

longFancyHeader
longtabuFancyHeader

Link Prediction In Directed Collaboration Networks

Masters Thesis
University of Montana

William Lyon

September 30, 2014

This page is probably the biographical bs stuff

William Lyon

Link Prediction In Directed Collaboration Networks

Keywords keywords here, probably remove this section

Abstract

William Lyon

Abstract here. 350 words I guess.

Key words: <insert key words>

Declaration

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been acknowledged.

<insert data and location>

<insert full name>

Contents

1	Introduction	1
1.1	Online Social Collaboration Networks	1
1.2	This project	1
2	Previous Work	3
2.1	Recommendation As Link Prediction	3
2.2	User-User Recommender Systems	3
2.2.1	Similarity Based Methods	4
2.2.2	Network Based Methods	4
2.3	Directed Social Networks	4
2.4	My Approach	4
3	Methods	7
3.1	Algorithms	7
3.1.1	Sample Graph	7
3.1.2	Collaborative Filtering	8
	Similarity Metrics	8
3.1.3	Triadic Closeness	8
3.2	Implementation	10
3.2.1	Graph Data Model	10
3.2.2	Graph Traversal Pattern	10
3.2.3	Graph Database And Querying Traversals	10
3.2.4	11
3.2.5	Steps	11
3.3	The new shit	11
3.4	Data	12
3.4.1	Data munging	12
4	Evaluation	15
4.0.2	Evaluation	15
4.1	Discussion	16

5	Summery and Outlook	17
	Appendix	25
A	First chapter of appendix	25
A.1	Parameters	25

CHAPTER 1

Introduction

Social networks are everywhere. Facebook, Twitter, LinkedIn, etc...

1.1 Online Social Collaboration Networks

Social networks based on our activity rather than who we know. Consider GitHub: Github is a software collaboration web application based on the git version control system. Using GitHub software developers can collaborate on projects, share software projects and follow what other developers are working on. GitHub can be thought of as a social collaboration network, a type of network that has both collaboration and social interactions between users.

1.2 This project

The goal of this thesis is to explore the link prediction problem, as applied to online social collaboration networks. Much of the previous literature in this area focuses on homogenous (single relationship) undirected social networks. We extend this research to focus on heterogenous (multiple relationship type) directed social collaboration networks, specifically the GitHub network. We duplicate the work done previously in this area and develop a novel approach to link prediction.

The remainder of this paper is outlined as follows:

Chapter 2 - Previous Work A review of the literature in this field

Chapter 3 - Methods An examination of the data used for this project as well as an in-depth explanation of the algorithms used for link prediction, in the context of the graph traversal pattern, which is also explained in this chapter. A novel approach for link prediction is proposed, a combined similarity and network structure method. Implementation details involving graph data modeling and graph databases are discussed.

Chapter 4 - Evaluation This new recommender system is evaluated relative to similarity based methods and network structure methods.

Chapter 5 - Summary Areas of further research are discussed.

Figure ?? shows the GitHub network modeled as a property graph [Rodriguez]. This project is concerned with only the *FOLLOWS* relationship: GitHub users can follow other users to express interest in another user and receive updates about that user's actions in the GitHub system. A user recommendation system is implemented which generates recommendations of other users to follow on GitHub. Since user recommendations are a prediction of future action, this can be thought of as a link prediction problem: given a vertex in the graph, can we predict which edges will be added in the future? Specifically, we are predicting *FOLLOWS* edges for specific users. A user-user collaborative filtering algorithm is implemented using the graph traversal pattern [Rodriguez] and evaluated using a leave-one-out method.

CHAPTER 2

Previous Work

Recommender system literature can typically be divided into two categories: 1) content (or item) recommendation and 2) user recommendation. An example of content recommender systems is Amazon product recommendations. These take the form of "users who bought x also bought y". An example of user recommendation is LinkedIn's "Do You Know..." feature, which is suggestingThe latter can be generalized as "Who To Follow" recommender systems and it is this type of recommendation on which this paper will focus.

2.1 Recommendation As Link Prediction

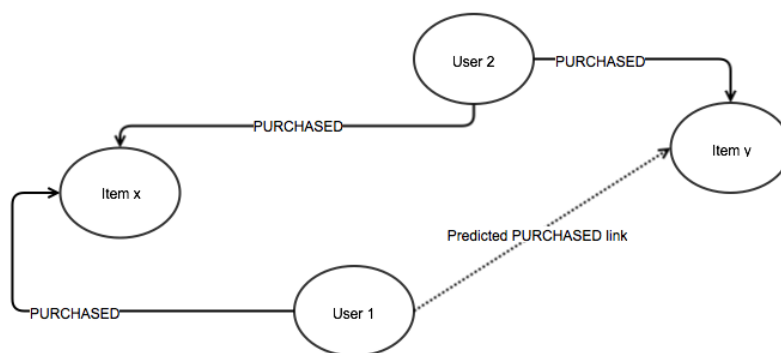


Figure 2.1: Here we see an example of an item recommender system modeled as a graph where the nodes are Users and Items and the edges indicate a purchase of an Item by a User. When modeled this way recommendation takes the form a predicted link in the graph.

2.2 User-User Recommender Systems

There are essentially two types of User-User recommender system algorithms: 1) Similarity based methods and 2) Network based methods.(CITATION NEEDED) Similarity based methods rely on the computation of a user-user similarity metric, which is then

used to make recommendations. This is based on the homophily principle, that users are more likely to be interested in users similar to them. Network based methods analyze the structure of the network to develop recommendations. Examples include PageRank, HITS, and SALSA (CITATION NEEDED).

2.2.1 Similarity Based Methods

Collaborative filtering systems produce user specific recommendations based on patterns of behavior observed from other users. Typically this involves observing ratings of items and using either latent factor models or a neighborhood based approach to generate item recommendations[cf]. With the rise of social network analysis however, often instead of user-item recommendations, we are more interested in generating user-user recommendations. User-user recommendations are the focus of this project. The underlying assumption of collaborative filtering is that of homophily: similar users like similar things. Collaborative filtering implementations can be problematic when applied to a large dataset. Most methods require a large sparse matrix for computation, the use of which is not always performant. Instead, the problem can be modeled as a graph, and make use of the graph traversal pattern as an alternative to the construction of a large sparse matrix [Rodriguez].

2.2.2 Network Based Methods

User-User recommender systems are often generalized as "Who To Follow" recommender systems, as made popular by the "Follow" social network action in the Twitter social network. A Follow action indicates a user expressing interest in another user in the network. This often involves

The Twitter Who To Follow system is an example of a link prediction system running in production. [wtf] details the design and implementation considerations for such a large scale system. The Twitter system operates on a custom in-memory graph processing engine which implements a PageRank-type algorithm known as Stochastic Approach for Link-Structure Analysis (SALSA). SALSA uses a random walk of the graph to generate link predictions.

2.3 Directed Social Networks

There is an important distinction to note between undirected social networks and directed social networks. Examples/explanation. Much of the literature has focused on undirected social networks only (CITATION NEEDED). In fact the similarity metrics shown above are all based on undirected social networks. A notable exception in the Triadic Closeness method described by (CITATION NEEDED).

2.4 My Approach

There is a clear gap in the literature making use of combined network based methods with similarity based methods. (CITATION Ensemble paper) My contribution to this field is to explore how these methods can be combined to improve the accuracy of such

recommendations. A linear combination of a collaborative filtering similarity based approach and a network based approach leveraging the use of directed networks is proposed, based somewhat on the work developed in (CITATION paper about novel similarity metric, weighted averaging).

CHAPTER 3

Methods

Here we describe the data used and detail the implementation. The network data examined is that of the GitHub network. GitHub is an online social collaboration network built around the git version control system. (CITATION NEEDED). GitHub allows software developers to collaborate on projects, share software projects, and follow other users. It is this "Follows" social action in which we are most interested.

3.1 Algorithms

3.1.1 Sample Graph

Draw a sample graph. Based on this graph we will examine specifically how to calculate CF using jaccard (and maybe some other similarity metrics from the table in Schall)

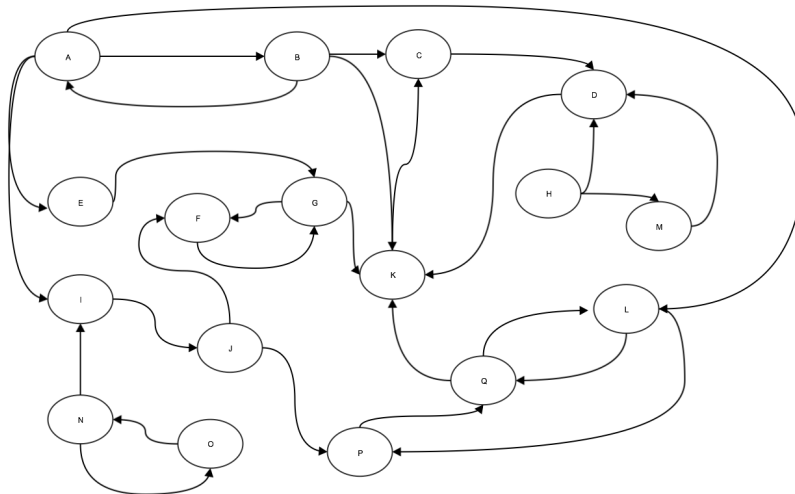


Figure 3.1: Sample social network of hypothetical

3.1.2 Collaborative Filtering

Collaborative filtering is a method of generating recommendations based on the homophily principle: users who are similar are likely to be interested in similar items. It is implemented by finding similar users, allowing each similar user to "vote" for recommendations and suggested items with the highest score. This is similar to the kNN algorithm used for classification.[**cf**]

Similarity Metrics

The Jaccard index is used to identify similar users. For two users, a and b , let A and B denote the sets of all users being followed by a and b , respectively. The Jaccard index is therefore as defined in Equation 3.1.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (3.1)$$

In this context, Jaccard is defined as the intersection of the users followed by a and b divided by the union of the users followed by a and b . This results in a number between 0 and 1, indicating the strength of similarity between users a and b .

3.1.3 Triadic Closeness

Graph theory proposes the concept of triadic closure, the hypothesis that the creation of an edge between u and v is related to the degree of overlapping neighbors in u and v 's respective networks. (CITATION NEEDED and expand on this - see one of the network analysis books I have) The concept of Triadic Closeness is an application of the theory of triadic closure, specifically taking into account the directed nature of social networks. For a given fully observed network, Triadic Closeness can be thought of as the ratio of the number of closed triads to the number of potentially closed triads. (CITATION NEEDED Schall 2013).

INSERT IMAGE OF A TRIAD

In a directed network there are 27 distinct configurations, or patterns that a triad can take on. See (CITATION TABLE of triad patterns). Table XXXXX (CITE ME) shows triad patterns that are open, that is no connection exists between nodes u and v . The pattern identifications ($T01, T02...$) are taken from (CITE PROPERLY) Schall 2013. Any open triad can be closed in one of three possible ways: $u \leftarrow v$, $u \rightarrow v$, or $u \leftrightarrow v$.

INSERT IMAGE OF 3 CLOSED TRIAD PATTERNS

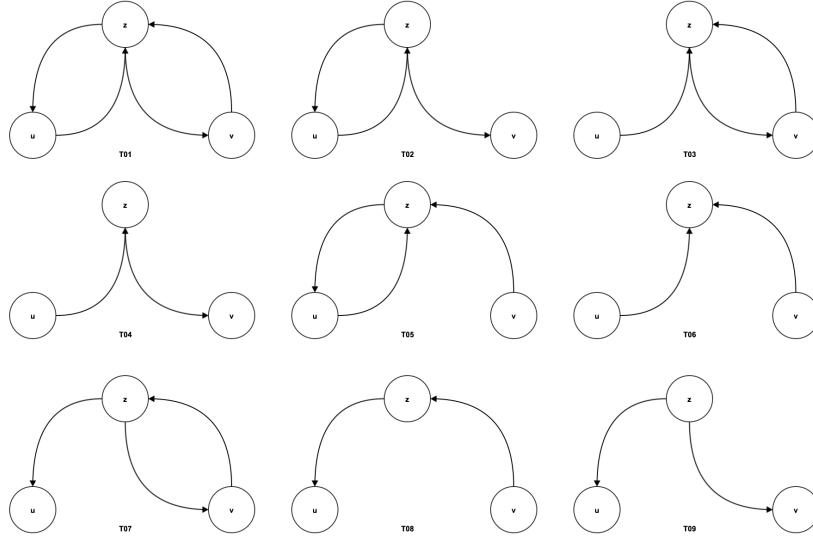


Figure 3.2: All possible open triad patterns are shown. A triad is said to be closed if a connection is established between u and v .

$$TC_{uv} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} w^P(u, v, z) \times w(z) \quad (3.2)$$

$$w^P(u, v, z) = \frac{F(T(u, v, z) + 10) + F(T(u, v, z) + 30)}{F(T(u, v, z))} \quad (3.3)$$

$$TC_{uv} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} w^P(u, v, z) \times \frac{1}{k_z} \quad (3.4)$$

Table 3.1: Open triad pattern frequency in the sample network shown in Figure CITATION NEEDED

ID	PATTERN	COUNT
T06	$u \rightarrow z \leftarrow v$	20
T04	$u \rightarrow z \rightarrow v$	14
T08	$u \leftarrow z \leftarrow v$	14
T02	$u \leftrightarrow z \rightarrow v$	8
T07	$u \leftarrow z \leftrightarrow v$	8
T09	$u \leftarrow z \rightarrow v$	8
T03	$u \rightarrow z \leftrightarrow v$	3
T05	$u \leftrightarrow z \leftarrow v$	3

Table 3.2: Closed triad pattern frequency in the sample network shown in Figure CITATION NEEDED

ID	PATTERN	COUNT
T18	$u \leftarrow z \leftarrow v \leftarrow u$	3
T14	$u \rightarrow z \rightarrow v \leftarrow u$	2
T16	$u \rightarrow z \leftarrow v \leftarrow u$	2
T19	$u \leftarrow z \rightarrow v \leftarrow u$	2
T15	$u \leftrightarrow z \leftarrow v \leftarrow u$	1
T17	$u \leftarrow z \leftrightarrow v \leftarrow u$	1
T24	$u \rightarrow z \rightarrow v \rightarrow u$	3
T26	$u \rightarrow z \leftarrow v \rightarrow u$	2
T28	$u \leftarrow z \leftarrow v \rightarrow u$	2
T29	$u \leftarrow z \rightarrow v \rightarrow u$	2
T22	$u \leftrightarrow z \rightarrow v \rightarrow u$	1
T23	$u \rightarrow z \leftrightarrow v \rightarrow u$	1
T34	$u \rightarrow z \rightarrow v \leftrightarrow u$	1
T38	$u \leftarrow z \leftarrow v \leftrightarrow u$	1

3.2 Implementation

3.2.1 Graph Data Model

The labeled property graph

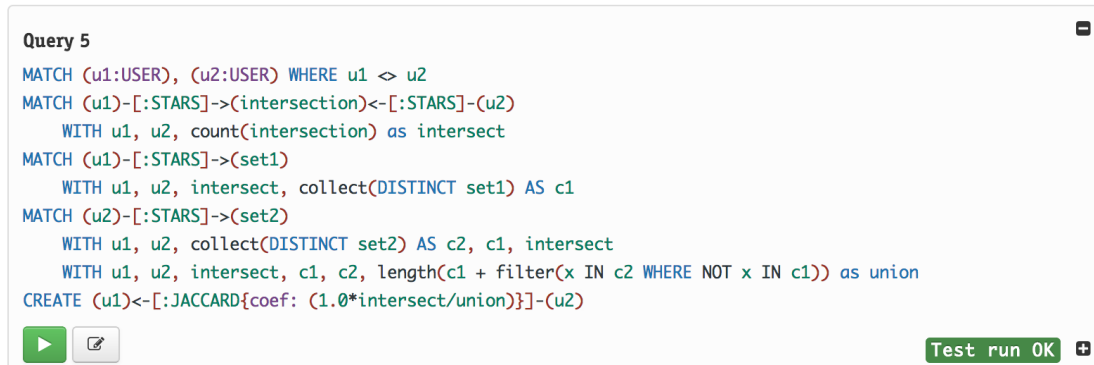
3.2.2 Graph Traversal Pattern

What is the graph traversal pattern? The graph traversal pattern allows us to design algorithms where the answer to our question is a traversal through the graph.

3.2.3 Graph Database And Querying Traversals

Why using a graph database makes sense. Performance benefits of using graph database. Compare to how this would be modeled in a RDBMS or a document DB.

3.2.4



```

Query 5
MATCH (u1:USER), (u2:USER) WHERE u1 <> u2
MATCH (u1)-[:STARS]->(intersection)-[:STARS]-(u2)
  WITH u1, u2, count(intersection) as intersect
MATCH (u1)-[:STARS]->(set1)
  WITH u1, u2, intersect, collect(DISTINCT set1) AS c1
MATCH (u2)-[:STARS]->(set2)
  WITH u1, u2, collect(DISTINCT set2) AS c2, c1, intersect
  WITH u1, u2, intersect, c1, c2, length(c1 + filter(x IN c2 WHERE NOT x IN c1)) as union
CREATE (u1)-[:JACCARD{coef: (1.0*intersect/union)}]-(u2)

```

Test run OK

Figure 3.3: Example of Neo4j Cypher query language used to calculate Jaccard metric for pairs of users.

The system is developed using Java and makes use of the Neo4j graph database. The Cypher query language is used to query the graph database and for defining graph traversals for processing. The program is configurable and implements k-fold cross validation.

3.2.5 Steps

The following describes implementation of the system at a high level. For each validation fold:

- Select p users at random
- For each user u in p :
 - Remove one *FOLLOWS* edge f for this user at random
 - Find k nearest neighbors using Jaccard index (ensure removed edge is not used for this calculation)
 - Calculate n followed users with greatest overlap (the most commonly followed users among the k nearest neighbors)
 - Return set of n users, these are the recommended users
 - Is f in n ? If yes, this run is counted as a valid prediction.
- Report summary accuracy metrics for this fold

3.3 The new shit

Linear combination of similarity metric (Jaccard) and network based method (triadic closeness). Why might this be beneficial?

3.4 Data

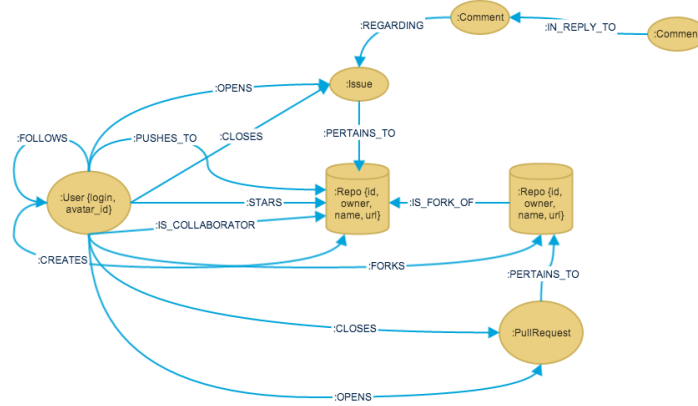


Figure 3.4: GitHub data model as a labeled property graph.

Data was collected from GitHub Archive[[githubarchive](#)], a service that maintains an archive of all public events emitted by the GitHub API[[github:Online](#)]. These include events such as creation of new repositories, pushes to repositories, repository stars, and user follows. Data was collected for the time period April 1st, 2013 - April 1st, 2014. For the purposes of this project, only *FOLLOWS* events were considered. This resulted in a total of 539893 users and 919489 follows. In terms of a graph, that equates to 539893 nodes and 919489 vertices between nodes.



Figure 3.5: The GitHub follow graph is a simple graph with User nodes and Follows edges.

A graph data model is used to represent this data as the data is highly connected: it is describing entities (users and repositories) and their interactions (stars, follows, pushes, etc). Figure 3.6 shows an example of a subgraph of user and repository nodes and the interactions among those entities, modeled as a graph.

3.4.1 Data munging

The data from GitHubArchive is available in streaming JSON format and includes all public events generated from the GitHub event API [[github:Online](#)] for a given time period. The one year of data collected for this project resulted in several thousand JSON files, each several megabytes in size resulting in a raw dataset of several hundred gigabytes. As this project is only concerned with the user-user *FOLLOWS* relationship, the data is parsed using a Python script to filter for only those *FOLLOWS* events. For

further efficiency, the data is reduced to an anonymized edgelist format:

```
1 2
3 4
5 6
7 8
9 10
11 12
11 13
14 15
16 17
18 19
...
```

This allows for the dataset to be anonymized using only arbitrary user ids and stored as compactly as possible. In the above example (and in the context of a directed graph) the first column corresponds to the source node and the second column the destination node. So the sample above represents: user 1 follows user 2, user 3 follows user 4, ...

This edgelist file is then used to populate a Neo4j graph database instance, using only the minimal information necessary to implement our link prediction system.

Table 3.3: Dataset descriptive statistics

	NODES (USERS)	539893
EDGES (:FOLLOWS RELATIONSHIPS)		919489
	MEAN DEGREE	3.4
STANDARD DEVIATION OF DEGREE		30.3

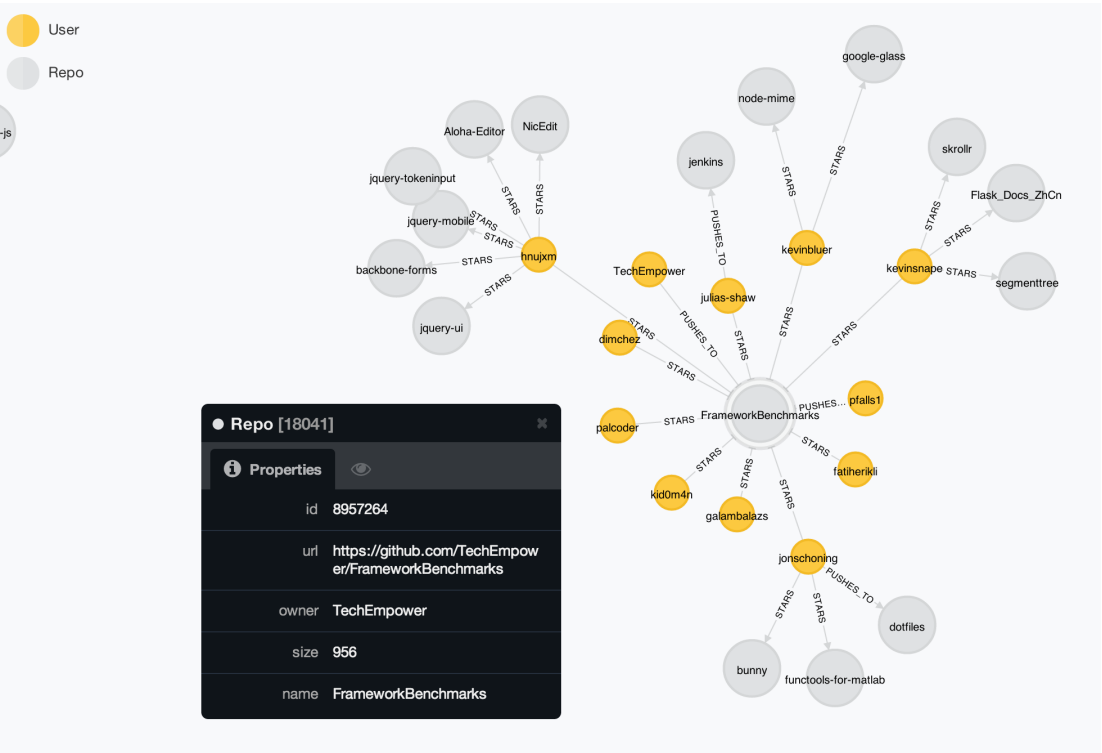


Figure 3.6: GitHub data model as a property graph. Screenshot from Neo4j graph database interface.

CHAPTER 4

Evaluation

The result of the algorithm is a set of user ids that are predicted destination nodes, given a specific source node. These are recommended users that the specified user might be interested in following.

4.0.2 Evaluation

The system is evaluated using several iterations for cross-validation with different configurations. Table 4.1 shows the results. Two methods of evaluation are identified as the standard for evaluating link prediction systems: precision and area under the receiver operating characteristic curve (AUC). Precision is defined as the ratio of the relevant items selected to the total number of items selected. AUC is a more involved calculation that requires an all-pairs comparison of the rank of the predicted link and *all* other possible unobserved links in the network. Due to the size of the GitHub network, calculating this metric was not seen as feasible. Instead of AUC, a simple accuracy metric is calculated: if the removed link is present in the array of predicted links we count the test run as correct, if not we consider it incorrect. Accuracy is therefore the number of correct predictions divided by the total number of runs. Due to the large number of possible links we expect these accuracy metrics to be quite low, relative to other recommender system results.

Table 4.1: Evaluation metrics for cross fold validation runs with varying parameters. K is number of neighbors allowed to vote. N is number of predicted links. Precision and accuracy are evaluation metrics, expressed in percentages. Random is the probability of at least one of the N links being relevant if N links are chosen at random.

K	N	PRECISION (%)	ACCURACY (%)	RANDOM (%)
5	25	0.61	15.0	0.0046
5	50	0.18	8.0	0.0093
5	100	0.19	15.0	0.0185
10	25	0.16	4.0	0.0046
10	50	0.14	7.0	0.0093
10	100	0.19	18.0	0.0185
25	25	0.24	6.0	0.0046
25	50	0.12	6.0	0.0093
25	100	0.11	11.0	0.0185
50	25	0.12	3.0	0.0046
50	50	0.14	7.0	0.0093
50	100	0.10	0.10	0.0185
100	25	0.28	7.0	0.0046
100	50	0.16	8.0	0.0093
100	100	0.05	5.0	0.0185
1000	25	0.32	8.0	0.0046
1000	50	0.24	12.0	0.0093
1000	100	0.14	14.0	0.0185
2000	25	0.24	6.0	0.0046
2000	50	0.18	9.0	0.0093
2000	100	0.10	10.0	0.0185

4.1 Discussion

Table 4.1 details the results of running our algorithm for various configurations of the parameters k and n, where k is the number of neighbors considered to generate recommended links and n is the number of predicted links generated for each user. We calculate both precision and accuracy for each run, where accuracy is defined as the number of users for which the removed link was observed to be in the set of predicted links divided by the total number of users in that run. In other words, we count a prediction as "correct" if v, the removed *FOLLOWS* edge is in the set of predicted edges. Using this same definition of correctness, we calculate the probability of accuracy if the links were chosen at random from our graph of existing GitHub users. We can see that for almost all configurations, the results are significantly (approximately 3000X) better than choosing links at random. Surprisingly, the best performing configurations (as measured by both accuracy and precision) are those with the lowest values of K (the number of neighbors used to generate recommendations).

CHAPTER 5

Summery and Outlook

List of Figures

2.1	Item Link Prediction	3
3.1	Sample network	7
3.2	triad patterns	9
3.3	Example of Neo4j Cypher query language used to calculate Jaccard metric for pairs of users.	11
3.4	GitHub graph data model	12
3.5	User-User data model	12
3.6	GitHub data model as a property graph. Screenshot from Neo4j graph database interface.	14

List of Tables

3.1	Open triad pattern frequency in the sample network shown in Figure CITATION NEEDED	10
3.2	Closed triad pattern frequency in the sample network shown in Figure CITATION NEEDED	10
3.3	Dataset descriptive statistics	14
4.1	Evaluation metrics for cross fold validation runs with varying parameters. K is number of neighbors allowed to vote. N is number of predicted links. Precision and accuracy are evaluation metrics, expressed in percentages. Random is the probability of at least one of the N links being relevant if N links are chosen at random.	16

Listings

APPENDIX A

First chapter of appendix

A.1 Parameters

Acknowledgments

I thank ?? and ?? for giving me the opportunity to write this bachelor/master/phd thesis at ??, and for their professional advise.

I thank in particular the ?? team who readily/willingly provided information at any time and ??.

I would also like to than all people who supported me in writing this thesis.

