2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust

Unsupervised Construction of Topic-based Twitter Lists

Francois de Villiers, McElory Hoffmann, Steve Kroon
Computer Science Division
Stellenbosch University
Private Bag X1, Matieland, 7602, South Africa
Email: fdv@ml.sun.ac.za, mcelory@sun.ac.za, kroon@sun.ac.za

***Abstract*—The Twitter lists feature was launched in late 2009 and enables the creation of curated groups containing Twitter users. Each user can be a list author and decide the basis on which other users are added to a list. The most popular lists are those that associate with a topic. Twitter lists can be used as a powerful organisation tool, but its widespread adoption has been limited. The two main obstacles are the initial setup time and the effort of continual curation. In this paper we attempt to solve the first problem by applying unsupervised clustering algorithms to construct topic-based Twitter lists. We consider k-means and affinity propagation (AP) as clustering algorithms and evaluate these algorithms using two document representation techniques. The selected representation techniques are the popular term frequency-inverse document frequency (TF-IDF) and the latent Dirichlet allocation (LDA) topic model. We calculate the similarities for the clustering algorithms using five well-known similarity measures that have been used extensively in the text domain. The adjusted normalised information distance (ANID) was used to compare the clustering result yielded by k-means and affinity propagation. We found that the careful selection of a similarity measure, combined with the LDA topic model can provide a user with a sensible starting point for list creation.**

I. Introduction

The demands of users have changed since the arrival of social networks. In the beginning, users were satisfied with being instantly connected to friends, family and colleagues. As the social networks of each user expanded, so did the number of messages, images and links that each user consumes on a daily basis [1].

Social networks, such as Facebook, Twitter and Google+ have recently introduced mechanisms to enable users to manage this vast flow of information. For example, Twitter introduced lists, which allow a user to categorise users and label them. However, initial setup cost is a common problem for a user creating these lists; a user of any system with many connections will find it tedious to sort through all his connections and organise them according to topics. The whole process would be simplified if the user could be provided with a good initial configuration.

In this paper, we propose clustering algorithms to construct topic-based Twitter lists. Clustering algorithms have been used extensively to find natural structure in data (see, for example [2]). They are applicable in any domain where the similarity, dissimilarity or distance between data points can be quantified. Clustering in the text domain is governed by the document's representation and the calculation of the similarity between two representations of documents.

The document representation process attempts to extract the most prominent features of a document. The term frequency-inverse document frequency (TF-IDF) [3] is a well-known technique, which represents a document as a vector of its most important terms and has been applied to short text documents [4] as well as full text documents [5]. Topic models [6] are another form of document representation techniques, which have recently gained in popularity. Latent Dirichlet allocation (LDA) [7] is such a topic model, which has been used for classification tasks [8], user recommendations [9] and clustering [10].

The second important aspect in the text domain is the choice of similarity measures. The most notable of these are the Euclidean distance, cosine similarity, Pearson correlation coefficient, Kullback-Leibler divergence and extended Jaccard coefficient, which have all been analysed in comparative studies [5], [11], [12].

The evaluation of unsupervised document clustering as an approach to constructing topic-based Twitter list has to the best of our knowledge not been performed before. Our goal is to provide users with a good initial partition of the users they follow, grouped according to the topics they tweet about. We select k-means [13] and affinity propagation (AP) [14] as our clustering algorithms and represent the documents with the LDA topic model and TF-IDF. We select the five previously mentioned similarity measures and report on the performance of each measure for clustering tweets.

We evaluate k-means and AP on two Twitter data sets, which contain 1737 and 2800 users respectively. The clustering result is evaluated with the adjusted normalised information distance (ANID) [15].

The rest of this paper is organised as follows: In Section II we discuss the TF-IDF and the LDA topic model. This is followed by an overview of the similarity measures in Section III. In Section IV, we discuss the k-means and AP clustering algorithms and in Section V we describe our approach to evaluating the quality of clustering results. Section VI introduces our experiments, where we discuss the data sets and our methodology, and give an analysis of our results. Finally, before concluding our work, we discuss related work in Section VII.

978-0-7695-4848-7/12 $26.00 © 2012 IEEE
DOI 10.1109/SocialCom-PASSAT.2012.64

II. Document Representation

A tweet[1] consists of a maximum of 140 characters and may contain mentions, hashtags and hyperlinks. A mention occurs when user A refers to user B in a tweet by including the username of B preceded by an "@". Hashtags usually relate to the topic of a tweet and consists of a term or concatenation of terms starting with a "#". A hashtag is used when users discuss an event or topic on Twitter. It simplifies the process of finding tweets related to an event or topic, since one can search Twitter for tweets containing a specific hashtag. Fig. 1 shows a tweet

Fig. 1. A screenshot of a tweet that contains a hyperlink, mentions and hashtags as part of the message.

containing both mentions [@grantimahari, @scully313] and hashtags [#FireFly, #LEGO]. Hong and Davison [16] showed that although a tweet is short in length, it may still convey rich meanings. The same study, discussed in Section VII, shows that representing a user by a single document consisting of a collection of tweets is a good representation for topic discovery and clustering performance.

We thus use this to represent each user as a user document consisting of his N latest tweets. Next, we discuss the two models we use to represent the user documents. The models are discussed in terms of a user document corpus $D = \{d_1, d_2, \ldots, d_n\}$. Each d_i in the corpus is a user document, which consists of user i's latest N tweets.

A. Vector Space Model

A popular way to represent documents is the bag of words model [3]. In this approach a document is represented by its terms, and it is assumed that terms are independent. Therefore the bag of words model does not take the order of terms or their position in consideration. The importance of each term can be measured by calculating the term frequency, since the terms in a document occur multiple times throughout the document. Representing a document using term frequencies effectively reduces the dimensionality of the document.

We can define our representation as follows. Let $T = \{t_1, t_2, \ldots, t_m\}$ be the set of distinct terms in the corpus. A document $d \in D$ is represented by an m-dimensional vector $t_d = (\mathrm{tf}(d, t_1), \mathrm{tf}(d, t_2), \ldots, \mathrm{tf}(d, t_m))$ where $\mathrm{tf}(d, t)$ is the number of occurrences of t in document d.

This representation implicitly assumes that terms that occur more frequently are more important, which is not always the case. If, for example, the terms "a" and "the" frequently appear in a document they will be considered important because of a high frequency count. The terms "a" and "the" do not generally convey any information about a document, due to their frequent occurrence in other documents in the corpus.

[1]A short text document on Twitter is referred to as a tweet.

To counteract this, the TF-IDF weighting scheme is used. The TF-IDF scheme assumes that terms which appear frequently in a small number of documents, but relatively infrequently in others, tend to be much more relevant for discriminating between documents in the corpus.

To calculate the TF-IDF score, the term frequency (TF) is adjusted based on the inverse document frequency (IDF):

$$\mathrm{tfidf}(d, t) = \mathrm{tf}(d, t) \times \log\left(\frac{|D|}{\mathrm{df}(t)}\right). \quad (1)$$

The term $\mathrm{df}(t)$ is the number of documents in which term t appears. The inverse document frequency, $\log\left(\frac{|D|}{\mathrm{df}(t)}\right)$, serves as a weighing factor applied to the term frequency, $\mathrm{tf}(d, t)$. As a result, a term occurring in few documents in the corpus but with a high frequency in those documents will have a higher relative importance in the corpus.

In this study we use the TF-IDF as one representation of a document.

B. Topic Models

Topic models [17] represent a document as a vector of topic proportions, with each topic modelled as a distribution over terms.

The set of topics derived from a set of documents D can be used to answer questions about the similarity of terms and documents: two terms are similar to the extent that they appear in the same topic, and two documents are similar to the extent that the same topics appear in those documents.

Latent Dirichlet allocation (LDA) is a generative probabilistic model, which represents documents as random mixtures over latent topics [7]. A topic is defined as a distribution over a fixed vocabulary of terms. Each document in a collection then consists of different proportions of a set of Q topics. It is natural to assume that a document collection exhibits some number Q of topics, because a collection tends to be heterogeneous with a number of central ideas and themes.

The generative model for LDA is shown in Fig. 2 using plate notation [7] for graphical models. Each topic q is represented by a distribution over terms, β_q, governed by a Dirichlet prior with parameter η. A document d is represented by a multinomial distribution, θ_d, over topics governed by a Dirichlet prior with parameter α. Furthermore, each term is assigned a latent topic, $Z_{d,n}$, drawn from the multinomial distribution θ_d, and then the term $W_{d,n}$ is drawn from the multinomial distribution $\beta_{Z_{d,n}}$. Finally, Q denotes the number of topics.

As stated previously, each document d_i is represented by a multinomial distribution over topics. We can view this representation as a vector of topic proportions present in the document. Thus the similarity between document d_i and d_j can be calculated by using the topic proportions.

III. Similarity Measures

We choose the similarity measures for this study based on a survey of literature.

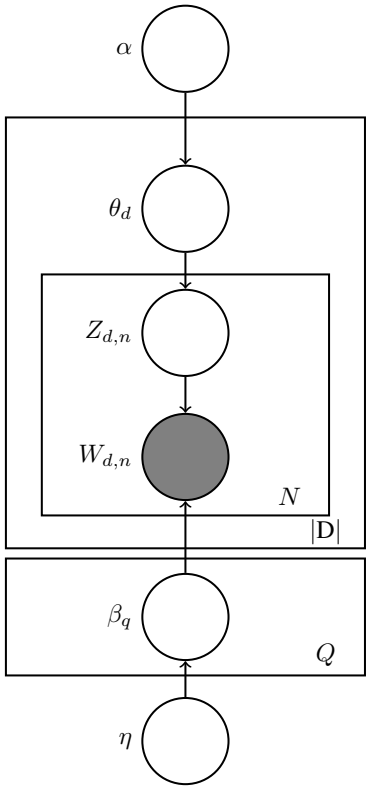

Fig. 2. A graphical model representation of latent Dirichlet allocation (LDA). Nodes denote random variables; edges denote dependence between random variables. Shaded nodes are observed random variables; unshaded nodes denote latent random variables. The rectangular boxes are "plate notation", which denote replication.

Let $D = \{d_1, d_2, \ldots, d_n\}$ be the document corpus and let $V = \{v_1, v_2, \ldots, v_m\}$ be the vocabulary of D. Each document d_i is represented by a vector of terms when processed with the TF-IDF and a vector of topics when processed with LDA. We refer to the elements of the vector as components.

A. Euclidean Distance

The Euclidean distance for document d_i and d_j is calculated as

$$\delta(d_i, d_j) = \sqrt{\sum_{k=1}^{m}(d_{ik} - d_{jk})^2}, \tag{2}$$

where m is the length of the document vector. The Euclidean distance has been used in a variety of studies [5], [11] as a basis for comparison as well as simple metric.

B. Cosine Similarity

The cosine similarity has been applied to document clustering [18], short text clustering [4] and user recommendations [9]. The cosine similarity measures the angle between the two given vectors. If the value is 0, the two vectors are orthogonal. If the value is 1, the two vectors have the same direction. The angle between document vectors d_i and d_j is

$$\cos(d_i, d_j) = \frac{\langle d_i, d_j \rangle}{||d_i|| \cdot ||d_j||} \tag{3}$$

We represent the cosine similarity as a distance measure for k-means by simply letting $\delta = 1 - \cos(d_i, d_j)$.

C. Pearson Correlation Coefficient

The Pearson correlation coefficient has been applied in a variety of domains, including recommendations [19] and document clustering [20]. It measures the strength and direction of the linear relationship between two variables. The value is in the range of -1 to 1 where 0 is no relation, 1 is the strongest positive correlation and -1 is the strongest negative correlation. The Pearson correlation coefficient for document d_i and d_j is

$$r = \frac{\text{cov}(d_i, d_j)}{\sigma_{d_i} \cdot \sigma_{d_j}}. \tag{4}$$

The Pearson correlation coefficient is also a similarity measure and we use it as a distance measure letting $\delta = 1 - r$ when $r \geq 0$ and $\delta = -r$ when $r < 0$.

D. Averaged Kullback-Leibler Divergence

The Kullback-Leibler Divergence (KL) is used to evaluate the difference between two probability distributions. Given two distributions P and Q, the KL divergence from distribution P to distribution Q is defined as

$$\delta_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \tag{5}$$

The KL divergence does not satisfy the symmetry requirement of a true metric, since in general $\delta_{KL}(P||Q) \neq \delta_{KL}(Q||P)$. The KL divergence has mostly been used in its symmetric form for topic models [21], and averaged form for document clustering [5].

For components, the divergence between two distributions is

$$\delta_{KL}(d_i||d_j) = \sum_{t=1}^{m}\left[d_{i,t} \times \log \frac{d_{i,t}}{d_{j,t}}\right], \tag{6}$$

where $d_{i,t}$ is the value for t'th component in document i and $d_{j,t}$ for document j. To obtain symmetry, we use the averaged form

$$\delta_{AvgKL}(d_i||d_j) = \sum_{t=1}^{m} \pi_1(t) \times \left(d_{i,t} \times \log \frac{d_{i,t}}{w_t}\right) + \pi_2(t) \times \left(d_{j,t} \times \log \frac{d_{j,t}}{w_t}\right), \tag{7}$$

where $\pi_1(t) = \frac{d_{i,t}}{d_{i,t}+d_{j,t}}$, $\pi_2(t) = \frac{d_{j,t}}{d_{i,t}+d_{j,t}}$ and $w_t = \pi_1(t)d_{i,t} + \pi_2(t)d_{j,t}$.

The average weighting between two vectors ensures symmetry; that is, the divergence from document i to document j is the same as the divergence from document j to i.

E. Extended Jaccard Coefficient

The binary Jaccard coefficient measures the similarity of two sets by computing the ratio of the number of shared attributes of d_i and d_j to the total number of attributes [22]. Strehl and Ghosh [23] extended the binary definition to continuous or non-negative features. The extended Jaccard coefficient is calculated as

$$\mathrm{EJ}(d_i, d_j) = \frac{\langle d_i, d_j \rangle}{||d_i||^2 + ||d_j||^2 - \langle d_i, d_j \rangle}. \quad (8)$$

As with our previous similarity measures we need to calculate the distance measure for k-means. We calculate it as $\delta = 1 - \mathrm{EJ}(d_i, d_j)$.

IV. Clustering Algorithms

The selected clustering algorithms, k-means and affinity propagation, are both partitional clustering algorithms. Objects are thus categorised into a number of non-overlapping subsets. The categorisation occurs around cluster prototypes such that each object is in some sense more similar to its cluster prototype than the other cluster prototypes [13].

A. k-Means

The k-means algorithm [13] is initialised with K user-selected initial points (centroids). By assigning each point to its nearest centroid, K clusters are formed. For each of the K clusters, the centroid is recomputed and the process of assigning each point to its nearest centroid repeated. This process is iterated until the centroid of each cluster stays the same for consecutive iterations or some maximum number of iterations is reached. For k-means, the centroids serve as the cluster prototypes.

An important consideration for k-means is how the initial starting points are selected. We opted for the k-means++ method [24], where a point is chosen with a probability proportional to its distance from its nearest centre. The first centre is sampled uniformly at random from the data points. It has been shown to perform much better than the random seeding of k-means [24].

B. Affinity Propagation

Affinity Propagation (AP) [14] treats all data points as potential exemplars, exchanging messages until the most suitable exemplars are found and clusters are formed. The resulting exemplars are the medoids[2] of the generated clusters, and serve as cluster prototypes.

The AP procedure takes as input a $N \times N$ matrix s of document similarities and exchanges two types of messages between data points, *responsibilities* and *availabilities*. A responsibility, $r(i,k)$, is sent from data point i to a candidate exemplar point k and reflects the evidence that data point k is suitable as an exemplar for data point i. The availability, $a(i,k)$, is sent from a candidate exemplar data point k to data point i and is the evidence for how appropriate it is for data point i to choose k as its exemplar. The responsibilities are computed as

$$r(i,k) \leftarrow s(i,k) - \max_{k' \text{ s.t. } k' \neq k} \{a(i,k') + s(i,k')\}, \quad (9)$$

where $s(i,k)$ is the similarity of data point i and j. The availabilities are calculated as

$$a(i,k) \leftarrow \min\Big\{0, r(k,k) + \sum_{i' \text{ s.t. } i' \notin \{i,k\}} \max\{0, r(i',k)\}\Big\}, \quad (10)$$

for $i \neq k$. The self-availability $a(k,k)$ is updated as

$$a(k,k) \leftarrow \sum_{i' \text{ s.t. } i' \neq k} \max\{0, r(i',k)\} \quad (11)$$

and reflects the accumulated evidence that data point k is an exemplar. This evidence is thus based on the positive responsibilities sent to candidate exemplar k from other data points.

The responsibilities and availabilities can be combined in order to identify the exemplars. For point i, the value of k that maximises $a(i,k) + r(i,k)$ identifies the data point that is the exemplar for point i.

AP allows us to set a self-preference value, p_i, for each data point. A data point with a higher self-preference value has a greater likelihood to be selected as a cluster exemplar. If the self-preference value is set equal to the same value, p, for all the data points, the number of clusters can be influenced by p. A large p in relation to the magnitude of the similarities will lead to many clusters while a small p will lead to fewer.

The algorithm described above can be terminated either after a fixed number of iterations, when changes to the messages fall below a threshold, or if the local decisions stay constant for some number of iterations.

In order to avoid numerical oscillations that can arise in some circumstances, each message can be set to λ times its value from the previous iteration plus $1-\lambda$ times its prescribed update value [14], as described in Equation 9 – 11. The damping factor value λ is set between 0 and 1.

V. Cluster Quality Evaluation

The measures used to evaluate the quality of a clustering are important since every clustering algorithm will tend to find clusters in a data set even if that data set has no natural clustering structure. In order to compare two clustering results, we can build a contingency table that summarises the overlap between two clusterings. This contingency table is used extensively in pair-counting-based measures [25].

Let S be a set of N data points where $U = \{U_1, U_2, \ldots, U_R\}$ and $V = \{V_1, V_2, \ldots, V_C\}$ are clusterings on S, that is partitions of S into disjoint subsets. The overlap between U and V can then be summarised in the $R \times C$ contingency table $M = [n_{ij}]_{j=1,\ldots,C}^{i=1,\ldots,R}$ where n_{ij} are the number of objects shared between subsets U_i and V_j. Table I illustrates the contingency table, where the clustering U represents the actual categories for each data set. The clustering U is thus the perfect clustering for a data set and we refer to each category in the data set as a class when discussing the evaluation

[2]The medoid of a group of points is a multi-dimension generalisation of the median.

techniques. The clustering V is the result of the k-means or affinity propagation algorithm.

TABLE I
THE STRUCTURE OF THE CONTINGENCY TABLE. HERE n_{ij} IS THE NUMBER OF OBJECTS SHARED BETWEEN SUBSETS U_i AND V_j.

U/V	V_1	V_2	...	V_C	Sums
U_1	n_{11}	n_{12}	...	n_{1C}	a_1
U_2	n_{21}	n_{22}	...	n_{2C}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U_R	n_{R1}	n_{R2}	...	n_{RC}	a_R
Sums	b_1	b_2	...	b_c	$\sum_{ij} n_{ij} = N$

A. Entropy and Purity

Entropy and purity are classification-oriented measures for cluster validity and are commonly used to evaluate the performance of classification models [12], [13]. In terms of cluster evaluation, the degree to which cluster labels correspond to class labels is measured.

The entropy measures the distribution of the various class labels in each cluster [12]. We calculate the entropy[3] for each cluster V_j from the contingency table as

$$E(V_j) = -\frac{1}{\log R}\sum_{i=1}^{R} p(U_i|V_j)\log p(U_i|V_j), \tag{12}$$

where $p(U_i|V_j) = \frac{n_{ij}}{b_j}$ is the probability that a document in cluster j belongs to class i. Here, the value b_j is the number of documents in cluster j. The total entropy of a clustering is defined as the weighted sum of the entropies for each cluster, with weights based on the cluster sizes, and is calculated as

$$E(V) = \sum_{j=1}^{C} \frac{b_j}{N} E(V_j). \tag{13}$$

The entropy takes on values in the range of $[0, 1]$. A entropy of 0 indicates that all documents in a cluster belong to the same class. If the document class distribution is uniform over the cluster, the entropy will be 1. Thus a smaller entropy value is better for a clustering result.

The purity measures the extent to which a cluster contains documents of a single class [13]. We calculate the purity of each cluster V_j as

$$P(V_j) = \frac{1}{b_j}\max_{i=1}^{R}\{n_{ij}\}. \tag{14}$$

The overall purity of a clustering is

$$P(V) = \sum_{j=1}^{C} \frac{b_j}{N} P(V_j), \tag{15}$$

a weighted sum of the purity of the clusters in V. A purity of 1 indicates that each cluster only contains documents of a single class.

[3]This is a normalisation of the general definition of entropy in Equation 17.

B. Normalised Information Distance

The Mutual Information (MI) [15] describes the amount of statistical similarity between two clusterings U and V [26]. We can calculate the MI from Table I as

$$I(U,V) = \sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N}\log\frac{n_{ij}/N}{a_i b_j/N^2}. \tag{16}$$

We can also define the entropies and conditional entropy, which are

$$H(U) = -\sum_{i=1}^{R}\frac{a_i}{N}\log\frac{a_i}{N} \tag{17}$$

and

$$H(U|V) = -\sum_{i=1}^{R}\sum_{j=1}^{C}\frac{n_{ij}}{N}\log\frac{n_{ij}/N}{b_j/N}. \tag{18}$$

The entropy $H(U)$ can be interpreted as the uncertainty in U and the conditional entropy $H(U|V)$ of cluster U given cluster V is the uncertainty in U given that V is already known.

It is of interest to see how much knowledge of V reduces the uncertainty in U. If V provides information that helps to reduce the uncertainty in U, then $H(U|V)$ will be smaller than $H(U)$. The MI quantifies the reduction in uncertainty: we have $H(U) - H(U|V) = I(U,V)$, which also holds in the reverse direction, $H(V) - H(V|U) = I(U,V)$.

Kraskov, Stögbauer, Andrzejak and Grassberger [27] showed that by modifying the mutual information, the resultant quantity will be a metric in the strict sense. We calculate the Normalised Information Distance (NID) by normalising the MI and converting it to a distance metric. The NID,

$$NID(U,V) = 1 - \frac{I(U,V)}{\max\{H(U), H(V)\}}, \tag{19}$$

is in the range $[0, 1]$.

Hubert and Arabie [25] showed that similarity measures for cluster evaluation should be corrected for chance. Their *corrected-for-chance* property states that an index should have a constant baseline value, which means that if objects are randomly assigned to the clusters, the index should expect to take on the value of 0. The resulting general form[4] is

$$\text{Adjusted\_Index} = \frac{\text{Index} - \text{Expected\_Index}}{\text{Max\_Index} - \text{Expected\_Index}}. \tag{20}$$

We thus use the Adjusted Mutual Information (AMI) as shown by Vinh, Epps and Bailey [15] to calculate the adjusted-for-chance NID (ANID). The ANID is

$$\begin{aligned} ANID(U,V) &= 1 - AMI(U,V) \\ &= 1 - \frac{I(U,V) - E\{I(U,V)\}}{\max\{H(U), H(V)\} - E\{I(U,V)\}}, \end{aligned} \tag{21}$$

where $E\{I(U,V)\}$ is the expected MI [15] between two clusterings U and V.

[4]The adjusted measure can be a negative value if the result is worse than the random allocation of data points to clusters.

In this study we use the ANID as our general evaluation metric, further analysing the best results using the purity and entropy.

VI. Experiments

We construct two data sets, one consisting of popular user-created Twitter lists and the other of the suggested categories provided by Twitter. Each user in the data sets is a member of a Twitter list or category and thus each user is labelled according to his/her list or category. Our goal is to apply k-means and AP on each data set, to find clusters corresponding to the original lists or categories in each data set.

A. Data sets

We use the website Listorious [28], a Twitter search engine, to identify the popular topic-based lists on Twitter and from those lists we construct our first data set.

TABLE II
The breakdown of categories and number of users in each category for the subscribed lists data set.

Categories	Members
travel	498
food	14
programmers	422
health	19
non-profits	40
design	98
marketing	51
quotes	53
tech-news-brands	497
social-media	102
all-winter-olympics	88
team	858
movies	49
most-influential-in-tech	277

For the first data set, which we call the *subscribed lists* set, we selected 2800 users who are members of the 14 lists shown in Table II. Each list in the data set has been created by a Twitter user, who manages the members of the list. We construct the subscribed lists data set, because it represents the user's perception of what a good clustering should be like.

The second data set, which we call the *suggested lists* set, is a collection of categories provided by Twitter and consists of 1737 users. Each category contains a number of users, which Twitter has decided to be relevant to the topic. The data set consists of 26 categories and has been used in a number of classification tasks [16]. The suggested lists data set is shown in Table III.

B. Methodology

In order to construct the subscribed lists data set, we view the "top 140" Twitter lists as indexed by Listorious. We retrieve the unique identifiers (ids) of each list in Table II from Listorious, and use it to request the list members using the Twitter API. The construction of the suggested list data set is simpler, since the Twitter API provides methods to retrieve all the categories and the members in each category.

For each of the data sets we iterate over categories or lists and request the ids of all the members in the category or list.

TABLE III
The breakdown of categories and number of users in each category for the suggested lists data set.

Categories	Members
music	106
sports	78
entertainment	79
twitter	49
funny	64
fashion	61
family	40
pga	132
technology	56
television	203
faith-and-religion	73
nhl	62
mlb	104
nba	135
us-election-2012	58
food-drink	67
news	56
art-design	70
books	63
business	49
science	51
health	48
travel	47
government	30
staff-picks	94
charity	60

We use the ids of each user to retrieve the 100 latest tweets of that user. At this stage of the process we remove users if

- they are part of multiple lists;
- their account is protected;
- their account is not specified as English; or
- they have tweeted less than one hundred times.

Next, we create the user documents by combining each user's tweets into a single document. The mentions, hashtags and links are removed from the tweets, as well as various stop words. We use the English stop words list provided by the Python Natural Language Toolkit (NLTK) [29].

We further apply the Porter stemmer [30] as part of our term normalisation process. At this stage each member is represented by a user document, which consists of his/her hundred latest tweets.

Next, we calculate the TF-IDF for each user document and limit the vocabulary to 2000 terms. The selected terms are those which appear with the highest frequency in the relevant data set. We train the LDA topic model on the user document set for each data set, using the same 2000 terms as the vocabulary. The LDA topic model takes as input the number of topics to train. We train each data set using 5, 15 and 30 topics.

Both the subscribed lists and suggested lists sets are now represented in four different ways. They consist of the TF-IDF representation, LDA trained on 5 topics, LDA trained on 15 topics and LDA trained on 30 topics. In our results we refer to these representations as TF-IDF, LDA5, LDA15, and LDA30 respectively.

We apply k-means and AP to each of the above representations for varying numbers of clusters, namely 5, 10, 15, 20, 25 and 30. The way in which K clusters are obtained differs

for k-means and AP. In the case of k-means, we can simply specify K, while for AP we adjust the self-preference value p. We adjust p empirically in a range of values to obtain K clusters for each representation in combination with the similarity measures.

In Section IV-B we discussed the damping factor as well as the termination conditions for the AP algorithm. We set the damping factor to $\lambda = 0.9$ throughout all our experiments. For AP, we set the maximum number of iterations to 1000 and set the termination condition for no update to the local result at 30 iterations. For k-means, we set the maximum number of iterations to 300.

C. Results

We include four tables, Tables IV – VII, two of which can be viewed as the "expected best result" tables with the other two containing the best results actually obtained. The "expected best result" table for a data set corresponds to the situation where the number of clusters is set closest to the number of lists in the data set.

TABLE IV
A COMPARISON OF THE ADJUSTED NORMALISED INFORMATION DISTANCE RESULTS, FOR THE SUGGESTED LISTS DATA SET, WITH 25 CLUSTERS.

	Adjusted Normalised Information Distance				
Representation	Euclidean	Cosine	Pearson	Jaccard	KL
	K-means for 25 clusters				
LDA5	0.6873	**0.6808**	0.6879	0.6868	0.7241
LDA15	0.6942	0.6392	0.7647	**0.6146**	0.7136
LDA30	0.7322	0.6295	0.7430	**0.5967**	0.6858
TF-IDF	0.9687	**0.6424**	0.7653	0.6543	0.9131
	Affinity Propagation for 25 clusters				
LDA5	0.6821	**0.6729**	0.8044	0.6738	0.8973
LDA15	0.5776	**0.5652**	0.9500	0.5695	0.9707
LDA30	0.5148	**0.5046**	0.9617	0.5062	0.9923
TF-IDF	0.8858	**0.5168**	0.8455	0.5372	0.9692

TABLE V
A COMPARISON OF THE ADJUSTED NORMALISED INFORMATION DISTANCE RESULTS, FOR THE SUGGESTED LISTS DATA SET, WITH 30 CLUSTERS.

	Adjusted Normalised Information Distance				
Data set	Euclidean	Cosine	Pearson	Jaccard	KL
	K-means for 30 clusters				
LDA5	0.6985	**0.6828**	0.6993	0.6887	0.7199
LDA15	0.6833	0.6420	0.7402	**0.6141**	0.7059
LDA30	0.7556	**0.6453**	0.7516	0.6482	0.6969
TF-IDF	0.9700	**0.6254**	0.7694	0.6393	0.9216
	Affinity Propagation for 30 clusters				
LDA5	0.6914	**0.6788**	0.8238	0.6872	0.8958
LDA15	0.5783	**0.5762**	0.9533	0.5807	0.9706
LDA30	0.5282	**0.5080**	0.9738	0.5203	0.9912
TF-IDF	0.8697	**0.4999**	0.8375	0.5182	0.9701

The suggested lists data set contains 26 lists and thus we include the results for 25 clusters in Table IV. The best clustering result is obtained for 30 clusters, which we show in Table V. The subscribed lists data set contains 14 lists and thus we include the results for 15 clusters in Table VI as well as the best results obtained using 10 clusters in Table VII.

In each table we list the ANID for each combination of representation, clustering algorithm and similarity measure. For each combination of representation and clustering algorithm we show the best performing similarity measure in bold. The table cell for the table's best result is highlighted in orange and the overall best result for each data set is highlighted in blue.

For the suggested lists data set, we see that the best result is obtained for the cosine similarity combined with the TF-IDF finding 30 clusters with AP. The second best result is obtained for the cosine similarity combined with LDA30 and finding 25 clusters with AP. The difference in the results, 0.4999 and 0.5046, is less than half a percentage.

The results for the subscribed lists data set show that the combination of the cosine similarity, LDA15 and 10 clusters with AP achieves the best result. The second best result is obtained for 15 clusters, LDA30 and the extended Jaccard coefficient. The difference in the results, 0.6117 and 0.6409, is greater than that of the suggested lists result.

TABLE VI
A COMPARISON OF THE ADJUSTED NORMALISED INFORMATION DISTANCE RESULTS, FOR THE SUBSCRIBED LISTS DATA SET, WITH 15 CLUSTERS.

	Adjusted Normalised Information Distance				
Data set	Euclidean	Cosine	Pearson	Jaccard	KL
	K-means for 15 clusters				
LDA5	0.6880	0.6776	0.7165	**0.6755**	0.7148
LDA15	0.7151	0.7003	0.7566	**0.6933**	0.7357
LDA30	0.7496	0.7269	0.7547	**0.7151**	0.7530
TF-IDF	0.9097	**0.7149**	0.7900	0.7321	0.9620
	Affinity Propagation for 15 clusters				
LDA5	0.6879	**0.6688**	0.8321	0.6803	0.8840
LDA15	0.6558	**0.6555**	0.9427	0.6588	0.9618
LDA30	0.6543	0.6426	0.9418	**0.6409**	0.9878
TF-IDF	0.8724	0.6872	0.9331	**0.6711**	0.9767

TABLE VII
A COMPARISON OF THE ADJUSTED NORMALISED INFORMATION DISTANCE RESULTS, FOR THE SUBSCRIBED LISTS DATA SET, WITH 10 CLUSTERS.

	Adjusted Normalised Information Distance				
Data set	Euclidean	Cosine	Pearson	Jaccard	KL
	K-means for 10 clusters				
LDA5	0.6667	**0.6338**	0.7189	0.6431	0.7206
LDA15	0.7227	**0.6763**	0.7657	0.7015	0.7468
LDA30	0.7756	0.7130	0.7436	**0.7104**	0.7498
TF-IDF	0.9633	**0.7371**	0.8215	0.7396	0.9710
	Affinity Propagation for 10 clusters				
LDA5	0.6560	0.6420	0.8809	**0.6417**	0.8786
LDA15	0.6988	**0.6117**	0.9420	0.6538	0.9279
LDA30	**0.7028**	0.7217	0.9586	0.7308	0.9269
TF-IDF	0.9941	**0.7918**	0.9492	0.8070	0.9736

We see that for both data sets, the cosine similarity tends to perform the best. We include two graphs, Fig. 3 and 4, which plot the ANID obtained by the cosine similarity against the number of clusters for each combination of representation and clustering algorithm.

We observe from Fig. 3 and 4 that the results tend to be similar around the cluster sizes further away from the correct number of clusters. The difference in the results closer to the correct number of clusters for a data set is greater than for the other sizes. Fig. 4 illustrates the tendency with a definitive gap visible between the results obtained with k-means and AP closer to the correct number of clusters.

Next, we interpret our results for the similarity measures, representation and clustering algorithms.

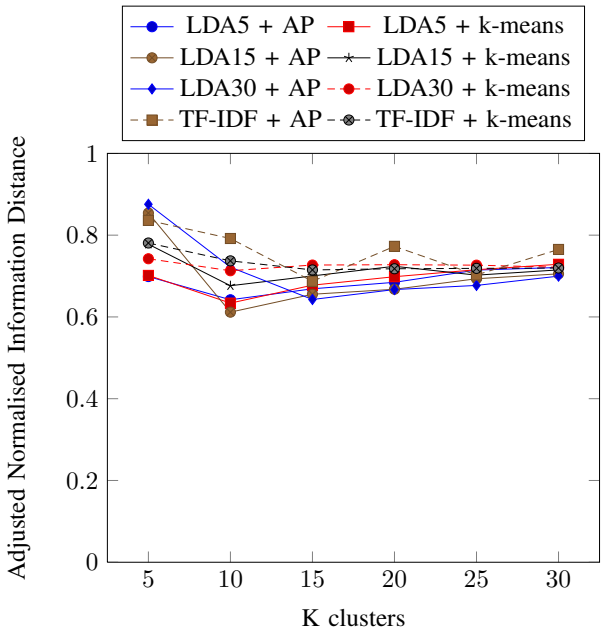

Fig. 3. A plot of the ANID, for the subscribed lists data set, obtained by the cosine similarity against the number of clusters for each combination of representation and clustering algorithm.

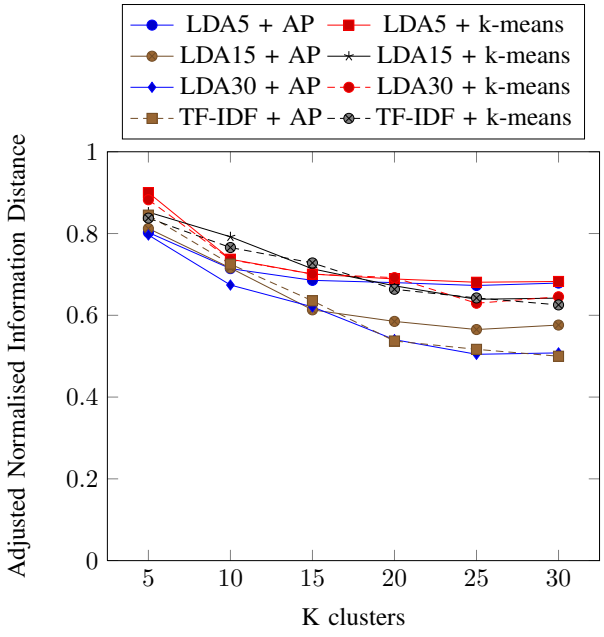

Fig. 4. A plot of the ANID, for the suggested lists data set, obtained by the cosine similarity against the number of clusters for each combination of representation and clustering algorithm.

1) Similarity Measures: In terms of the performance of the similarity measures, we see that the cosine similarity and extended Jaccard coefficient tend to perform better than the other similarity measures. The ANIDs achieved for the cosine similarity and extended Jaccard coefficient generally differ by less than 0.05, with a maximum difference of 0.09.

We see that the Pearson correlation coefficient and Kullback-Leibler divergence perform mostly worse than the cosine similarity and extended Jaccard coefficient. If we compare the results of the Euclidean distance in combination with TF-IDF we see that the result is poor, we also see that combined with LDA the performance is much improved. The poor performance of the Euclidean distance can be attributed to the relatively small number of features shared between document vectors.

An interesting result is the poor performance of the Pearson correlation coefficient and Kullback-Leibler divergence for the AP clustering algorithm even though their result for k-means is not much worse when compared to the other measures.

If a similarity measure is able to clearly distinguish clusters in a group of points, we would expect a histogram of pairwise similarity values for these points to be multimodal. However, the pairwise similarity values of the Pearson correlation coefficient and Kullback-Leibler divergence are concentrated in a small range. The evidence suggest that k-means performs better than AP when data points are relatively close to each other.

2) Representation: If we compare the document representation techniques, we observe that the LDA representation tends to outperform the TF-IDF representation if the number of topics is close to the correct number of lists in the data set. The TF-IDF performs exceptionally well for the suggested list data set, where only the performance of the LDA30 is comparable. This is in contrast with the results for the subscribed lists data set, where the TF-IDF is outperformed by most of the different topic sizes for k-means and affinity propagation. The surprisingly good performance of TF-IDF on the suggested list data set could be evidence that Twitter [31] uses a representation similar to the TF-IDF to generate these lists.

3) Clustering Algorithms: Finally, in terms of clustering algorithms, we see that AP consistently achieves better results than k-means for the Euclidean distance, cosine similarity and extended Jaccard coefficient. For the Pearson correlation coefficient and Kullback-Leibler divergence, k-means tend to perform better than AP. We see that the improved performance of AP over k-means is more pronounced close to the correct number of clusters for each data set.

4) Further Analysis: We have mentioned earlier that the best result for the suggested lists data set was achieved for the TF-IDF using 30 AP clusters, while LDA30 using 25 AP clusters was a close second. In order to get a intuitive feel for the quality of each clustering and to make a better decision about which one performs best, we analyse the result further in terms of the purity and entropy.

A purity of 0.5947 and entropy of 0.4406 is achieved for

the TF-IDF using 30 AP clusters with an ANID of 0.4999. For LDA30 using 25 AP clusters with an ANID of 0.5046 a purity of 0.5427 and entropy of 0.4577 is achieved. It is clear from our further analysis that the TF-IDF representation combined with 30 AP clusters achieves a better clustering result.

We perform the same analysis on the best result for the subscribed lists data set. A purity of 0.6162 and entropy of 0.4652 is achieved which in combination with an ANID of 0.6117 cements the result.

TABLE VIII
DESCRIPTION OF EACH CLUSTER ACCORDING TO THE MOST PROMINENT HASHTAGS.

Cluster	Size	Keywords
4. programmers	170	#devsummit, #inday, #neo4j, #sxsw, #psw12
0. team	414	#fb, #12for12k, #mwsf, #hackweek, #sopa
5. tech-news-brands	280	#tech, #cloud, #ipad, #iphone, #jobs
2. travel	218	#travel, #cruise, #traveltuesday, #ff, #fb
3. tech-news-brands	147	#augmented, #bbcclick, #ccevent, #wwfus, #broadband
7. programmers	134	#traveler, #icann, #flash, #io2011, #viagem
9. design	92	#freelance, #ttgtsummit, #socialtv, #design, #wordpress
1. travel	193	#travel, #ff, #tni, #ttot, #hisms
6. most-influential-in-tech	218	#bestofwikipedia, #titanic, #buffettrule, #tedmed, #sxsw
8. team	881	#travel, #sxsw, #fb, #ff, #jointheflock

We present the clustering result in Table VIII. In this table, the first column contains the cluster id, which we further label with the name of the list whose users are dominant in the cluster. Based on the purity value above, we can conclude that on average 61.62% of users in each cluster is part of the list with which the cluster is labelled. The second column lists the number of users in each cluster and the third column provides the five most frequent hashtags in each cluster. We observe from the table that even though there are 10 distinct clusters only six distinct labels are used. If we compare the result to the distribution of users in lists as shown in Table II, we see that the result is dominated by the lists containing the most users. The result is not unexpected, since clustering algorithms tend to find clusters of roughly equal size. It follows that by finding clusters containing users from the larger lists provides better results.

Based on the analysis of the results we can conclude that AP in combination with the LDA topic model, using either the cosine similarity or extended Jaccard coefficient can provide a sensible starting point for a user to create his/her Twitter lists.

VII. RELATED WORK

A number of classification [32], [33] and clustering [4], [10] tasks have been applied to short text documents from Twitter. The classification approach has been used for user and tweet recommendations [32] as well as tweet classification tasks [33]. A potential problem for classification tasks on Twitter is the limited availability of annotated data sets for training a classifier.

In our study we show that it is possible to construct topic-based Twitter lists using clustering techniques, without the necessity of an annotated training set.

The process of clustering data is affected by three decisions: how we choose to represent a document, how we calculate the document similarities and the clustering algorithm that we select. The other studies that applied clustering algorithms to short text documents have been focused on similarity measures [4], document representation [16] and clustering algorithm selection [4].

In terms of similarity measures Rangrej, Kulkarni and Tendulkar [4] analysed three clustering algorithms and two similarity measures for the task of clustering short text documents from Twitter. The selected algorithms — k-means, AP [14] and a SVD-based approached — were evaluated on a Twitter data set constructed by the authors. The data set consisted of 611 handpicked tweets and each tweet was associated with a topic. The two measures, cosine similarity and Jaccard coefficient, were used to calculate the similarity between two documents represented by a TF-IDF vector. In this study, we expand on their results by defining a new problem of clustering users and extending the number of similarity measures. We further compare TF-IDF to the LDA topic model and increase the number and size of the data sets on which we evaluate the implementation.

The way in which a short text document is represented affects the performance of clustering algorithms. Hong and Davison [16] performed a study on the different document representation techniques for short text messages. They focused on LDA and how to train the topic model on a collection of tweets. Three different techniques for training document representation were compared, namely:

- each document is a tweet;
- each document is an aggregate of user tweets; or
- each document is an aggregate of tweets sharing similar terms.

They compared each representation technique on a classification and clustering task, using the Normalised MI [15] to evaluate the results. The clustering task consisted of constructing a data set of 274 Twitter users from the categories provided by Twitter. They clustered users according to the topic with maximum value in the user document topic distribution. They found that representing each user as a document consisting of an aggregation of the users tweets achieved the best result.

We based our choice of user representation on their results and expanded the clustering process by applying clustering algorithms to the document topic distributions.

VIII. CONCLUSION

We investigated the use of unsupervised techniques to address the construction of topic-based Twitter lists. Our investigation evaluated combinations of five similarity measures, two document representation techniques and two clustering algorithms.

Our results indicated that the cosine similarity and extended Jaccard coefficient outperform the other similarity measures considered. We observed that the performance of the Pearson correlation coefficient and Kullback-Leibler divergence is worse for AP than k-means.

We found that in terms of document representation, the LDA topic model tends to outperform the TF-IDF when the number of topics is close to the number of lists in the data set. We found that for our subscribed lists data set that the LDA topic model outperforms the TF-IDF, but for the suggested lists data set the TF-IDF performs on par and in some cases better than LDA.

The results achieved by k-means and AP shows that AP outperforms k-means for the task of clustering Twitter users based on topics. We verified that clustering algorithms that achieve good results in the document domain can also achieve good clustering results for short text documents.

A number of practical issues exists, of which finding the correct number of topic and clusters to achieve the best result is of primary concern. We analysed our results over numbers of topics and clusters to find the best combination. A solution must still be implemented to find the optimal number of topics and clusters for a data set. AP has the ability to determine the number of clusters based on the self-preference parameter, but setting the self-preference is still a problem.

Nevertheless, we conclude that one can provide a Twitter user with a good starting point for creating lists by applying the AP clustering algorithm in combination with LDA and calculating similarities with the cosine similarity measure.

ACKNOWLEDGMENT

The financial assistance of MIH and the NRF towards this research is hereby acknowledged. Opinions and conclusions arrived at is those of the author and are not necessarily to be attributed to MIH or the NRF.

REFERENCES

[1] C. Borgs, J. Chayes, B. Karrer, B. Meeder, R. Ravi, R. Reagans, and A. Sayedi, "Game-theoretic models of information overload in social networks," *Algorithms and Models for the Web-Graph*, vol. 6516, pp. 146–161, 2010.

[2] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[3] G. Salton and M. McGill, *Introduction to modern information retrieval*, ser. McGraw-Hill computer science series. McGraw-Hill, 1983.

[4] A. Rangrej, S. Kulkarni, and A. Tendulkar, "Comparative study of clustering techniques for short text documents," in *Proceedings of the 20th International Conference Companion on World Wide Web*. ACM, 2011, pp. 111–112.

[5] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the sixth New Zealand Computer Science Research Student Conference*, 2008, pp. 49–56.

[6] A. Srivastava and M. Sahami, *Text Mining: Classification, Clustering, and Applications*, ser. Data Mining and Knowledge Discovery Series. CRC Press, 2009.

[7] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[8] D. Ramage, S. Dumais, and D. Liebling, "Characterizing microblogs with topic models," in *Proceedings of the International AAAI Conference on Weblogs and Social Media*. The AAAI Press, 2010.

[9] M. Pennacchiotti and S. Gurumurthy, "Investigating topic models for social media user recommendation," in *Proceedings of the 20th International Conference Companion on World Wide Web*. ACM, 2011, pp. 101–102.

[10] Y. Wu, Y. Ding, X. Wang, and J. Xu, "Topic based automatic news recommendation using topic model and affinity propagation," in *Proceedings of the Internation Conference on Machine Learning and Cybernetics*, vol. 3. IEEE, 2010, pp. 1299–1304.

[11] N. Sandhya, Y. Sri Lalitha, A. Govardan, and K. Anuradha, "Analysis of similarity measures for text clustering," *International Journal of Data Engineering*, vol. 2, no. 4, 2008.

[12] Y. Zhao and G. Karypis, "Empirical and theoretical comparisons of selected criterion functions for document clustering," *Journal of Machine Learning*, vol. 55, no. 3, pp. 311–331, 2004.

[13] P. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*, ser. Pearson International Edition. Pearson Addision Wesley, 2006.

[14] B. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[15] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[16] L. Hong and B. Davison, "Empirical study of topic modeling in twitter," in *Proceedings of the 1st Workshop on Social Media Analytics*. ACM, 2010, pp. 80–88.

[17] D. Blei and J. Lafferty, "Topic models," *Text mining: classification, clustering, and applications*, pp. 71–94, 2009.

[18] R. Subhashini and V. Kumar, "Evaluating the performance of similarity measures used in document clustering and information retrieval," in *Proceedings of the First International Conference on Integrated Intelligent Computing*. IEEE, 2010, pp. 27–31.

[19] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W. Ma, "Recommending friends and locations based on individual location history," *ACM Transactions on the Web*, vol. 5, no. 5, 2011.

[20] G. Torres, R. Basnet, A. Sung, S. Mukkamala, and B. Ribeiro, "A similarity measure for clustering and its applications," *International Journal of Electrical, Computer and Systems Engineering*, vol. 3, no. 3, 2009.

[21] T. Landauer, *Handbook of latent semantic analysis*, ser. University of Colorado Institute of Cognitive Science Series. Lawrence Erlbaum Associates, 2007.

[22] N. Ye, *The Handbook of Data Mining*, ser. Human Factors and Ergonomics. Taylor & Francis, 2004.

[23] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Workshop on Artificial Intelligence for Web Search*. AAAI, 2000, pp. 58–64.

[24] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[25] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[26] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra, "Clustering on the unit hypersphere using von Mises-Fisher distributions," *Journal of Machine Learning Research*, vol. 6, no. 2, pp. 1345–1382, 2006.

[27] A. Kraskov, H. Stögbauer, R. Andrzejak, and P. Grassberger, "Hierarchical clustering using mutual information," *EPL (Europhysics Letters)*, vol. 70, p. 278, 2005.

[28] "Listorious," http://listorious.com, 2012, [Online: Accessed 9-May-2012].

[29] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, ser. O'Reilly Series. O'Reilly, 2009.

[30] M. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 40, no. 3, pp. 211–218, 2006.

[31] J. Elman, "The power of suggestions," http://blog.twitter.com/2010/01/power-of-suggestions.html, 2010.

[32] J. Hannon, M. Bennett, and B. Smyth, "Recommending twitter users to follow using content and collaborative filtering approaches," in *Proceedings of the 4th ACM conference on Recommender systems*. ACM, 2010, pp. 199–206.

[33] K. Lee, D. Palsetia, R. Narayanan, M. Patwary, A. Agrawal, and A. Choudhary, "Twitter trending topic classification," in *Proceedings of 11th International Conference on Data Mining*. IEEE, DEC 2011, pp. 251–258.