An Adaptive Ensemble Method For Link Prediction In Multi-Modal Directed Social Collaboration Networks Using The Graph Traversal Pattern

William Lyon

October 12, 2014

This page is probably the biographical bs stuff

William Lyon

Link Prediction In Directed Collaboration Networks

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

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. [Goo94]

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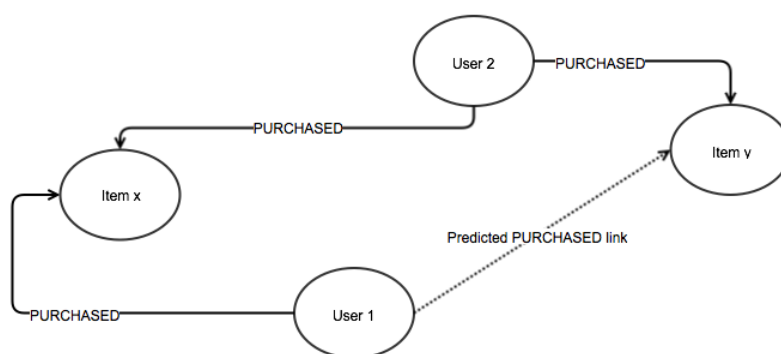
Figure 2.1: Here we see an example of an item recommender system modeled as a graph where the nodes are Users and Items and the edges indicate a purchse of an Item by a User. When modeled this way recommendation takes the form a predicted link in the graph.

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 homopholy 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. Example 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 [Rod].

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 levering 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. We first examine in detail two methods for link prediction: collaborative filtering using the Jaccard similarity metric and the Triadic Closeness method. We then show how these two methods can be used together in an ensemble predictor using adaptive weights. The data used in this experiment is examined. Finally, implementation is discussed.

3.1 Algorithms

3.1.1 Sample Graph

Consider the network shown in figure 3.1. This is a sample network which was randomly generated and does not represent any real world observed data, however we shall refer to this network to demonstrate the techniques used in this project.
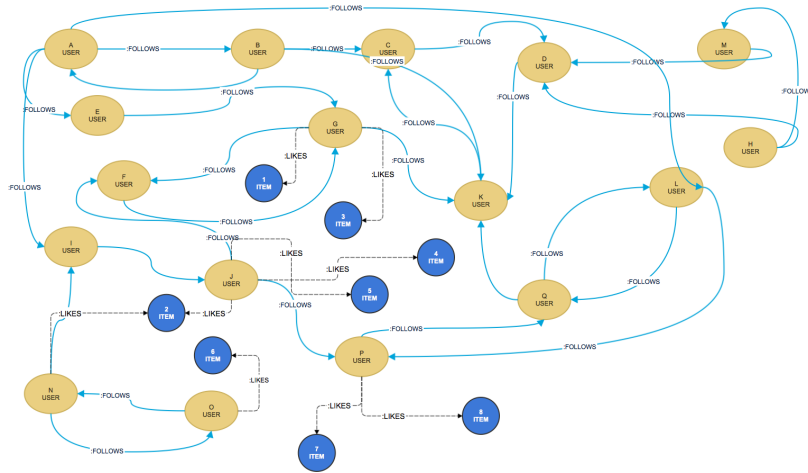

Figure 3.1: Sample social network of hypothetical

We will generate recommendations for a user in the sample network using each of the

Table 3.1: Descriptive statistics for sample network

Metric	value
Num nodes	17
Num edges	28
Avg degree	1.67
OTHERS	??

three methods: 1) collaborative filtering, 2) triadic closeness, and 3) a new approach combining triadic closeness and multi-modal relationship types that has been presented here for the first time.

As mentioned previously, generating recommendations can be thought of as link prediction: we are identifying links that do not appear in the network and predicting links that will either 1) appear in the future or 2) in the case of a partially observed network, links that exist but are not observed.

For the purposes of the next three sections we will consider link prediction for user J. We proceed through each algorithm manually, ignoring some implementation details for now that will explored in depth in the proceeding section.

3.1.2 Collaborative Filtering

TODO: REWRITE THIS PIECE TO MAKE USE OF REPOS / STARS

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**]

Collaborative filtering is usually used in the context of user-item recommendation, so here we simply treat other users as items (as necessary).

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 3.7.

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

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.

To generate recommendations for user J, we first must identify all friend-of-friend nodes, that is nodes that share a neighbor in common with J. That gives us the set

---

Algorithm 1 Collaborative filtering algorithm

1: $input : G(U, E), x, N$ ▷ explain inputs here
2: $usersSample \leftarrow getRandomUsers(U,x)$
3: $results \leftarrow \{\}$
4: **for** $each user in usersSample$ **do**
5:     $validationEdge \leftarrow getRandomEdge(G, user)$
6:     $removeEdge(validationEdge,G)$
7:     fofs $\leftarrow getFOFS(user, G)$
8:     fofRanks $\leftarrow \{\}$
9:     **for** each fof in fofs **do**
10:         j $\leftarrow jaccard(user, fof)$ fofRanks $\leftarrow results + \{j : fof\}$
11:     **end for**
12:     knn $\leftarrow topk(fofRanks, k)$
13:     aggregated $\leftarrow \{\}$
14:     **for** each kn in knn **do**
15:         possible $\leftarrow getFollows(kn)$
16:         **for** each p in possible **do**
17:             **if** p in aggregated.keys **then** $aggregated[p]+ = 1$
18:             **else**
19:                 $aggregated[p] = 1$
20:             **end if**
21:         **end for**
22:     **end for**
23:     $predictions \leftarrow topXSortedByTC(pred, N)$
24:     $hit \leftarrow is validationEdge in predictions?$
25:     $addEdge(validationEdge, G)$
26:     $results \leftarrow results + \{hit, pred, u, v, validation_e dge\}$
27: **end for**
28: **return** results

$\{G, L\}$. Our possible recommendations are now reduced to G, L. We will now compute the Jaccard similarity metric for the pairs (J, G) and (J, L):

$$J(J,G) = \frac{|J \cap G|}{|J \cup G|} \tag{3.2}$$

$$J(J,G) = \frac{|\{F\}|}{|\{P, F, K\}|} \tag{3.3}$$

$$J(J,G) = \frac{1}{3} \tag{3.4}$$

and for J(J,L):

TODO: adjust network so Jaccard calculation is different

$$J(J,L) = \frac{|J \cap L|}{|J \cup L|} \tag{3.5}$$

$$J(J,G) = \frac{|\{P\}|}{|\{F,P,Q\}|} \tag{3.6}$$

$$J(J,G) = \frac{1}{3} \tag{3.7}$$

We now take the top k nodes that have the highest Jaccard score and allow each to vote for new outgoing links to form from J. Here we will select L and recommend any outgoing links from L as destination nodes for predicted links emminating from J. In this case we predict the link $J \rightarrow Q$.

As you can see, the collaborative filtering link prediction process for a given user x involves finding other users most similar to user x, then finding items those similar users are most interested in. In this sense collaborative filtering can be thought of as very similar to k-nearest neighbors, where the distance calculation is based on some similarity metric.

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). A triad consists of three nodes u, z, v where edges (ignoring direction) u,z and z,v exist. Edges between u and v may exist, however the concept of triadic closure posits that an implicit connection exists between u and v.
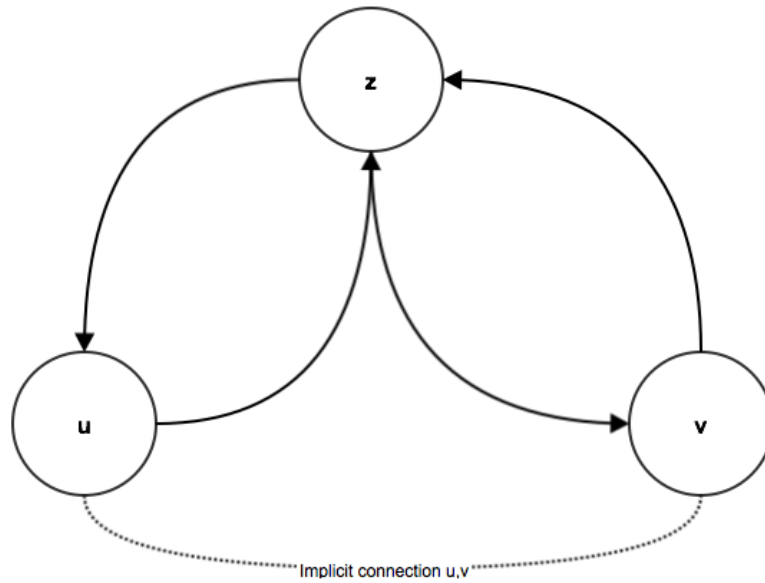
Figure 3.2: A triad is considered to be closed if an edge exists between u and v.

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 $u \leftrightarrow v$.

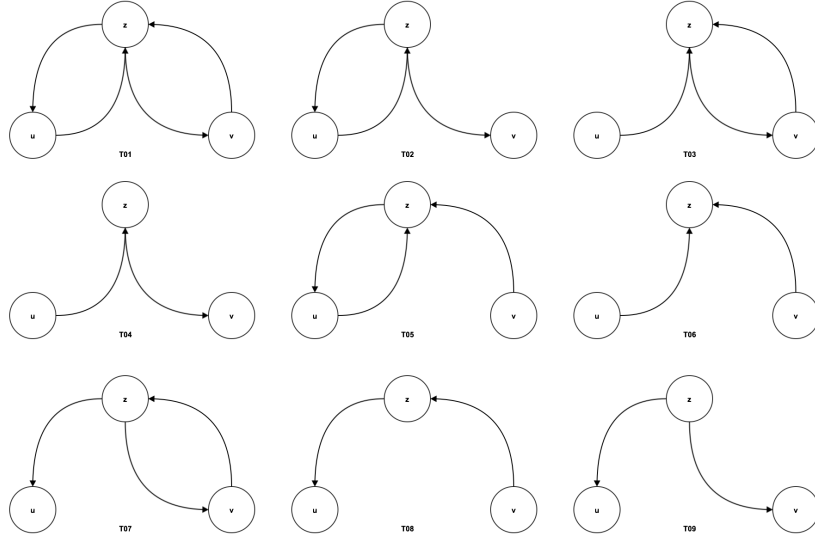INSERT IMAGE OF 3 CLOSED TRIAD PATTERNS

Figure 3.3: All possible open triad patterns are shown. A triad is said to be closed is a connection is established between u and v.
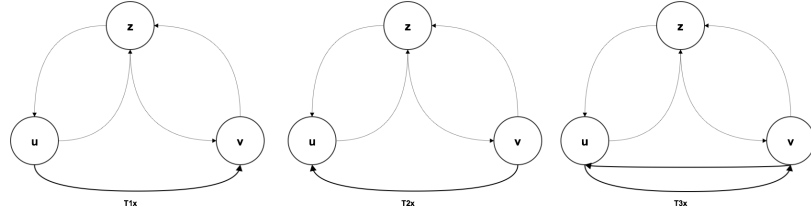

Figure 3.4: A triad is considered to be closed if an edge exists between u and v.

Figure 3.4 shows the three distinct ways in which an open triad can be closed. Either the creation of an edge $u \rightarrow u$, the creation of an edge $u \leftarrow v$, or the creation of two edges $u \rightarrow v$ and $u \leftarrow v$. In terms of the triad pattern id, the first digit indicates if the triad is open (0), closed with an edge $u \rightarrow v$ (1), closed with an edge $u \leftarrow v$ (2), or closed with two edges $u \rightarrow v$ and $u \leftarrow v$ (3). With this nomenclature we are now able to represent each possible triad pattern with a two digit identifier.

$$TC_{uv} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} w^P(u, v, z) \times w(z) \tag{3.8}$$

$$w^P(u,v,z) = \frac{F(T(u,v,z) + 10) + F(T(u,v,z) + 30)}{F(T(u,v,z))} \tag{3.9}$$

Table 3.2: 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 3.3: 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

$$TC_{uv} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} w^P(u,v,z) \times \frac{1}{k_z} \tag{3.10}$$

Having collected these triad pattern frequencies, we can now generate recommendations as we did in section (REFERENCE SECTION COLLABORATIVE FILTERING) for user J.

The first step is identify all open triads of the form u,z,v where J is u and no link between u,z exists. Those triads are:

J,F,G J,I,N J,P,Q

(INSERT GRAPH HERE OF THE THREE TRIADS AND THEIR PATTERNS)

Thus the three possible recommendations that we might generate are $J \to G$, $J \to N$, and $J \to Q$. To determine the rank of the predictions, we must calculate the triadic closeness metric for the pairs (J,G), (J,N) and (J,A).

$$TC_{JG} = \sum_{z \in \Gamma(J) \cap \Gamma(G)} w^P(J, G, z) \times w(z) \tag{3.11}$$

$$TC_{JG} = w^P(J, G, F) \times w(F) \tag{3.12}$$

$$w^P(J,G,F) = \frac{F(T(J,F,G) + 10) + F(T(J,F,G) + 30)}{F(T(J,F,G))} \tag{3.13}$$

We can see that T(J,F,G) = T03, so we have:

$$w^P(J,G,F) = \frac{F(T03 + 10) + F(T03 + 30)}{F(T03)} \tag{3.14}$$

$$w^P(J,G,F) = \frac{0 + 0}{3} \tag{3.15}$$

$$w^P(J,G,F) = 0 \tag{3.16}$$

$$TC_{JG} = 0 \tag{3.17}$$

Next for J,N:

$$TC_{JG} = \sum_{z \in \Gamma(J) \cap \Gamma(G)} w^P(J, N, z) \times w(z) \tag{3.18}$$

$$TC_{JG} = w^P(J, I, N) \times w(I) \tag{3.19}$$

$$w^P(J,G,F) = \frac{F(T(J,I,N) + 10) + F(T(J,I,N) + 30)}{F(T(J,I,N))} \tag{3.20}$$

We can see that T(J,I,N) = T08, so we have:

$$w^P(J,G,F) = \frac{F(T08 + 10) + F(T08 + 30)}{F(T08)} \tag{3.21}$$

Using table (TRIAD FREQ PATTERN REF TABLE)

$$w^P(J,G,F) = \frac{3 + 1}{14} \tag{3.22}$$

$$w^P(J,G,F) = 0.286 \tag{3.23}$$

$$w(I) = 1/3 \tag{3.24}$$

$$TC_{JN} = 0.286 \times 1/3 = 0.095 \tag{3.25}$$

and finally, calculate TC(J,Q):

$$TC_{JQ} = \sum_{z \in \Gamma(J) \cap \Gamma(Q)} w^P(J, Q, z) \times w(z) \tag{3.26}$$

$$TC_{JG} = w^P(J, P, A) \times w(Q) \tag{3.27}$$

$$w^P(J,P,Q) = \frac{F(T(J,P,Q) + 10) + F(T(J,P,Q) + 30)}{F(T(J,P,Q))} \tag{3.28}$$

We can see that T(J,P,Q) = T04, so we have:

$$w^P(J,P,Q) = \frac{F(T04 + 10) + F(T04 + 30)}{F(T04)} \tag{3.29}$$

$$w^P(J,P,Q) = \frac{2 + 0}{14} \tag{3.30}$$

$$w^P(J,P,Q) = 0.143 \tag{3.31}$$

$$w(P) = 1/3 \tag{3.32}$$

$$TC_{JQ} = 0.143 \times 1/3 \tag{3.33}$$

$$TC_{JQ} = .048 \tag{3.34}$$

We can then sort our recommendations by TC and the most likely edge we will recommend is $J \to Q$.

3.1.4 An adaptive ensemble method

Linear combination of similarity metric (Jaccard) and network based method (triadic closeness). Why might this be beneficial?
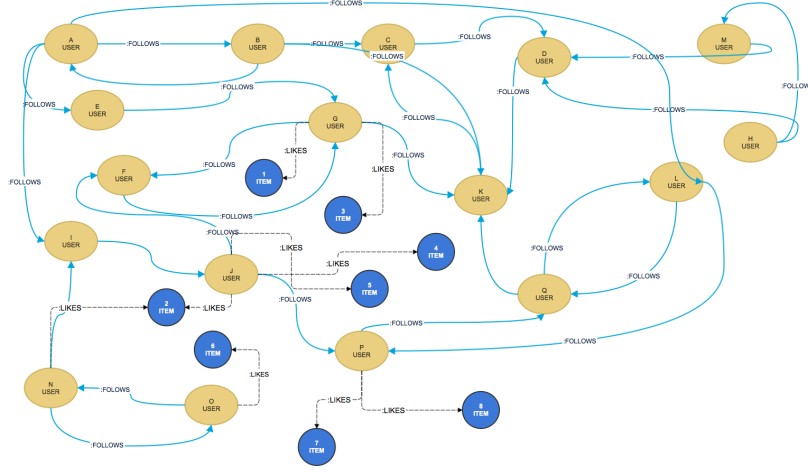
Figure 3.5: We now

$$new_s hit_{uv} = \sum_{z \in \Gamma(u) \cap \Gamma(v)} w^P(u, v, z) \times w(z) \tag{3.35}$$

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

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 Triad detection

Triad detection involves scanning the entire graph to identify triads in the network, both open and closed. This process is

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

3.2.6 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

---

Algorithm 2 Link prediction algorithm

1: $input : G(U, E), x, N$ ▷ explain inputs here
2: $usersSample \leftarrow getRandomUsers(U, x)$
3: $results \leftarrow \{\}$
4: **for** $each\ user\ in\ usersSample$ **do**
5:     $validationEdge \leftarrow getRandomEdge(G, user)$
6:     $removeEdge(validationEdge, G)$
7:     $triads \leftarrow getTriads(user, G)$
8:     $pred \leftarrow \{\}$
9:     **for** $u, v\ in\ triads$ **do**
10:         $tc \leftarrow calcTC(u, v, G)$
11:         $pred \leftarrow pred + \{tc, u, v\}$
12:     **end for**
13:     $predictions \leftarrow topXSortedByTC(pred, N)$
14:     $hit \leftarrow is\ validationEdge\ in\ predictions?$
15:     $addEdge(validationEdge, G)$
16:     $results \leftarrow results + \{hit, pred, u, v, validation_edge\}$
17: **end for**
18: **return** results

3.3 Data
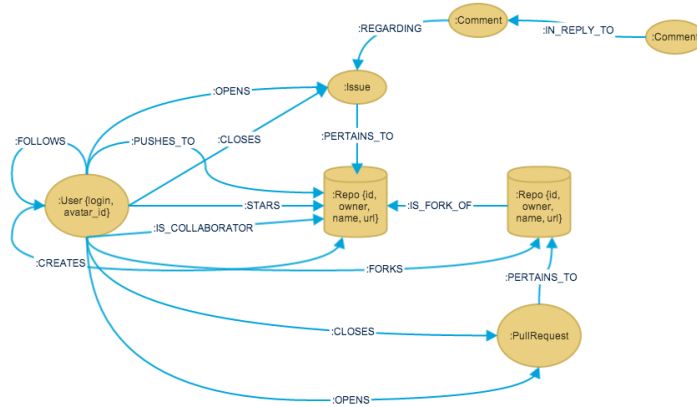
Figure 3.7: GitHub data model as a labeled property graph.

Data was collected from GitHub Archive[Git], 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 3.8: 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 3.9 shows an example of a subgraph of user and repository nodes and the interactions among those entities, modeled as a graph.

3.3.1 Github Archive

FollowEvent

```json
{
  "created_at": "2013-07-11T15:03:05-07:00",
  "payload": {
    "target": {
      "id": 4602587,
      "login": "smarquez1",
      "followers": 1,
      "repos": 1,
      "gravatar_id": "42eb6556201588fa7641bf2f0bf615e6"
    }
  },
  "public": true,
  "type": "FollowEvent",
  "url": "https://github.com/smarquez1",
  "actor": "matiasalvarez87",
  "actor_attributes": {
    "login": "matiasalvarez87",
    "type": "User",
    "gravatar_id": "0ee1a5bec013545c91ad05c451fb9715",
    "name": "Matias Alvarez Duran",
    "company": "NaN Labs",
    "blog": "http://ar.linkedin.com/pub/matias-emiliano-alvarez-duran/17/39b/
a96",
    "location": "Argentina",
    "email": "matiasalvarez87@gmail.com"
  }
}
```

Listing 3.1: LaTeX Typesetting By Example

WatchEvent (Stars)

```json
{
  "created_at": "2013-07-11T15:01:56-07:00",
  "payload": {
    "action": "started"
  },
  "public": true,
  "type": "WatchEvent",
  "url": "https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-
Bayesian-Methods-for-Hackers",
  "actor": "cebe",
  "actor_attributes": {
    "login": "cebe",
    "type": "User",
    "gravatar_id": "2ebfe57beabd0b9f8eb9ded1237a275d",
    "name": "Carsten Brandt",
    "company": "cebe.cc",
    "blog": "http://cebe.cc/",
    "location": "Berlin, Germany",
    "email": "mail@cebe.cc"
  },
  "repository": {
    "id": 7607075,
    "name": "Probabilistic-Programming-and-Bayesian-Methods-for-Hackers",
    "url": "https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-
Bayesian-Methods-for-Hackers",
    "description": "aka \"Bayesian Methods for Hackers\": An introduction to
Bayesian methods + probabilistic programming with a computation/understanding-
first, mathematics-second point of view. All in pure Python ;)  ",
    "homepage": "http://camdavidsonpilon.github.io/Probabilistic-Programming-
and-Bayesian-Methods-for-Hackers",
    "watchers": 3353,
    "stargazers": 3353,
    "forks": 444,
    "fork": false,
    "size": 1264,
    "owner": "CamDavidsonPilon",
    "private": false,
    "open_issues": 21,
    "has_issues": true,
    "has_downloads": true,
    "has_wiki": true,
    "language": "Python",
```

```
38      "created_at": "2013-01-14T07:46:28-08:00",
39      "pushed_at": "2013-07-04T17:08:47-07:00",
40      "master_branch": "master"
41    }
42  }
```

Listing 3.2: peep this JSON object dawg

3.3.2 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.

3.3.3 Data Analysis

Table 3.4: Dataset descriptive statistics

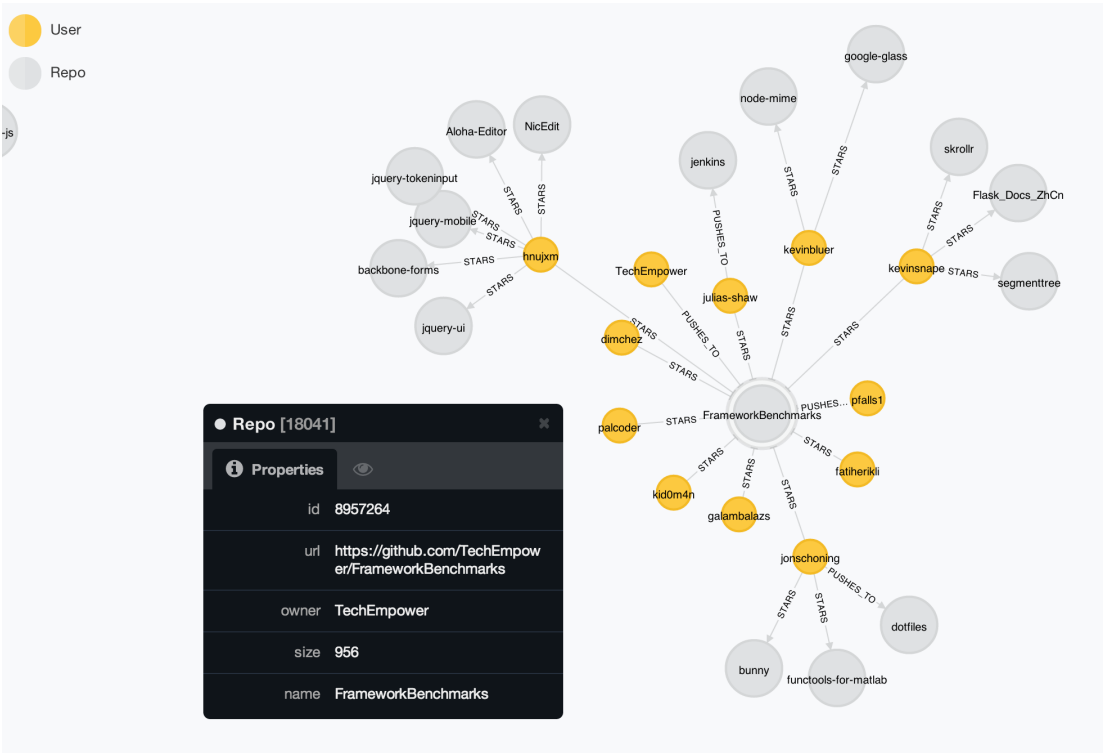|	
Nodes (users)	539893
Edges (:FOLLOWS relationships)	919489
Mean degree	3.4
Standard deviation of degree	30.3

Figure 3.9: GitHub data model as a property graph. Screenshot from Neo4j graph database interface.

Table 3.5: Here we see the most "central" Users per their PageRank rankings. This is based on the graph created by user-user follows edges.

User	PageRank
funkenstein	413.14
mojombo	300.01
torvalds	248.21
rippleFoundation	220.29
visionmedia	140.52
paulirish	129.59
BYVoid	114.29
schacon	112.17
JakeWharton	110.55
defunkt	106.86
mattt	99.38
worrydream	87.33
hakimel	83.05
pjhyett	80.89
addyosmani	80.59
mbostock	75.63
mdo	70.40
LeaVerou	66.92
tekkub	62.24
nf	60.93

Table 3.6: Top 20 central GitHub repositories by PageRank.

Repository	PageRank
https://github.com/vhf/free-programming-books	455.07
https://github.com/twbs/bootstrap	335.46
https://github.com/jquery/jquery	289.42
https://github.com/resume/resume.github.com	251.99
https://github.com/mandatoryprogrammer/Octodog	233.41
https://github.com/angular/angular.js	202.18
https://github.com/mbostock/d3	149.55
https://github.com/torvalds/linux	133.48
https://github.com/FortAwesome/Font-Awesome	121.47
https://github.com/twitter/bootstrap	111.42
https://github.com/laravel/laravel	106.75
https://github.com/papers-we-love/papers-we-love	102.25
https://github.com/joyent/node	101.27
https://github.com/rethinkdb/rethinkdb	92.34
https://github.com/neovim/neovim	91.52
https://github.com/libgit2/libgit2	90.99
https://github.com/rogerwang/node-webkit	88.23
https://github.com/github/gitignore	88.08
https://github.com/dypsilon/frontend-dev-bookmarks	86.73
https://github.com/zurb/foundation	84.25

3.4 Evaluation Metrics

Each method described above in section (PROPER SECTION CITATION) is evaluated using the same evaluation metrics for comparison.

3.4.1 Precision / Recall

3.4.2 Hit Ratio

3.4.3 Operator Receiver Curve

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.

Table 4.3 on page 29 demonstrate the creation of a pleasant appearing table, which helps to read the table without attracting to much attention by the use of shaded colors. The caption uses the additional short caption in square brackets [ ], which is used in the list of tables, see page 37.
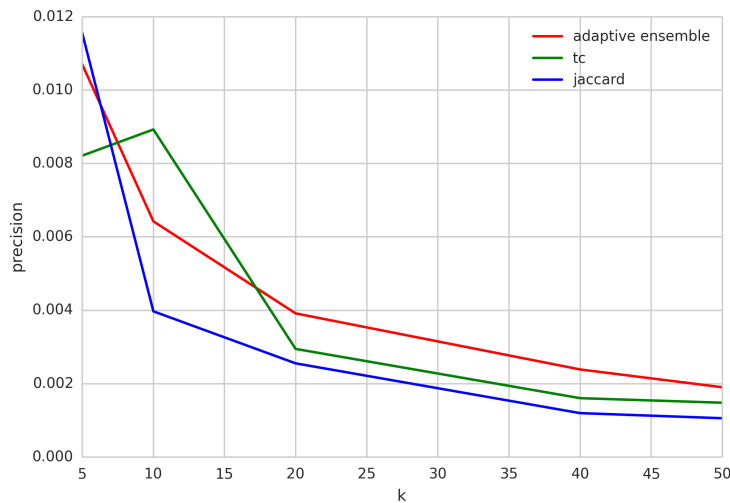
Figure 4.1: Test image for television (Origin of the image: `http://de.wikipedia.org/wiki/Testbild`).

All possibilities of grouping pictures side by side, on top or in matrices can be realized. Each subfigure is created in the same way as a graphic inside a figure, just enclosed by

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

a figure environment, as shown in fig. 4.3.

For complex subfigure constructs and correct alignment of the subcaption the `floatrow` provides powerful commands.

4.0.4 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

**Table 4.2:** Numbers of Computers in the department, By Type.

Mac (Apple)	2
Windows XP, 7	60
Linux (Server)	10

Table 4.3: Comparison of the mean-field predictions for the critical temperature of the Ising model with exact results and the best known estimates for different spatial dimensions d and lattice symmetries.

lattice	d	q	T_{mf}/T_c
square	2	4	1.763
triangular	2	6	1.648
diamond	3	4	1.479
simple cubic	3	6	1.330
bcc	3	8	1.260
fcc	3	12	1.225
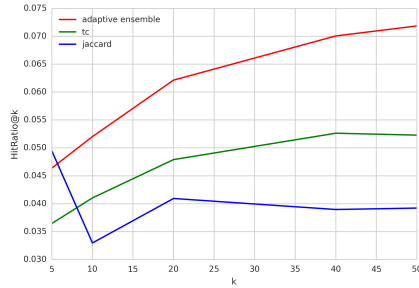
(a) The first subfigure.
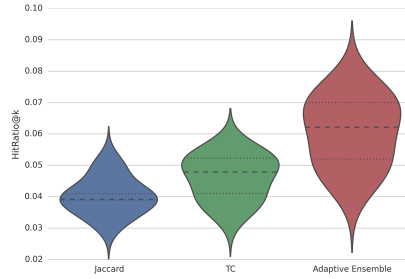
(b) The second subfigure.

Figure 4.2: Demonstration of the *subfigure* environment inside a figure environment

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 *a*ll 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.
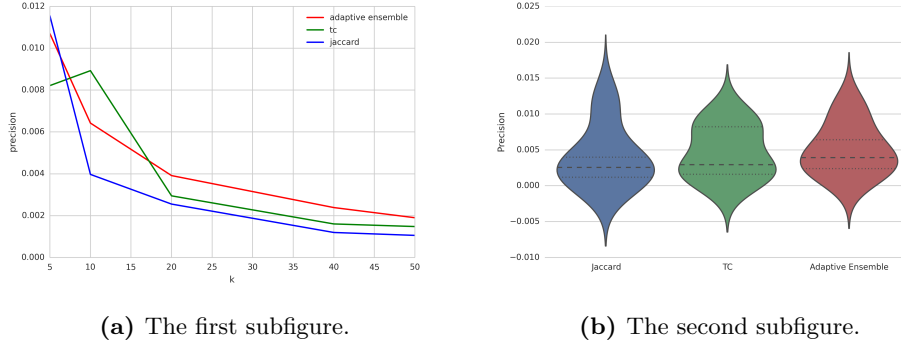
(a) The first subfigure.

(b) The second subfigure.

Figure 4.3: Demonstration of the *subfigure* environment inside a figure environment

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

Bibliography

[Git]   URL: http://www.githubarchive.org/ (cit. on p. 19).

[Goo94]   MICHEL GOOSSENS, FRANK MITTELBACH, and ALEXANDER SAMARIN: *The LaTeX Companion*. 1st ed. Reading, Mass.: Addison-Wesley, 1994. 528 pp. (cit. on p. 1).

[Rod]   MARKO A RODRIGUEZ and PETER NEUBAUER: 'The Graph Traversal Pattern'. In (), vol.: pp. 1–18 (cit. on pp. 2, 4).

List of Figures

List of Tables

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.