Thesis: Graph-oriented data mining

Social recommendation in Twitter

Submitted for a degree in Computer Engineering

Author: Richard A. J. Weiss, z3252543

Supervisor: Mike Bain

Assessor: Xiongcai Cai

October 2012

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering

Abstract

With the proliferation of users on social networks, it has become more and more important for network providers to be able to generate high quality recommendations for users to follow or befriend. Machine learning techniques are highly applicable to it, and this thesis presents one approach to this problem. My approach is based on several novel features of the social graph which are combined with commonplace machine learning algorithms, to provide recommendations.

Contents

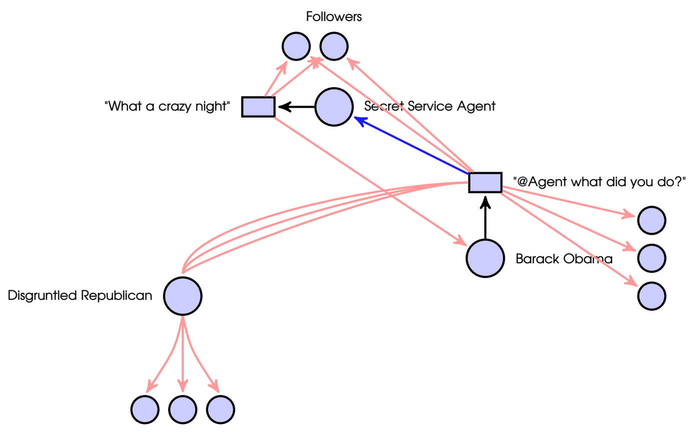


Figure 1: Figure: Twitter example

1 Introduction

Social networks are becoming larger and larger, with more diverse user bases. As such, it is important that users are able to find similar users to follow or befriend, and machine learning can help with this problem. Twitter is a social network which has experienced rapid growth in the last few years; founded in 2006, it has grown to more than 100 million users. My thesis aims to use features of the Twitter social graph to recommend users to one another, specifically on Twitter.

1.1 Twitter

The social network Twitter was launched in March 2006. It has few features: users can post messages that are broadcast to their followers; these messages are called "tweets" and are limited to 140 characters in length. Any user can follow another user; unlike some services, these follow relationships need not be reciprocal. Users can re-broadcast tweets from other users; this is called "re-tweeting". Users can also respond directly to tweets by including one or more usernames, prefixed with an @ symbol.

1.1.1 How Twitter works

Figure 1 shows basic Twitter functionality. Users are drawn as circles, and red arrows show the relationships between users. Tweets appear as boxes, with arrows to identify the authors.

The user 'Secret Service Agent' posted a tweet, "What a crazy night". This tweet doesn't refer to any specific user, and is broadcast to all of the Agent's followers. One follower, Barack Obama, replies by including "@agent" in his tweet. The Agent receives a notification of this reply (marked here as a blue arrow). Finally, a user, Disgruntled Republican, retweets Obama's tweet: this broadcasts Obama's tweet to all of the Disgruntled Republican's followers.

Tweets by a user, or tweets including the user's username, appear on the user's Twitter profile page and on their followers' feed pages. This makes the set of users that one follows meaningful, as it determines which user's Tweets appear in the feed page, and controls the visibility of their tweets.

1.1.2 Social and interest recommendation

Evidence from Java et al. (2007) suggests that there may be two kinds of recommendation: social recommendation and interest recommendation. The difference between the two is that social recommendation aims to recommend friends to users, people with whom they are likely to have a social relationship. Interest recommendations aim to provide users who the active user will follow out of interest for their content, rather than out of an interest in the person producing them. These two forms of recommendation are quite different and require different approaches.

1.2 Structural recommendation

My thesis focuses on structural recommendation, which seeks to recommend users primarily through their existing connections to users, rather than the content they produce or consume. A side effect of this is that my approach will be far more useful for social recommendation than for interest recommendation.

Java et al. (2007) found that three kinds of Twitter users:

1. social users, who use Twitter to communicate with their friends
2. information seekers, who use Twitter as a source of information
3. information providers, who post to Twitter but do not frequently read other's posts.

Of these, the strongest graph structure will appear for social users (Watts and Strogatz (1998b)). This structure appears because social groups tend to be very cohesive, with friends sharing many mutual friends. Social users tend to follow one another, forming near-cliques. By contrast, in networks where users follow information sources, we cannot deduce anything more than their interest in the topic that the source provides. There may be patterns in this behaviour—for example, a user who follows many sources on the same topic—but ultimately less information is available for analysis, and the social graph of a purely interest-based network would be less cohesive and interlinked.

This thesis aims to produce a set of recommendations. These recommendations are of users that the active user should consider following, and would be presented as “people you may know” or similar on the Twitter website. The aim is to have this set of people be people that the active user would want to follow.

1.3 Value

This research is valuable because it expands our ability to identify and understand the ways that people interact socially. There are considerable implications for social networking businesses. As these businesses expand, adapt and compete against one another, it is increasingly important for them to understand why and how their users interact with each other. This insight allows them to build more engaging systems that facilitate positive social interaction. In other words, the social networks that thrive will be the ones that foster broad, deep social communities. This research is a part of that development.

The value of this research comes from its expansion of our understanding of the reasons and ways that people interact socially. With the recent expansion of social networks it is becoming increasingly important that those creating these systems understand why and how people interact with each other. With such insight, they can build more engaging systems that facilitate this interaction.

2 Literature review

2.1 Other recommendation approaches

The existing research on Twitter recommendation can be arranged by how much it considers the user's followers or followers, compared to how much it considers the content of their tweets as a source of data.

I will also review literature that describe some general data mining approaches that are useful for the problem.

Content based Some recommendation systems draw their conclusions solely from textual data, mined from all users. For example, Esparza et al. (2012) explains a system to recommend products from reviews in a Twitter-like service called Blippr.

The authors create two indices: products and users. The product index links terms to products, and the user index links users to the keywords in their reviews.

Both indicies have weightings assigned to the terms. The authors propose two weighting schemes, 'term-frequency inverse document frequency' (Salton and McGill (1986)) and BM25 (Robertson et al. (1996)). Recommendations taken by comparing the weighted term of user and products , or by finding the recommended products of similar users. This approach can be applied to Twitter by treating user's tweets as products or by treating any unidirectional friend recommendation as a product recommendation.

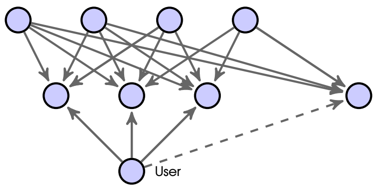
A similar approach is taken in Deshpande and Karypis (2004) where a matrix of item-to-item similarity is constructed. The similarity of products can be found in a number of ways, but the authors suggest cosine-based similarity between items.

Deshpande and Karypis also suggest that similarity can come from the conditional probability that a user buys item j, given that they have purchased item i.

Once the similarity matrix has been constructed, finding the recommendations is a matter of multiplying the similarity matrix with a vector of user purchases, where the entry corresponding to a purchased value is given the value of one. The resulting vector represents the recommendation strengths for the items.

Ma et al. (2011) is an example of the integration of both content and social features. The authors use an approach similar to Deshpande and Karypis (2004) but augment it with social trust information.

Local social graph based Recommendation algorithms that draw on a user's social graph are more aligned with my proposed approach. Armentano et al. (2011) look at a user's followees followers:



The dotted line is the resulting recommendation, because all of the people the follow the people the user follows also follow the recommended user.

The dotted line is the resulting recommendation, because the active user follows the three users in the middle, and those three users are followed by the four users on the top row, and these users also follow the recommended users.

Further research into graph-based mining was undertaken by Naruchitparames et al. (2011) - while the above studies looked for interest relations, these authors focus on *social recommendations* (see Section 1.1.2 for an explanation of the terms). They approach the problem by finding social features, specifically shared friends; location; age range; general interest from user's profile; co-occurrence at events, in photos and in groups (the research was with Facebook data, which includes such information); movies users like; education; and similarities in religion and politics. The authors then build a "social genome"—which reflects how similarity in these features predicts a user's friendships. Finally, the social genome is used to recommend other users. Their approach worked especially well when they pre-filtered the users for consideration based on the Facebook social graph. Results were improved markedly when they limited the candidates to friends of friends, suggesting that the social graph is very useful in giving social recommendations.

General social graph based Kim and Shim (2011) use the social graph almost exclusively. Their approach uses PageRank to find topics and users to recommend. Using PageRank means that their results include implicit information about the recommended user's influence in the network, as PageRank is an influence determination algorithm. Similarly, the recommendation of tweets in this way considers the reach and influence of the tweet being recommended, which will presumably improve recommendation through collaborative filtering. Their results shown the approach to be effective and scalable, and it will return results that have good social and interest utility.

2.2 Assisting literature

The following literature does not deal directly with recommendation, but provides research that I feel will assist my thesis.

Twitter's follow relationship can be unidirectional, but a unidirectional relationship does not mean that two users do not have a social connection. These relationships are useful in inferring other social connections and as sources of recommendations in themselves. Kuan et al. (2008) describes a method to find such relationships by searching for cliques. The problem with searching for cliques is that a clique of size one is made of users with one connection to the other clique members, equivalent to saying that it is a bidirectional follow relationship (and there is nothing to recommend). A clique of size two is too large, because it includes any two users with a common friend (which yields many irrelevant results). They propose a 1.5 clique, where the distance between user u and v plus the distance between v and u is no greater than three for all members of the clique. This and similar approaches may be useful for finding inferred friendships.

Much of my thesis is built on the assumption that good social recommendations can be found with graph clusters or closeness, as suggested in Watts and Strogatz (1998a). There are several ways that I may approach this problem. The local clustering coefficient outlined in Watts and Strogatz (1998a) is an easily implemented option Local clustering is appropriate because of the size of the social networks involved. Local clustering scales well in these cases; in contrast many global clustering algorithms are NP hard. There are some approximations that perform reasonably well, both in speed and results: Blondel et al. (2008) outlines a greedy algorithm that finds fuzzier clusters by grouping neighbouring nodes until the modularity gain from doing so becomes negative. This algorithm also accepts weighted edges, which will make it trivial to augment the groupings with other data, such as the social features outlined in Section 7.1.

2.3 Machine learning algorithms

Non-negative matrix factorisation is a relatively new machine learning approach which I chose to apply to this problem. It was introduced in Lin (2007) although the method is clearer in Kim and Park (2008). Non-negative matrix factorisation aims to factor a matrix V, which has only positive, real entries, into two approximating matrices, W and H:

$$V = WH$$

The inner dimension of W and H is manually specified for the algorithm.

2.4 Graphs

2.4.1 PageRank

PageRank gives nodes a score (their PageRank) which represents how likely it is that a random surfer would land on a particular while they randomly follow links. This is useful for finding celebrities, who have high PageRanks. See (Page et al., 1999) for details on the exact algorithm.

2.4.2 HITS

The HITS algorithm tries to label nodes as “Hubs” and “Authorities”. A hub is a node that points to many good authorities, and an authority is a node that is pointed to by many good hubs. Given the adjacency matrix $\mathbf{A}$, a node’s hub score ($\mathbf{x}$) is given by:

$$\mathbf{x} = \alpha \mathbf{A} \mathbf{y}$$

and its authority ($\mathbf{y}$) score by:

$$\mathbf{y} = \beta \mathbf{A} \mathbf{x}$$

To compute this, one needs to find the eigenvectors of

$$\mathbf{A}\mathbf{A}^T$$

and

$$\mathbf{A}^T\mathbf{A}$$

that have the same eigenvalue, and from there all that needs to be done is derive $\mathbf{x}$ and $\mathbf{y}$ from:

$$\mathbf{A}^T\mathbf{A}(\mathbf{A}^T\mathbf{x}) = \lambda(\mathbf{A}^T\mathbf{x})$$

and

$$\mathbf{y} = \mathbf{A}^T\mathbf{x}$$

HITS is described in Salton and McGill (1986) and Newman (2010b).

3 Ethics and legal

Since this project uses information created by others, some ethical and legal issues must be addressed.

To create a data set for this project, I mined data from Twitter users without their explicit knowledge or consent. This is expressly permitted by Twitter's terms of service, to which all users have agreed. One condition of data use is that mined data, and any specific results, should not be published. General results are allowed. This satisfies the requirements of my evaluation. (Twitter Inc. (2012)).

There have been cases where users were offended by the suggestions of friend-suggestion algorithms. As I have no plans to publish the recommendations arising from this project, this issue will not arise.

4 Problem formulation

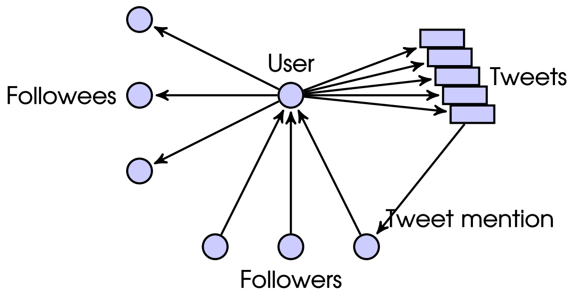
4.1 Problem statement

Given a user's Twitter graph, and a number of social features, provide recommendations that will have a high social utility.

4.2 Problem inputs

The Twitter graph, $\mathcal{G}$, is similar to the social graph: both contain the user's connections (follow relationships) with others. The Twitter graph includes a user's tweets, the people that are referenced by the tweet, and any hashtags or URLs in the tweets.

The diagram shows a subsection of a user's tweet graph.



The Twitter graph under analysis can be arbitrarily smaller than the entire Twitter network (which would include many millions of users and tweets), subject to the requirements of the algorithms being applied.

A recommendation is a suggestion that user a follow user b, which is given at time t, and forms a triple: $\langle a, b, t \rangle$, and may be denoted $R_{a \to b, t}$.

Social utility of a following b at time t for social graph g is given by $\mathcal{S}_g(a, b, t)$. The social utility cannot be accessed directly, but is mirrored to some unknown extent by the presence of mutual friends, bi-directional communication and communications involving social keywords or cues. Part of my thesis will be the discovery and justification of a function from these social queues to $\mathcal{S}^*_g(a, b, t)$, which approximates social utility.

4.3 Evaluation

4.3.1 Dropped relations

To evaluate the system's performance I will drop connections between users and see whether the connections are re-inferred by my algorithms. Many of the studies outlined in section 2 use this approach. Another option would be to collect two data-sets, separated by some period of time and look at the connections users make in that period. I chose not to pursue this path because of the risk that there will be too few new relationships formed in the periods between collection, and therefore it won't be possible to draw valid conclusions from the data.

Another option, albeit one that would generate a smaller data set, is to manually tag connections as social or non—social, or allocate each connection a social score. This will certainly be useful for the classification of social and non-social users, where the datasets required do not have to be as large as for the link prediction case.

Once the various algorithms have been run, the results must be compared in some way. The simplest approach is the training/test set approach, commonly used to evaluate machine learning algorithms. This approach produces correct and incorrect classifications, from which metrics are easy to produce. While classification accuracy is one common metric, it is not very informative in this particular case. Accuracy by itself is sensitive to the skew of the data. When link prediction is run on a large, sparsely connected graph, predicting "no connection" every time is a simple and easy way to make good predictions. In Twitter's case, the average user has 126 followers (Cara Pring, 2012), amongst 140 million users, so the majority classifier produces an accuracy of 99.9991%. To avoid this problem, I instead chose to collect two metrics: Lift and F_1 score (Salton and McGill, 1986). Lift is the improvement of a particular algorithm compared to another. I used the simple majority class case for the algorithm to compare against, so the figure is:

$$\text{Lift}_a = \frac{A_a - A_m}{1 - A_m}$$

where A_a is the accuracy of the classifier in question and A_m is the accuracy of the majority classifier. This metric makes improvements in accuracy much clearer, and gives a better sense of the performance of the algorithms.

The F_1 score is the harmonic mean of a classifier's precision and recall:

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Precision and recall are calculated for the ”link exists” case, as this makes it harder for the heavy skew in the data to obscure a poor classifier. The use of a harmonic mean ensures that it is not possible for the algorithm to ”cheat” by always suggesting a link exists or by never predicting links.

5 Problem approach

The general outline of my approach is:

1. Build a dataset from Twitter, including users and their tweets.
2. Label a subset of these users as ‘primarily social’, ‘news reader’, ‘news provider’ or ‘uncertain’.
3. Build features on this data that I believe will be useful for link prediction.
4. Build features on this data that I believe will be useful for ‘social’, ‘news reader’ and ‘news provider’ link prediction.
5. Use machine learning algorithms to try to predict links.
6. Use machine learning algorithms to try to predict social/non-social preferences.
7. Order the list of predicted links by a combination of strength of prediction and social preferences.

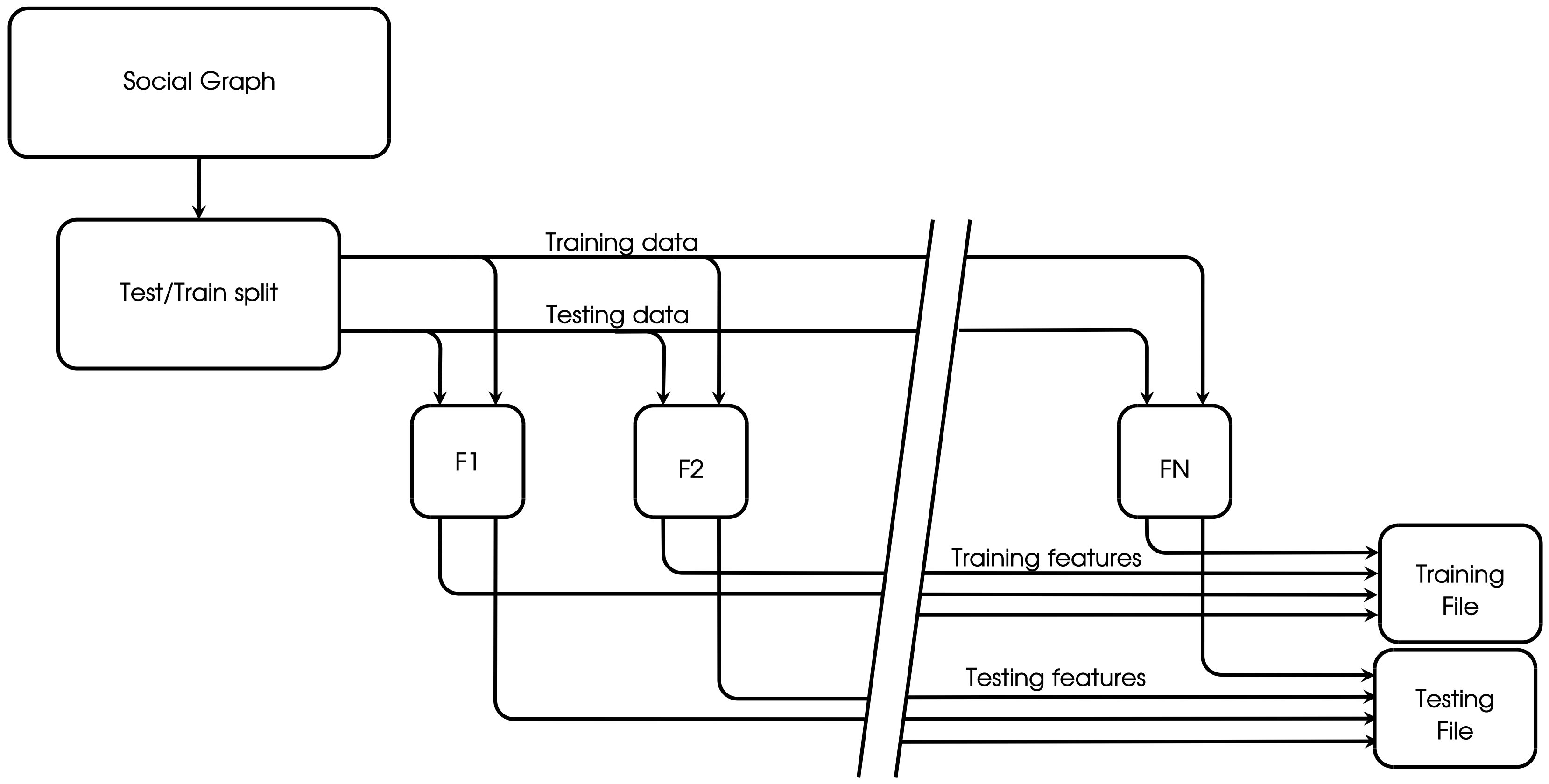


Figure 2: System design, feature generation

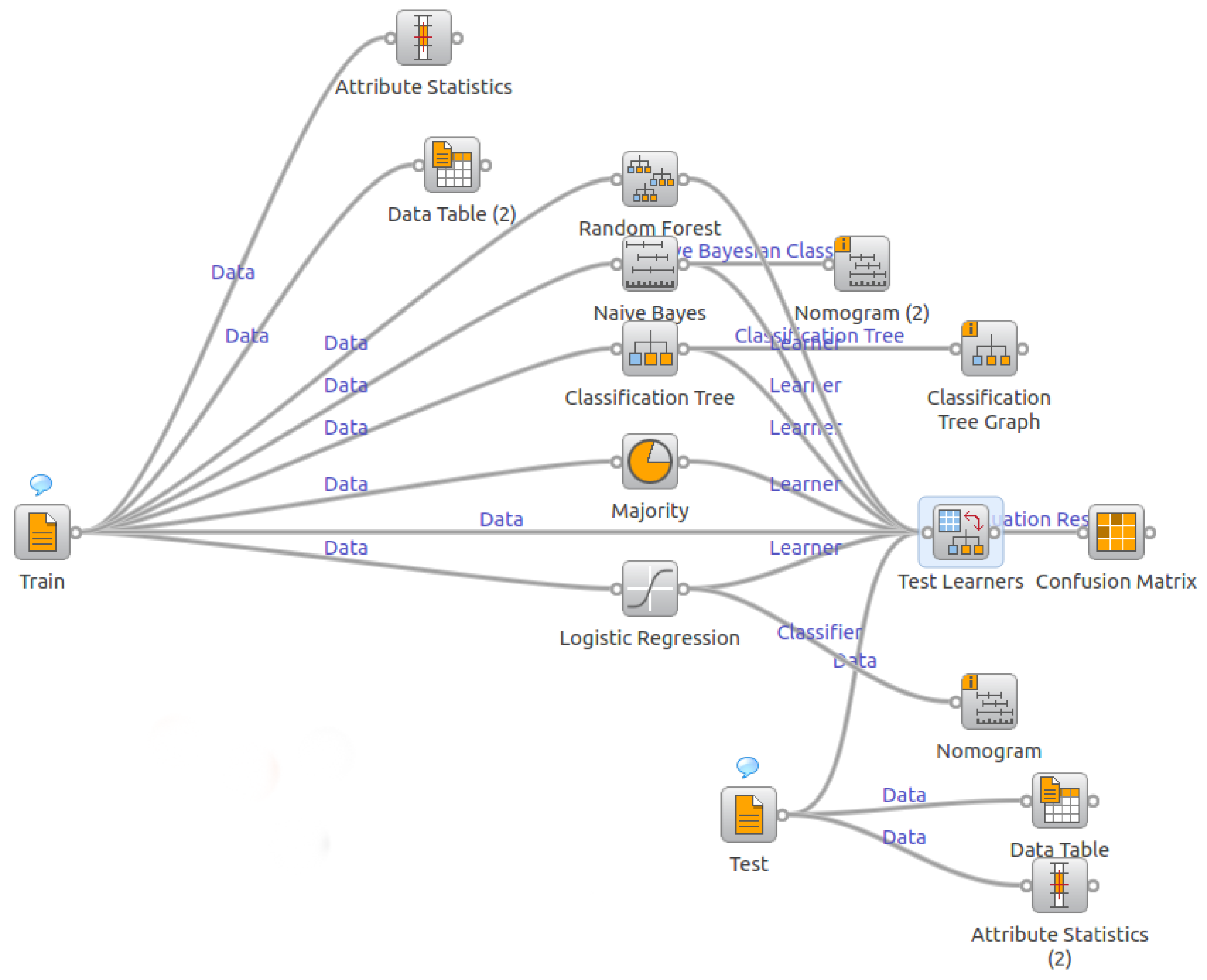


Figure 3: System design, machine learning

6 Data gathering

Testing and development requires a data set. At the beginning of this project, it was assumed that this data set would include users, their tweets and who they follow. A simple approach is to expand the social graph in a breadth first manner, which easily captures a user's social graph and friends.

Mining an individual user begins by collecting their Twitter username, screen name, and the numbers of their friends and followers. Twitter limits the number of items in any request; the per-request limit for tweets is 20. Thus, the user's first 20 tweets are retrieved, then their first 300 friends. (Some users have far more than 300 friends: President Barack Obama follows nearly 700,000 people. For the purposes of this project, I have limited data mining to their first 300 friends, which hopefully will cover most, if not all, of a typical user's friends.)

Each mined user and their 20 tweets are entered into the database, and the `todo` table is updated. The `todo` table is a priority queue, used to decide what action the system should take next. Its options are to mine a new user, or continue mining this user's tweets. Twitter keeps the last 3000 tweets of a particular user, and they can be retrieved in groups of 20, by specifying the id of the most recent tweet in the group, so the `todo` table can contain either an entry like

$$\langle \text{user id}, \text{NULL}, \text{priority}, \text{“user”} \rangle$$

for a user to mine, or

$$\langle \text{user id}, \text{last tweet id}, \text{priority}, \text{“tweet”} \rangle$$

to continue mining a user's tweets. Whichever entry in the `todo` list has a lower priority is acted upon next, and once it is finished, the `todo` table is updated. When mining a user, all of the user's first 300 friends are placed in the `todo` table, with a new, lower priority. When mining a tweet, the next 20 tweets are retrieved; if at least some new tweets are retrieved, then there may still be other tweets by that author. In that case, the `todo` list is updated with the same user id, a new last tweet id, and a lower priority.

This approach allows a flexible tradeoff between preferences for tweet content and

Algorithm 1 Download tweets

```
todo $\leftarrow$ A priority queue with the seed entries
while todo is not empty do
    toMine $\leftarrow$ todo.pop()
    priority $\leftarrow$ toMine[3]
    if toMine[4] = “user” then
        newUsers $\leftarrow$ friends of toMine[1]
        tweets $\leftarrow$ the most recent 20 tweets of toMine[1]
        Insert the new user into the database
        Insert the tweets into the database
        lastTweet $\leftarrow$ tweets[length(tweets)]
        for user in newUsers do
            todo.push($\langle$ user.id, NULL, priority+1, “user”$\rangle$)
        end for
        todo.push($\langle$ user.id, lastTweet.id, priority $\times$ 1.025, “user”$\rangle$)
    else
        tweets $\leftarrow$ the most recent 20 tweets of toMine[1] following and including toMine[2]
        lastTweet $\leftarrow$ tweets[length(tweets)]
        If lastTweet is the same as the tweet we take from the todo list
        we can be sure we have mined all a user’s available tweets
        if lastTweet.id $\neq$ toMine[2] then
            Insert the tweets into the database
            todo.push($\langle$ user.id, lastTweet.id, priority $\times$ 1.025, “user”$\rangle$)
        end if
    end if
end while
```

the social graph. It prevents the system becoming "stuck" while mining a celebrity with many followers. This approach can mine tens of thousands of users in a day.

7 Features

7.1 Features: user similarity

This project includes several features that I feel are informative with regards to social relationships.

7.1.1 Community detection

I suspect that social users, as opposed to news providers, tend to befriend each other and have many overlapping friends. A number of algorithms exist to group nodes of a graph into communities. This will separate distinct social groups into different communities, to prevent recommending users from "distant" groups. Most of an active user's friends, it is suspected, will be one or two steps away from the user.

The grouping algorithm chosen was the one described in Blondel et al. (2008) which seeks to greedily minimise the measure:

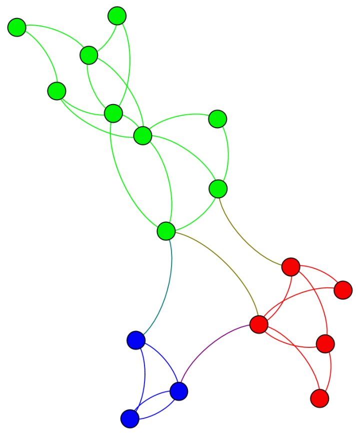
$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

where m is the number of nodes in the graph, A is the adjacency matrix that describes the graph and i and j are indexes into it, k_i is the degree of the i^{th} node, and δ is the Kronecker delta function. In this case the function is applied to c_i, and c_j are the communities of the two nodes. There are fast incremental versions of this expression, which makes the algorithm fast enough to be practical in application, even in very large networks.

Once the locally optimal assignment of communities has been found, the network's nodes are compressed, with all the members of a community becoming a single node, preserving its connections to other communities. (See figure 4, as this is much clearer in picture form.) From here, two different values are produced: a simple true or false value for whether a user is in the same community as the active user, and the distance between the active and target users in the compressed graph.

In the featuresets produced by the program, this feature is called `modClass`, and it is given by the formula: modClass

$$\text{modClass} = \begin{cases} 0 & \text{where there is no path} \\ \frac{1}{1+d} & \text{where there is a path of length } d \end{cases}$$


(a) A simple network, coloured by modularity class

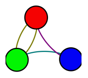

(b) The same network, where the modularity class has been "compressed"

Figure 4: The modularity class feature

It also appears as `simpleCon`, which is given the value 'one' where the the target and active users are in the same community, and 'zero' otherwise.

simpleCon

7.1.2 Co-citation and Bibliographic coupling

A user's friends are typically friends with each other. To try to detect these sorts of relationships, I look for common users: finding the number of friends and followers that each user shares with anyone I am considering for a recommendation. Co-citation and bibliographic coupling are both measures that reflect this.

These two measures are quite similar in many ways. Both concern the number of shared neighbours that two nodes have. Co-citation looks at the number of shared friends, while bibliography coupling looks at the number of shared followers. Where A is the adjacency matrix, these values are:

$$C = AA^T$$
$$B = A^TA$$

The matrices under consideration are extremely large. Instead of finding the entirety of C and B, it is more efficient to find the column for the active user. This reduces the size of the computation needed significantly.

Once these vectors have been found, the features produced for each target user j, for active user i are matrix entries C_{ij} and B_{ij}.

Note that these measures have been used by other authors Armentano et al. (2011). with some refinements, and have yielded good results.

In the featuresets, these values were called `cocite` and `biblio`, and were simply the relevant entries in C and B.

cocite
biblio

7.1.3 Similarity measures

Similarity measures are predicated on the assumption that users will share similar neighbours, and that by comparing users based on their shared friends and followers, I will be able to infer other users they will like.

Cosine similarity From the social graph's adjacency matrix it is possible to easily compute the cosine similarity of two users. Selecting two users, I take their rows in the adjacency matrix. This means that we are considering the cosine similarity of the people that they follow, that is, what proportion of all followers are shared by two users. In this way it is similar to the co-citation measure.

For active and target users with rows A_a and A_t, the cosine similarity is given by:

$$\frac{A_a \cdot A_t}{||A_a|| \cdot ||A_t||} \tag{1}$$

It is also easy to compute the cosine similarity based on followers, rather than friends, by using the above formula with columns of the adjacency matrix instead of rows, which gives a bibliographic measure of similarity.

On a technical note, many people in the social graph have undefined cosine similarity. This happens when they have no followers or no friends, so the denominator of equation 1 is zero. In these cases the similarity is set to zero—that is, no correlation and no similarity—which notes the lack of information about them.

In the featureset cosine similarity appears twice, as `bibcosine` and `cocosine`, for the follower and friend similarity, respectivley.

bibcosine
cocosine

Pearson similarity Pearson similarity differs from cosine similarity in that it aims to normalize the similarity of two users being compared by the expected number of neighbours they would have if they picked at random. See Newman (2010a) for details. The equation for this metric is:

$$r_{ij} = \frac{(\sum_k A_{ik} - \langle A_i \rangle)(\sum_k A_{jk} - \langle A_j \rangle)}{\sqrt{\sum_k (A_{ik} - \langle A_i \rangle)^2} \sqrt{\sum_k (A_{ik} - \langle A_j \rangle)^2}}$$

Where $\langle A_m \rangle$ is the mean of the m^{th} row.

As with cosine similarity, it is trivial to perform this calculation on columns instead of rows, to give a measure of the similarity between a user's followers, rather than their friends.

Where the denominator is zero, the similarity is set to zero rather than being undefined. This reflects that there is no information about the similarity between the two users.

In the featureset pearson similarity appears twice, as `bibpearson` and `copearson`, for the follower and friend similarity, respectivley.

bibpearson
copearson

7.1.4 Non-negative matrix factorisation

Non-negative matrix factorisation aims to decompose a matrix V (of size $n \times m$) of only positive entries into two other matrices, W and H, the product of which is close to V. W has size $n \times r$ and H has size $r \times m$. Various ways to define the difference exist. For this project I chose to use the frobenius norm, sum of squared errors:

$$e = \sum_{i=1}^{n} \sum_{j=1}^{m} (V_{ij} - (WH)_{ij})^2$$

This is one of the most straightforward cases of NMF, which simplified the implementation of the algorithm.

Initially, W and H are set to random values, and update rules applied iteratively until the matrices W and H approach V.

$$H_{ij} \leftarrow H_{ij} \frac{(W^T V)_{ij}}{(W^T W H)_{ij}}$$
$$W_{ij} \leftarrow W_{ij} \frac{(V H^T)_{ij}}{(W H H^T)_{ij}}$$

Non-negative matrix factorisation can be viewed as a way to eliminate noise from data with some underlying pattern. The pattern here would be the groupings in the social groups on Twitter, which is discussed further in later chapters.

I used a custom implementation of non-negative matrix factorisation for my thesis, which supports distributed and local computation. This is essential for larger data sets where the matrices become too large for memory. The distributed version uses naïve matrix multiplication algorithms, but appears to give acceptable performance. The distributed version has not been needed for the smaller datasets I used for development and testing.

The use of NMF is based on the assumption that user's choices of friends are motivated by common structural factors shared by a group of friends. Within the social graph these will be expressed in the friends that people have. By factorising the matrix, I hope to uncover these structural factors, and remove network "noise" caused by idiosyncrasies in individual friend relationships. When V and H are multiplied together it yields a new social graph where users are connected with users who would be good matches for them.

The choice of the inner dimension r is important. Too small, and the structural factors will not appear; too large, and the factorisation will not generalise enough, and take too long to process.

In the featureset, this appears as the feature `nmfUser`. nmfUser

7.2 Features: news/social differentiation

A simple way to generate social recommendations would be to build a list of broad recommendations then penalise 'news sources'. This project hopes to head off news sources at the pass. It separates social users from news sources by concatenating a user's tweets into one large piece of text, then applying several machine learning algorithms to these sets of tweets.

This project used several simple features to differentiate social and news providers:

Followers The absolute number of a user's followers. This data is taken directly from Twitter.

Friends The absolute number of a user's friends. This data is taken directly from Twitter.

Friends to followers The user's number of friends divided by the user's number of followers, and followers divided by friends.

Linguistic novelty feature The number of unique words divided by the total number of words in a user's tweets.

Slang words The proportion of slang words in all the user's tweets. The list of slang words was taken from `www.noslang.com`. As well as absolute values for slang, the ratios of slang to total words and total words to slang are also included.

URLs The number of URLs in a user's tweets, as an absolute value; and the ratios between URLs and total words, and total words and URLs.

User-directed tweets The number of times a user uses the "`@`" symbol, which on Twitter generally denotes a tweet directed at another user. As with all the word-based features, the ratio between this number and the total number of words, and the total number of words and this number, are reported.

Presence of words In order to try to find words that appear more commonly in social or news sources, the top 200 most common words across all tweets are found. Then, for any user, it is reported whether or not they use the word, and also the proportion of that word amongst either all common words, or all words in the text.

The algorithm for this is:

Algorithm 2 Compute word features for documents

```
document: the document being examined, as a list of tokens
wordFeatures: the list of the 200 most common words in all documents
function DocumentWordFeatures(document, wordFeatures)
    documentWords ← set(document)
    discreetFeatures ← []
    for word in wordFeatures do
        discreetFeatures.append(word $\in$ documentWords)
    end for
    return discreetFeatures
end function
```

Total retweets The number of times a user uses the phrase 'RT', which in Twitter denotes a retweet.

PageRank The PageRank algorithm was run over the entire network, using a damping parameter of 0.85 (a common default). The PageRank for each user is then reported.

Hub and Authority score The HITS algorithm was run on every user in the network and each user's hub and authority score are reported. See 2.4.2 for a brief explanation of the HITS algorithm, or Newman (2010b).

HITS may be useful in detecting whether users are social users or news providers. Presumably news providers are more like authorities, and users with a particular, focused interest may be analagous to readers with particular interests.

8 Machine learning algorithms

Once the features are collated, they are passed to machine learning algorithms. The models that they create form the system predictions. The system did not produce many features per data point, which means that highly-dimensional approaches are not needed. The machine learning is handled by “Orange”, a machine learning program (Curk et al. (2005)).

8.1 Link Prediction

Here, link prediction is attempted by naïve bayes, Orange’s classification tree implementation, a simple majority implementation, logistic regression and random forests.

All the features are continuous features, with the exception of the simple community feature, which is a true/false value.

I use two approaches to build the set of links for prediction. The first approach is simply to apply the feature generators to every link in the social graph. The second way is to only apply the features to a subset of the links. For the active user U_a, I find the set of users who are distance 3 or less from this user and include them in the data set, and go no further than this.

In order to have valid results, I created two data sets, a training and a testing data set. It is very important that this be done *before* the features are produced, because some features are based on the network structure. For example, the modularity class feature gives the distance in hops between the active user and the target user. If the testing users are not treated as such in the feature construction stage, they will end up either zero or two hops away from the active user. This is because either they will be within the active user’s class, or in some other class, but this and the active user’s class must be connected, because of the way the class connection graph is constructed (see section 7.1.1. To avoid this, I select a set of test set users and then modify the social graph passed in to remove connections between the active user and the test set.

8.2 Social/News user separation

Classifying and separating social and news users requires first removing “reader users” from the data set. While this does reduce the accuracy of the classifications overall, it allows me to use algorithms like logistic regression, which are normally a true/false classifier. Readers are extremely rare because of the way that the data set is built, so this will not lead to any serious problems.

The classification prediction is attempted by naïve bayes, Orange's classification tree, logistic regression, majority class and random forests.

8.3 Algorithms

The following algorithms were applied to both problems.

Naïve bayes classifier This was chosen because of its simplicity, and because in Orange it is possible to inspect its models and get some intuition for what features are important and how they are correlated with links. (Kuncheva (2006))

Classification trees Orange's inbuilt implementation (see Quinlan (1986) for general information) was used for tree classification

The inbuilt classifier was configured to exploit the binary nature of the problem. There are only two classes, whether a link exists or not, and with that extra information better classification trees are possible. To avoid overfitting data, the tree size is limited. Experimentation has shown diminishing returns after more than six levels of the tree, so this is set as a maximum.

Classification trees give good performance, especially if they are allowed to grow very large, and they are easy to interpret, which also made debugging some of the features much easier.

Random trees The random tree algorithm is one of the most powerful algorithms available in orange, and according to Breiman (1999), they do no overfit data, which makes it a good standard for comparison and an excellent classifier.

Logistic regression Of all the algorithms used, this is the second simplest and the one with the second largest bias in the hypotheses it can form. While this puts it at a disadvantage compared to the other systems, it is very useful for inspecting the data and looking at which factors are important and how they are important.

Majority The data that is being analyzed has a *very* large skew. Because a feature is produced for every user in the graph other than the active user, most of the features will be negative for where a connection does not exist, while only features corresponding to the active user's friends will be positive cases. This give a skew of between 98 percent and 99 percent negative cases. The Majority classifier is an excellent comparison because it sets a baseline accuracy for any classifier.

9 Results

I ran the above approach in a number of different configurations, with different paramaters. The configurations are outlined in Table 5.

A1	Link prediction without NMF
A2	Link prediction with NMF, $r = 3$
A3	Link prediction with NMF, $r = 10$
A4	Link prediction with NMF, $r = 20$
B1	Link prediction with local NMF, $r = 3$
B2	Link prediction with local NMF, $r = 10$
C1	Social/Non-social without common word analysis
C2	Social/Non-social with true/false for common word presence

Figure 5: System configurations. r is the inner dimension for the matrix factorisation

The r setting varies the inner dimension of the two factored matrices the are produced by the non-negative matrix factorisation. While a lower inner dimension makes the resulting factorisation less accurate, this may be an advantage, up to a point. The aim of NMF here is to try to find patterns that may exist in the adjacency matrix, and limit the inner dimension constrains the complexity of the pattern. An overly complex pattern is a form of overfitting which makes it impossible for the model to generalise and find useful patterns.

A matrix factorisation can be run in a “local” way or a “global" way. The local approach takes a subset of the social graph centered around the active user, builds outward in a breath first manner (see Figure 6), and then applies the various machine learning features to this. The “global” way apples NMF to the entire socal graph at once. These two approaches give significantly different outcomes, with performance improvements in the local case.

[h]

One of the features in the social/non-social analysis is a check on common words within the dataset. This may lead to overfitting in some cases, so it is present in different ways or not present at all. This is in order to build a case that it is not leading to overfitting, which happened when the system was run with smaller data sets.

Finally, the various learning algorithms are also compared.

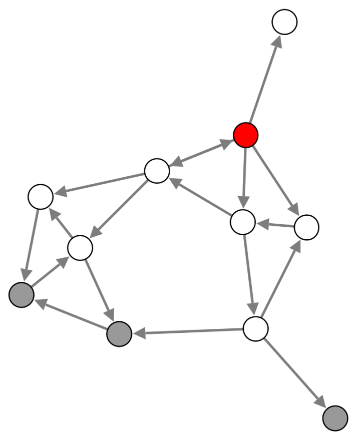


Figure 6: Breath first subset of a user's social graph. Grey users are not included, and the red user is the center user.

9.1 Link prediction: global NMF

See Tables 7, 8, 9 and 10 for the performance of these algorithms.

The best performing algorithms were often the simpler ones which created simpler hypotheses, rather than highly flexible ones like classification and random trees. This indicates that there is some difference between the training and the test sets that prevents machine learning from being applicable to the problem, as will be discussed in section 10.1.2.

The link prediction algorithms had trouble generalising the results of learning to correct hypotheses about the data. This may be because of the inclusion of non-negative matrix factorisation. NMF based approaches appear to work much better in the training data set than they do in the test set, as is disucssed later. This meant that the learing algorithms would attach an overly high weight to the results of the matrix factorisation. This would cause a drop in performance when the system worked with dropped relations used for the test set.

The greatest impovement in accuracy over the majority case was using the classification tree with the NMF with inner dimension three, an improvement of around 12 %, and this occured when NMF was not included. The worst failure was a drop of 22 % for case A2, where the NMF inner dimension was 3. These large falls are indicative of undergeneralisation and differences in the training/test data.

Algorithm	Accuracy	Lift	F1
Classification Tree	99.08%	-13.58%	0.1639
Logistic Regression	99.20%	12.35%	0.0220
Majority	99.19%	0%	
Naive Bayes	99.25%	7.407%	0.1942
Random Forest	96.22%	3.704%	0.1042

Figure 7: Table: Configuration A1—Global link prediction without NMF

Algorithm	Accuracy	Lift	F1
Classification Tree	99.01%	-22.22%	0.1538
Logistic Regression	99.20%	1.253%	0.0220
Majority	99.19%	0%	
Naive Bayes	99.25%	7.047%	0.1782
Random Forest	99.23%	4.938%	0.1584

Figure 8: Table: Configuration A2—Link prediction with global NMF, inner dimension 3

Algorithm	Accuracy	Lift	F1
Classification Tree	99.20%	1.235%	0.2764
Logistic Regression	99.21%	2.469%	0.0435
Majority	99.19%	0%	
Naive Bayes	99.23%	4.938%	0.2609
Random Forest	99.25%	7.407%	0.1782

Figure 9: Table: Configuration A3—Link prediction with global NMF, inner dimension 10

Algorithm	Accuracy	Lift	F1
Classification Tree	99.08%	-13.58%	0.1356
Logistic Regression	99.21%	2.469%	0.0435
Majority	99.19%	0%	
Naive Bayes	99.26%	8.652%	0.1961
Random Forest	96.22%	3.704%	0.1569

Figure 10: Table: Configuration A4—Link prediction with global NMF, inner dimension 20

9.2 Link prediction: local NMF

The results of configurations B1, B2 and B3 are in Tables 11, 12 and 13.

These results are far more promising than the results drawn through global NMF. The greatest improvement occured with the inclusion of a matrix factorisation, with inner dimension 3. In this case there was an 86.67 percent improvement by the random forest classifier over the baseline majority classifier. Even without matrix factorisations, the results were markedly better for localized machine learning; the largest improvement was larger than any of the global cases. This suggests that there are important structural factors that are more apparent in subsets of the network.

9.3 Social/Non-social prediction

The social/non-social prediction shows only a small increase in performance over the baseline. The best performance is found where common words are not used. Because the two experiments are the same except for the inclusion of common words, this suggests that there is overfitting occuring across the different validation folds.

Algorithm	Accuracy	Lift	F1
Classification Tree	98.57%	13.33%	0.6638
Logistic Regression	98.44%	5.455%	0.1053
Majority	98.35%	0%	
Naive Bayes	98.42%	4.242%	0.1569
Random Forest	98.62%	16.36%	0.2857

Figure 11: Table: Configuration B1—Link prediction on local network, no NMF

Algorithm	Accuracy	Lift	F1
Classification Tree	99.71%	82.42%	0.9158
Logistic Regression	99.25%	54.55%	0.7172
Majority	98.35%	0%	
Naive Bayes	98.40%	3.030%	0.1553
Random Forest	99.72%	86.67%	0.9294

Figure 12: Table: Configuration B2—Link prediction on local network, NMF inner dimension 3

Algorithm	Accuracy	Lift	F1
Classification Tree	99.74%	84.24%	0.9213
Logistic Regression	98.75%	24.24%	0.4848
Majority	98.35%	0%	
Naive Bayes	98.46%	6.767%	0.1600
Random Forest	99.49%	69.09%	0.8158

Figure 13: Table: Configuration B3—Link prediction on local network, NMF inner dimension 10

Algorithm	Accuracy	Lift	F1
Classification Tree	90.02%	38.88%	0.6541
Majority	83.67%	0%	
Naive Bayes	88.57%	30.00%	0.6135
Random Forest	88.38%	28.84%	0.6098

Figure 14: Table: Configuration C1—Social/Non-social prediction, without common words

Algorithm	Accuracy	Lift	F1
Classification Tree	89.30%	34.48%	0.6467
Majority	83.67%	0%	
Naive Bayes	85.12%	8.879%	0.3881
Random Forest	87.30%	22.22%	0.4355

Figure 15: Table: Configuration C2—Social/Non-social prediction, with common words

10 Discussion

10.1 Link prediction

The link prediction's most surprising outcome was the difference in performance of globalised approaches compared to localised ones. This raises many questions about the factors the algorithms used most heavily, and whether these factors differ between global and local methods. Finally, the applicability and outcomes of the matrix factorisation are worth discussing.

10.1.1 Important factors

Orange allows an experimenter to examine the trees that it produces to try to understand the hypothesis that it generates. Parts of four different trees are presented in Figures 16, 17, 18 and 19, taken from cases A1, A2, B1 and B2, respectively. Cases A1 and A2 are global approaches wheres B1 and B2 are local ones, and both cases A1 and B1 use no matrix factorisation, while A2 and B2 use NMF with inner dimension three.

From these trees it is clear that in all cases the modularity distance is a very important factor, however the relations act the what I expected them to: many links are formed with people where there is no connection between the modularity classes. This is probably an artifact of the way the data was mined. Some of the users who are analysed may not have as many connections as others, so there will not by paths to the other communities of the graph. This problem disappears when the mining is constrained to users who are more thoroughly mined.

The trees' structure begins to diverge from each other at this point. Those configurations that included NMF (Figures 17 and 19) both have choices based on the matrix factorisation at level two, while the other two (Figures 16 and 18) begin to use similarity or community measures. NMF is interpreted in the expected way by the classification tree: high values are indicative of the existence of a link.

This supports my initial hypothesis that NMF would be a useful way to classify and predict links, and this confirms the power of structural factors in their prediction.

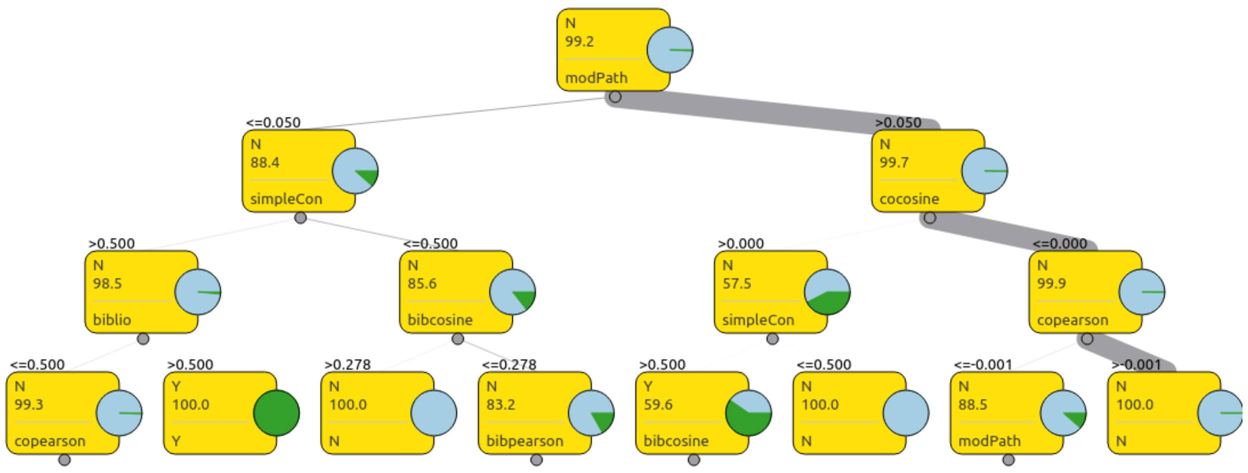


Figure 16: Tree produced by configuration A1 (truncated to four levels)

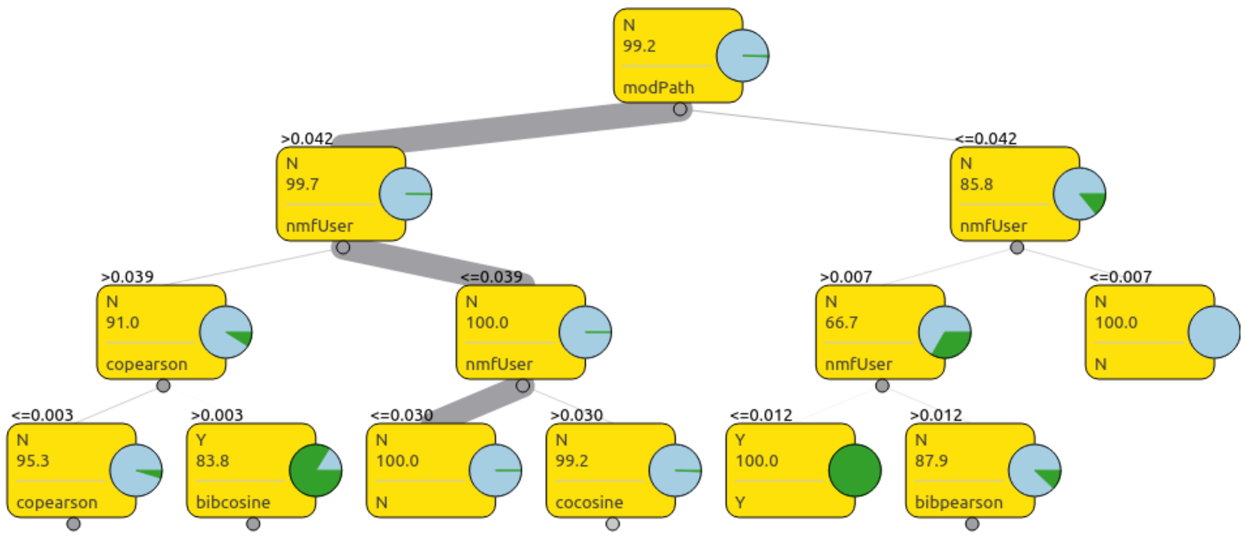


Figure 17: Tree produced by configuration A2 (truncated to four levels)

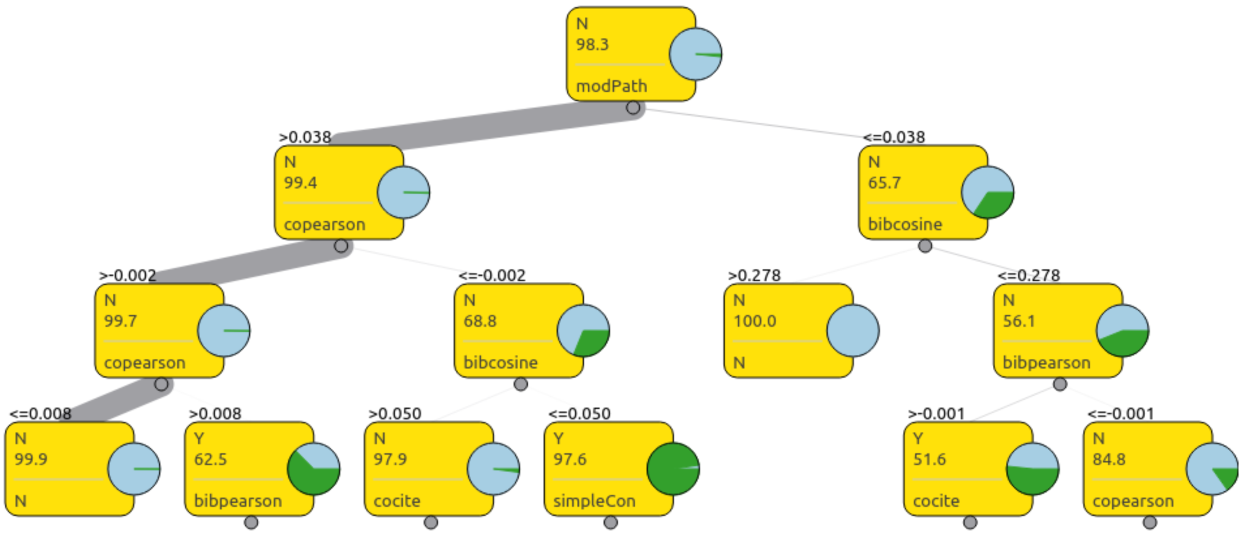


Figure 18: Tree produced by configuration B1 (truncated to four levels)

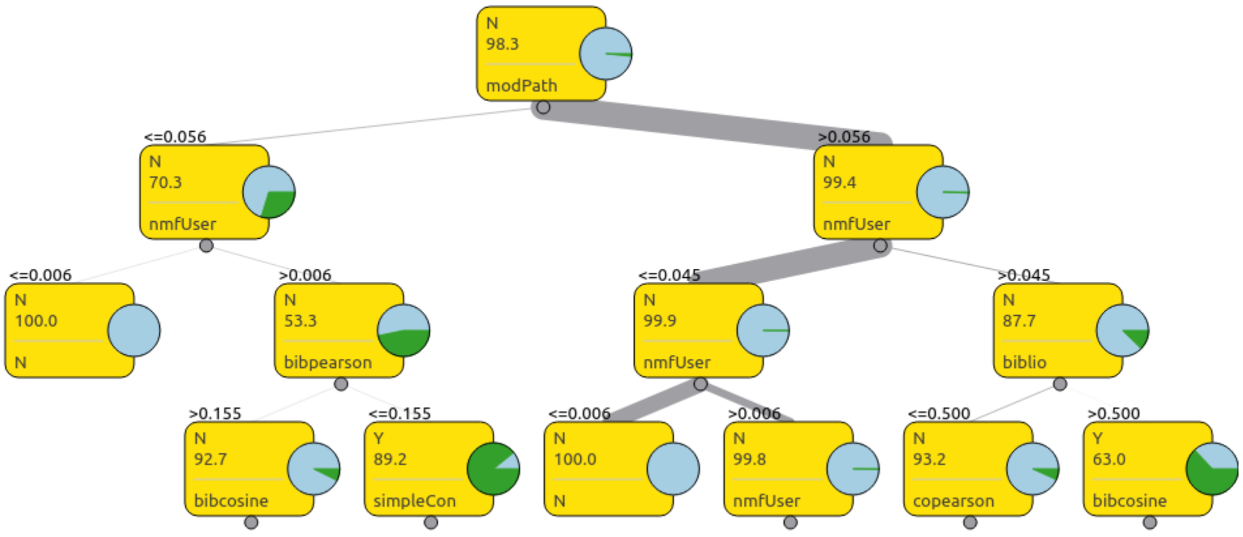


Figure 19: Tree produced by configuration B2 (truncated to four levels)

10.1.2 Matrix factorisation and clustering

Non-negative factorisation can be viewed in a number of ways. One way, and possibly one of the explanations for the success of NMF in this application, is in NMF as a way to find patterns that underlie data. These patterns san be viewed as either a set of vectors and linear combinations of these vectors that predict a user's membership in the graph. Roughly, each user expresses a preference for different kinds of users, and these vectors form the rows of W. Then, the rows of H are expressions of the properties of each of the members of the graph. The product of W and H then approximates the social graph. In approximating it it is able to eliminate some idiosyncratic connections. These idiosyncratic connections could be either connections that exist but should not (unexpected friendships) and connections that do not exist but should (candidates for recommendation).

Matrix factorisations are often forms of clustering, often more flexible than K-means, because they can form more complex hypotheses about group membership. These hypotheses are linear combinations of the vectors in H that set the position of a particluar item in the high-dimensional space of the adjacency matrix. K-means only allows for a group membership based on proximity to the centroids that it finds.

As a form of unsupervised clustering, NMF requires that the user set the number of clusters, as the inner dimension provided to the algorithm. The more clusters that the algorithm is provided, the more closely the approximated matrix gets to the adjacency matrix. This means that the choice of inner dimension can be critical to the performance of the factorisation in predicting links. The particular matrix factorisation chosen does not have good computational performance for large inner dimensions, which creates a trade-off between accuracy of prediction and computing time.

There will also be a trade off between accuracy and generalization. With higher and higher inner dimensions, the factorised matrix becomes a better and better clustering for groups, but its performance as a predictor for non-existent connections will peak then fall, as the now more accurate matrix factorisation fails to generalize the clusters it finds to useful results.

The performance of matrix factorisation jumped substantially when a limited subset of the graph was considered, instead of the entire social graph. This may be because of a poor choice of inner dimension: if the clusters are not specific enough, the factorisation will overgeneralise, providing no useful information. While this may be the case, there was no improvement in the quality of results for the full social graph even with a very large inner dimension. An inner dimension of three yields good results for the local graph. The overall graph is only twice as large in terms of nodes as the local graphs (which were all about the same size). When NMF was run with inner dimensions 10 and 20 there was no appreciable improvement in performance on the global graph. It is possible that even more clusters are needed, though this is probably not the case.

The poor performance may also be an artifact of the way that the graph was built. The test users were chosen to be reasonably well connected and central. This means the subset that is centered around those users will be denser than the overall graph. In comparison, the global graph will be harder to cluster or factorize because it will be primarily dangling nodes. Dangling nodes do not present much "information" to the machine learning algorithms, because their dangling is not the result of a genuine preference on their part, but rather the result of the way that the dataset was built.

This is confirmed by the density of the adjacency matrix for all of the test user-centered graphs, which is 0.018 percent, while the density of the overall graph is 0.0093 percent, a difference of a factor of two. See Figure 20 for an example of this artefact. Note that the sizes of the two types of graph differ by a factor of two as well, *suggesting strongly that most non-included users are dangling ones* that have not been explored. This may be the reason for the poor performance.

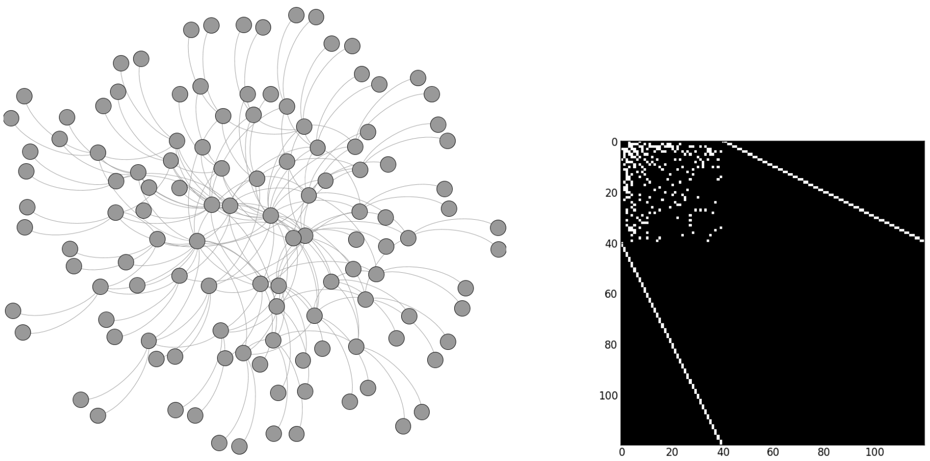
10.2 Social analysis

The results of the social analysis step were not as good as expected, making them less useful for the filtering of the lists produced in the link prediction step.

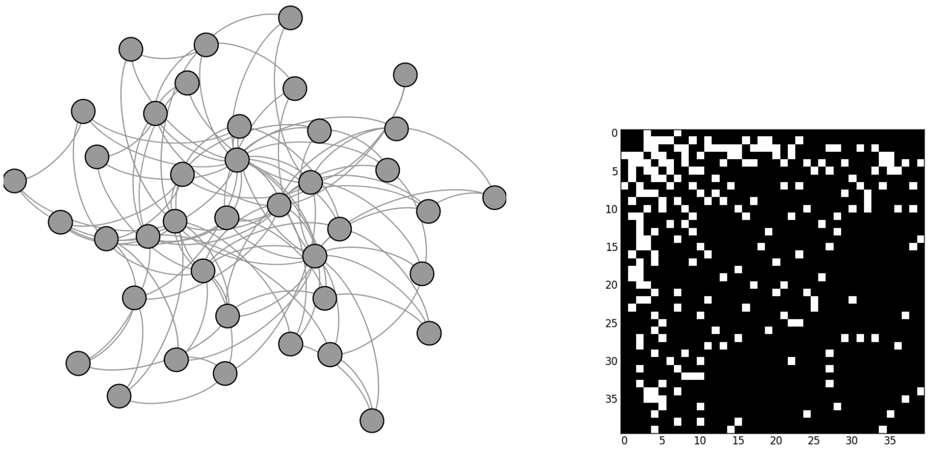
In general, the social analysis step was based on the assumption that there is a significant difference between the way that socail users and celebrities act. This may not be the case, or the difference may not be as large as was first hoped.

It was initially thought that there were three types of Twitter users:

- Social users: who use Twitter to communicate with friends and family.
- News sources: who exist to broadcast news to their followers.
- News readers: who subscribe to news sources for information.



(a) Graph and adjacency matrix of the full social graph. Note the many unconnected components.



(b) Graph and adjacency matrix of a subset of the social graph.

Figure 20: This is an example of the odd structure of the data, built from a random model with edge-nodes added (note that this is an undirected model). The outer nodes of a are only connected to one other node. This is due to the way the data was mined.

It was first believed that these categories were reasonably separate, which would make their classification quite easy. In reality, there are many users who are less distinct. For example, there are some users that clearly exist to broadcast news, but who also have interactions with their followers that are obviously more social in nature. For example, a musician's Twitter account may aim to promote the musician to their fans, but this happens through a mix of information and social interaction. This is motivated by a belief amongst marketing experts that social interaction builds a stronger following than pure information dissemination.

10.3 Important factors

Of the factors that were used in the model, some of the more important ones were:

1. inverse of the number of '`@`' symbols used
2. number of friends
3. number of retweets
4. linguistic novelty feature.

Often these were not correlated in the way I expected. Of the features mentioned above, I was surprised by the first and the third. I expected social users to tweet at their friends reasonably often, and to have less original content (and more retweets) compared to news providers, but this appears not to be the case.

As expected, people with many friends are more likely to be new providers, as they tend to try to collect followers; befriending others assists with this.

Also, as expected, social users tend to use more diverse words than news providers, since news providers are generally themed in some way. For example, President Barack Obama's Twitter feed currently repeats many of the same words very often, such as "budget", "election" and "president". By contrast, social users tend to tweet on diverse topics.

11 Conclusion

Using the methods I described earlier, I have found an effective way to perform link prediction in social networks. This link prediction is based primarily on structural factors, which makes it an effective predictor for social relationships. Matrix factorisation methods have been shown to be useful in this problem, but there are still some unanswered questions about the best ways to apply them to the problem. Even so, the system outlined above is relatively simple, and could probably be improved through further tweaks or extensions and through a larger data set.

For the project I implemented several powerful features over the social graph:

1. community detection
2. cocitation and Bibliographic coupling
3. cosine and Pearson similarity
4. non-negative matrix factorisation.

These turned up to be an effective set of features for link prediction, especially the matrix factorisation, which appears to be quite effective in inferring missing connections. The results of these classifiers would, on their own, be sufficient for implementation into social networking systems that wish to provide recommendations for their users.

My social/non-social classifer also achieved some accuracy, and would be useful in conjunction with the link prediction for providing specific types of recommendations. This would be most useful in situations where the people running the social network are trying to cultivate a particular culture or set of use patterns on their network, and part of that was either a tendancy for people to form social connections or a tendancy for news connections.

My results also reinforce the importance of the social graph, and the power that it has in predicting or explaining people's interactions over social networks.

References

Marcelo G. Armentano, Daniela Godoy, and AnalÃa Amandi. Recommending information sources to information seekers in twitter. 2011.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008+, October 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/P10008. URL http://dx.doi.org/10.1088/1742-5468/2008/10/P10008.

Leo Breiman. Random forests – random features. Technical Report 567, Statistics Department, University of California, Berkeley California, September 1999.

Cara Pring. 99 new social media stats for 2012. http://thesocialskinny.com/99-new-social-media-stats-for-2012/, May 2012.

Tomaž Curk, Janez Demšar, Qikai Xu, Gregor Leban, Uroš Petrovič, Ivan Bratko, Gad Shaulsky, and Blaž Zupan. Microarray data mining with visual programming. *Bioinformatics*, 21:396–398, February 2005. ISSN 1367-4803. URL http://bioinformatics.oxfordjournals.org/content/21/3/396.full.pdf.

Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004. ISSN 1046-8188. doi: 10.1145/963770.963776. URL http://doi.acm.org/10.1145/963770.963776.

Sandra Garcia Esparza, Michael P. O'Mahony, and Barry Smyth. Mining the real-time web: A novel approach to product recommendation. *Knowledge-Based Systems*, 29(0):3 – 11, 2012. ISSN 0950-7051. doi: 10.1016/j.knosys.2011.07.007. URL http://www.sciencedirect.com/science/article/pii/S095070511100147X. Artificial Intelligence 2010.

Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why We Twitter: Understanding Microblogging Usage and Communities. *Procedings of the Joint 9th WEBKDD and 1st SNA-KDD Workshop 2007*, August 2007.

H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008. doi: 10.1137/07069239X. URL http://epubs.siam.org/doi/abs/10.1137/07069239X.

Younghoon Kim and Kyuseok Shim. Twitobi: A recommendation system for Twitter using probabilistic modeling. *Data Mining, IEEE International Conference on*, 0:340–349, 2011. ISSN 1550-4786. doi: http://doi.ieeecomputersociety.org/10.1109/ICDM.2011.150.

Shih-Ta Kuan, Bang-Ye Wu, and Wan-Jui Lee. Finding friend groups in blogosphere. In *Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 1046 –1050, march 2008. doi: 10.1109/WAINA.2008.156.

Ludmila I. Kuncheva. On the optimality of naÃŕve bayes with dependent binary features. *Pattern Recognition Letters*, 27(7):830 – 837, 2006. ISSN 0167-8655. doi: 10.1016/j.patrec.2005.12.001. URL http://www.sciencedirect.com/science/article/pii/S0167865505003582.

Chih-Jen Lin. On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *Neural Networks, IEEE Transactions on*, 18(6):1589 –1596, nov. 2007. ISSN 1045-9227. doi: 10.1109/TNN.2007.895831.

Hao Ma, Tom Chao Zhou, Michael R. Lyu, and Irwin King. Improving recommender systems by incorporating social contextual information. *ACM Trans. Inf. Syst.*, 29(2):9:1–9:23, April 2011. ISSN 1046-8188. doi: 10.1145/1961209.1961212. URL http://doi.acm.org/10.1145/1961209.1961212.

Jeffrey Naruchitparames, Mehmet Hadi Gunes, and Sushil J. Louis. Friend recommendations in social networks using genetic algorithms and network topology. In *IEEE Congress on Evolutionary Computation'11*, pages 2207–2214, 2011.

M. E. J. Newman. *Networks*. Oxford University Press, New York, 2010a.

M. E. J. Newman. *Networks*. Oxford University Press, New York, 2010b.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120.

J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, March 1986. ISSN 0885-6125. doi: 10.1023/A:1022643204877. URL http://dx.doi.org/10.1023/A:1022643204877.

S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. pages 109–126, 1996.

Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986. ISBN 0070544840.

Twitter Inc. Twitter terms of service. `https://twitter.com/tos`, May 2012.

D.J. Watts and S.H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393: 440–442, June 1998a. doi: 10.1038/30918.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of ’small-world’ networks. *Nature*, 393(6684):440–442, June 1998b. ISSN 0028-0836. doi: 10.1038/30918. URL `http://dx.doi.org/10.1038/30918`.