A Novel Method for Unsupervised and Supervised Conversational Message Thread Detection

Giacomo Domeniconi[1], Konstantinos Semertzidis[2], Vanessa Lopez[3], Elizabeth M. Daly[3], Spyros Kotoulas[3] and Gianluca Moro[1]

[1]*Department of Computer Science and Engineering (DISI), University of Bologna, Bologna, Italy*

[2]*Department of Computer Science and Engineering, University of Ioannina, Ioannina, Greece*

[3]*IBM Research - Ireland Damastown Industrial Estate Mulhuddart, Dublin 15, Ireland*

{*giacomo.domeniconi, gianluca.moro*}*@unibo.it, ksemer@cs.uoi.gr,*

{*vanlopez, elizabeth.daly, spyros.kotoulas*}*@ie.ibm.com*

Keywords: Clustering Algorithms, Conversation Threads, Topic Detection.

Abstract: Efficiently detecting conversation threads from a pool of messages, such as social network chats, emails, comments to posts, news etc., is relevant for various applications, including Web Marketing, Information Retrieval and Digital Forensics. Existing approaches focus on text similarity using keywords as features that are strongly dependent on the dataset. Therefore, dealing with new corpora requires further costly analyses conducted by experts to find out new relevant features. This paper introduces a novel method to detect threads from any type of conversational texts overcoming the issue of previously determining specific features for each dataset. To automatically determine the relevant features of messages we map each message into a three dimensional representation based on its semantic content, the social interactions in terms of sender/recipients and its timestamp; then clustering is used to detect conversation threads. In addition, we propose a supervised approach to detect conversation threads that builds a classification model which combines the above extracted features for predicting whether a pair of messages belongs to the same thread or not. Our model harnesses the distance measure of a message to a cluster representing a thread to capture the probability that a message is part of that same thread. We present our experimental results on seven datasets, pertaining to different types of messages, and demonstrate the effectiveness of our method in the detection of conversation threads, clearly outperforming the state of the art and yielding an improvement of up to a 19%.

1 INTRODUCTION

In recent years, online texting has become a part of most people's everyday lives. The use of email, web chats, online conversations and social groups has become widespread. It is a fast, economical and efficient way of sharing information and it also provides users the ability to discuss different topics with different people. Understanding the context of digital conversations supports a wide range of applications such as marketing, social network extraction, expert finding, the improvement of email management, ranking content and others (Jurczyk and Agichtein, 2007; Coussement and den Poel, 2008; Glass and Colbaugh, 2010; F. M. Khan and Pottenger, 2002).

The increased use of online fora leads to people being overwhelmed by information. For example, this can happen when a user has hundreds of new unread messages in a chat or one needs to track and organise posts in forums or social groups. To instantly have a clear view of different discussions, which requires expensive and tedious efforts for a person, we need to automatically organise this data stream into threads.

There has been considerable effort in extracting topics from document sets, mainly through a variety of techniques derived from Probabilistic Latent Semantic Indexing (pLSI) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). However, the problem of detecting threads from conversational messages differs from document topic extraction for several aspects (Shen et al., 2006; Huang et al., 2012; F. M. Khan and Pottenger, 2002; Adams and Martell, 2008; Yeh, 2006): (i) conversational messages are generally much shorter than usual documents making the task of topic detection much more difficult (ii) thread detection strongly depends on social interactions between the users involved in a message exchange, (iii) as well the time of the discussion.

Domeniconi, G., Semertzidis, K., Lopez, V., Daly, E., Kotoulas, S. and Moro, G.
A Novel Method for Unsupervised and Supervised Conversational Message Thread Detection.
In *Proceedings of the 5th International Conference on Data Management Technologies and Applications (DATA 2016)*, pages 43-54
ISBN: 978-989-758-193-9

Other studies deal with thread tree reconstruction where given a known thread they seek to construct the conversation tree. In particular, in (X. Wang and Chen, 2008) the authors use headers extracted from emails to create the parent/child relationships of a thread, and in (Aumayr et al., 2011) the authors propose an algorithm which uses reply behaviours in forum threads in order to output the threads' structure.

This paper addresses the efficient detection of conversation threads from pools of online messages - for example from social groups, pages chats, email messages etc. - namely the detection of sets of messages related with respect to contents, time and involved users. The thread detection problem is different from thread tree reconstruction, since the latter requires that the conversation threads are known. In contrast the former, seeks to find the messages which form a conversation thread. In other words, thread detection is the essential initial step of thread tree reconstruction.

We consider a three dimensional representation (Zhao and Mitra, 2007) which consists of text content, temporal information, and social relations. In Figure 1, we depict the three dimensional representation which illustrates 3 threads with different colours and shapes, that yields to total of 14 messages. The green circles and red squares threads have the same social and content dimensions but not time. While the blue diamonds thread consists of different topics and users, but it occurs in the same time frame of the green circles one. The use of the three dimensional representation leads to emphasis of thread separation.

We propose several measures to exploit the messages features, based on this three dimensional representation. Then, the generated features are embedded into a metric distance in density and hierarchical clustering algorithms (Ester et al., 1996; Bouguettaya et al., 2015) which cluster messages in threads. In order to enhance our approach to efficiently detect threads in any type of dataset, we build a classification model from a set of messages previously organised in threads. The classifier exploits the same features used in the clustering phase and it returns the probability that a pair of messages belong to the same thread. In other words, a binary supervised model is trained with instances, each referring to a pair of messages. Each instance uses the same features described previously, and a label describing whether the two messages belong to the same thread or not. This model provides a probability of being in the same thread for a pair of messages, we propose to use this probability as a similarity distance in clustering methods to detect the threads. We observe that the classifiers output can help the clustering process to achieve higher accuracy

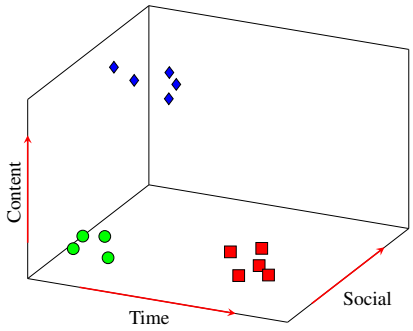

Figure 1: Three dimensional representation of threads messages.

by identifying the threads correctly. We have extensively evaluated our approach with real world datasets including emails and social group chats. Our experimental results show that our method can identify the large majority of the threads in several type of dataset, such as web conversation including emails, chats and posts.

In summary, this paper presents the following contributions:

- a three dimensional message representation based on textual semantic content, social interactions and time to generate features for each message;
- clustering algorithms to detect threads, on top of the features generated from the three dimensional representation;
- combination of the generated features to build a classifier that identifies the membership probability of pair of messages to the same thread and this probability is used as a distance function for the clustering methods to detect threads;
- the combined classification technique with clustering algorithms provides a higher accuracy than using clustering alone..

The rest of this paper is structured as follows. In Section 2, we present related work, while in Section 3, we formally define the thread detection problem. In Section 4, we introduce our model and our algorithms for thread detection. Section 5 presents the experimental results on real datasets and Section 6 concludes the paper.

2 RELATED WORK

As discussed in the previous section, thread detection has received a lot of attention, including content and metadata based approaches. Metadata based approaches refers to header fields that are contained in

emails or forum posts (e.g. send-to, reply-to). Content based approaches focus on text analysis on subject and content text. In this paper, we differentiate from the existing works by generalizing the problem of detecting threads in different types of datasets, not only in email sets like the most of related work (Wu and Oard, 2005; Yeh, 2006; X. Wang and Chen, 2008; Erera and Carmel, 2008; Joshi et al., 2011). The authors of (Wu and Oard, 2005) focus on detecting conversation threads from emails using only the subject. They cluster all messages with the same subject and at least one participant in common. Here, we also handle cases where messages belong to the same thread but have different subject. Similarly, in (X. Wang and Chen, 2008) the authors detect threads in emails using the extracted header information. They first try to detect the parent/child relationships using Zawinski algorithm [1] and then they use a topic-based heuristic to merge or decompose threads to conversations. Another approach for detecting threads in emails is proposed in (Erera and Carmel, 2008), where clustering into threads exploits a similarity function that considers all relevant email attributes, such as subject, participants, text content and date of creation. Quotations are taken into account in (Yeh, 2006) where combined with several heuristics such as subject, sender/recipient relationships among email and time, and as a result can construct email threads with high precision. Emails relationships are also considered in (Joshi et al., 2011) where the authors use a segmentation and detection of duplicate emails and they group them together based on reply and forwarding relationships.

The work most closely related to ours is that of (Dehghani et al., 2013), that studies the conversation tree reconstruction, by first detecting the threads from a set of emails. Specifically, they map the thread detection problem to a graph clustering task. They create a semantic network of a set of emails where the nodes denote emails and the weighted edges represent co-thread relationships between emails. Then, they use a clustering method to extract the conversation threads. However, their approach is focus only on email datasets and their results are strongly bound with the used features, since when they do not take into account all features they have a high reduction in their accuracy. In contrast here, we consider general datasets and by using our classification model we are able to detect threads even when there are missing features. Although, it is not clear which graph clustering algorithm is used and how it detects the clusters. We conduct an extensive comparison between our approach and the study of (Dehghani et al., 2013) in Section 5.

Another line of research addresses mining threads from online chats (F. M. Khan and Pottenger, 2002; Adams and Martell, 2008; Huang et al., 2012; Shen et al., 2006). Specifically, the study of (F. M. Khan and Pottenger, 2002) focuses on identifying threads of conversation by using pattern recognition techniques in multi-topic and multi-person chat-rooms. In (Adams and Martell, 2008) they focus on conversation topic thread detection and extraction in a chat session. They use an augmented $tf.idf$ to compute weights between messages' texts as a distance metric exploiting the use of Princeton WordNet[2] ontology, since related messages may not include identical terms, they may in fact include terms that are in the same semantic category. In combination with the computed distance between messages they use the creation time in order to group messages with high similarity in a short time interval. In (Shen et al., 2006), they propose three variations of a single-pass clustering algorithm for exploiting the temporal information in the streams. They also use an algorithm based on linguistic features in order to exploit the discourse structure information. A single-pass clustering algorithm is also used in (Huang et al., 2012) which employs the contextual correlation between short text streams. Similar to (Adams and Martell, 2008), they use the concept of correlative degree, which describes the probability of the contextual correlation between two messages, and the concept of neighboring co-occurrence, which shows the number features co-existing in both messages.

Finally, there also exists a line of research on reconstructing the discussion tree structure of a thread conversation. In (Wang et al., 2011), a probabilistic model in conditional random fields framework is used to predict the replying structure for online forum discussions. The study in (Aumayr et al., 2011) employs conversation threads to improve forum retrieval. Specifically, they use a classification model based on decision trees and given a variety of features, including creation time, name of authors, quoted text content and thread length, which allows them to recover the reply structures in forum threads in an accurate and efficient way. The aforementioned works achieve really high performance (more than 90% of accuracy) in the conversation tree reconstruction, while the state of the art in threads detection obtains lower performance, about 80% for emails data and 60% for chats and short messages data. To this end, in this study we focus on improving thread detection performance.

[1] https://www.jwz.org/doc/threading.html

[2] http://wordnet.princeton.edu/

3 METHOD DESCRIPTION

In this section, we outline a generic algorithm for detecting messages which belong to the same thread from a set of messages $\mathcal{M}$, such as emails, social group posts and chats. As an intermediate step, the algorithm addresses the problem of computing the similarity measure between pairs of messages. We propose a suite of features and two methods to combine them (one unsupervised and one supervised) to compute the similarity measure between two messages. We also present clustering algorithms which detect threads based on this similarity measure in Section 3.3.

3.1 Data Model

We consider a set of messages $\mathcal{M} = \{m_1, m_2, ...\}$ that refers to online texts such as emails, social group chats or forums. Each message is characterized by the following properties: (1) textual data (content and subject in case of emails), (2) creation time, and (3) the users involved (authors or sender/recipients in case of emails). We represent each message as a three-dimensional model (Zhao and Mitra, 2007; Zhao et al., 2007) to capture all these components. Thus, a message $m \in \mathcal{M}$ can be denoted as a triplet $m = <c_m, \mathcal{U}_m, t_m>$, where c_m refers to text content, $\mathcal{U}_m = \{u_1, u_2, ...\}$ refers to the set of users that are involved in m, and t_m refers to the creation time. Some dimensions can be missing, for instance chat, groups and forum messages provide only the author information, without any recipients.

A conversation thread is defined as a set of messages exchanged on the same topic among the same group of users during a time interval, more formally, the set of messages $\mathcal{M}$ is partitioned in a set of conversations $\mathcal{C}$. Each message $m \in \mathcal{M}$ belongs to one and only one conversation $c \in \mathcal{C}$. The goal of the thread reconstruction task is to automatically detect the conversations within a pool of messages. To this aim, we propose a clustering-based method that relies on a similarity measure between a pair of messages, called $SIM(m_i, m_j)$. In the following sections, we define different proposed approaches to calculate the similarity measure. In the rest of the paper, we will use the notation $\Omega = \{\omega_1, \omega_2, ...\}$ to refer the predicted extracted conversations.

3.2 Messages Features

Social text messages, like emails or posts, can be summarized by three main components: text content, temporal information, and social relations (Zhao and Mitra, 2007). Each of the three main components can be analyzed under different points of view to compute the distance between a pair of messages, which involves the creation of several features. The function $SIM(m_i, m_j)$ relies on these features and returns a similarity value for each pair of messages (m_i, m_j), which is used by the clustering algorithm that returns the finding threads. We now present the extracted features used to measure the similarity between two messages.

The content component relies on the semantics of the messages. There are two main sources: the messages text and the subject, if present (e.g., social network posts do not have this information). The first considered feature is the similarity of the messages text content. We make use of the common *Bag of Words* (*BoW*) representation, that describes a textual message m by means of a vector $\mathcal{W}(m) = \{w_1, w_2, ...\}$, where each entry indicates the presence or absence of a word w_i. Single words occurring in the message text are extracted, discarding punctuation. A stopwords list is used to filter-out all the words that are not informative enough. The standard Porter stemming algorithm (Porter, 1980) is used to group words with a common stems. To estimate the importance to each word, there exist several different weighting schemes (Domeniconi et al., 2016), here we make use of the commonly used *tf.idf* scheme (Salton and Buckley, 1988).

Using *BoW* representation, the similarity between two vectors m_i, m_j can be measured by means of the commonly used *cosine similarity* (Singhal, 2001):

$$f_{C_T}(m_i, m_j) = \frac{\mathcal{W}(m_i) \cdot \mathcal{W}(m_j)}{\|\mathcal{W}(m_i)\| \|\mathcal{W}(m_j)\|}$$

Since by definition the *BoW* vectors have only positive values, the $f_{C_T}(m_i, m_j)$ takes values between zero and one, being zero if the two vectors do not share any word, and one if the two vectors are identical. In scenarios where the subject is available, the same process is carried out, computing the similarity cosine $f_{C_S}(m_i, m_j)$ of words contained in the messages subject.

The cosine similarity allows a lexical comparison between two messages but does not consider the semantic similarity between two messages. There are two main shortcomings of this measure: the lack of focus on keywords, or semantic concepts expressed by messages, and the lack of recognition of lexicographically different words but with similar meaning (i.e. synonyms), although this is partially computed through the stemming. In order to also handle this aspect, we extend the text similarity by measuring the correlation between entities, keywords and con-

cepts extracted using AlchemyAPI [3]. AlchemyAPI is a web service that analyzes the unstructured content, exposing the semantic richness in the data. Among the various information retrieved by AlchemyAPI, we take into consideration the extracted topic keywords, involved entities (e.g. people, companies, organizations, cities and other types of entities) and concepts which are the abstractions of the text (for example, "My favorite brands are BMW and Porsche = "Automotive industry"). These three information are extracted by Alchemy API with a confidence value ranging from 0 to 1. We create three vectors, one for each component of the Alchemy API results for keywords, entities and concepts for each message and using the related confidence extracted by AlchemyAPI as weight. Again we compute the cosine similarity of these vectors, creating three novel features:

- $f_{C_K}(m_i, m_j)$: computes the cosine similarity of the keywords of m_i and m_j. This enables us to quantify the similarity of the message content based purely on keywords rather than the message as a whole.
- $f_{C_E}(m_i, m_j)$: computes the cosine similarity of the entities that appear in m_i and m_j focusing on the entities shared by the two messages.
- $f_{C_C}(m_i, m_j)$: computes the cosine similarity of the concepts in m_i and m_j, allowing the comparison of the two messages on a higher level of abstraction: from words to the expressed concepts.

The second component is related to the social similarity. For each message m, we create a vector of involved users $\mathcal{U}(m) = \{u_1, u_2, ...\}$ defined as the union of the sender and the recipients of m (note that the recipients information is generally not provided in social network posts). We exploit the social relatedness of two messages through two different features:

- The similarity of the users involved in the two messages $f_{S_U}(m_i, m_j)$, defined as the Jaccard similarity between $\mathcal{U}(m_i)$ and $\mathcal{U}(m_j)$:

$$f_{S_U}(m_i, m_j) = \frac{|\mathcal{U}(m_i) \cap \mathcal{U}(m_j)|}{|\mathcal{U}(m_i) \cup \mathcal{U}(m_j)|}$$

- The neighborhood Jaccard similarity $f_{S_N}(m_i, m_j)$ of the involved users. The neighborhood set $\mathcal{N}(u)$ of an user u is defined as the set of users that have received at least one message from u. We also include each user u in its neighborhood $\mathcal{N}(u)$ set. The neighborhood similarity of two messages m_i and m_j is defined as follows:

$$f_{S_N}(m_i, m_j) = \frac{1}{|\mathcal{U}(m_i)||\mathcal{U}(m_j)|} \sum_{\substack{u_i \in \mathcal{U}(m_i) \\ u_j \in \mathcal{U}(m_j)}} \frac{|\mathcal{N}(u_i) \cap \mathcal{N}(u_j)|}{|\mathcal{N}(u_i) \cup \mathcal{N}(u_j)|}$$

[3] http://www.alchemyapi.com/

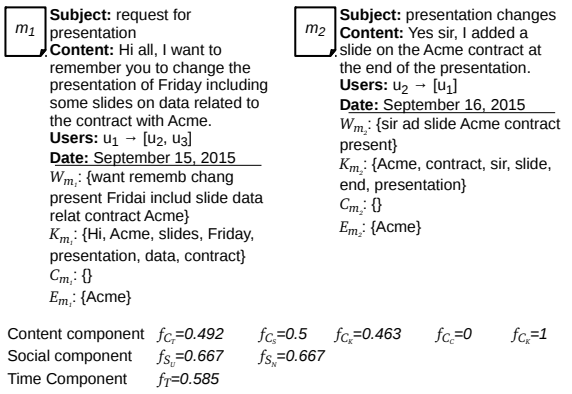

Figure 2: Example of features calculation for a pair of messages. Message components: Subject, Content, Users (sender → recipients) and creation date. $\mathcal{W}(m_i)$ refers to the bag of words of a message obtained after the tokenization, stopwords removal and stemming. The vectors of keywords ($\mathcal{K}(m_i)$), concepts ($\mathcal{C}(m_i)$) and entities ($\mathcal{E}(m_i)$) extracted from AlchemyAPI are shown. In the bottom the values for each proposed feature are also shown. For simplicity, we assume binary weight for components.

Finally, the last component relies on the time of two messages. We define the time similarity as the logarithm of the inverse of the distance between the two messages, expressed in days, as follows:

$$f_T(m_i, m_j) = \log_2(1 + \frac{1}{1 + |t_{m_i} - t_{m_j}|})$$

We use the inverse normalization of the distance in order to give a value between zero and one, where zero correspond to a high temporal distance and one refers to messages with low distance.

As a practical example, Figure 2 shows two messages, with the related properties, and the values of the features generated from them.

3.3 Clustering

In this section, we present the clustering methods used to detect the threads. Based on the set of aforementioned features $\mathcal{F} = \{f_{C_T}, f_{C_S}, f_{C_K}, f_{C_E}, f_{C_C}, f_{S_U}, f_{S_N}, f_T\}$, we define a distance measure that quantifies the similarity between two messages:

$$SIM(m_i, m_j) = \Pi_{f \in \mathcal{F}}(1 + f(m_i, m_j)) \qquad (1)$$

We compute a $N \times N$ matrix with the similarities between each pair of messages (m_i, m_j) and we use density based and hierarchical clustering algorithms, being the two most common distance-based approaches.

3.3.1 Density-based Clustering

We use the DBSCAN (Ester et al., 1996) density-based clustering algorithm in order to cluster messages to threads because given a set of points in some space, DBSCAN groups points that are closely packed together (with many nearby neighbors). DBSCAN requires two run time parameters, the minimum number *min* of points per cluster, and a threshold θ that defines the neighborhood distance between points in a cluster. The algorithm starts by selecting an arbitrary point, which has not been visited, and by retrieving its θ-neighborhood it creates a cluster if the number of points in that neighborhood is equals to or greater than *min*. In situations where the point resides in a dense part of an existing cluster, its θ-neighbor points are retrieved and are added to the cluster. This process stops when the densely-connected cluster is completely found. Then, the algorithm processes new unvisited points in order to discover any further clusters.

In our study, we use messages as points and we use weighted edges that connect each message to the other messages. An edge (m_i, m_j) between two messages m_i and m_j is weighted with the similarity measure $SIM(m_i, m_j)$. When DBSCAN tries to retrieve the θ-neighborhood of a message m, it gets all messages that are adjacent to m with a weight in their edge greater or equal to θ. Greater weight on an edge indicates that the connected messages are more similar, and thus they are closer to each other.

3.3.2 Hierarchical Clustering

This approach uses the Agglomerative hierarchical clustering method (Bouguettaya et al., 2015) where each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. Running the agglomerative method requires the choice of an appropriate linkage criteria, which is used to determine the distance between sets of observations as a function of pairwise distances between clusters that should be merged or not. In our study we examined, in preliminary experiments, three of the most commonly used linkage criteria, namely the *single*, *complete* and *average linkage* (Manning et al., 2008). We observed that average linkage clustering leads to the best results. The average linkage clustering of two clusters of messages Ω_y and Ω_z is defined as follows:

$$avgLinkCl(\Omega_y, \Omega_z) = \frac{1}{|\Omega_y||\Omega_z|} \sum_{\substack{\omega_i \in \Omega_y \\ \omega_j \in \Omega_z}} SIM(\omega_i, \omega_j)$$

The agglomerative clustering method is an iterative process that merges the two clusters with highest average linkage score. After each merge of the clusters, the algorithm starts by recomputing the new average linkage scores between all clusters. This process runs until a cluster pair exists with a similarity greater than a given threshold.

3.4 Classification

The clustering algorithms described above rely on the similarity measure *SIM*, that combines with a simple multiplication several features, to obtain a single final score. This similarity measure in eq. 1 gives the same weight, namely importance, to each feature. This avoids the requirement to tune the parameters related to each feature, but could provide an excessively rough evaluation and thus bad performance. A different possible approach, is to combine the sub components of similarity measure *SIM* as features into a binary supervised model, in which each instance refers to a pair of messages, the features are the same described in the Section 3.2 and the label is one if the messages belonging to the same thread and zero otherwise. At runtime, this classifier is used to predict the probability that two messages belong to the same thread, using this probability as the distance between the pairs of messages into the same clustering algorithms. The benefit of such approach is that it automatically finds the appropriate features to use for each dataset and it leads to a more complete view of the importance of each feature. Although it is shown in (Aumayr et al., 2011) that decision trees are faster and more accurate in classifying text data, we experimented with a variety of classifiers.

The classification requires a labeled dataset to train a supervised model. The proposed classifier relies on data in which each instance represents a pair of messages. Given a set of training messages $\mathcal{M}_{Tr}$ with known conversation subdivision, we create the training set coupling each training message $m \in \mathcal{M}_{Tr}$ with n_s messages of $\mathcal{M}_{Tr}$ that belong to the same thread of m and n_d messages belonging to different threads. We label each training instance with one if the corresponding pair of messages belong to same thread and zero otherwise. Each of these coupled messages are picked randomly. Theoretically we could create $(|\mathcal{M}_{Tr}| \cdot |\mathcal{M}_{Tr} - 1|)/2$ instances, coupling each message with the whole training set. In preliminary tests using *Random Forest* as the classification model, we notice that coupling each training message with a few dozen same and different messages can attain higher performances. All the experiments are conducted using $n_s = n_d = 20$, i.e. each message is coupled with at maximum 20 messages of the same conversation

Table 1: Characteristics of datasets.

Dataset	Messages type	#messages	#threads	#users	Peculiarities
BC3	Emails	261	40	159	Threads contain emails with different subject
Apache	Emails from mailing list	2945	334	113	Threads always contain emails with same subject
Redhat	Emails from mailing list	12981	802	931	Threads always contain emails with same subject
WhoWorld	Posts from Facebook page	2464	132	1853	Subject and recipients not available
HealthyChoice	Posts from Facebook page	1115	132	601	Subject and recipients not available
Healthcare Advice	Posts from Facebook group	3436	468	801	Subject and recipients not available
Ireland S. Android	Posts from Facebook group	4831	408	354	Subject and recipients not available

and 20 of different ones. In the rest of the paper we refer to the proposed clustering algorithm based on a supervised model, as *SVC*.

As it will be shown in the Section 4.3, the Agglomerative hierarchical clustering achieves better results with respect to the DBSCAN, thus, we use this clustering algorithm in the *SVC* approach.

4 EVALUATION

In this section, we compare the accuracy of the clustering methods described in Section 3 in terms of detecting the actual threads.

4.1 Datasets

For evaluating our approach we consider the following seven real datasets:

- The *BC3* dataset (Ulrich et al., 2008), which is a special preparation of a portion W3C corpus (Soboroff et al., 2006) that consists of 40 conversation threads. Each thread has been annotated by three different annotators, such as extractive summaries, abstractive summaries with linked sentences, and sentences labeled with speech acts, meta sentences and subjectivity.
- The *Apache* dataset which is a subset of Apache Tomcat public mailing list[4] and it contains the discussions from August 2011 to March 2012.
- The *Redhat* dataset which is a subset of Fedora Redhat Project public mailing list[5] and it contains the discussions that took place in the first six months of 2009.
- Two Facebook pages datasets, namely *Healthy Choice*[6] and *World Health Organizations*[7], crawled using the Facebook API[8]. They consist of real posts and relative replies between June and August 2015. We considered only the text content of the posts (discarding links, pictures, videos, etc.) and only those written in English (AlchemyAPI is used to detect the language).
- Two Facebook public groups datasets, namely *Healthcare Advice*[9] and *Ireland Support Android*[10], also crawled using the Facebook API. They consist of conversations between June and August 2015. Also for this dataset we considered only the text content of the posts written in english.

We use the first three datasets that consist of emails in order to compare our approach with existing related work (Dehghani et al., 2013; Erera and Carmel, 2008; Wu and Oard, 2005) on conversation thread reconstruction in email messages. To our knowledge, there are no publicly available datasets of social network posts with a gold standard of conversation subdivision. We use the four Facebook datasets to evaluate our method in a real social network domain.

The considered datasets have different peculiarities, in order to evaluate our proposed method under several perspectives. *BC3* is a quite small dataset (only 40 threads) of emails, but with the peculiarity of being manually curated. In this dataset is possible to have emails with different subjects in the same conversation. However, in *Apache* and *Redhat* the messages in the same thread, have also the same subject.

With regards to Facebook datasets, we decided to use both pages and groups. Facebook pages are completely open for all users to read and comment in a conversation. In contrast, only the members of a group are able to view and comment a group post and this leads to a peculiarity of different social interaction nets. Furthermore, each message - post - in these datasets has available only the text content, the sender and the time, without information related to subject and recipients. Thus, we do not take into account the similarities that use the recipients or subject. Table 1 provides a summary of the characteristics of each dataset.

[4]http://tomcat.apache.org/mail/dev
[5]http://www.redhat.com/archives/fedora-devel-list
[6]https://www.facebook.com/healthychoice
[7]https://www.facebook.com/WHO
[8]https://developers.facebook.com/docs/graph-api
[9]https://www.facebook.com/groups/533592236741787
[10]https://www.facebook.com/groups/848992498510493

In the experiments requiring a labeled set to train a supervised model, the datasets are evaluated with 5-fold cross-validation, subdividing each of those in 5 thread folds.

4.2 Evaluation Metrics

The *precision*, *recall* and F_1*-measure* (Manning et al., 2008) are used to evaluate the effectiveness of the conversation threads detection. Here, we explain these metrics in the context of the conversation detection problem. We evaluate each pair of messages in the test set. A true positive (TP) decision correctly assigns two similar messages to the same conversation. Similarly, a true negative (TN) assigns two dissimilar messages to different threads. A false positive (FP) case would be when the two messages do not belong to the same thread but are labelled as co-threads in the extracted conversations. Finally, false negative (FN) case is when the two messages belong to the same thread but are not co-threads in the extracted conversations. Precision (p) and recall (r) are defined as follows:

$$p = \frac{TP}{TP+FP} \qquad r = \frac{TP}{TP+FN}$$

The F_1*-measure* is defined by combining the precision and recall together, as follows:

$$F_1 = \frac{2 \cdot p \cdot r}{p+r}$$

We also use the *purity* metric to evaluate the clustering. The dominant conversation, i.e. the conversation with the highest number of messages inside a cluster, is selected from each extracted thread cluster. Then, purity is measured by counting the number of correctly assigned messages considering the dominant conversation as cluster label and finally dividing by the number of total messages. We formally define purity as

$$purity(\Omega, C) = \frac{1}{|\mathcal{M}|} \sum_k \max_j |\omega_k \in c_j|$$

where $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ is the set of extracted conversations and $C = \{c_1, c_2, ..., c_j\}$ is the set of real conversations.

To better understand the purity metric, we refer to the example of thread detection depicted in Figure 3. For each cluster, the dominant conversation and the number of related messages are: $\omega_1 : c_1, 4$, $\omega_2 : c_2, 4$, $\omega_3 : c_3, 3$. The total number of messages is $|\mathcal{M}| = 17$. Thus, the purity value is calculated as $purity = (4 + 4 + 3)/17 = 0.647$.

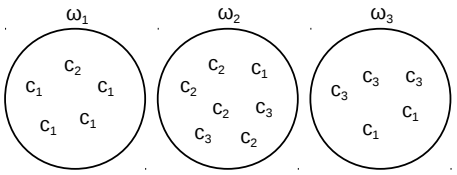

Figure 3: Conversation extraction example. Each ω_k refers to an extracted thread and each c_j corresponds to the real conversation of the message.

A final measure of the effectiveness of the clustering method, is the simple comparison between the the number of detected threads ($|\Omega|$) against the number of real conversations ($|C|$).

4.3 Results

In Table 2 and 3, we report the results obtained in the seven datasets. All the results in these tables are obtained using the Weka (Hall et al., 2009) implementation of *Random Forest* algorithm, using a 2×2 cost matrix with a weight of 100 for the instances labeled with one. The reported results are related to the best tuning of the threshold parameter of the clustering approaches, both for DBSCAN and Agglomerative. Further analysis on the parameters of our method will be discussed in the next section.

Table 2 shows the results on the email datasets, on which we can compare our results (SVC) with other existing approaches, such as the studies of Wu and Oard (Wu and Oard, 2005), Erera and Carmel (Erera and Carmel, 2008) and the lastest one of Dehghani et al (Dehghani et al., 2013). The first two approaches (Wu and Oard, 2005; Erera and Carmel, 2008) are unsupervised, as the two clustering baselines, while the approach in (Dehghani et al., 2013) is supervised, like our proposed SVC; both this supervised methods are evaluated with the same 5-fold cross-validation, described above. All of the existing approaches use the information related to the subject of the emails, we show in the top part of the table a comparison using also the subject as feature in our proposed approach. We want point out that in *Apache* and *Redhat* dataset, the use of the subject could make the clusterization effortless, since all messages of a thread have same subject. It is notable how our supervised approach obtains really high results, reaching almost perfect predictions and always outperforming the existing approaches, particularly in *Redhat* and *Apache* dataset. In our view, the middle of Table 2 is of particular interest, where we do not considered the subject information. The results, especially in *Redhat* and *Apache*, have a little drop, remaining anyhow at high levels, higher than all existing approaches that take into consideration the subject. Including the subject

Table 2: Conversation detection results on email datasets. (Wu and Oard, 2005; Erera and Carmel, 2008), DBSCAN and Agglom. are unsupervised methods, while (Dehghani et al., 2013) and SVC are supervised. The top part of the table shows the results obtained by methods using subject information, the middle part shows those achieved without such feature, finally the bottom part shows the results obtained with SVC method considering only a single dimension. With + and – we indicate respectively the use or not of the specified feature (s: subject feature, a: the three Alchemy features). For clustering and SVC approach we report results with best threshold tuning.

	BC3			Apache			Redhat		
Methods	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
Wu and Oard (Wu and Oard, 2005)	0.601	0.625	0.613	0.406	0.459	0.430	0.498	0.526	0.512
Erera and Carmel (Erera and Carmel, 2008)	0.891	0.903	0.897	0.771	0.705	0.736	0.808	0.832	0.82
Dehghani et al. (Dehghani et al., 2013)	0.992	0.972	0.982	0.854	0.824	0.839	0.880	0.890	0.885
DBSCAN (+s)	0.871	0.737	0.798	0.359	0.555	0.436	0.666	0.302	0.416
Agglom. (+s)	**1.000**	0.954	0.976	0.358	0.918	0.515	0.792	0.873	0.83
SVC (+s)	**1.000**	**0.986**	**0.993**	**0.998**	**1.000**	**0.999**	**0.995**	**0.984**	**0.989**
DBSCAN (–s)	0.696	0.615	0.653	0.569	0.312	0.403	0.072	0.098	0.083
Agglom. (–s)	**1.000**	0.954	**0.976**	0.548	0.355	0.431	0.374	0.427	0.399
SVC (–s)	**1.000**	0.952	0.975	**0.916**	0.972	**0.943**	**0.966**	**0.914**	**0.939**
SVC (–s –a)	0.967	**0.979**	0.973	0.892	**0.994**	0.940	0.815	0.699	0.753
SVC (content)	**1.000**	**0.919**	**0.958**	**0.954**	**0.974**	**0.964**	**0.988**	**0.984**	**0.986**
SVC (content –s)	0.964	0.902	0.932	0.604	0.706	0.651	0.899	0.872	0.885
SVC (content –s –a)	**1.000**	0.828	0.905	0.539	0.565	0.552	0.68	0.558	0.613
SVC (social)	0.939	0.717	0.813	0.345	0.361	0.353	0.360	0.045	0.08
SVC (time)	0.971	0.897	0.933	0.656	0.938	0.772	0.376	0.795	0.511

Table 3: Conversation detection results on Facebook post datasets (subject and recipient information are not available). The top part of the table shows the results obtained considering all the dimensions, the bottom part shows the results obtained with SVC method considering only a single dimension. For clustering and our approach we report results with best threshold tuning.

	Healty Choice			World Health Org.			Healthcare Advice			Ireland S. Android		
Methods	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
DBSCAN	0.027	0.058	0.037	0.159	0.043	0.067	0.206	0.051	0.082	0.201	0.002	0.004
Agglom.	0.228	0.351	0.276	0.154	0.399	0.223	0.429	0.498	0.461	0.143	0.141	0.142
SVC	**0.670**	0.712	**0.690**	**0.552**	0.714	0.623	**0.809**	0.721	0.763	0.685	0.655	0.67
SVC (–a)	0.656	**0.713**	0.683	0.543	**0.742**	**0.627**	0.802	**0.733**	**0.766**	**0.708**	**0.714**	**0.711**
SVC (content)	0.308	0.032	0.058	0.406	0.120	0.185	0.443	0.148	0.222	0.127	0.042	0.063
SVC (content –a)	0.286	0.025	0.046	0.376	0.11	0.171	0.414	0.127	0.195	0.105	0.033	0.050
SVC (social)	0	0	0	0	0	0	0.548	0.188	0.280	0.155	0.234	0.186
SVC (time)	**0.689**	**0.670**	**0.679**	**0.531**	**0.750**	**0.622**	**0.638**	**0.769**	**0.697**	**0.667**	**0.703**	**0.685**

or not, the use of a supervised model to evaluate the similarity between two messages, brings a great improvement to the clustering performances, compared to the use of a simple combination of each feature as described in Section 3.3. In the middle part of Table 2 is also shown the effectiveness of our SVC predictor without the three features related to AlchemyAPI information; these features lead to an improvement of results especially in *Redhat*, which is the largest and more challenging dataset.

The aforementioned considerations, are valid also for the experiments on social network posts. To the best of our knowledge, there is not any related work on such type of datasets. In Table 3, we report the results of our approach on the four Facebook datasets. These data do not provide the subject and recipients information of messages, thus the reported results are obtained without the features related to the subject and neighborhood similarities, namely $f_{C_S}(m_i, m_j)$ and $f_{S_N}(m_i, m_j)$. We notice that the pure unsupervised clustering methods, particularly DBSCAN, achieve

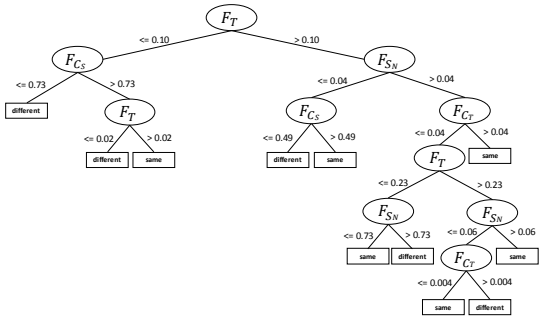

Figure 4: Decision trees created for *BC3* dataset.

low *precision* and *recall*. This is due to the real difficulties of these post's data: single posts are generally short with little semantic information. For example suppose we have two simultaneous conversations *t1: "How is the battery of your new phone?" - "good!"* and *t2: "how was the movie yesterday?" - "awesome!"*. By using only the semantic information of the content, it is not possible to associate the

replies to the right question, thus the time and the social components become crucial. Although there is a large amount of literature to handle grammatical errors or misspelling, in our study we have not taken into account these issues. Despite these difficulties, our method guided by a supervised model achieves quite good results in such data, with an improvement almost always greater than 100% with respect the pure unsupervised clustering. Results in Table 3 show the difficulties also for AlchemyAPI to extract valuable information from short text posts. In fact, results using the AlchemyAPI related features does not lead to better results.

The results achieved by the SVC method for each dimension are reported at the bottom of the Tables 2 and 3, in particular those regarding the content dimension have been produced with all features, excepts the subject and the Alchemy related features. In Table 2 is notable that considering the content dimension together with the subject feature leads, as expected, to the highest accuracy. By excluding the subject feature, SVC produces quite good results with each dimension, however they are lower than those obtained by the complete method; this shows that the three dimensional representation leads to better clusterisation.

Table 3 shows the differentiation in the results related to the Facebook datasets. In particular, the social dimension performs poorly if used alone, in fact the author of a message is known, whereas not the receiver user; also the text content dimension behaves badly if considered alone. In these datasets, the time appears to be the most important feature to discriminate the conversations, however the results achieved only with this dimension are worse than those of the SVC complete method.

From these results, achieved using each dimension separately from the others, we deduce that SVC is robust to different types of data. Moreover the use of a supervised algorithm allows both to detect the importance of the three dimensions and to achieve a method that can deal with different datasets without requiring ad-hoc tuning or interventions.

The learning algorithm we used, *Random Forest*, builds models in form of sets of decision trees, which predicts the relationship between an input features vector by walking across branches to a leaf node. An example of such a decision tree is reported in Figure 4: to predict the relation between a pair of messages, the classifier start from the root and follow branches of the tree according to the computed features. The numbers in the edges are the threshold of feature values which classifier checks in order to choose which branch of the tree should pick. We shown that the time similarity is the most significant feature. In particular, when we include the subject as a feature, we notice that the second significant features are the text cosine similarity and the Jaccard neighborhood. However, when we exclude the subject, involved users Jaccard similarity takes the place of text cosine similarity.

Table 4: Results varying the supervised model used to compute the distance between two email.

Model	Purity	Precision	Recall	F_1	$\|\Omega\|$
BC3 ($\|\mathcal{C}\| = 40$)					
LibSVM	0.980	0.962	0.984	0.973	40
Logistic	**1.000**	**1.000**	0.965	**0.982**	45
J48	0.961	0.915	**0.993**	0.952	39
RF	**1.000**	**1.000**	0.961	0.980	45
RF:100	**1.000**	**1.000**	0.952	0.975	46
Apache ($\|\mathcal{C}\| = 334$)					
LibSVM	0.785	0.584	0.583	0.584	500
Logistic	0.883	0.904	0.883	0.893	275
J48	0.851	0.865	0.931	0.897	281
RF	0.862	0.885	**0.979**	0.930	255
RF:100	**0.920**	**0.916**	0.972	**0.943**	286
Redhat ($\|\mathcal{C}\| = 802$)					
LibSVM	0.575	0.473	0.674	0.556	450
Logistic	0.709	0.619	0.697	0.656	572
J48	0.672	0.614	0.516	0.561	1330
RF	0.89	0.888	0.900	0.894	762
RF:100	**0.954**	**0.966**	**0.914**	**0.939**	818
Facebook page: **Healty Choice** $\|\mathcal{C}\| = 132$)					
LibSVM	0.766	0.657	0.694	0.675	187
Logistic	**0.788**	0.676	**0.724**	**0.699**	211
J48	0.777	0.621	0.710	0.662	219
RF	0.771	**0.682**	0.656	0.668	218
RF:100	0.787	0.670	0.712	0.690	214
Facebook page: **World Health Organization** ($n_c = 132$)					
LibSVM	0.628	0.444	**0.805**	0.573	118
Logistic	0.755	0.566	0.702	**0.627**	198
J48	**0.774**	**0.603**	0.615	0.609	260
RF	0.731	0.536	0.718	0.614	186
RF:100	0.747	0.552	0.714	0.623	222
Facebook group: **Healthcare Advice** ($\|\mathcal{C}\| = 468$)					
LibSVM	0.692	0.502	0.768	0.607	383
Logistic	0.840	0.699	0.761	0.729	548
J48	0.775	0.610	0.671	0.639	573
RF	0.766	0.596	**0.773**	0.673	467
RF:100	**0.909**	**0.809**	0.721	**0.763**	714
Facebook page: **Ireland Support Android** ($\|\mathcal{C}\| = 408$)					
LibSVM	0.655	0.460	0.744	0.568	356
Logistic	0.814	0.654	**0.723**	**0.687**	573
J48	0.758	0.590	0.636	0.612	658
RF	0.786	0.646	0.641	0.644	627
RF:100	**0.821**	**0.685**	0.655	0.670	663

4.3.1 Parameter Tuning

Parameter tuning in machine learning techniques is often a bottleneck and a crucial task in order to obtain good results. In addition, for practical applications, it is essential that methods are not overly sensitive to

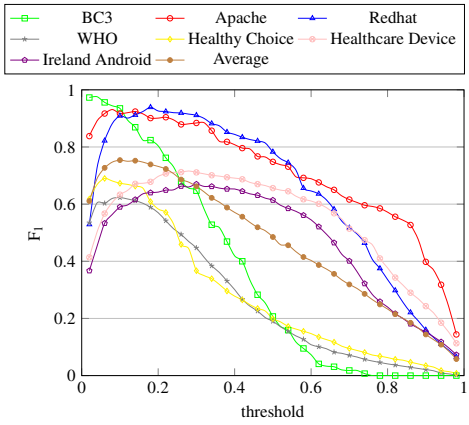

Figure 5: F_1 measure for varying number of threshold.

parameter values. Our proposed method requires the setting of few parameters. In this section, we show the effect of changing different parameter settings. A first investigation of our SVC regards the supervised algorithm used to define the similarity score between a pair of messages. We conducted a series of experiments on the benchmark datasets varying the model. Namely, we used decision trees (*Random Forest* and *J48*), SVM (*LibSVM*) and *Logistic Regression*. For all the algorithms we used the Weka implementation using the default parameter values. Considering the intrinsic lack of balance of the problem (i.e. each message has a plenty of pairs with messages that belong to different threads and just few in the same one) we also experimented with a cost-sensitive version of Random Forest, setting a ratio of 100 for instances with messages belonging to the same thread. Table 4 shows the results, it is notable that the cost sensitive Random Forest always outperforms the standard Random Forest. Logistic regression and cost sensitive Random Forest achieve better results, with a little predominance of the latter.

The main parameter of our proposed method regards the threshold value used in the clustering algorithms. We experimented with the use of a supervised model in the DBSCAN clustering algorithm, but we noticed the results were not good. This is not surprising if we consider how DBSCAN works: it groups messages in a cluster iteratively adding the neighbors of the messages belonging to the cluster itself. This leads to the erroneous merge of two different conversations, if just one pair of messages is misclassified as similar, bringing a sharp decline to the clustering precision. The previous issue, however, does not affect the agglomerative clustering, because of the use of average link of two messages inside two clusters, to decide whether to merge them or not. In this approach the choice of the threshold parameter is crucial, namely the stop merge criterion. Figure 5 shows the F_1 trend varying the agglomerative threshold, using the weighted Random Forest as the supervised model. Is notable that all the trends have only one peak that corresponds to a global maximum, thus with a simple gradient descent is possible to find the best threshold value. Furthermore, our method is generally highly effective for threshold values ranging from 0.1 to 0.3, as shown in Figure 5. This is also confirmed by the average trend, that has a peak with a threshold equal to 0.1.

4.4 Towards Subject-based Supervised Models

In this section, we discuss the possibility of creating an - incomplete - training set from which to create the supervised model of SVC, with the peculiarity that is not a labeled set known a priori. This proposed method is particularly suitable for email datasets. The main assumption is that a conversation of emails can be formed by emails with the same subject and also by emails with different ones. It is quite common, but not always true, that emails with the same subject refer to the same conversation (Wu and Oard, 2005). This intuition can provide preliminary conversation detection of a message pool in an unsupervised way. Our proposal is to create the training set using this simple clusterization as labels for distinguishing messages belonging to the same or different threads (i.e. if the two messages have same subject). We can train the classifier with this labeled set using this model inside the clustering method as described in section 3.4. Since the subject is a known feature of an email dataset, this approach guarantees the use of a supervised classifier even if no labeled dataset is given.

From the benchmark datasets we considered, only BC3 provides emails with different subject in the same threads. To test this approach, there is no need of a crossfold validation: the whole dataset is initially labeled based only on the subject of emails, a classifier is trained with this data and then used to clusterize the starting dataset. The results are extremely promising: we obtain a *purity* and a *precision* equal to 1, a *recall* equal to 0.986 and the resulting F_1 *measure* is equal to 0.993, higher than the state of art.

5 CONCLUSIONS

In this paper, we focus on the problem of detecting threads from a pool of messages that correspond to social network chats, mailing list, email boxes, chats,

forums etc. We address the problem by using a three-dimensional representation for each message, which involves textual semantic content, social interactions and creation time. Then, we propose a suite of features based on the three dimensional representation to compute the similarity measure between messages, which is used in a clustering algorithms to detect the threads. We also propose the use of a supervised model which combines these features using the probability to be in the same thread estimated by the model as a distance measure between two messages. We show that the use of a classifier leads to higher accuracy in thread detection, outperforming all earlier approaches.

For future work, an interesting variation of the problem to consider is the conversation tree reconstruction, where we have to detect the reply structure of the conversations inside a thread.

REFERENCES

Adams, P. H. and Martell, C. H. (2008). Topic detection and extraction in chat. In *ICSC 2008*, pages 581–588.

Aumayr, E., Chan, J., and Hayes, C. (2011). Reconstruction of threaded conversations in online discussion forums. In *Weblogs and Social Media*.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Bouguettaya, A., Yu, Q., Liu, X., Zhou, X., and Song, A. (2015). Efficient agglomerative hierarchical clustering. *Expert Syst. Appl.*, 42(5):2785–2797.

Coussement, K. and den Poel, D. V. (2008). Improving customer complaint management by automatic email classification using linguistic style features as predictors. *Decision Support Systems*, 44(4):870–882.

Dehghani, M., Shakery, A., Asadpour, M., and Koushkestani, A. (2013). A learning approach for email conversation thread reconstruction. *J. Information Science*, 39(6):846–863.

Domeniconi, G., Moro, G., Pasolini, R., and Sartori, C. (2016). A comparison of term weighting schemes for text classification and sentiment analysis with a supervised variant of tf.idf. In *Data Management Technologies and Applications (DATA 2015), Revised Selected Papers*, volume 553, pages 39–58. Springer.

Erera, S. and Carmel, D. (2008). Conversation detection in email systems. In *ECIR, Glasgow, UK, March 30-April 3, 2008.*, pages 498–505.

Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *(KDD-96), Portland, Oregon, USA*, pages 226–231.

F. M. Khan, T. A. Fisher, L. S. T. W. and Pottenger, W. M. (2002). Mining chatroom conversations for social and semantic interactions. In *Technical Report LU-CSE-02-011, Lehigh University*.

Glass, K. and Colbaugh, R. (2010). Toward emerging topic detection for business intelligence: Predictive analysis of meme' dynamics. *CoRR*, abs/1012.5994.

Hall, M. A., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

Hofmann, T. (1999). Probabilistic latent semantic indexing. In *ACM SIGIR*, pages 50–57. ACM.

Huang, J., Zhou, B., Wu, Q., Wang, X., and Jia, Y. (2012). Contextual correlation based thread detection in short text message streams. *J. Intell. Inf. Syst.*, 38(2):449–464.

Joshi, S., Contractor, D., Ng, K., Deshpande, P. M., and Hampp, T. (2011). Auto-grouping emails for faster e-discovery. *PVLDB*, 4(12):1284–1294.

Jurczyk, P. and Agichtein, E. (2007). Discovering authorities in question answer communities by using link analysis. In *CIKM, Lisbon, Portugal, November 6-10, 2007*, pages 919–922.

Manning, C. D., Raghavan, P., Schütze, H., et al. (2008). *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Shen, D., Yang, Q., Sun, J., and Chen, Z. (2006). Thread detection in dynamic text message streams. In *SIGIR, Washington, USA, August 6-11, 2006*, pages 35–42.

Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43.

Soboroff, I., de Vries, A. P., and Craswell, N. (2006). Overview of the TREC 2006 enterprise track. In *TREC, Gaithersburg, Maryland, USA, November 14-17, 2006*.

Ulrich, J., Murray, G., and Carenini, G. (2008). A publicly available annotated corpus for supervised email summarization. In *AAAI08 EMAIL Workshop*.

Wang, H., Wang, C., Zhai, C., and Han, J. (2011). Learning online discussion structures by conditional random fields. In *In SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 435–444.

Wu, Y. and Oard, D. W. (2005). Indexing emails and email threads for retrieval. In *SIGIR*, pages 665–666.

X. Wang, M. Xu, N. Z. and Chen, N. (2008). Email conversations reconstruction based on messages threading for multi-person. In *(ETTANDGRS '08)*, volume 1, pages 676–680.

Yeh, J. (2006). Email thread reassembly using similarity matching. In *CEAS, July 27-28, 2006, Mountain View, California, USA*.

Zhao, Q. and Mitra, P. (2007). Event detection and visualization for social text streams. In *ICWSM, Boulder, Colorado, USA, March 26-28, 2007*.

Zhao, Q., Mitra, P., and Chen, B. (2007). Temporal and information flow based event detection from social text streams. In *AAAI, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1501–1506.