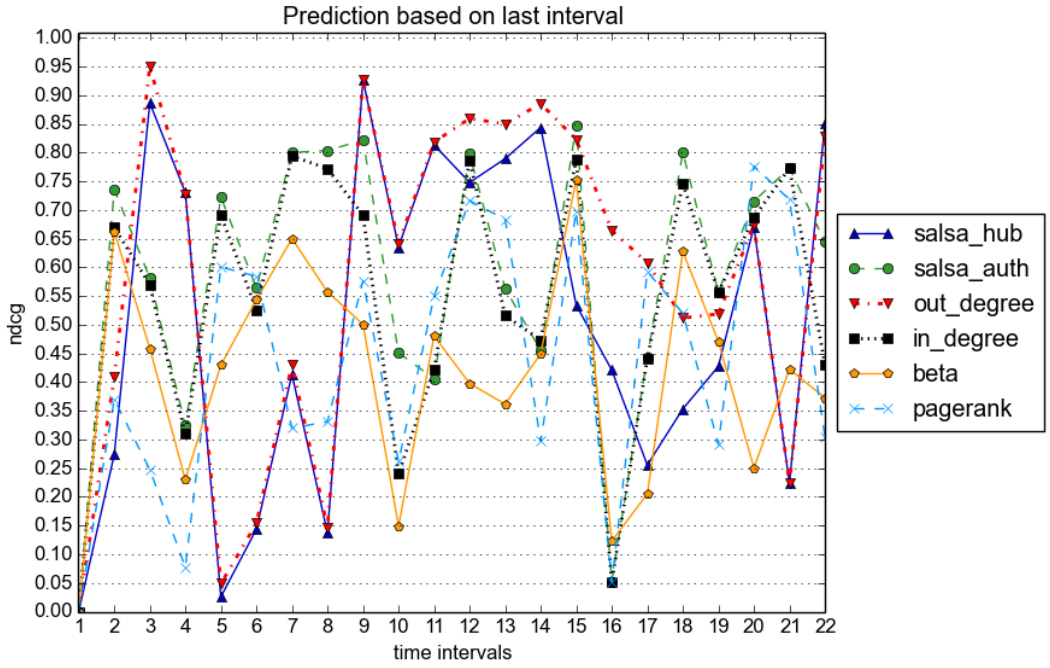


Figure 7.2: Quality of the previous interval based baseline prediction for *occupy* ($k = 10$).

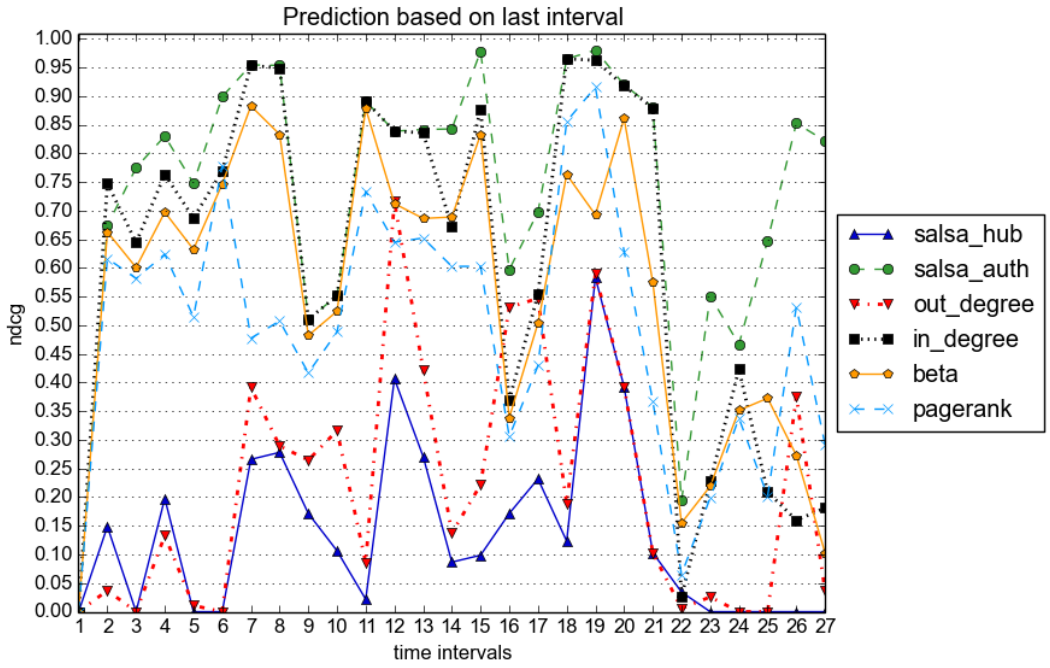


Figure 7.3: Quality of the previous interval based baseline prediction for *20-n* ($k = 10$).

for several centrality measures, than the baseline prediction introduced in this section. Moreover, we also recover the main feature driving centrality in the next interval. These are the homophily edges occurring in the next time frame. The precise definition and the related experiments are presented in the next section.

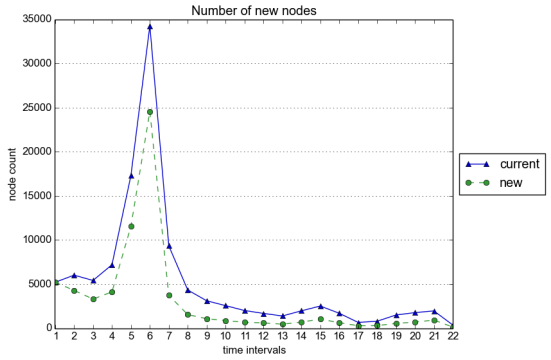
7.3 Link prediction results

In this thesis, we examined link prediction for a special case. Our goal was to infer the edges of G_i from the data of the first $(i-1)$ time interval. We proposed online stochastic gradient descent (SGD) for this task. The result can be found in Section 7.3.2. We also discovered, that a special type of edges, called homophily edges, dominate the centrality in G_i. This definition is explained in Section 7.3.1.

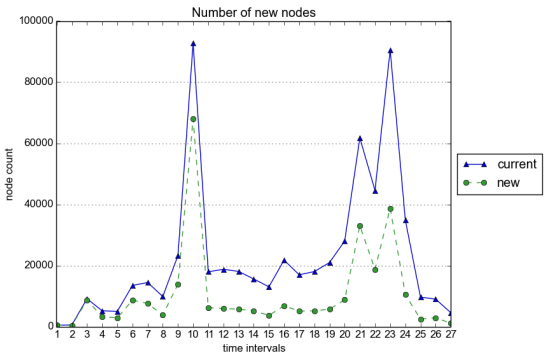
7.3.1 Homophily edges dominate centrality

First, for each interval i group the edges according to, whether any endpoints of an arbitrary $uv \in G_i$ directed edge, have been seen in the first $(i-1)$ time interval.

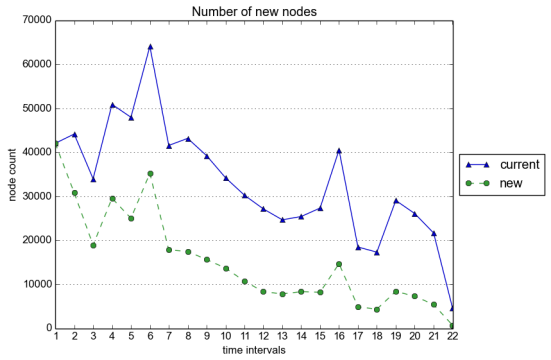
- **former edges:** uv directed edge occurred before the i^{th} time interval as well.
- **new edges:** uv directed edge did not occurred before the i^{th} time interval.
  - Neither u nor v occurred before
  - Only the source vertex u occurred before
  - Only the target vertex v occurred before
  - **homophily edges:** Both u and v occurred before the i^{th} time interval.



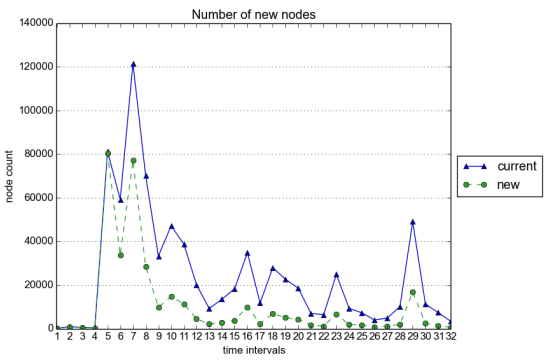
(a) *15-o*: $\Delta t = 1$ day



(b) *20-n*: $\Delta t = 1$ day



(c) *occupy*: $\Delta t = 1$ day



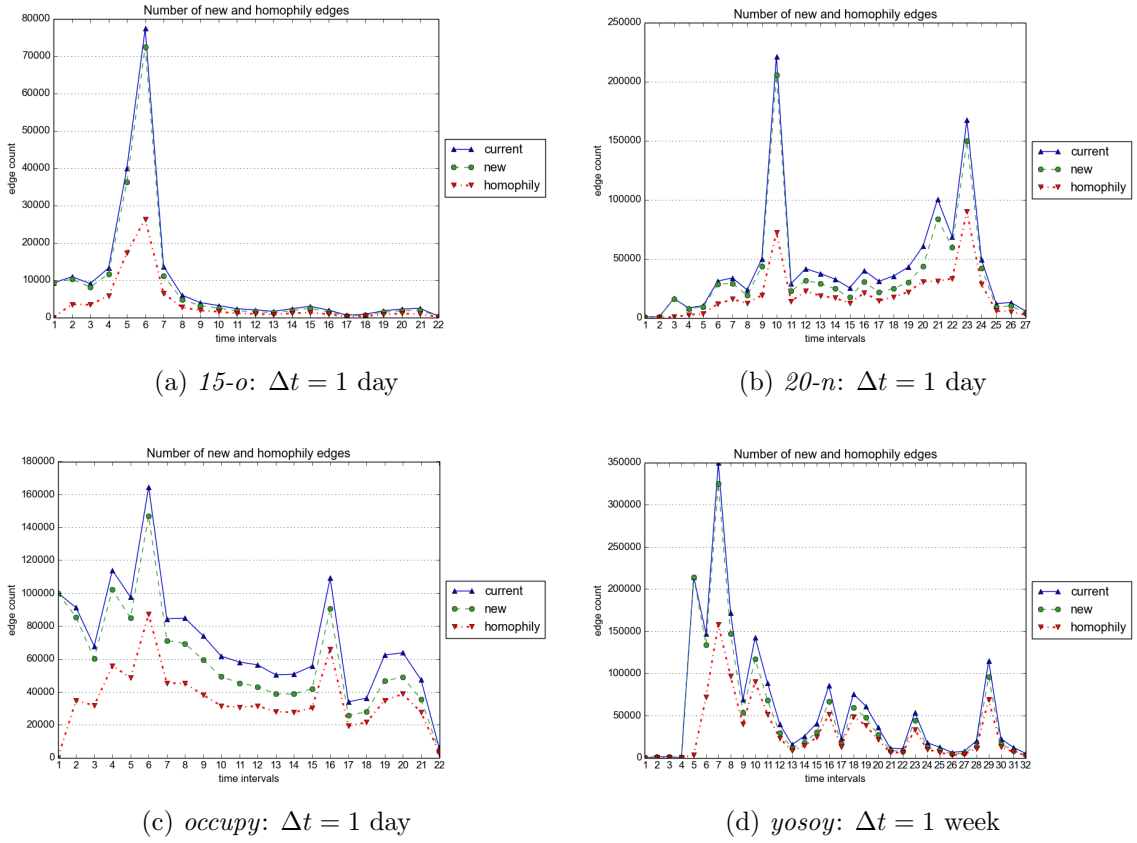
(d) *yosoy*: $\Delta t = 1$ week

Figure 7.4: The number of new vertices compared to the current number of vertices in each time interval for the given mention graphs.

In Figure 7.4 the number of new vertices are shown for each interval in the examined Twitter mention graphs. Naturally, a vertex is new in interval i, if it never occurred before.

Figures 7.4a, 7.4b and 7.4c show, that in average nearly half of the users of each interval are new for the mention graphs related to *15-o*, *20-n* and *occupy* datasets. In Figure 7.4d it can be observed, that the dynamic network constructed from *yosoy* dataset contains less new vertices.

That is the reason, why the number of new edges are also high in Figure 7.5. Although in case of dynamic graphs, the appearance of edges connected to new vertices cannot be predicted, but from the data of previous intervals one may be able to infer homophily edges. Furthermore, it is an important question whether the new homophily edges or the former edges dominate the centrality in G_i.



(a) *15-o*: $\Delta t = 1$ day

(b) *20-n*: $\Delta t = 1$ day

(c) *occupy*: $\Delta t = 1$ day

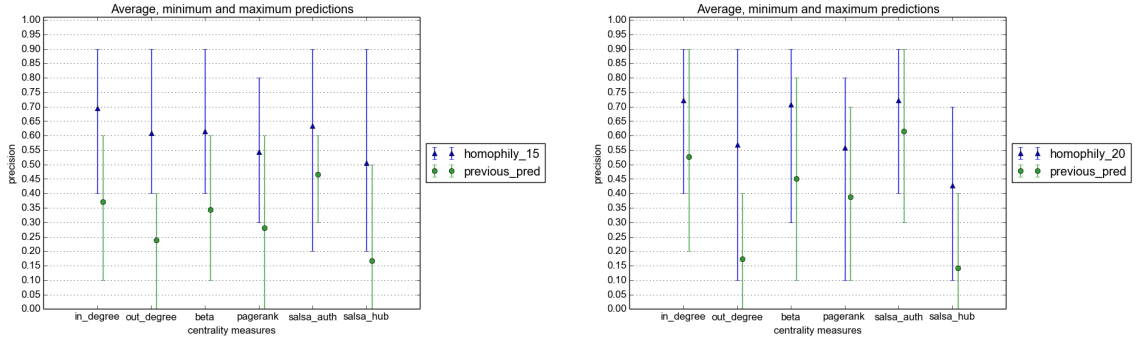
(d) *yosoy*: $\Delta t = 1$ week

Figure 7.5: The number of new edges and homophily edges compared to the current number of edges in each time interval for the given mention graphs.

Now, let $\hat{G}_i$ be the subgraph of G_i that is induced by only the homophily edges of the i^{th} interval. In order to access the potential of the homophily edges, we conducted an experiment, where G_i was approximated by $\hat{G}_i$. Then, the formerly mentioned centrality measures were computed on $\hat{G}_i$. The result of this experiment are shown in Figure 7.6. We found that the average precision (P@10) nearly doubled for each measure compared to the baseline prediction. Therefore, it worth to put an emphasis on homophily edge prediction.

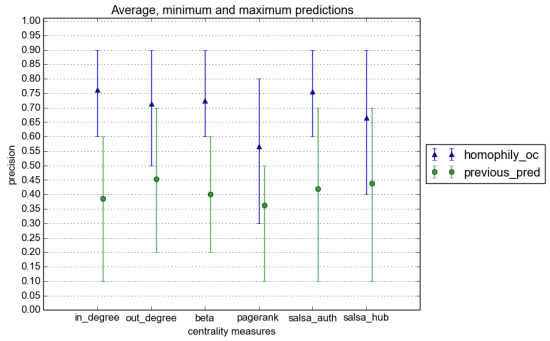
7.3.2 Link prediction results for online SGD

In this section, we present link prediction results for our model. For each experiment, a global timeline is generated from a given mention graph. This timeline contains all mention events related to an uv directed edge for all u, v pairs of nodes. From this global timeline, we generate a toplist of length L for each interval using online stochastic

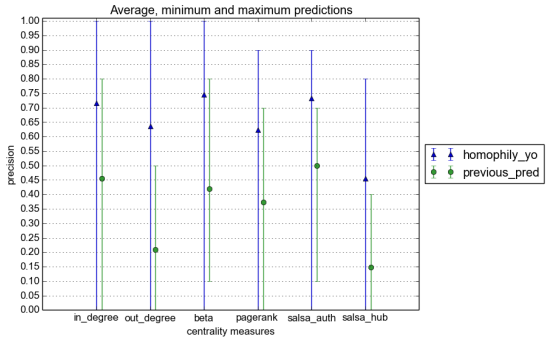


(a) *15-o*

(b) *20-n*



(c) *occupy*



(d) *yosoy*

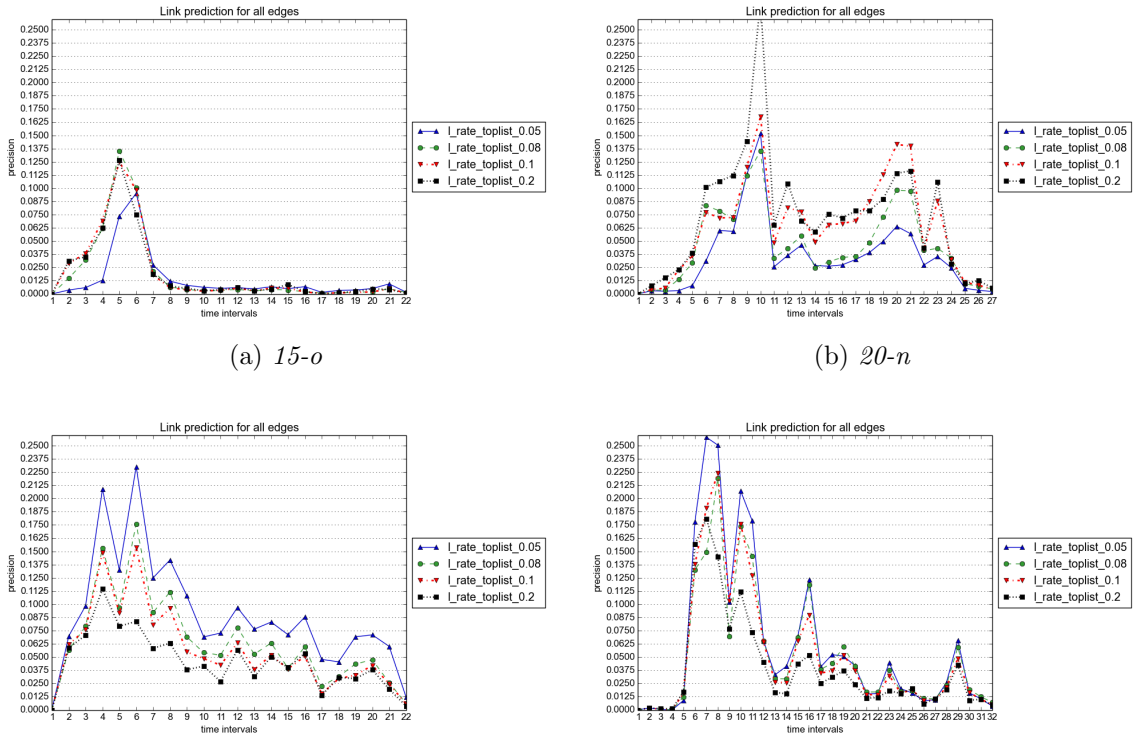
Figure 7.6: Average P@10 for centrality "predictions" based on current homophily edges. Here, *previous_pred* denotes the result of the baseline model discussed in Section 7.2

gradient descent. Let T_i^L denote this toplist for the i^{th} interval, that contains the predicted links for G_i. T_i^L is ordered according to the scores, that the edges achieved in the factor model described in Section 5.2.2. Every toplist was generated by a C++ recommender framework, that was developed by the Data Mining and Search Group of the Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI) [23]. There are three tunable parameters for the online SGD model, that we use. The *learning rate* (l_rate), the *negative sample rate* (n_rate), and the *regularization rate* (r_rate), that were both mentioned in Section 5.2.2. In our experiment, we do proper parameter tuning for each variable.

First, we execute some experiments to find the best learning rate, that turns out to be different for several datasets. Figure 7.7 presents some of the l_rate-s, that we experiment with. Here, the number of predicted edges are set $L = 10000$ for all movement data. We find that choosing the appropriate learning rate can indeed improve the quality of the factor model. For these experiments, $n\_rate = 100$ was fixed as well.

After finding the best learning rates for the datasets, we try to improve the link predictions by changing the the number of generated negative edges for each ui real edge. Negative sample generation was properly described in Section 5.2.2. Figure 7.8 shows some of the n_rate-s, that was used in the experiments. The experiments show that without negative samples the performance of the model decreases abruptly. Especially, it can be observed in Figure 7.8d.

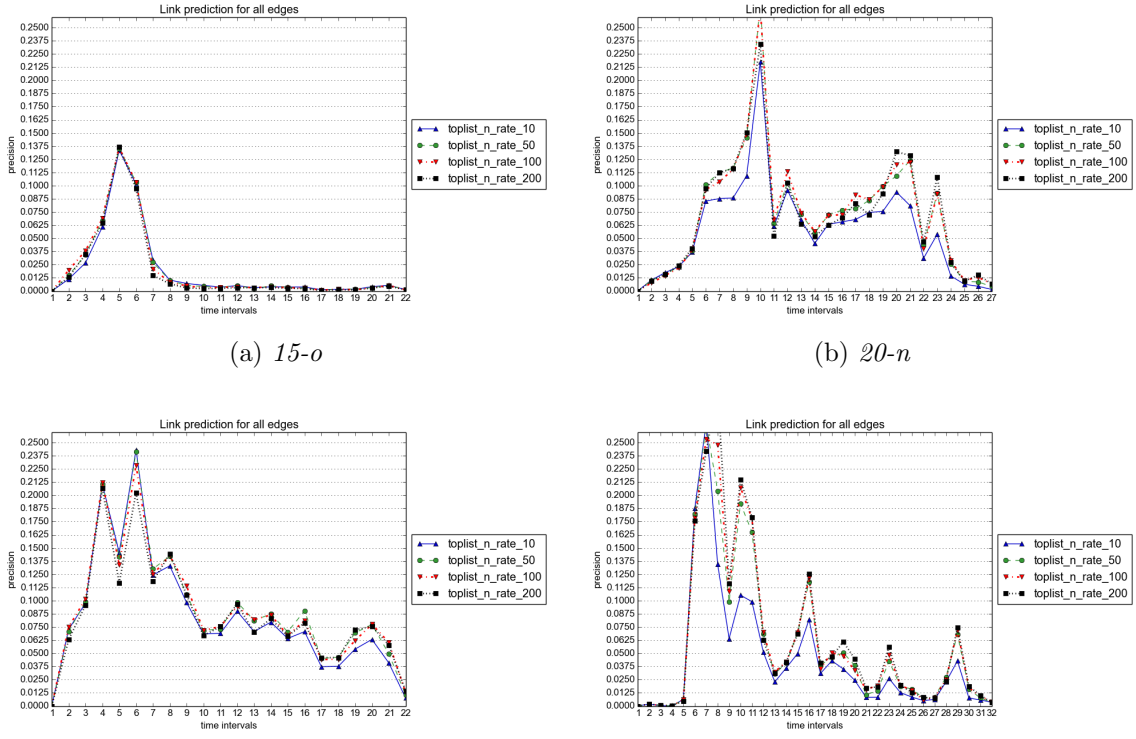
Finally, we examine whether regularization can improve the factor model. Figure 7.9



(a) *15-o* (b) *20-n*

(c) *occupy* (d) *yosoy*

Figure 7.7: Precision of the online SGD link predictions with $l\_rate \in \{0.05, 0.08, 0.1, 0.2\}$.



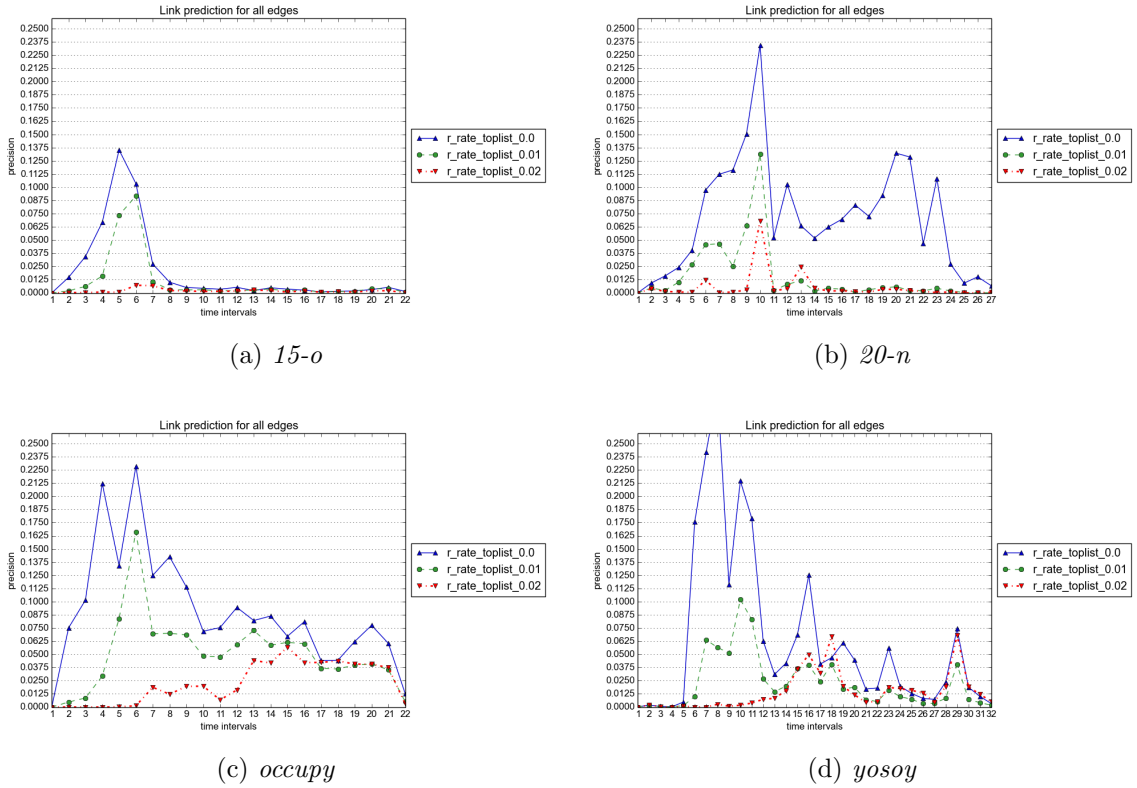
(a) *15-o* (b) *20-n*

(c) *occupy* (d) *yosoy*

Figure 7.8: Precision of the online SGD link predictions with $n\_rate \in \{10, 50, 100, 200\}$. Here we used the best l_rate for each dataset.

shows that the best parameter was $r\_rate = 0.0$ for all datasets. In other words, regularization only decreases the performance of the model. Moreover, the precision for link

prediction drops significantly even for a slight change in the regularization rate. Similarly to the former experiments, we predict $L = 10000$ edges in each time interval.



(a) *15-o* (b) *20-n*

(c) *occupy* (d) *yosoy*

Figure 7.9: Precision of the online SGD link predictions with $r\_rate \in \{0.0, 0.01, 0.02\}$. Here we used the best l_rate and n_rate from former experiments.

In this section, we were able to improve the link prediction with proper parameter tuning for online SGD. Choosing the appropriate learning rate resulted in the most significant gain in precision.

7.4 Centrality prediction with online SGD

In this section, we present our results that we achieved for centrality prediction on dynamic mention graphs. These networks were constructed from the datasets described in Chapter 6. In the former section, we were able to determine the best parameters for online stochastic gradient descent. Now, we compute indegree, outdegree, negative β-measure, PageRank and SALSA centrality measures on each inferred $\hat{G}_i$ interval subgraphs. Then, we extract the top $k = 10$ central nodes to $\hat{L}_i^M(10)$ for each time frame i. M denotes our centrality prediction model that is based on link prediction with online SGD.

Before presenting our results, we show that each centrality measure can be interpreted as a model of importance for Twitter users.

1. **Indegree**: In this model, the users who are mentioned by many other users are important, e.g. mass media sources or celebrities.

2. **Outdegree**: Here, users who mention many other users are relevant. Although, this aspect can be deceptive, as spammer users could be identified as important members of the network.

3. **Negative β-measure**: The former mentions of the users are known. In this model, we suppose that every user u chooses user v uniformly from the nodes he formerly mentioned. Then a $u \to v$ mention occurs. The important users have high expected number of mentions.

4. **PageRank**: Initially, all user get the same importance score. In each step a user uniformly distributes his score among the users he formerly mentioned. These iterations are repeated until the process converges. Finally, users with high importance scores are relevant.

5. **SALSA**: Here, every user has an authority score and a hub score (Section 1.2.6). The hub score reflects whether a given user u usually mentions relevant users. The authority score represents whether u was mentioned by users with high hub scores. After computing these variables with an iterative process, central users are presumed to have high authority scores. Although, the salsa hub score cannot be corrupted as easily as outdegree, but important users are not identified according to their hubbiness.

In Figures 7.10, 7.12 the experiments show that our model is capable of identifying central nodes for indegree, negative β-measure, PageRank and SALSA authority score. On the contrary, the predicted central nodes for outdegree and SALSA hub score are completely indifferent. Nevertheless, it may not be a problem as important users are usually not defined according to their outdegree and hub score in dynamic mention graphs.

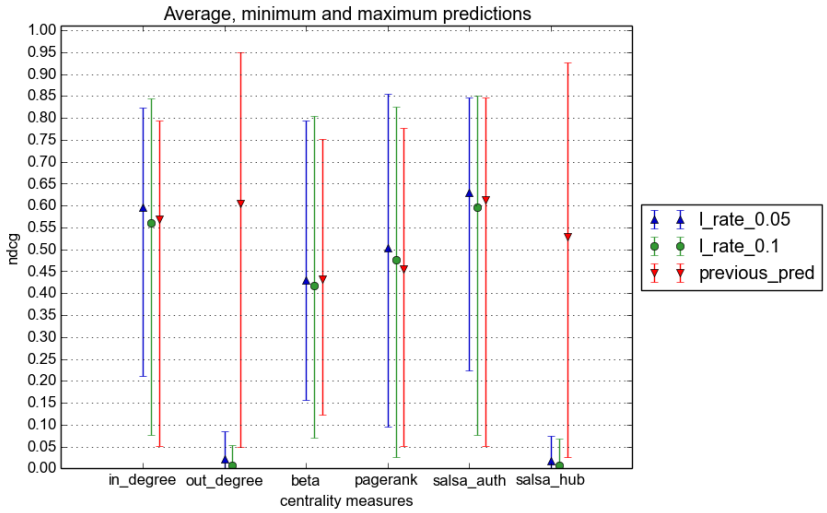
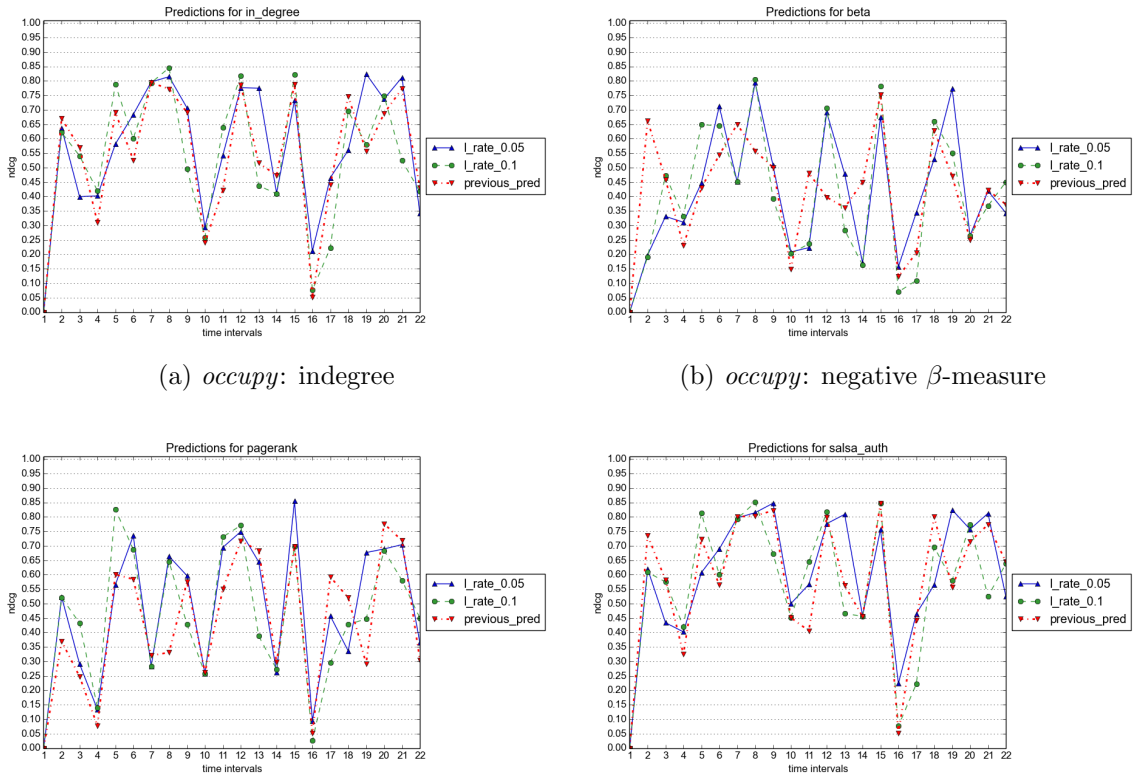


Figure 7.10: Average, minimum and maximum NDCG@10 for the days of *occupy* with different learning rates. Here, *previous_pred* denotes the result of the baseline model discussed in Section 7.2.

In Section 7.3, we were able to improve our link prediction model by finding the appropriate learning rate. Figure 7.10 show that the centrality predictions were indeed better for the best *l_rate* in case of *occupy*. The results are presented in Figure 7.11 for each time interval related to the formerly mentioned centrality indices. Similar increases can be observed in NDCG for the other datasets as well.

Moreover, we tried to improve the performance of our factor model based centrality prediction with changing the number of inferred edges L. Although, in Figure 7.12c we find that for *occupy* predicting more links cannot improve the model significantly, while



(a) *occupy*: indegree

(b) *occupy*: negative β-measure

(c) *occupy*: PageRank

(d) *occupy*: SALSA authority score

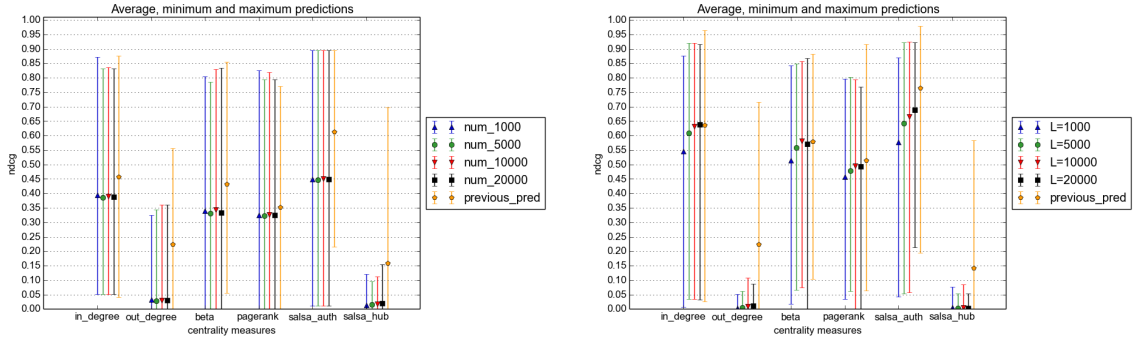
Figure 7.11: Centrality prediction for different centrality measures related to *occupy*. Here, $\Delta t = 1$ day and *previous_pred* denotes the result of the baseline model discussed in Section 7.2. The changes in the learning rate ($l\_rate \in \{0.05, 0.1\}$) improves the centrality predictions as well.

for *20-n* and *yosoy* it turned out to be a good idea. Furthermore, the results show that for *occupy* the performance with $L = 5000$ is nearly as good as with $L = 20000$ predicted edges. It is good to know that our model is capable of extracting central nodes from 5000 predicted edges compared to the average number of edges, that is approximately 60000 for this movement data (Figure 6.1). Unfortunately for *20-n*, even with this improvement we cannot achieve the performance of the baseline model described in Section 7.2. These results are presented in Figure 7.12b.

	15-o	*occupy*	*yosoy*	*20-n*
l_rate	0.08	0.05	0.05	0.2
n_rate	50	100	200	200
r_rate	0.0	0.0	0.0	0.0
L	10000	10000	20000	20000

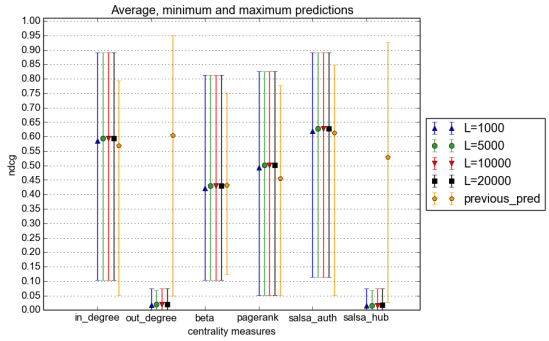
Table 7.1: Best configurations for each dataset.

Finally, Table 7.1 contains the optimal parameters for our model related to each dataset. These are the results of the proper parameter tuning that we carried out for the learning rate, the negative sample rate, the regularization rate and the number of predicted edges. It is a common observation that regularization should not be used for our case because it results in poor predictions. On the contrary, an appropriate learning rate can indeed improve centrality predictions for all datasets. After setting the best *l_rate*, we find that negative samples are crucial for our model. For small *n_rate* values

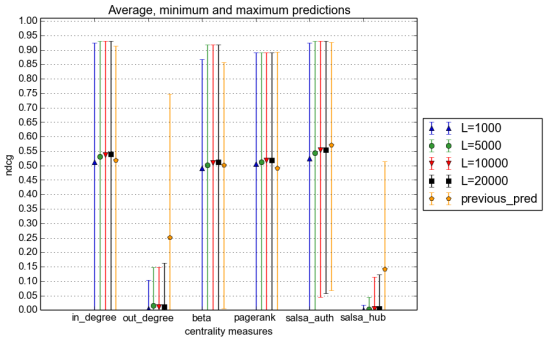


(a) *15-o*

(b) *20-n*



(c) *occupy*



(d) *yosoy*

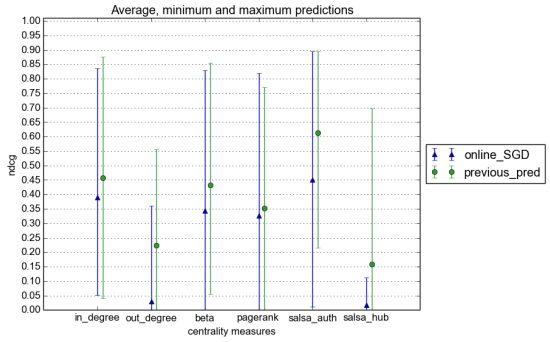
Figure 7.12: Centrality predictions with different number of predicted edges. Here, *previous_pred* denotes the result of the baseline model discussed in Section 7.2.

the performance of our online SGD based link prediction model decreases. Altering the number of predicted edges also proves to be effective for some datasets.

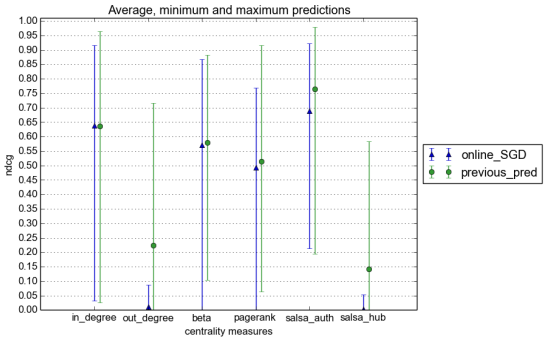
	15-o	*occupy*	*yosoy*	*20-n*
Indegree	-	2,5%	2%	-
Negative β-measure	-	-	1%	-
PageRank	-	5%	3%	-
SALSA authority	-	1,5%	-	-

Table 7.2: In several cases our online SGD based model can improve the baseline prediction. The gain in $NDCG$ is presented for the configurations mentioned in Table 7.1.

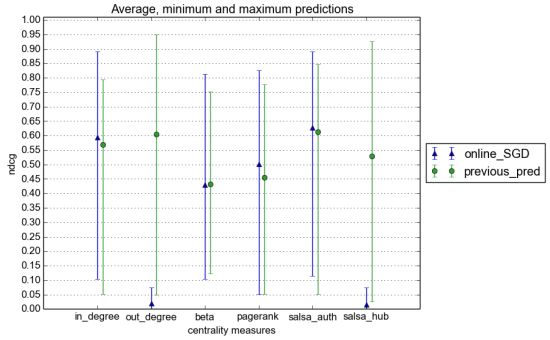
Altogether, we could improve the baseline prediction for some centrality measures. The results are presented in Table 7.2. Our model has best performance on the *occupy* movement data. For PageRank there is a 5% gain in the average NDCG of time intervals. Moreover, we could also improve the prediction for indegree and SALSA authority score as well. The online SGD based model also improved the prediction for PageRank with 3% in case of *yosoy*. For this dataset, there was a 2% gain for indegree as well. For *20-n* the baseline model is surprisingly strong. However, the baseline proves to be better for all measures, but our model can also identify central users as for indegree, negative β-measure and PageRank the results are only a bit worse. Our centrality prediction model has poor performance on *15-o*. It is important to note that *15-o* has significantly lower average edge count compared to the other movement datasets (Figure 6.3). It is likely that the lack of edges causes the poor performance of our model for this movement data.



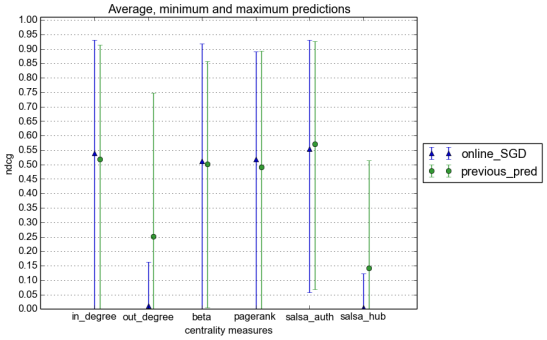
(a) *15-o*



(b) *20-n*



(c) *occupy*



(d) *yosoy*

Figure 7.13: Centrality predictions where the parameters are set according to Table 7.1 for each datasets. These are the best results compared to the baseline model.

Summary

In this thesis, our goal was to propose a model that can identify central users of dynamic graphs in advance. We focused on Twitter mention graphs that were constructed from datasets related to global movements from 2011 and 2012. In these graphs the nodes represents Twitter users. Our experiments showed that these graphs are highly dynamic as the fraction of the new incoming users is high for each time interval. However, we realized that homophily edges can be the key to effective centrality predictions.

Our model first solves the link prediction problem for these temporal graphs with matrix factorization. For this task, we used online stochastic gradient descent. Using negative samples, this factor model could recover $6-7\%$ of the edges in average for most of the datasets. From the edges that our model predicted for interval i we constructed a directed graph $\hat{G}_i$ that tries to approximate the original subgraph G_i of the related time frame. After we computed indegree, outdegree, negative β-measure, PageRank and SALSA centrality measures for $\hat{G}_i$, we gave centrality predictions for interval i. For a given measure it is the top k node with highest centrality score in $\hat{G}_i$.

For the proper evaluation of our model we introduced a baseline model. It predicts the top k central users of G_{i-1} for interval i. The results show that our model could improve the baseline centrality prediction for indegree, negative β-measure, PageRank and SALSA authority score for some datasets. Our online stochastic gradient descent based model performed the best for PageRank. For this measure it achieved a 5% gain in the average NDCG of time intervals compared to the baseline model.

In our experiments, we tried to improve link prediction with regularization, negative sample generation and changing the number of predicted edges. There are other techniques to improve the quality of the underlying factor model which were not tested in this thesis. One possibility is to introduce biases into the model. It may improve the performance of online SGD. Altogether, we found that our model can identify central vertices of Twitter mention graphs related to social events. Therefore, it can be useful for predicting key users in global movements.

Bibliography

[1] Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 65–74. ACM, 2011.

[2] Abraham Berman and Robert J Plemmons. Nonnegative matrices. *The Mathematical Sciences, Classics in Applied Mathematics*, 9, 1979.

[3] Paolo Boldi, Marco Rosa, and Sebastiano Vigna. *Robustness of social networks: Comparative results based on distance distributions*. Springer, 2011.

[4] Paolo Boldi and Sebastiano Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014.

[5] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[6] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[7] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630. ACM, 2009.

[8] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. Can cascades be predicted? pages 925–936, 2014.

[9] Flavio Chierichetti, Jon Kleinberg, Ravi Kumar, Mohammad Mahdian, and Sandeep Pandey. Event detection via communication pattern analysis. In *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[10] Nick Craswell. Gov2 test collection. Online; accessed: 2015-04-11; `http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm`.

[11] Charles H Hubbell. An input-output approach to clique identification. *Sociometry*, pages 377–399, 1965.

[12] Bernardo A Huberman, Daniel M Romero, and Fang Wu. Social networks that matter: Twitter under the microscope. *Available at SSRN 1313405*, 2008.

[13] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[14] Edward H Kaplan. What are the risks of risky sex? modeling the aids epidemic. *Operations Research*, 37(2):198–209, 1989.

[15] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[17] Ronny Lempel and Shlomo Moran. The stochastic approach for link-structure analysis (salsa) and the tkc effect. *Computer Networks*, 33(1):387–401, 2000.

[18] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[19] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[20] Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)*, 27(1):2, 2008.

[21] Seth A Myers and Jure Leskovec. The bursty dynamics of the twitter information network. In *Proceedings of the 23rd international conference on World wide web*, pages 913–924. International World Wide Web Conferences Steering Committee, 2014.

[22] Egerváry Research Group on Combinatorial Optimization. Lemon graph library. Online; accessed 2015-05-28;

`http://lemon.cs.elte.hu/trac/lemon`.

[23] Róbert Pálovics, András A Benczúr, Levente Kocsis, Tamás Kiss, and Erzsébet Frigó. Exploiting temporal influence in online recommendation. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 273–280. ACM, 2014.

[24] Manuel Gomez Rodriguez, Jure Leskovec, David Balduzzi, and Bernhard SchÖlkopf. Uncovering the structure and temporal dynamics of information propagation. *Network Science*, 2(01):26–65, 2014.

[25] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al. *Introduction to data mining*, volume 1. Pearson Addison Wesley Boston, 2006.

[26] Sebastiano Vigna. Spectral ranking. *arXiv preprint arXiv:0912.0238*, 2009.

[27] Jacco Wallinga and Peter Teunis. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *American Journal of Epidemiology*, 160(6):509–516, 2004.

[28] Wikipedia. 20-n. Online; accessed 2015-05-28;

`http://en.wikipedia.org/wiki/15_October_2011_global_protests`.

[29] Wikipedia. 20-n. Online; accessed 2015-05-09;

`http://en.wikipedia.org/wiki/20-N`.

[30] Wikipedia. Occupy wall street. Online; accessed 2015-05-09;

`http://en.wikipedia.org/wiki/Occupy_Wall_Street`.

[31] Wikipedia. Yo soy 132. Online; accessed 2015-05-09;

`http://en.wikipedia.org/wiki/Yo_Soy_132`.

[32] Eva Zangerle, Wolfgang Gassler, and Günther Specht. Recommending#-tags in twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings*, volume 730, pages 67–78, 2011.

Appendix A

Experimental C++ framework

Overview

Our framework is based on the LEMON open source project [22]. LEMON is a C++ library that aims to provide efficient algorithms and data structures related to graphs. I implemented additional graph objects for computing centrality measures. These objects incorporate iterative methods for computing PageRank and SALSA. These features are not part of the original LEMON source code.

Beside providing objects for centrality computation, our framework supports the temporal analysis of dynamic graphs. The start date, the number and the length of the time intervals can be set as parameters for the experiments. The centrality results are written to JSON output files which can be parsed easily.

Usage

The graph objects and the runner environment have several parameters. Thus, due to space constraints the documentation of the framework is presented in my online GitHub repository (`https://github.com/ferencberes/msc-thesis`). The source code is accessible through Git. To download the repository use the following command.

```
git clone https://github.com/ferencberes/msc-thesis.git
```

About dependencies

The framework is developed under Unix environment. There are scripts for installing most of the dependencies. The details can be found in the formerly mentioned online GitHub repository.