Social & Information Network Analysis

CS 224W

Final Report

Alexandre Becker
Jordane Giuly
Sébastien Robaszkiewicz

Stanford University
December 2011

1 Introduction

The microblogging service Twitter today encompasses more than 200 million users, who send more than a billion tweets every week. A tweet is a short message that contains at most 140 characters. It can include one or several *hashtags* such as "#CS224W". The hashtags enable to easily search messages on a content, look at trends, or precise the topic of a given message. Given the short length of messages, these hashtags indeed provide an insightful information on the topic of a tweet.

In order to characterize user habits and interests, one can try to identify what are the hashtags that best describe him, and more generally what hashtags he is likely to use.

The goal of our project is to predict what hashtags are likely to be used and analyze the influence of hashtag categories in this prediction. More precisely, our project has consisted in:

- building a bipartite graph between users and the hashtags that they have used
- implementing a link prediction algorithm on the bipartite graph between users and graph
- implementing a clustering algorithm on the hashtag graph, obtained by projecting the initial bipartite graph on the hashtag nodes
- combining the hashtag clustering with the link prediction and observing how this affects the link prediction precision
- analyzing the link prediction precision on the bipartite graph between users and hashtag clusters

As an application of our results, one could predict the trends of hashtags and hashtag categories, *i.e.* forecast what are the hashtags that people are the most likely to use in the future.

2 Methods

2.1 Data preparation

The data on which we will be working will be extracted from a collection of Tweets, containing the time, sender name and tweet, collected from June 2009 by J. Yang and J. Leskovec [1].

1. We modeled our dataset on Twitter into a bipartite weighted graph with U user nodes and H hashtag nodes. An edge (u, h) in $U \times H$ has a weight equal to the number of times the user u has sent a tweet containing the hashtag h.
2. We preprocessed hashtags by removing special characters, and deleting hashtags whose lengths are below 2.
3. Instead of considering all hashtags, we limited our dataset to the 1,000 most sent hashtags for several reasons:

- the top sent hashtags reflect general trends of interest among users
- most hashtags do not have any english meaning, and can be considered as noise
- if we were an advertising company who would try to extract information from Twitter, focusing our attention on the most mentioned subjects makes more sense.

4. Then we reduced our set of users to the top 10,000 senders of these 1,000 hashtags in the past. Again, it makes no sense to consider users who never used our set of hashtags, and in order to reduce computational cost, we focused only on the top 10,000 senders.

5. From this choice of hashtags and users, we derive three smaller datasets:

   - 100 hashtags; 1,182 users; 10,047 edges; 361 missing edges;
   - 100 hashtags; 2,571 users; 30,081 edges; 1,154 missing edges;
   - 1000 hashtags, 1,967 users; 30,001 edges; 1,240 missing edges;

6. For each data set, we removed the edges that appeared the most recently. Our link prediction algorithm works on predicting these removed edges.

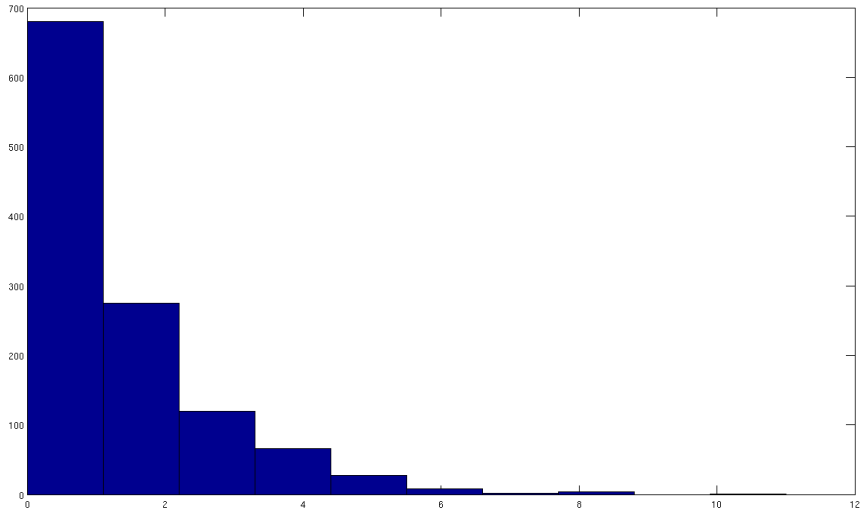

Figure 1: Number of different hashtags posted by users.

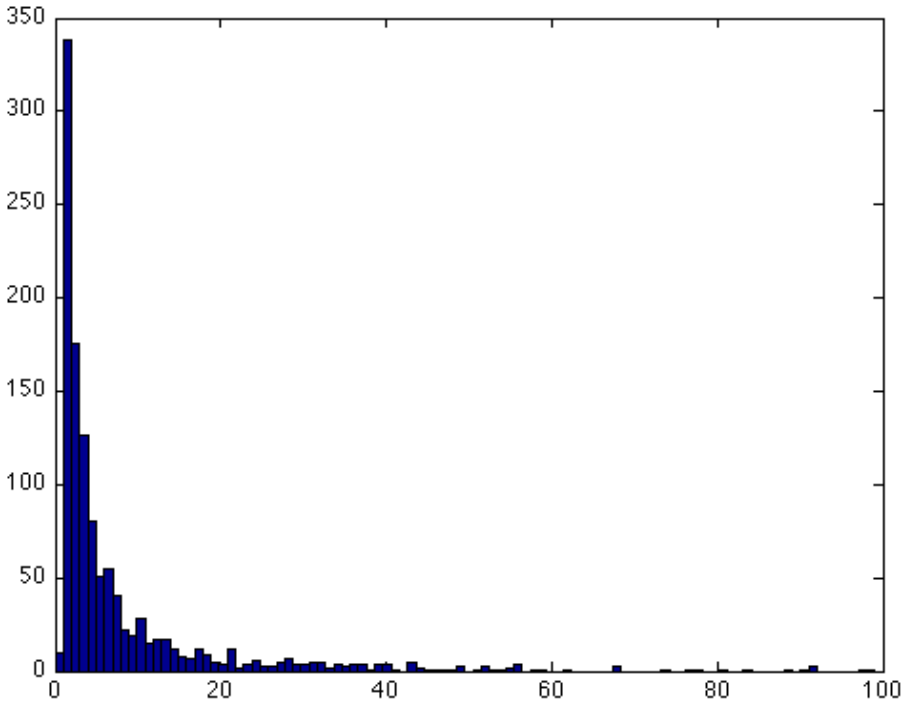

Figure 2: Degree distribution of the users.

2.2 Link Prediction

Given the dataset, our analysis then focused on Link Prediction. In other words, the problem that we addressed can be stated as: given a set of previous edges (user, hashtag), what are the most likely hashtags to be tweeted and by who?

2.2.1 Prior work

The link prediction problem has been widely studied and our analysis relied on this legacy. In particular, our approach combined methods from [2] and [3], as we explain below.

In Liben-Nowel & Kleinberg's article [2], the authors predict links that are likely to appear in the future using different manners to measure the proximity of nodes in a network. More precisely, they assign scores to every possible link and these links are then ranked by their score in decreasing order. Only the edges with the best score will form the set of predicted edges.

Unfortunately, their approach cannot be transposed to our study, as the tests and experiments of this paper are done on large co-authorship networks, and not on a bipartite

graph as in our case. For that reason, we analyzed the method presented in [3], which present a link prediction approach to bipartite graphs called collaborative filtering.

It consists in computing a score between a user u and a hashtag h, for every pair (u, h). The score is based on the neighbors of u (which belong to the set of hashtags) and their similarity to h, as we will describe later.

However, the algorithm presented in [3] is used for a recommendation problem, whereas we would like our algorithm to address a link prediction problem. Therefore, we had to combine the two approaches presented in [2] and [3]. Furthermore, both of these articles deal with unweighted graph and, for that reason, we developed our own way of predicting edges in a weighted graph.

2.2.2 Our model

Building on the articles previously presented, the method we developed consists of the different steps below:

1. Compute the projection of the bipartite graph in the set of hashtags. The result, called **projection matrix**, indicates the similarity between hashtags
2. Compute the score of every (user, hashtag) edge, based on the similarity between a hashtag with the hashtags already linked to a user
3. Computing the weight associated with each of these scores
4. Ranking the edges by decreasing weight and output the best k edges.

We will now present each of these points more in depth.

2.2.3 Hashtag similarity

The similarity between two hashtags is inferred by projecting the bipartite graph between users and hashtags into a graph with hashtag nodes. The method is presented in [3] and consists in computing a weight between two hashtags according to a certain score.

More precisely, the weight may be defined as the number of (top) neighbors that two hashtags u and v have in common in the bipartite graph. This weight is called the *sum* metrics and is defined as:

$$\sigma(u, v) = |N(u) \cap N(v)|.$$

Besides, we have also used the *Jaccard* coefficent, presented in [2]:

$$\gamma(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}.$$

2.2.4 Score of user-hashtag pair

If the similarity between two hashtags is defined as ω, the score between a user u and a hashtag h is then:

$$S(u, h) = \sum_{i \in N(u)} \omega(h, i).$$

2.2.5 Weight of predicted links

The next step consists in computing a weight associated with a specific edge. We have taken two approaches:

1. Uniform weight of 1
2. Linear function of the score:

$$\text{weight} = \max(1, \lfloor a_1 * \text{score} + a_0 \rfloor),$$

where a_0 and a_1 are parameters.

The rationale behind the second approach is that, as we are using a multi-edge graph, a (user, hashtag, score) tuple with a very high score is likely to correspond to an edge used multiple times. The need for such a method comes from the fact that the score can differ by several orders of magnitudes, and because, as presented below, the predicted edges often have a weight larger than 1.

2.3 Clustering

Clustering hashtags could bring improvement to our predictions in two ways. First, if we consider a hashtag h and a cluster C in which it appears, a link between h and a user is more likely to appear if the user has used several hashtags from C before. Secondly, instead of evaluating scores for every edge from h, we would preferably consider edges with users who have sent hashtags that are in C.

2.3.1 Prior work

The problem of graph clustering, intuitive at first sight, is actually not well defined. The main elements of the problem themselves, *i.e.* the concepts of community and partition, are not rigorously defined, and require some degree of arbitrariness and/or common sense. In the literature, we can see two main trends for defining communities: local and global.

Local Communities are parts of the graph with a few ties with the rest of the system. To some extent, they can be considered as separate entities with their own autonomy. So, it makes sense to evaluate them independently of the graph as a whole. Local definitions focus on the subgraph under study, including possibly its immediate neighborhood, but neglecting the rest of the graph.

Global definition Communities can also be defined with respect to the graph as a whole. This is reasonable in those cases in which clusters are essential parts of the graph, which can not be taken apart without seriously affecting the functioning of the system, which is the case in social networks. The literature offers many global criteria to identify communities. The most popular model, the modularity, has been introduced by Girvan and Newman [4]. It is based on the idea that a random network is not expected to have community structure, so the possible existence of community structure of a given network is revealed by comparison between the number of intra-community edges in this given network and that number in a

random network. Though modularity suffers from the resolution limit problem, it has been widely accepted as a *de facto* standard. The **modularity** is defined by:

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - P_{ij}) \delta(C_i, C_j),$$

where A is the adjacency matrix, m the number of edges, P_{ij} the expected number of edges between the vertices i and j in the null model, and where δ is the Kronecker symbol (which equals 1 if $C_i = C_j$, *i.e.* i and j are in the same community, and 0 else). Then, the modularity can be understood as a function that evaluates the goodness of partitions of a graph into clusters.

However, testing an algorithm essentially means applying it to a specific problem whose solution is known and comparing such solution with that delivered by the algorithm. In the case of graph clustering, a problem with a well-defined solution is a graph with a clear community structure. This concept is not trivial, however. Many clustering algorithms are based on similar intuitive notions of what a community is, but different implementations. For that reason, it is hard to check the relevance of our clustering algorithm on our dataset in particular.

Then, when dealing with real networks, it is useful to solve their community structure with different clustering techniques, to cross-check the results and make sure that they are consistent with each other, as in some cases the answer may strongly depend on the specific algorithm adopted. However, one has to keep in mind that there is no guarantee that "reasonable" communities, defined on the basis of non-structural information, must coincide with those detected by methods based only on the graph structure.

2.3.2 Method & Algorithm

As we couldn't reasonably expect to implement a significant number of different clustering algorithms to verify the relevance of our clusterisation, we intend to implement the Fast Modularity Optimization that we saw in class. Indeed, after reviewing the performance of several clustering algorithms on a range of datasets, the Fast Modularity Algorithm appears to be the one that performs best in general; moreover, the bigger the graph is, the wider the gap is between this algorithm and the others. On top of that, due to the complexity of other algorithms (for example, the Girvan-Newman algorithm is $\mathcal{O}(n^3)$) this optimization is currently one of the few algorithms that can be used to estimate the modularity maximum on such large graphs: its complexity is $\mathcal{O}(n^2 \log n)$. All in all, the Fast Modularity Algorithm seemed to be the more relevant to use here.

Fast Modularity Algorithm

Before running the algorithm, we have to calculate the Modularity Matrix of the graph we want to cluster. As we would use the projected graph of hashtags (which is a multigraph), we had to take into account the weight/number of edges in the adjacency matrix. Thus, the coeficients of the Modularity Matrix B were given by:

$$B_{ij} = A_{ij} - \frac{s_i s_j}{2W},$$

Where s_k represents the strength of the vertex k (*i.e.* the sum of the weights of edges adjacent to the vertex) and W is the sum of all weights.

The Fast Modularity Algorithm is a recursive algorithm: it starts on the whole graph, tries to divide it into two clusters and, if this division is relevant—*i.e.* if the total modularity is increased—, starts the process on each one of these two clusters. The only argument needed is then the indices of B that belong to the current cluster we are working on (*i.e.* we are trying to divide into two smaller clusters).

Here is the algorithm, where a is an array of indices and $B[a, a]$ represents the submatrix of B made with the lines and the columns whose indices are in a.

```
1  fastModularity(a)
2      B' = B[a, a]
3      calculate the eigenvector u of B' associated to the greatest
           eigenvalue with the power method
4      for i = 1 to length(a)
5          if u(i) > 0
6              add the value a_i in the array a_+
7          else
8              add the value a_i in the array a_-
9      X = sum_{i,j in a} B_ij
10     Y = sum_{i,j in a_+} B_ij + sum_{i,j in a_-} B_ij
11     if Y > X
12         clusters = [fastModularity(B, a_+), fastModularity(B, a_-)]
13     else
14         clusters = a
```

Run

The Fast Modularity Algorithm was run on two adjacency matrices of the projected graph (sum and Jaccard), and we present the results in the following section.

2.3.3 Results

As we have seen above, it is difficult to measure the accuracy of a clustering algorithm. We tried to verify the performance of our algorithm with empirical measures: check manually the clusters on small datasets, and plot the distribution of the size of the clusters.

To do that, we used the smaller dataset (100 hashtags and 10,000 edges) we had and studied the results of the clustering algorithm over the two hashtag projections (sum and Jaccard).

	Sum	Jaccard
Number of clusters	23	16
Maximum size of a cluster	20	13
Minimum size of a cluster	1	1
Average size of a cluster	4.30	6.19
Median size	2	6

Table 1: Cluster statistics

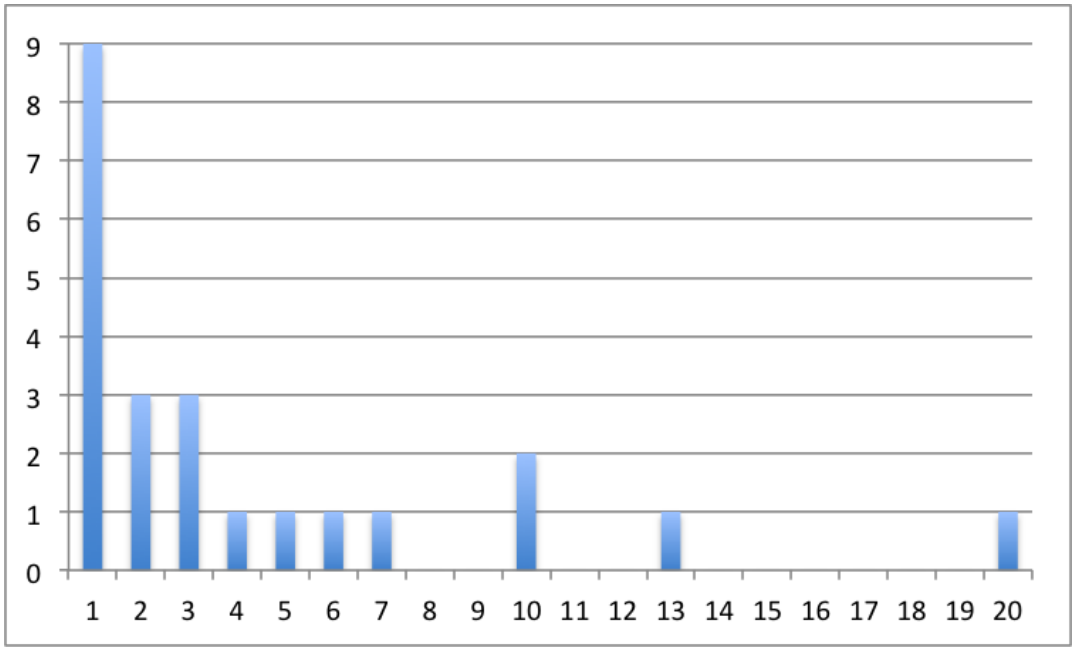

Figure 3: Cluster size distribution on sum projected hashtags.

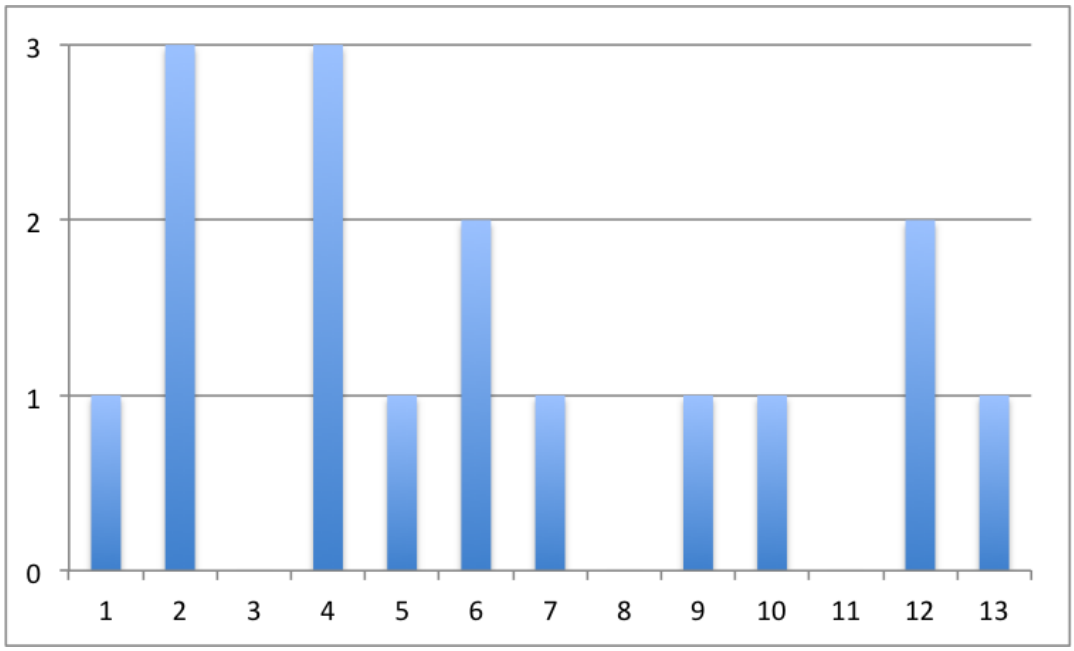

Figure 4: Cluster size distribution on Jaccard projected hashtags.

In both cases, we have encouraging results about the hashtags gathered in the different clusters. For instance, we have the cluster {free, firefox, web, blog, photoshop, webdesign, photog, mac, media, linux, film, science} with the Jaccard Adjacency Matrix [1]. However, and it seems (empirically) that the Jaccard Adjacency Matrix outperforms a bit the Sum Adjacency Matrix: its clusters seem to be more relevant.

We can also notice that in both cases, the distribution of the sizes of the clusters is broad (cf. Figures 3 and 4), and we can see that there are more clusters with a small size than with a bigger size. Also, the bigger cluster appears only once in both cases. This is consistent with what is observed in social networks. Social networks are usually made of one big cluster and several peripheral elements with a smaller size. Taking into account that the hashtags used by the users of a social network are likely to be representative of the social network in itself, our clustering distribution makes sense.

All in all, it seems that our clustering algorithm produced fairly good results on the hashtag network. This accuracy cannot be measured in a systematic manner but empirical measurements are satisfying.

2.4 Combining Link Prediction and Clustering

In order to combine Link Prediction and Clustering, we want to favour hashtags that belong to the same cluster. We then create the Clustering Matrix C where $C_{ij} = 1$ if i and j belong to the same cluster and $C_{ij} = 0$ else. We then have to add this Clustering Matrix to the

[1] For further details about the clustering, please refer to our Excel document in the ZIP file.

projected Adjacency Matrix when computing the LP algorithm. We can adjust the weight of this matrix with a β coefficient such that:

$$A' = A + \beta C.$$

We evaluate a credible value for β by assessing the average edge weight in A and the median value: β is chosen in order to have the same order of magnitude as the numbers in A. In order to assess the value of β that gives the best value, we also run a benchmark over β given the previous statistics.

3 Results and Analysis

3.1 Link prediction

3.1.1 Training parameters

In the section 2.2.5 on page 6, we explained that instead of assigning a uniform weight of 1 to all the predicted edges, we considered a method using a linear function of the score as the weight of the predicted edge:

$$\text{weight} = \max(1, \lfloor a_1 * \text{score} + a_0 \rfloor)$$

In order to determine the parameters a_0 and a_1, we have run the algorithm on a training set and determined the optimum parameters with a benchmark. The optimum parameters that we have found for the smallest dataset in the case of the Jaccard sum are: $a_1 = 0.06$ and $a_0 = 0.2$.

The following plot (Figure 5 on page 12) shows the scores against the future (real) weight among the predicted edges. Although it may not be obvious in the plot, using a linear function of the score does improve the precision, as shown in the precision results. Besides, this plot also illustrates the high propotion of false positives, *i.e.* edges predicted by our algorithm which aren't actually appearing in the future.

3.1.2 Precision results

We present in this section the precision of our methods on the different datasets

	Uniform weight of 1	Linear function of score
Sum	27.4%	30.2%
Jaccard	29.1%	36%

Table 2: Precision results on the dataset with: 1,182 users; 100 hashtags; 10,047 edges; 361 missing edges.

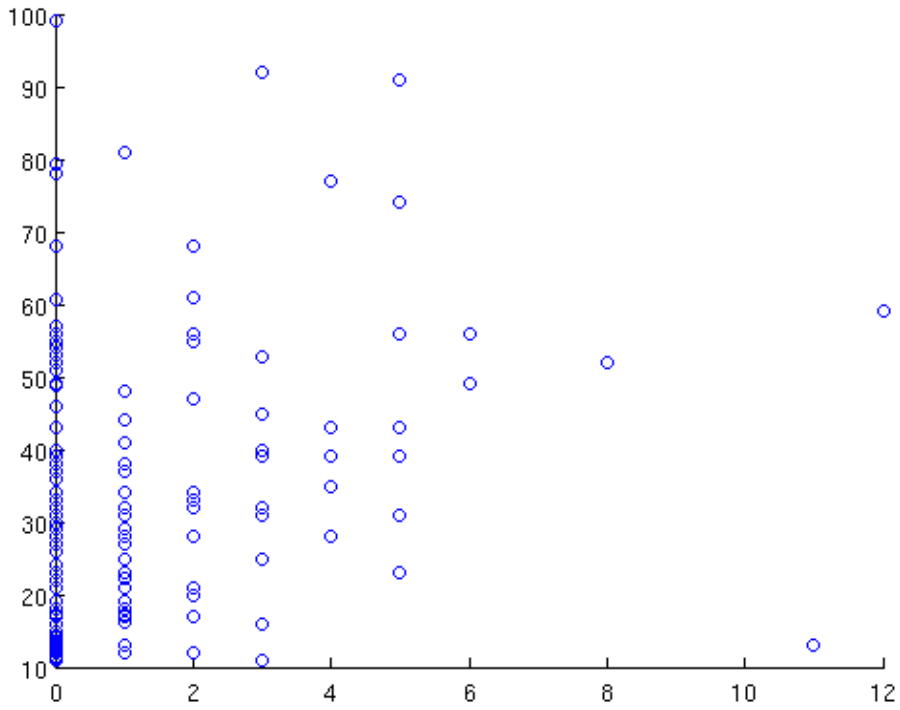

Figure 5: Scores by the future weight among the predicted edges the dataset with 100 hashtags and 361 missing edges. Out of the 196 distinct predicted edges, only 74 are actually future edges

	Uniform weight of 1	Linear function of score
Sum	23.3%	34.7%
Jaccard	24.5%	40.7%

Table 3: Precision results on the dataset with: 2,571 users; 100 hashtags; 30,081 edges; 1,154 missing edges.

	Uniform weight of 1	Linear function of score
Sum	17.3%	24.6%

Table 4: Precision results on the dataset with 1,967 users; 1,000 hashtags; 30,001 edges; 1,240 missing edges.

Our results are encouraging. Our approach in computing the predicted edge weights is validated by the fact that it does improve the precision, in some cases by 15%.

3.2 Link Prediction combined with Clustering

We computed, for various values of β, the accuracy of the link prediction helped by clustering information.

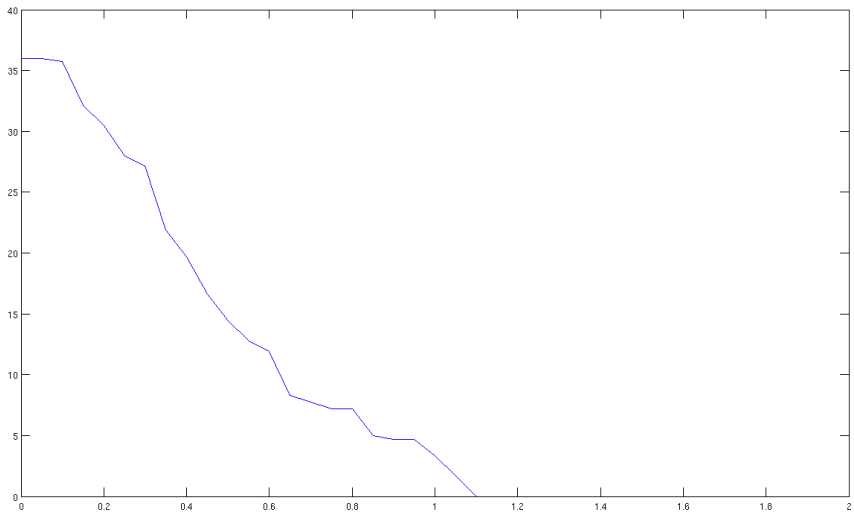

Figure 6: Precision of the Link Prediction algorithm combined with clustering. The x-axis is the coefficient.

From the past observations, we draw the conclusion that the clustering is very inefficient in increasing link prediction accuracy. We can give several interpretations to this unfortunate conclusion.

First all of, the way we add the clustering matrix to the projected matrix has the effect of changing hashtag neighborhood. But, the following histogram (Figure 7) shows that hashtag neighborhood before and after clustering are not significantly changed. Given the fact that the edge cost calculation between a user and a hashtag relies on the hashtag neighborhood, we can expect only a small difference in the final edge cost ranking.

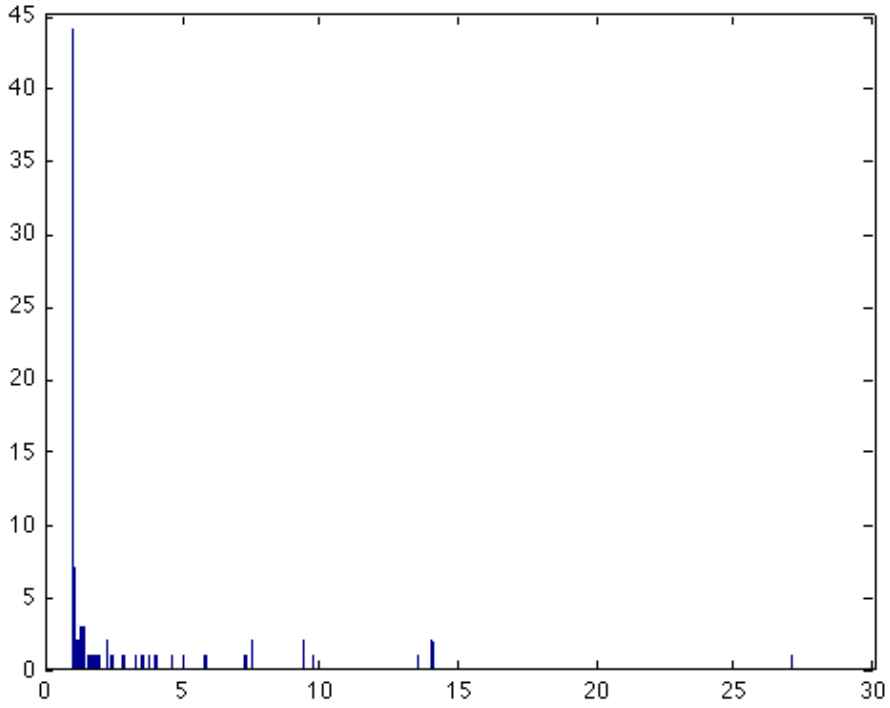

Figure 7: Ratio between the size of the neighbourhood ∪ cluster and the size of the neighborhood (step: 1000).

Because clustering brings minor changes to hashtag neighborhood in the projected matrix, we can conclude that clustering information is redundant with the information brought by our projection matrix. Besides, we even loose information, because all our clustering coefficient, *i.e.* the bonus to the edge cost between two hashtags, brought by the fact that they are in the same cluster, is constant, and do not change from one cluster to another. A further study would consist of keeping bonuses depending on the cluster sizes.

In our dataset, users send preferably hashtags in the same clusters. Therefore, we can assume that there is a preferential attachment phenomenon, where users tend to send

more hashtags from the cluster they used before. As a consequence, clustering may be more efficient for a recommendation problem. When we add clustering information to our projected matrix, we emphasise on links between a user and hashtags from the user preferred clusters. This reasoning would perfectly fit in a recommendation problem. However, in a link prediction problem, our current clustering information does not help us to decide between users. Because we think that keeping the symmetry between user and hashtags could be preferable in a link prediction problem, we consider computing a clustering algorithm on users, thanks to the network provided by tweet references and subscribtions.

Finally, we think that the degree distribution in our data set is not appropriate to benefit from all the information brought by hashtag clustering. Because more than 80% of our users are using less than two different hashtags, our network information is concentrated in edge multiplicities. We think that hashtag clustering would bring more information to a denser network, where most edges have the same weight.

3.3 Link Prediction on Clusters

The final part of our report presents the results of link prediction on clusters. Instead of considering edges between users and hashtags, we have merged all the hashtags from the same cluster in a single node. Therefore, we now consider edges from users to clusters, and try to predict which edges will appear next, *i.e.* to which clusters users will be more likely to post tweets.

Our results are the following:

Data set with: 1,182 users; 100 hashtags; 10,000 edges; 361 missing edges:

- $a = 0$, $b = 0$: 30.2%
- $a = 0.06$, $b = 0.2$: 36.0%

Dataset with: 2,571 users; 100 hashtags; 30,081 edges, 1,154 missing edges

- $a = 0$, $b = 0$: 25.0%
- $a = 0.06$, $b = 0.2$: 41.0%

Dataset with: 1,976 users; 1,000 hashtags; 30,001 edges, 1,240 missing edges

- $a = 0$, $b = 0$: 21.2%
- $a = 0.06$, $b = 0.2$: 37.0%

Because accuracy in both case (link prediction on hashtags and clusters) are really close to each other, we can draw the conclusion that cluster brings only little information on link prediction. Basically, we managed to improve, given a user, which edges he will be more likely to use in the future, but hashtag clustering does not bring any information on how to select among users the ones that will actually send a new tweet (and therefore create an edge we want to predict). The results of link prediction on clusters confirm the fact that, in order to improve accuracy, we need also more information about users.

We have saturated our algorithms with information on hashtags. But because link prediction does not make any difference between users and hashtags, we also need more information about users to improve link prediction accuracy.

4 Conclusion

The goal of our project was to predict the hashtags likely to be used in the Twitter network and test if the information from our hashtag clustering would increase the link prediction accuracy.

We firstly reduced the large graph of Twitter to a smaller set of interesting hashtags, and corresponding users. Then, we have separately run the link prediction over this set, and the fast modularity algorithm, to come up with a hashtag clustering.

Our prior results are encouraging. We managed to reach a satisfying level of accuracy for our different link predictions, and the clusters we get from the modularity algorithm tend to divide hashtags in pertinent topics.

Then, we tried to include the information brought by the hashtag clustering to the link prediction algorithm. In this case, our results were a little disappointing: the accuracy does not significantly increase when the hashtag clusters are taken into account.

The network structure and the fact that the clustering information might be redundant are reasons that could explain our results.

As an improvement to our current work on link prediction in the Twitter network, we would therefore consider the following additional studies:

- Try hashtag recommendation on a larger dataset (with more hashtags).
- Take into account the information on the user network: we could for example add clustering on users, whether thanks to the follower/followee graph or thanks to a projected graph of @-mentions in the tweets.
- Try different measures for the projected graph since we have seen that Jaccard measure performed significantly better than the naive one.
- Try not to add a constant bonus to edges in the projected graph. For example, the bonus could depend on clusters (cluster size being the most straighforward parameter).
- Compute clusters with overlapping hashtags: we saw that our cluster algorithm produced good results in itself but was not very efficient for the combination with th slink prediction. Maybe a different clustering method would change the effect on link prediction.

References

[1] Jaewon Yang and Jure Leskovec. Patterns of temporal variation in online media. In *ACM International Conference on Web Search and Data Minig (WSDM)*, pages 177–186. Stanford InfoLab, 2011.

[2] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58:1019–1031, May 2007.

[3] Oussama Allali, Clémence Magnien, and Matthieu Latapy. Internal link prediction: a new approach for predicting links in bipartite graphs, 2011.

[4] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004.