Information fusion methods for the automatic creation of Twitter lists

Mengjiao Wang*, Donn Morrison and Conor Hayes

Digital Enterprise Research Institute IDA Business Park,
Co. Galway, Ireland
Email: mengjiao.wang@deri.org
Email: donn.morrison@deri.org
Email: conor.hayes@deri.org
*Corresponding author

Abstract: Many microblogging services provide tools that allow users to organise the people they follow into groups for easier information access and filtering. However, the uptake of these tools is low with a likely reason being that curating groups of followees is a time consuming task. This paper proposes methods for automatically clustering followees into groups so that users can use these groups as their user lists. As social microblogging services contain both textual content posted by users and directed followee relationships between users, members in the same list usually share common interests and/or have dense followee relationships. Under this assumption, this paper first applies separate content- and graph-based methods to cluster users. Next, we propose several novel information fusion configurations that combine textual and network features. We evaluate these approaches using both an offline evaluation and a user evaluation on datasets crawled using the Twitter API.

Keywords: document clustering; community detection; early fusion; late fusion.

Reference to this paper should be made as follows: Wang, M., Morrison, D. and Hayes, C. (2015) 'Information fusion methods for the automatic creation of Twitter lists', *Int. J. Social Network Mining*, Vol. 2, No. 1, pp.19–43.

Biographical notes: Mengjiao Wang has been a Master student in Digital Enterprise Research Institute in the National University of Ireland, Galway (Ireland). He received his Master degree in 2013. From 2006 to 2010, he has been an undergraduate student in the School of Software in Dalian University of Technology (China). His current position is a researcher at the SAP China Labs.

Donn received his BSc in Computer Science in 2000 at the University of Victoria, Victoria, British Columbia and MSc in Computer Science in 2005 at Massey University, Palmerston North, New Zealand. In 2011, he completed his PhD at the University of Geneva, Geneva, Switzerland. His thesis was entitled 'Latent variable modelling of user interaction in image retrieval'. His research interests are unsupervised machine learning techniques such as latent variable and topic models, user modelling, and collaborative filtering in information retrieval.

Copyright © 2015 Inderscience Enterprises Ltd.

Conor Hayes is a Lecturer in the IT discipline at NUI Galway and leader of DERI's Information Mining and Retrieval Unit (UIMR). He is a principal investigator on a number of DERI projects such as Clique, ROBUST, ADVANSSE, SociaLens as well as being a funded investigator in the new INSIGHT National Centre for Data Analytics.

This paper is a revised and expanded version of a paper entitled 'Early and late fusion methods for the automatic creation of Twitter lists' presented at Proceedings of the 1st International Workshop of Social Knowledge Discovery and Utilization (SKDU '2012) Workshop at IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM, Istanbul, Turkey, 26–29 August 2012.

1 Introduction

As a broadcast medium and communication platform, microblogging is becoming more and more popular around the world. It is an online service with the function of both social networking and information sharing that allows users to exchange short content such as textual messages, images or links. Users of microblogging services can follow their real-life acquaintances such as classmates and colleagues, as well as popular artists, scientists, politicians and celebrities. As users in social media follow more and more people, the information stream quickly becomes cluttered, which makes it easy to miss important information and opportunities to create engaging conversations. Many microblogging services provide features that allow users to organise contacts such that messages from specific groups of people can be highlighted rather than the whole information stream. For instance, the popular Twitter microblogging service provides a feature called *Twitter lists* that enables a user to organise other Twitter users into lists.

The membership information of such lists has proven useful for many data mining and machine learning applications such as curating messages into meaningful themes (Greene et al., 2011), inferring user characteristics and interests (Dongwoo et al., 2010). As the list members need to be manually provided by the user, creating and managing user lists is a time consuming task. For example, a user wants to group classmates. First he or she must find those users and then put them into the group one by one. We hypothesise that the required labour contributes to a low percentage of users using this feature and we believe that many more users would adopt the user lists feature if the lists could be created and recommended to them automatically.

Clustering techniques have been well studied in various literatures. There exist studies on clustering users into groups based on the properties of each user (Jain et al., 1999; Agarwal and Kempe, 2008). There also exist studies on clustering networked users into groups based on the connections between users (Fortunato, 2010; Lancichinetti and Fortunato, 2009; Lancichinetti et al., 2011). In microblogging services, various information sources like network structure and user content can be seen as a graph where users have attributes. A new challenge consists in combining content and network structure. Does content plus structure improve user clustering in microblogs? Addressing this question is important, as it gives us a better understanding of the information sources in order to design a user clustering method that can be used as a starting point for users to manage their followees.

The primary assumption in this work is that users create lists to filter information based two main criteria: content and relationships. The first is that a user will want to group his or her followees based on the content or topic of their messages (Dongwoo et al., 2010). For example, a user interested in music may create a list containing users who are well known in that community and tweet often about that topic. The second criterion is based on the notion of relationship or membership. For example, a user might create a list that contains members of family, workplace or sport club and there is a high likelihood that the members of this list will also follow each other.

In this work we formalise these assumptions and propose methods for automatically creating users lists. To this end, we apply the non-negative matrix factorisation (NMF) method based on named entities extracted from tweets and the order statistics local optimisation method (OSLOM) (Lancichinetti et al., 2011) for community detection based on the followee graph. We then propose early and late fusion methods to combine content features and the followee graph.

2 Related work

In this section, we classify microblog user clustering methods into three main categories based on the information sources these methods used:

1 content-based which is a data clustering problem treating content as a feature vector of user

2 structure-based which is a community detection problem finding community structure from network of followee relationships

3 fusion-based which combines both content and network structure.

2.1 Content-based

User clustering based on content is related to the area of document clustering as we aggregate all messages from a user into one document and represent a user with his or her corresponding document. Recently, much research has focused on document clustering for use in different areas of text mining and information retrieval, such as improving the performance of search engines by clustering web pages (Zeng et al., 2002). Methods based on *k*-means (Steinbach et al., 2000), hierarchical clustering (Fung et al., 2003) and NMF (Lee and Seung, 2001) are some of the most commonly used methods for document clustering. Simplicity and speed are the main advantages of k-means when applied to large datasets. Hierarchical clustering usually has better clustering quality than *k*-means, but it is limited by its quadratic time complexity (Steinbach et al., 2000). NMF learns latent topic vectors of each documents and the clusters are assigned directly according to latent topic vectors. The main advantage of NMF is that it provides a direct representation of topics in user content. Karandikar (2010) applied a topic model approach for Twitter user clustering. They manually chose 21 well known Twitter users across seven different domains. They then generated topic models from unlabelled tweets posted by those users and proposed a way to cluster them using the topic distributions of each user with the *k*-means method. Experimental results showed that most of the clusters were accurately found. While this method is not appropriate for user list creation because

users that are to be clustered in this method should be carefully selected to make sure that most of their messages are related to their own domain, while user list creation targets followees of any given user and those followees typically talk about multiple topics. In other words, this method is suited for users whose contents are quite distinct from each other. Furthermore, user list creation is a method that seeks to support soft clustering (i.e., users are assumed to post about multiple topics) while *k*-means is a hard clustering method that does not support overlapping clusters.

2.2 Structure-based

User clustering based on network structure is related to the research area of community detection (Newman, 2004; Lancichinetti et al., 2011; Lancichinetti and Fortunato, 2009). Communities are groups of vertices within which connections are dense but between which they are sparser (Newman, 2004). The followee relationships in microblogs can be extracted and represented as a network upon which community detection algorithms find groups of densely connected users. An example is the graph cuts algorithm (Hao and Orlin, 1992; Chekuri et al., 1997). In graph theory, a cut of a graph is a partition of the vertices and a minimum cut is the partition whose size or weight is not larger than any other cut. There are several algorithms to find minimum cuts of a graph. Hao and Orlin (1992) proposed one of the fastest algorithm based on max-flow. A graph can be divided into communities by iteratively applying the minimum-cut. The advantage of the minimum cut method for community detection is that it is fast and easy to implement. However, it requires a pre-determined number of cuttings which makes this method less flexible. Girvan and Newman (2002) proposed another commonly used method for finding communities by removing edges with maximum betweenness centrality. They defined the betweenness of an edge as the number of shortest paths between pairs of vertices that run along it. This method tends to converge slowly as it re-calculates betweenness of each remaining edges after removing the one with maximum value. In general, the Girvan-Newman method takes $O(m^2n)$ time on a network of n vertices and m edges. Another widely used method for community detection is modularity maximisation (Agarwal and Kempe, 2008; Cafieri et al., 2011). Modularity measures the strength of division of a network into modules (also called communities or clusters). A high modularity network has dense connections within modules but sparse connections between modules. Theoretically, the modularity maximisation method first enumerate all possible divisions of a network and then calculates modularity for each division. Practical ways to find the division with highest modularity is based on optimisation techniques such as greedy algorithms or spectral optimisation (Fortunato, 2010). For the network of microblog, Greene et al. (2011) proposed methods for creating user lists around news stories. Given a small seed list of manually curated users, the methods expand this set by using friend-, mention-, retweet- and co-listed-networks of seed users. The expanded user list provides a more comprehensive coverage of the story. Qin et al. (2012) proposed an algorithm for mining users' friends in microblog and dividing them into different social circles automatically based on the closeness of their relationships. While algorithm considers only followees with two-way relationship and it is not easy to be generalised to all followees.

2.3 Fusion-based

In the context of information fusion, there are some previous works that study the use of both content and social structure. Agarwal and Kempe (2008) investigated the problem of clustering Twitter users based on user interests. They calculated user similarity by linearly aggregating five different user similarity approaches which included similarity of tweet text, URLs, hashtags, followee relationship and retweeting relationship. The difference between their approach and ours is that they cluster users globally and our work clusters followees of a given user. Steinhaeuser and Chawla (2008) proposed a clustering method that combines user attributes and relations in cellular phone network. They calculated a similarity metric with user attributes and use it as a weight the corresponding edge. Afterwards, any community detection method can be applied on this weighted graph. This method is similar to the early fusion proposed by us, but their method does not add new edges whereas our approach does. Mei et al. (2008) proposed a framework for topic modelling with network structure by assuming that connected documents have similar topic distributions. Their method used a statistical topic model with a harmonic regulariser learned from network structure and can also be applied to community discovery. However, this method does not adapt to user list creation in microblogs as it returns only topical communities.

3 Methodology

In this section we present the methodology followed in this work. We first introduce the datasets collected. Next, we justify our assumptions for clustering users based on content and network structure. Then, we introduce the clustering methods which include the baselines and the information fusion approaches.

3.1 Datasets

To evaluate our work we crawled two datasets using the Twitter API. The first dataset was collected to evaluate the methods in the offline setting. The second dataset was used to evaluate the methods using in the user study.

3.1.1 Dataset 1 (offline)

Since Twitter uses an integer number for the user identifier, users were chosen by uniformly randomly sampling integers within the range 1 and 500,000,000. Next, we crawled the Twitter lists created by each user. While our techniques are not specific to a single language, for the evaluation in this chapter we focus on English language tweets. To filter non-English users, we used a language detection library (Shuyo Nakatani, Language Detection Library for Java) and retain only those tweets for which English is the detected language. For each sampled user, we crawl his or her Twitter lists and for each list, we crawl the list members, the followee relationships between members and recent tweets of each list member.

The crawl took place between 22nd April 2012 and 29th April 2012 and yielded a total of 2,311 Twitter lists, 67,083 list members, 2,022,372 followee relationships and 26,694,692 tweets from 490 sampled users (see Table 1). Most sampled users have less than ten Twitter lists (see Figure 1) and most lists have less than 100 members (see Figure 2). Twitter lists are overlapping sets because Twitter allows users to add a followee to more than one list. In our dataset, 64% of sampled users have overlapping lists. An overlapping member is a member that belongs to more than one list of a given sampled user. The percentage of overlapping members of a given sampled user is the ratio of the number of such members to the number of all followees of the given user. In the pre-collected dataset, the average percentage of overlapping members of the sampled users is 9.6% and most of them (72%) have less than 10% of overlapping members.

Table 1 General description of the offline evaluation dataset

Sampled users	490
Twitter lists	2,311
List members	67,083
Followee relationships	2,022,372
Tweets	26,694,692

Figure 1 #Lists per user

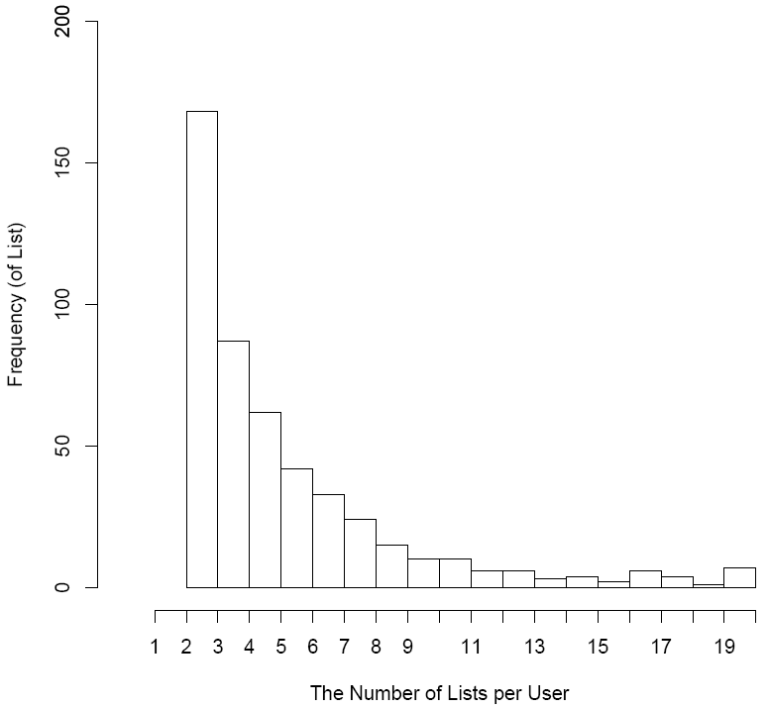
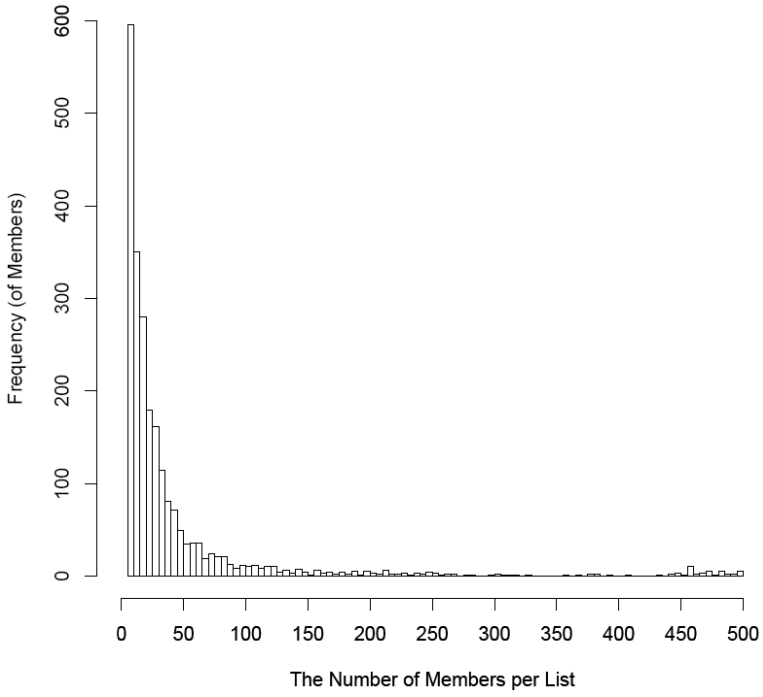


Figure 2 #Members per list



In our dataset, there are 3,386 members who have no tweets and there are 7,014 members (10%) that have less than 50 tweets. Content-based user clustering methods will perform poorly on user profiles where there is very little content information to use. There are also 1,561 (2%) members that do not follow or are not followed by other members in the same list. In other words, these members have no relationships to other members in the list; they are only followed by the users we randomly sampled for crawling. In our dataset, there are 402 (82%) users having at least one such members and also 12 (2%) users have more than 100 such member. Network-based user clustering methods will fail to find community structures for those users as they are unreachable on the network. Information Fusion methods cope with this incomplete information problem by combing information from different sources. For example, if a followee has no content but is reachable on the network, the method is still able to assign this followee to a list.

3.1.2 Dataset 2 (user study)

There are 31 volunteers within a research institute (in anonymisation is due to submission guidelines) participated in this evaluation. The crawl was carried out as above but using the user accounts of the 31 study participants and yielded a total of 10,099 followees, 704,484 followee relationships and 1,234,093 tweets. The average number of followees per participant is 325 and the average number of tweets per followee is 122.

3.2 Assumptions

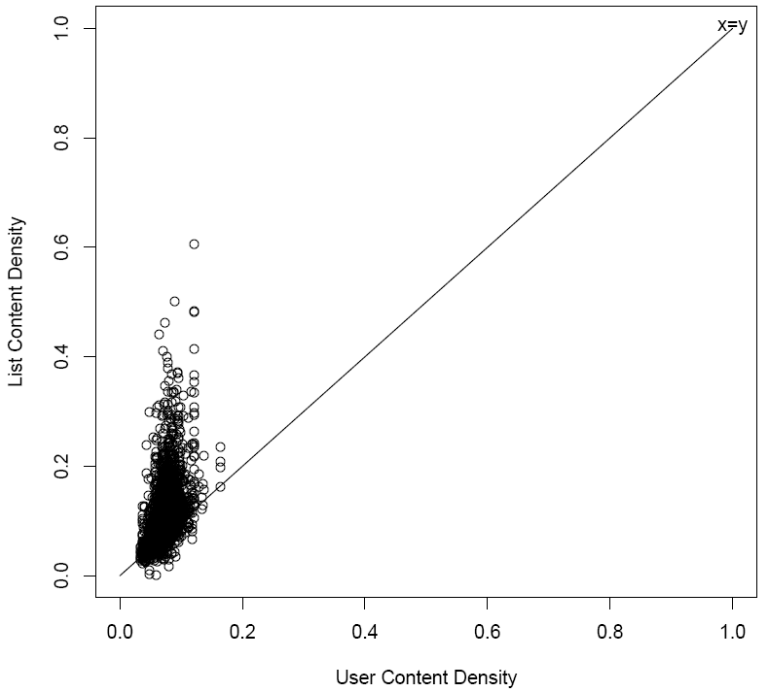
In the introduction we briefly outlined the assumptions that underlie this work. The first assumption is that users create lists of followees who post messages relating to a common topic. The second is that users create lists of followees who in turn tend to follow each other. Here we quantitatively justify these assumptions by examining the content and structure of Dataset 1. We calculate a cosine similarity matrix $\mathbf{S}_u \in \mathbb{R}^{N_u \times N_u}$ from the term-user matrix $\mathbf{V}_U \in \mathbb{R}^{M_t \times N_u}$, where M_t is the number of terms. Each element in $\mathbf{S}_u$ is in the interval of [0, 1]. We then represent the followee graph for user u as an unweighted adjacency matrix $\mathbf{A}_u \in \mathbb{R}^{N_u \times N_u}$, where N_u is the number of followees of user u and each binary element in $\mathbf{A}_u$ indicates whether there is a connection between two users.

3.2.1 Content density

To test if followees in the same list post similar content or not, we measure the similarity between followees based on named entities extracted from their content. First we calculate the cosine similarity matrix ($\mathbf{S}_U$) from the user-term matrix. Then we define content density (CD) as the mean cosine similarity of all followees of user $\mathbf{U}$:

$$CD(U) = \frac{\sum_{i \neq i \wedge i,j \subseteq U} S_{ij}}{n(n-1)}. \tag{1}$$

Figure 3 Content density of sampled users: 1,859 out of 2,311 points are above the line $x = y$, which shows that users in the same list are more similar in content to each other than to those in other lists



We measure two types of content density: user content density and list content density. *User content density* $CD(U_{all})$ is defined as the content density of all members in all the lists of a sample user and *list content density* $CD(U_l)$ is defined as the content density of all members in one list of a sample user. For a user list l, $CD(U_l) > CD(U_{all})$ means that users in l are more similar in content to each other than to those in other lists. Figure 3 shows the plot between user content density and list content density of each list in the pre-collected dataset.

3.2.2 Graph density

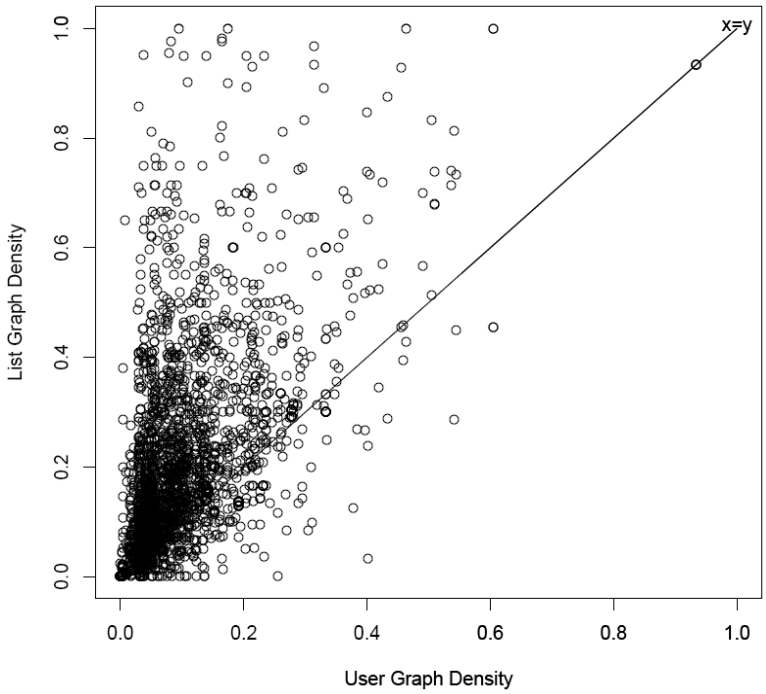
We posit that members from the same Twitter lists have denser connections. To measure this density, we use the definition of graph density (*GD* for a directed graph *G*:

$$GD(G) = \frac{|E|}{|V|\left(|V| - 1\right)}, \tag{2}$$

where $|E|$ is the number of edges in the graph and $|V|$ is the number of nodes in the graph of a single sampled user.

We define *user graph Gu* as the followee graph of all members in all the lists of a sample user. Similarly, we define *list graph* G_l as the followee graph of members only in list l. Thus, $GD(G_u)$ is the density of G_u and $GD(G_l)$ is the density of G_l.

Figure 4 Graph density of sampled users: 1,871 out of 2,311 points are above the line $x = y$, which shows that users in the same list have denser connections between each other than between those in other lists



For each sample user in our dataset, we calculate the *user graph density* and for each list of this sample user, we calculate its *list graph density*. For a user list l, $GD(G_l) > GD(G_u)$ means that users in the same list have denser connections between each other than between those in other lists. Figure 4 shows the plot between user graph density and list graph density of each list in the pre-collected dataset. 1,871 out of 2,311 points are above the line $x = y$ which indicates that users in the same list have denser connections within the list because the mass lies above the $x = y$ slope.

3.2.3 Mann-Whitney's U test

To further examine our assumptions, we perform a Mann-Whitney's U tests over 100 randomly sampled pairs of user content density and list content density. The reason for sampling is that Mann-Whitney test has a high computational complexity on large data. The hypothesis in this test is set to be that list content density is stochastically greater than user content density. The p-value (p-value = 4.957e–10) in this test indicates that the likelihood of the difference occurring due to chance is very low. Similarly, the p-value of Mann-Whitney's U tests over 100 randomly sampled pairs of user graph density and list graph density is 9.447e–09, which also suggests that list graph density is stochastically greater than user graph density.

3.3 Preliminaries

3.3.1 Hard clustering and soft clustering

According to clustering output, clustering techniques categorised into two types, *hard clustering* and *soft* or *fuzzy clustering*. In hard clustering, users are divided into distinct clusters, where each user belongs to exactly one cluster. In soft clustering, users will be assigned membership degrees and they can belong to more than one cluster.

3.3.2 Membership degree

To represent membership degree, we build a membership matrix named $\mathbf{M} \in \mathbb{R}^{M \times N}$ ($\mathbb{R}$ is the real number set) whose elements are numbers in the range [0, 1], and represent the degree of membership between users and clusters. Similarly, membership degree matrix of a hard clustering is a special case of soft clustering whose elements are either 1 or 0.

3.4 Baselines

Twitter lists are overlapping, meaning members can belong to multiple lists of a user. To address this, we use two baselines:

1 NMF (Lee and Seung, 2001) which is a soft clustering method that gives each user (represented by textual content) a latent semantic vector for list membership determination

2 the OSLOM (Lancichinetti et al., 2011) which detects overlapping clusters based on statistical significance of clusters accounting for edge direction and weight.

3.4.1 Non-negative matrix factorisation

NMF is a group of algorithms where a matrix $\mathbf{V_U}$ is factorised into two non-negative matrices *W* and *H*. There are several ways in which the *W* and *H* may be obtained. In this thesis, we apply the NMF proposed by Lee and Seung (2001) as it has been a popular method due to the simplicity of implementation. This method obtains an approximation of $\mathbf{V_U}$ by computing a (*W*, *H*) pair to minimise the Euclidean distance of the difference:

$$min \| \mathbf{V_U} - WH \|, \tag{3}$$

where $\mathbf{V}_U \in \mathbb{R}^{M_t \times N_u}$ is the user-term matrix. Each dimension of the row vector in matrix *W* is a base latent topic of the user, and each user is jointly represented by the base latent topics.

The iterative method to minimise the above function is as follows:

- initialise *W* and *H* with random non-negative values;
- iterate for each *k*, *j*, and *i* until convergence or after *l* iterations according to:

$$H_{kj} \leftarrow H_{kj} = \frac{\left(W^T \mathbf{V_U}\right)_{kj}}{\left(W^T WH\right)_{kj}} \tag{4}$$

and

$$W_{ik} \leftarrow W_{ik} = \frac{\left(\mathbf{V_U}^T W\right)_{ik}}{\left(W^T HH^T\right)_{ik}}, \tag{5}$$

where *k* is the number of topics and $0 \le i \le MU$, $0 \le j \le NU$.

Non-negative row vectors factorised by NMF can be directly used to find clusters in user collections. Similar techniques like principal components analysis and vector quantisation also learn basis vectors, but negative entries in the basis vectors of the latter methods make interpretation difficult. In this paper, we apply the nonnegative matrix factorisation as a baseline method to cluster users based on content. It accepts TF-IDF weighted user content matrix as input. The user latent topic matrix *W* returned by the NMF will be used to generate clusters.

3.4.2 Order statistics local optimisation method

Lancichinetti et al. (2011) proposed an algorithm called OSLOM that supports finding overlapping community structures in weighted and directed networks. It optimises locally the statistical significance of clusters with respect to a global null model during community expansion. The algorithm starts from a configuration model (Molloy, 1995), which is designed to build random networks with a given distribution of the number of neighbours of a vertex. To expand it, OSLOM calculates the significance of each neighbour in the current community and adds corresponding neighbours whose significance is higher than a given tolerance to the current community. OSLOM supports overlapping communities because each significant cluster is detected independently of the others and as such some vertices may belong to different clusters. This is desired

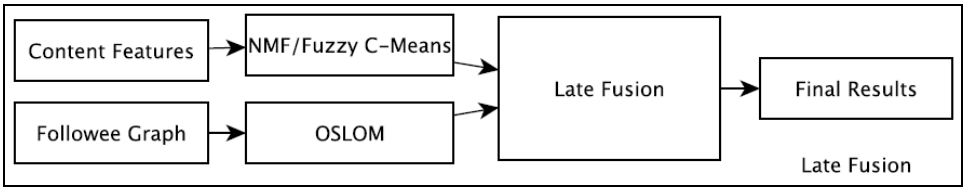
because user lists in microblogs are also overlapping, with some lists sharing followees for a given user. OSLOM can also be generalised to weighted and directed graphs by adding edge weight and direction to the cluster finding procedure. Homeless vertices are those followees that do not belong to any clusters. Although OSLOM allows homeless vertices by default, it can also assign each node to at least one cluster. The cluster is selected based on the significance score of the homeless vertices with respect to existing clusters.

In this paper, OSLOM is the second baseline method and it is used to compare with baseline methods based on content as well as information methods. The algorithm accepts directed edges as input and returns communities upon vertices.

3.5 Early fusion

The key idea in our early fusion approach is to augment the followee graph for a given user by adding weighted links between followees based on content similarity. The new denser graph obtained by this augmentation will contain both graph and content information. We then apply a community detection algorithm on the denser graph to obtain communities either sharing similar topics or having densely connected members. A work-flow of early fusion is shown in Figure 5.

Figure 5 Work-flow of early fusion method: early fusion first combines the feature spaces and then applies the OSLOM algorithm to obtain user clusters



We represent the followee graph for user u as an unweighted adjacency matrix $\mathbf{A}_u \in \mathbb{R}^{N_u \times N_u}$, where N_u is the number of followees of user u and each binary element in $\mathbf{A}_u$ indicates whether there is a connection between two users. We then calculate a cosine similarity matrix $\mathbf{S}_u \in \mathbb{R}^{N_u \times N_u}$ from the term-user matrix $\mathbf{V}_U \in \mathbb{R}^{M_t \times N_u}$, where M_t is the number of terms. Each element in $\mathbf{S}_u$ is in the interval of [0, 1].

The similarity values between two users are very small when they have little common content. A large number of small values in the followee graph will increase the complexity of the community detection. Thus, the early fusion does not consider small values in similarity matrix $\mathbf{S}_u$. An empirical threshold is used for content feature pruning. The threshold is selected according to the following steps:

1 sort elements in matrix $\mathbf{S}_u$ in descending order (as the similarity matrix $\mathbf{S}_u$ is symmetric, we only need to sort half of the elements)

2 take first p% of elements

3 set the value of the threshold to the smallest value in the first p% of elements.

In addition, the diagonal of $\mathbf{S}_u$ will not be considered in early fusion because the element in the diagonal is the similarity of a corresponding user with himself or herself.

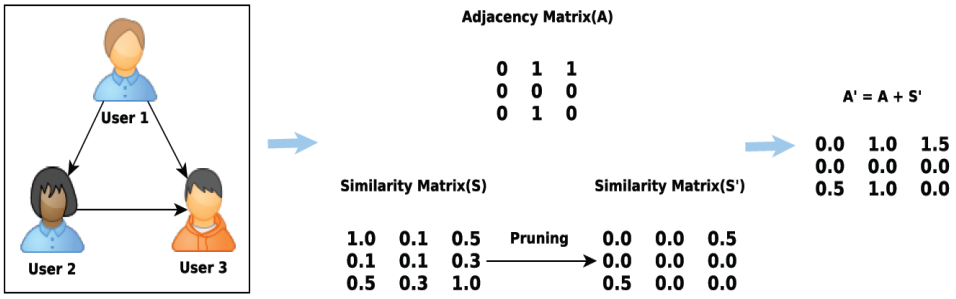
We merge content into the followee graph by linearly combining the content similarity matrix with the adjacency matrix of the followee graph. Where no followee relationship exists and the textual similarity between two users is greater than a threshold, a new edge is created with the weight equal to the corresponding element in $\mathbf{S}_u$. The steps of the edge weight assignment are:

1 normalise $\mathbf{S}_u$.: set $diag(\mathbf{S}_u) = 0$; set $\mathbf{S}_u(i, j) = 0$ if $\mathbf{S}_u(i, j) < Threshold$ (as $\mathbf{S}'_u$ is symmetric, for each $\mathbf{S}_u(i, j) < Threshold$, we add directed edges between i and j and also between j and i)

2 fusion: define new adjacency matrix $\mathbf{A}'_u = \mathbf{A}_u + \mathbf{S}'_u$.

Consider an example where we have three users in a network (see Figure 6). The similarity matrix is $\mathbf{S}_u$ and the adjacency matrix $\mathbf{A}_u$ can be obtained from user content and followee graph. Early fusion first removes values in the similarity matrix $\mathbf{S}_u$ under a given threshold, say 0.5 in this example, yielding the new similarity matrix $\mathbf{S}'_u$. Then, the new adjacency matrix $\mathbf{A}'_u = \mathbf{A}_u + \mathbf{S}'_u$.

Figure 6 Example of early fusion method on three users: (1) feature vectorisation, (2) feature pruning, (3) fusion stage (see online version for colours)



Finally, we run OSLOM on the new weighted graph, which is represented by the adjacency matrix $\mathbf{A}'_u$. The final communities can then be obtained by running OSLOM algorithm on matrix $\mathbf{A}'$.

3.6 Late fusion

For our late fusion approach, we combine clustering results of the content-based method and the network-based method into new clusters with a greedy algorithm. Suppose we have clustering results $\mathbf{C}_A = \{c_{A1}, c_{A2}, \cdots, c_{AI}\}$ and $\mathbf{C}_B = \{c_{B1}, c_{B2}, \cdots, c_{BI}\}$ generated by different clustering methods A and B, where c_{AI} and c_{BJ} are clusters with a number of users in it. Our greedy algorithm will merge $\mathbf{C}_A$ into $\mathbf{C}_B$ into a new result C = $\{c_1, c_2, \cdots, c_K\}$ with the following steps:

For each cluster c_{AI} in $\mathbf{C}_A$

a find c_{AI}'s most similar cluster c_{BJ} in $\mathbf{C}_B$

b merge c_{AI} into c_{BJ} to get a larger cluster c_K

c add c_K to $\mathbf{C}$.

The outcome of the late fusion approach are a set of merged clusters from both C_A and C_B. We use the Jaccard index (other similarity measures such as Euclidean distance and cosine similarity are also applicable) to measure cluster similarity which is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{6}$$

Two types of configuration are possible with this greedy algorithm:

a merging content-based results into network-based results

b merging network-based results into content-based.

We investigate both configurations in the offline evaluation.

4 Evaluation and results

This section presents the results of the baselines and fusion approaches introduced in the previous section for two evaluation scenarios:

1 the offline evaluation, where the automatically created lists are compared against ground truth lists

2 the user evaluation, where users are presented with their clustered followees and asked to adjust the lists until they are satisfied.

4.1 Offline evaluation

We conducted the offline evaluation to measure and compare the accuracies of the proposed user list creation methods against the ground truth lists obtained by crawling Dataset 1.

4.1.1 Evaluation metric: normalised mutual information

In order to validate user clustering results, we compare the automatically created clusters with the existing Twitter lists (ground truth) obtained via the Twitter API. Lancichinetti and Fortunato (2009) proposed an adaptation of the normalised mutual information (NMI). The mutual information measures the information that two random variables share, which is extended for overlapping clusters. In NMI, the mutual information is transformed to a value between 0 and 1. A high NMI value (close to 1) means that created clusters are similar to existing lists. This measure has become quite popular for comparing community finding algorithms in social network analysis. Lancichinetti's adaptation can be summarised as follows:

$$NMI(X \mid Y) = 1 - \frac{1}{2}\left(\frac{H(X \mid Y)}{H(X)} + \frac{H(Y \mid X)}{H(Y)} \right), \tag{7}$$

where $H(X)$ is the entropy of the overlapping sets X and $H(Y|X)$ is the conditional entropy of X given Y and vice versa.

As such, it yields a mean score for each user u.

$$Mean(NMI) = \frac{\sum_{i=1}^{N} NMI_{u_i}}{N}, \tag{8}$$

where NMI_{u_i} is the NMI score of user u_i and N is the total number of users.

4.1.2 Evaluation results

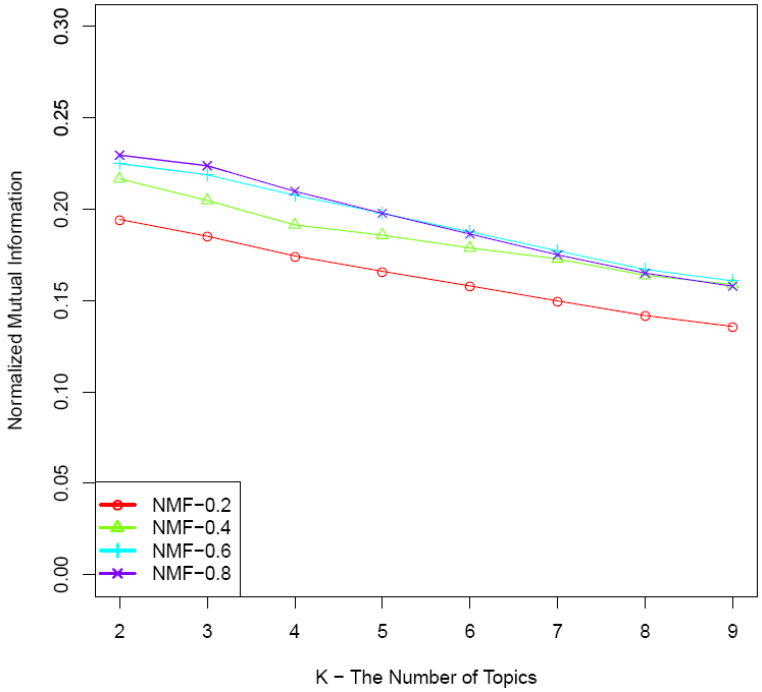
This part introduces the configuration of user clustering methods and presents the mean NMI for each of them. As NMF and OSLOM are stochastic and rely on randomly initialised values, we run each method five times with different initial values and calculate the mean NMI for each method. The notation we use for each method is presented in Table 2. For NMF, we use the default maximum number of iterations (100) in the R implementation and calculate mean NMI values of NMF methods with different number of topics K and different membership thresholds.

Table 2 Description and notation of different clustering methods

Method notation	*Description*
NMF-x	The NMF when we use x as threshold
OSLOM-1	The OSLOM with homeless vertices
OSLOM-2	The OSLOM without homeless vertices
Early Fusion-1	The method that runs OSLOM-1 on weighted graph
Early Fusion-2	The method that runs OSLOM-2 on weighted graph
Late Fusion-1	The method that merges NMF-0.8 into OSLOM-1
Late Fusion-2	The method that merges NMF-0.8 into OSLOM-2
Late Fusion-3	The method that merges OSLOM-1 into NMF-0.8
Late Fusion-4	The method that merges OSLOM-2 into NMF-0.8

K is the number of user lists we will create with non-negative matrix factorization methods and we test K ranging from one to nine for NMF methods. Figure 7 shows mean NMI values of each configuration. NMF-0.8 gives the highest mean NMI value when the number of topics is 2. The NMF methods have an explicit trend when increasing K and threshold. Increasing the number of topics K results in a decrease in the mean NMI values, so there is a preference for a smaller number of topics (NMI is higher), indicating that Twitter users tend to track a small number of topical lists. On the other hand, increasing the membership threshold results in a decrease in the mean NMI values, indicating that Twitter users tend to create lists without too much overlap (see Table 1).

Figure 7 NMF results (see online version for colours)



OSLOM is run in two different modes: with and without homeless vertices. The default mode is with homeless vertices which will ignore vertices whose statistical significance values are smaller than 0.10 (default tolerance). The mode without homeless vertices will assign each homeless vertex to at least one community (Lancichinetti et al., 2011). The evaluation results of both mode are shown in Table 3.

Table 3 OSLOM results

Method	*NMI*
OSLOM-1	0.253
OSLOM-2	0.244

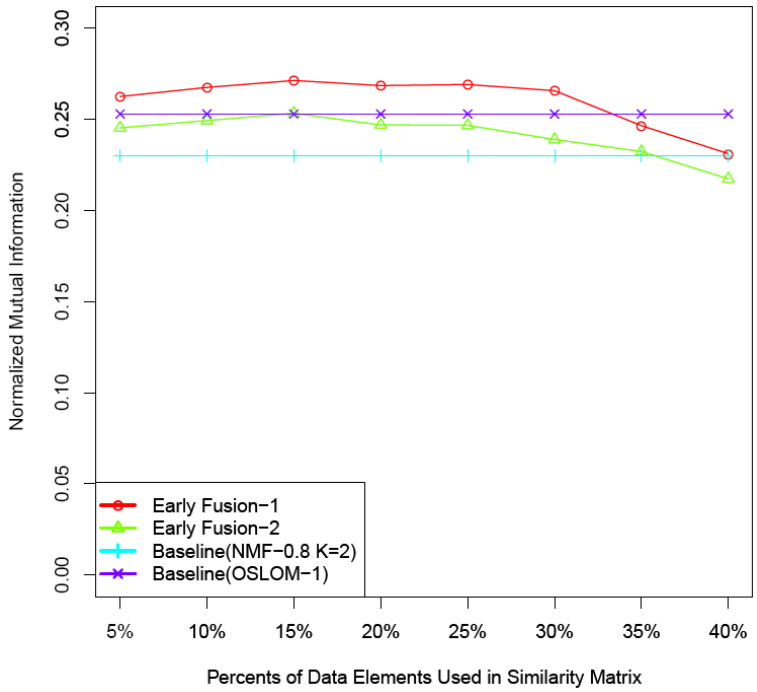
The result shows that the variant that ignores homeless vertices gives a slightly higher NMI score than the variant that handles homeless vertices. An explanation is that assigning each user to at least one list introduces noise.

For early fusion, the parameter is the percentage of data elements to add to the followee graph from the content similarity matrix. For these experiments, we set the value of this parameter between 5% to 40%. Once the combined feature matrix is built, we run OSLOM algorithm on it with two modes: with and without homeless vertices.

Figure 8 shows the results of our early fusion approach. The results of methods that have the highest mean NMI values are (NMF-0.8 when K = 2 and OSLOM-1) also included in Figure 8 for a convenient comparison. Early Fusion-1 outperforms Early Fusion-2. Early Fusion-1 reaches the highest NMI value when the percentages of users we take from user similarity matrix is 15%. This indicates that we should introduce a

limited amount of content information into the followee graph and it does not necessarily mean that the more content information we introduce, the higher NMI we obtain.

Figure 8 Early fusion results (see online version for colours)



For the late fusion approach, we use NMF as the content-based method and OSLOM as the network-based method. The parameter K of late fusion is from NMF, which is the number of topics as well as the number of user lists. The late fusion takes the result of NMF-0.8 when $K = 2$ as it gives highest mean NMI value among other NMF configurations.

We show the results for our late fusion approach in Figure 9. Similar to early fusion, we plot three other methods that have the highest mean NMI values for a convenient comparison. We can see that the Late Fusion-1 has the highest NMI value when the number of topics is 2. The trend shows that increasing the number of topics K does not yield further improvement in NMI and OSLOM that ignores homeless vertices gives higher mean NMI score than OSLOM that handles homeless vertices.

4.1.3 Overall results

We compare the NMI results of different methods in Table 5. Late fusion slightly outperforms early fusion and both fusion methods outperform methods based on a single information source. To clarify the stability, we plot error bars of mean values and standard deviations for each running of our experiments. Table 5 shows that the variances of NMI scores of NMF, OSLOM, Early Fusion-1 and Late Fusion-1 methods are low compared to the mean values. Further, to account for the stability of NMF, OSLOM and

Fusion methods we perform Kruskal-Wallis test of NMI scores on the five runs from NMI-0.8, OSLOM-1, Early Fusion-1 and Late Fusion-1 (for notations see Table 2). We set the null hypothesis to be that NMI scores of the tested algorithms originate from the same distribution (not stable). The p-value in this test turns out to be nearly zero (0.0004781). Hence we reject the null hypothesis.

Figure 9 Late fusion results (see online version for colours)

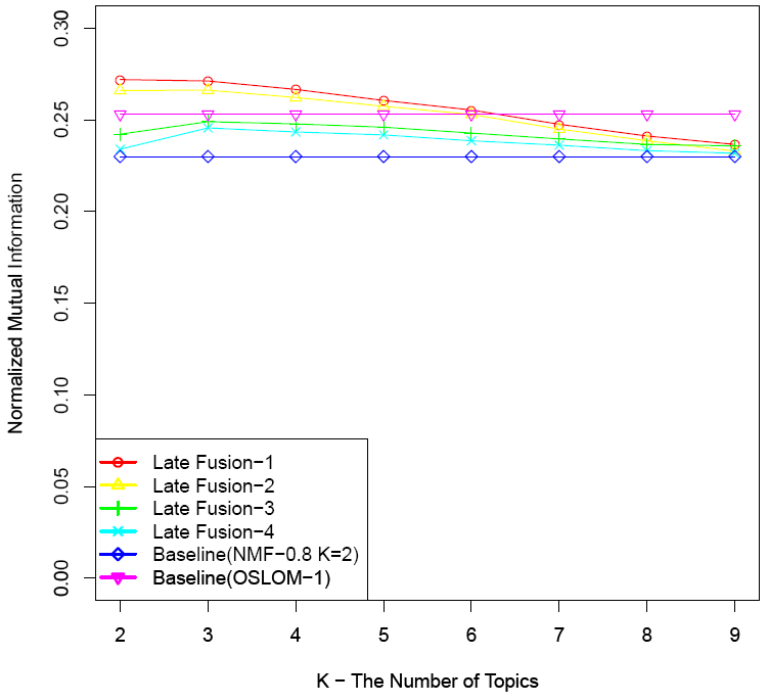


Table 4 Mean NMI values of the best configuration of NMF and OSLOM

Configuration	*Highest NMI*
OSLOM-1	0.253
NMF-0.8 $K = 2$	0.229

Table 5 Overall NMI results of different methods

Method	*Mean NMI ± standard deviation*
NMF-0.8 when $K = 2$	0.229 ± 0.0005
OSLOM-1	0.253 ± 0.0018
Early Fusion-1	0.271 ± 0.0024
Late Fusion-2	0.272 ± 0.0011

4.2 User evaluation

The user list creation is a subjective task, as users in microblog may have diverse ways of managing followees depending on their domain, knowledge, and attitude towards their followees. In this respect, ground truth user lists existing in real microblogs are not the unique solution for users to manage their followees. Thus, the offline evaluation may fail to measure the accuracy when our list creation methods give acceptable but different lists from the ground truth. To further validate our user list creation methods, we conduct an user evaluation which allows users to select lists created by our method as a starting point to manage their followees. Then we measure the changes between the original recommendation and the user modified version. A corresponding evaluation interface is developed to support user list management.

4.2.1 Evaluation metric: minimum number of operations

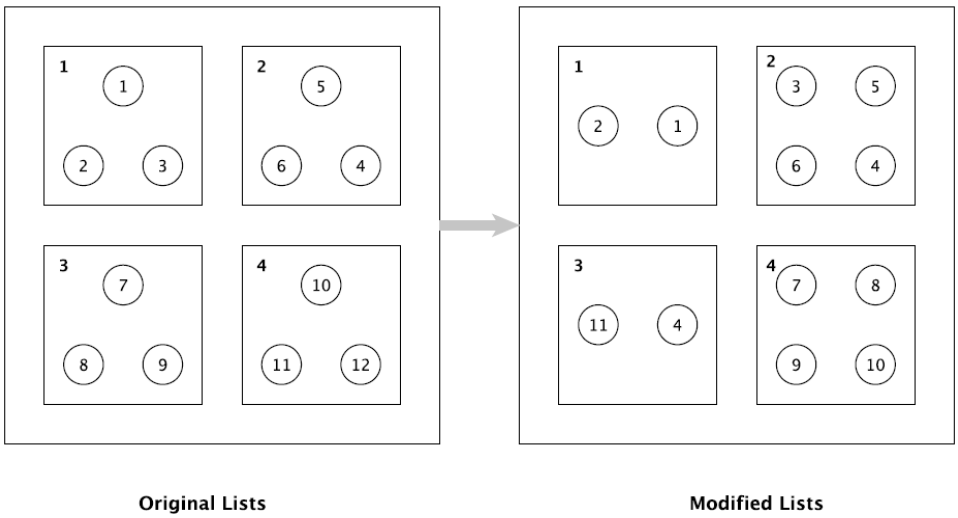
This quantitative measurement calculates the difference between two values:

1 $\mathbf{NUM_{op\_man}}$ which is the minimum number of operations for users to create user list without any accessibility tool

2 $\mathbf{NUM_{op\_auto}}$ which is the minimum number of operations for users to create user list from the starting point provided by our method.

The two types of operation considered in this paper are adding a list or a member and removing a list or a member.

Figure 10 User list modification: this example shows the user lists before and after user's modification



Note: The modification here means add or delete a user.

$\mathbf{NUM_{op\_man}}$ is the same as the sum of the number of members in each list. $\mathbf{NUM_{op\_auto}}$ can be obtained by comparing the user modified lists with the original lists recommended by our method. For example, in Figure 10, we recommend four lists for a user to modify.

After the user's modification, the members in those four lists have changed. $\mathbf{NUM}_{\mathbf{op\_man}}$ in this example is 12 as there are 12 members in those four lists. In order to calculate $\mathbf{NUM}_{\mathbf{op\_auto}}$, we proposed a backtracking algorithm. The algorithm first builds an operation matrix (**OM**) in which each element represents the minimum of adding and removing operations required to convert a list from the original recommendation to the modified version. In the example in Figure 10, $\mathbf{OM}_{1,1} = 1$ as there is one removing operation needed to convert list 1 in the original recommendation to list 1 in the modified version. The operation matrix (**OM**) is built with equation (9).

$$\mathbf{OM}_{ij} = \left|\mathbf{L}_i \cup \mathbf{L}'_j\right| - \left|\mathbf{L}_i \cap \mathbf{L}'_j\right|, \tag{9}$$

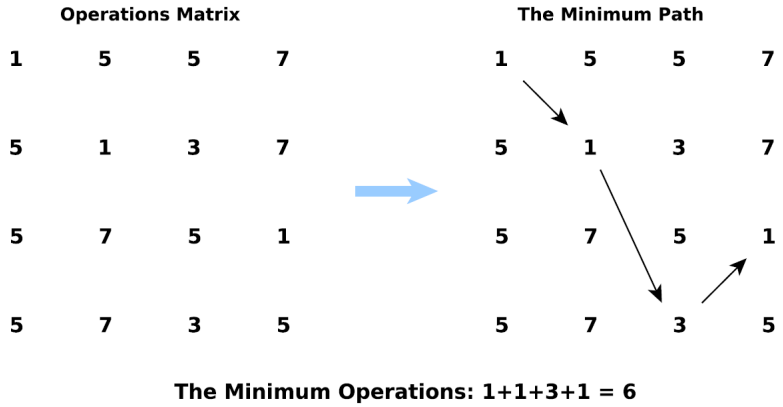
where $\mathbf{L}_i$ is the i^{th} list in the original recommendation and $\mathbf{L}'_j$ is the j^{th} list in the modified version. $|\mathbf{L}_i \cup \mathbf{L}'_j|$ is the number of distinct users in $\mathbf{L}_i$ and $\mathbf{L}'_j$. $|\mathbf{L}_i \cap \mathbf{L}'_j|$ is the number of users appear in both $\mathbf{L}_i$ and $\mathbf{L}'_j$.

Finding the minimum number of operations over all lists means finding a path on the operation matrix (**OM**) which goes through each row and column exactly one time and minimised the sum of the reached elements (see Figure 11). This is a typical backtracking problem which systematically searches for all possible solutions to a problem among all available options. In this paper, we use backtracking to search all possible paths and calculate the sum of the reached elements. The minimum number of operations is the minimum sum among all possible paths. The backtracking algorithm is composed as the following steps:

Algorithm 1 Backtracking for finding all possible paths

```
1:  min ← integer.max
2:  procedure check_column(column)          ▷ the column to be tested
3:      if column = N then                  ▷ last column
4:          sum ← calculate_sum()           ▷ the sum of reached elements on this path
5:          if sum < min then
6:              min ← sum                   ▷ find the minimum sum
7:          end if
8:      else
9:          for k in 1 : N do
10:             if IS_REACHABLE(column, k) then   ▷ test if column k is already reached
11:                 SET_REACHED(k)          ▷ set column k to reached
12:                 CHECK_COLUMN(column+1)  ▷ check next the column
13:             end if
14:         end for
15:     end if
16: end procedure
```

Figure 11 The minimum number of operations: we denote O_i as list i from original lists and M_j as list j from modified lists (see online version for colours)



Notes: The modification consists of four steps: (1) M_1 is modified from O_1 with 1 deletion; (2) M_2 is modified from O_2 with 1 adding; (3) M_3 is modified from O_4 with 1 adding and 2 deletion; (4) M_4 is modified from O_3 with 1 adding. In total, there are six operations required to perform this modification.

The above part shows how to calculate $\mathbf{NUM}_{\mathbf{op\_auto}}$ when the number of lists in the original recommendation (N_1) is the same as the number of lists in the modified version (N_2). The following part will introduce how to calculate $\mathbf{NUM}_{\mathbf{op\_auto}}$ when N_1 does not equal to N_2. The difference between N_1 and N_2 is due to the operations of adding or removing a user list rather than list members. As the evaluation interface allows users to add and remove the whole list by one click, only one operation required each time they add or remove a list. An alternative way to remove a list is removing all the members of this list one by one, and it requires many more operations than removing the whole list. Thus, we can first make N_1 equals to N_2 by adding empty lists and calculate $\mathbf{NUM}_{\mathbf{op\_auto}}'$ between original recommendation and modified version with empty lists. Then the $\mathbf{NUM}_{\mathbf{op\_auto}}$ can be obtained by summing the number of empty lists added to $\mathbf{NUM}_{\mathbf{op\_auto}}'$.

Another possible measurement is to track the real number of operations on the evaluation interface. This work does not apply this measurement as it also includes misoperations of users due to carelessness and the misoperations will make the measurement less accurate.

- if $\mathbf{NUM}_{\mathbf{op\_man}} > \mathbf{NUM}_{\mathbf{op\_auto}}$, the list creation method make it easy for users to manage their followees by reducing the number of operations
- if $\mathbf{NUM}_{\mathbf{op\_man}} < \mathbf{NUM}_{\mathbf{op\_auto}}$, the list creation method fails to make it easy for users to manage their followees, because the method increases the number of operations.

Further, the percentage of operations reduced (**RE**) by list creation method for a given user **u** is defined as:

$$\mathbf{RE}(\mathbf{u}) = \frac{\mathbf{NUM}^{\mathbf{u}}_{\mathbf{op\_man}} - \mathbf{NUM}^{\mathbf{u}}_{\mathbf{op\_auto}}}{\mathbf{NUM}^{\mathbf{u}}_{\mathbf{op\_man}}} \times 100\%.$$

4.2.2 Evaluation results

The clustering method we apply in user evaluation is the early fusion method with 15% of content data element. The reason we why choose the early fusion is two-fold:

1 it gives relative higher mean NMI score in offline evaluation among all the user clustering methods

2 it does not require prior knowledge of the number of user clusters.

The mean NMI score of user evaluation is 0.581, which is two times higher than the mean NMI score in the offline evaluation. The offline evaluation measures how similar are the lists created by us with the existing lists on Twitter. The problem faced in the offline evaluation is that the existing lists on Twitter are not the unique solution for users to manage their followees. The user evaluation measures the degree of acceptance of the lists created by us regardless of any ground truth. The acceptance of participants here is a highly subjective standard that is influenced by many factors such as knowledge and attitudes toward followees. A higher mean NMI score in user evaluation indicates that users accept the lists created by us including those lists which are different from the ground truth. Table 6 shows the range of the NMI scores. Most users are in the range between 0.8 and 1, but there are still 11 users whose NMI scores are lower than 0.4. This indicates that the early fusion method does not always generate acceptable results. One possible reason might be that users have different attitudes towards our user lists so that casual users tend to modify less but serious users tend to modify more.

Table 6 NMI result of user evaluation: the table shows five ranges of NMI scores and the corresponding number of users in each range

NMI range	*# Users*	*NMI range*	*# Users*
0.0−0.2	4	0.6−0.8	2
0.2−0.4	7	0.8−1.0	12
0.4−0.6	6	-	-

The early fusion method creates 187 user lists for all participants in this evaluation. There are seven participants who remove at least one user list and 14 participants who add at least one user list. In other words, 21 out of 31 users change the number of lists we created. This indicates that the user clustering method is not sufficient for users to automatically manage followees because users still need some more manual efforts.

To measure the manual efforts needed for users, we count the minimum number of operations required for users to create ($\mathbf{NUM}_{\mathbf{op\_man}}$) as well as to modify ($\mathbf{NUM}_{\mathbf{op\_man}}$) their user lists. There are 25 participants whose $\mathbf{NUM}_{\mathbf{op\_auto}}$ are less than $\mathbf{NUM}_{\mathbf{op\_man}}$ and six users whose $\mathbf{NUM}_{\mathbf{op\_auto}}$ are greater than or equal to $\mathbf{NUM}_{\mathbf{op\_man}}$. It indicates that the user clustering method makes the list creation easier by reducing the minimum number of operations required to manage their followees. There are some users whose $\mathbf{NUM}_{\mathbf{op\_man}}$ is

much greater than $\mathbf{NUM}_{\mathbf{op\_auto}}$ (e.g., User 1, User 8, User 11 and User 22) because the lists proposed do not require substantial modification.

Table 7 shows the range of **RE** scores. Most users have a high **RE** score (**RE** > 80%), which indicates that those users do not modify our recommended lists substantially. However, the early fusion method also makes six users harder to manage user lists, as they did more manual work than they needed to create lists without the early fusion method. The possible reason is that casual users tend to accept most of the lists we created and serious users tend to create their own.

Table 7 **RE** result of user evaluation: the table shows six ranges of **RE** values and the corresponding number of users in each range

RE range	*# Users*	*RE range*	*# Users*
⩽0%	6	40%−60%	1
0%–20%	1	60%–80%	4
20%–40%	4	80%–100%	15

5 Conclusions

In this paper, we addressed the problem of the tedious manual creation of user list in microblogging services. To solve this problem, we first applied several methods based on the textual content and the followee graph to generate user clusters. Next, we proposed several early and late fusion configurations to combine information from both textual content and followee graph. To validate the above methods, we conducted both offline evaluation and user evaluation. The main observations are described as follows:

- In microblogging services, users in the same list are more similar in content to each other than to those in other lists and also have denser connections between each other than between those in other lists: This is a basic assumption in our user clustering method, we analysed 2,311 user lists in our pre-collected dataset in the offline evaluation. The results show that the assumption is supported by most of the cases, as users in the same list usually have a higher cosine similarity in content to each other (Figure 3) and also have a higher graph density (Figure 4).
- *The method using followee graph information performs better than that using textual content information*: We applied one content-based methods (NMF) and one network-based method (OSLOM) in the offline evaluation, the OSLOM method gives a higher mean NMI score than NMF. Thus, we can conclude that user clusters created by the method using followee graph give more similar results to the real user lists in microblogs. One of the reasons might be topics in the user content in microblogs change over time quickly while the changes in followee graph are less time sensitive. However, we can not be sure the improvement is due to the method or simply the data because it is not straightforward to apply NMF to graph data nor is it to apply OSLOM to tweet content.

- *The information fusion methods give better user clustering results than any single method relying on a single information source*: We evaluated both the early fusion method and the late fusion method in the offline evaluation and compared them with purely content-based and network-based methods. The mean NMI results show that the fusion methods give higher mean NMI scores than purely content-based and network-based methods.
- *The list creation method makes it easier for user to manage their followees*: In this part, we adopted the early fusion method in the user evaluation and calculated the minimum number of operations required to create user lists. The early fusion method makes it easier for user to manage their followees by reducing the number of operations required to create user lists.

Acknowledgements

This work has been made possible by funding by Science Foundation Ireland under the Clique project (SFI/08/SRC/I1407).

References

Agarwal, G. and Kempe, D. (2008) 'Modularity-maximizing graph communities via mathematical programming', *The European Physical Journal B – Condensed Matter and Complex Systems*, Vol. 66, No. 3, pp.409–418.

Cafieri, S., Hansen, P. and Liberti, L. (2011) 'Locally optimal heuristic for modularity maximization of networks', *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, Vol. 83, No. 5, p.056105 (8p).

Chekuri, C.S., Goldberg, A.V., Karger, D.R. and Levine, M.S. (1997) 'Experimental study of minimum cut algorithms', *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp.324–333.

Dongwoo, K., Yohan, J., Il-Chul, M. and Alice, O. (2010) 'Analysis of Twitter lists as a potential source for discovering latent characteristics of users', *Proceedings of the Workshop on Microblogging at the ACM Conference on Human Factors in Computer Systems*.

Fortunato, S. (2010) 'Community detection in graphs', *Physics Reports*, Vol. 486, pp.75–174, Elsevier B.V.

Fung, B.C.M., Wang, K. and Ester, M. (2003) 'Hierarchical document clustering', *Encyclopedia of Data Warehousing and Mining*, pp.970–975.

Girvan, M. and Newman, M.E.J. (2002) 'Community structure in social and biological networks', *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 99, pp.7821–6.

Greene, D., Reid, F., Sheridan, G. and Cunningham, P. (2011) 'Supporting the curation of Twitter user lists', *NIPS 2011 Workshop on Computational Social Science and the Wisdom of Crowds*.

Hao, J. and Orlin, J.B. (1992) 'A faster algorithm for finding the minimum cut in a graph', *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pp.165–174.

Jain, A.K. and Murty, M.N. and Flynn, P.J. (1999) 'Data clustering: a review', *ACM Computing Surveys*, Vol. 31, No. 3, pp.264–323.

Karandikar, A. (2010) *Clustering Short Status Messages: A Topic Model based Approach*, Master Thesis, University of Maryland.

Lancichinetti, A. and Fortunato, S. (2009) 'Detecting the overlapping and hierarchical community structure in complex networks', *New Journal of Physics*, Vol. 11, No. 3, p.033015 (18p).

Lancichinetti, A., Radicchi, F., Ramasco, J.J. and Fortunato, S. (2011) 'Finding statistically significant communities in networks', *PLoS ONE*, Vol. 6, No. 4, p.e18961, doi:10.1371/journal.pone.0018961.

Lee, D. and Seung, H.S. (2001) 'Algorithms for non-negative matrix factorization', *Proceeding of the Neural Information Processing Systems: Natural and Synthetic*, Vol. 13, pp.556–562.

Mei, Q., Cai, D., Zhang, D. and Zhai, C. (2008) 'Topic modeling with network regularization', *Proceeding of the 17th International Conference on World Wide Web*.

Molloy, M. (1995) 'A critical point for random graphs with a given degree sequence', *Random Structures and Algorithms*, Vol. 6, Nos. 2–3, pp.161–179.

Newman, M. (2004) 'Fast algorithm for detecting community structure in networks', *Physical Review E*, Vol. 69, No. 6, p.066133 (5p).

Qin, H., Liu, T. and Ma, Y. (2012) 'Mining user's real social circle in microblog', *2012 International Conference on Advances in Social Networks Analysis and Mining*, pp.348–352.

Steinbach, M., Karypis, G. and Kumar, V. (2000) 'A comparison of document clustering techniques', *Proceeding of the Workshop on Text Mining at the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Steinhaeuser, K. and Chawla, N.V. (2008) 'Community detection in a large real-world social network', *Proceeding of the International Conference on Social Computing*, *Behavioral Modeling and Prediction*.

Zeng, H., Chen, Z. and Ma, W. (2002) 'A unified framework for clustering heterogeneousweb objects', *Proceedings of the Third International Conference on Web Information Systems Engineering*, pp.161–170.