Identifying Conversational Message Threads by Integrating Classification and Data Clustering

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

[1]Dept. of Computer Science and Engineering (DISI), University of Bologna at Cesena, Italy
[2]Dept. of Computer Science and Engineering, University of Ioannina, Greece
[3]IBM Research - 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

Abstract. Conversational message thread identification regards a wide spectrum of applications, ranging from social network marketing to virus propagation, digital forensics, etc. Many different approaches have been proposed in literature for the identification of conversational threads focusing on features that are strongly dependent on the dataset. In this paper, we introduce a novel method to identify threads from any type of conversational texts overcoming the limitation of previously determining specific features for each dataset. Given a pool of messages, our method extracts and maps in a three dimensional representation the semantic content, the social interactions and the timestamp; then it clusters each message into conversational threads. We extend our previous work by introducing a deep learning approach and by performing new extensive experiments and comparisons with classical learning algorithms.

Keywords: Conversational message, Thread identification, Data Clustering, Classification

1 Introduction

Nowadays, online conversations have become widespread, such as email, web chats, online conversations and social groups. Online chatting, 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 finds a wide spectrum of applications such as marketing, social network extraction, expert finding, the improvement of email management, ranking content and others [1, 2, 3, 4].

The contiguous increase of digital content leads people being overwhelmed by information. For example, imagine the case where a user has hundreds of new unread messages in a chat or a mailbox or in a situation where the same user

* This work was partially supported by the european project "TOREADOR" (grant agreement no. H2020-688797).

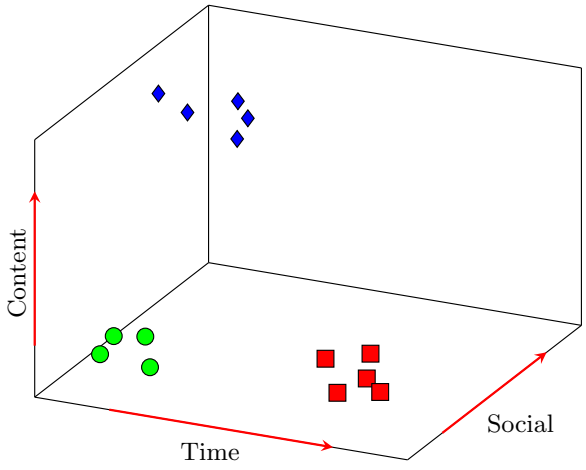

Fig. 1. Three dimensional representation of threads messages

needs to track and organise posts in forums or social groups. In order to instantly have a clear view of different discussions, avoiding expensive and tedious human efforts, we need to automatically organise this data stream into threads.

Many different approaches have been proposed in the related literature to extract topics from document sets, mainly through a variety of techniques derived from Probabilistic Latent Semantic Indexing (pLSI) [5] and Latent Dirichlet Allocation (LDA) [6]. However, the problem of identifying threads from conversational messages differs from document topic extraction for several aspects [7, 8, 4, 9, 10]: (i) conversational messages are generally much shorter than usual documents making the task of topic identification much more difficult (ii) thread identification strongly depends on social interactions between the users involved in a message exchange, (iii) as well the time of the discussion.

In our previous work [11] we addressed the problem of efficiently identifying conversational threads from pools of online messages - for example from emails, social groups, chats etc. In other words, we looked for the sets of messages that are related to each other with respect to text content, time and involved users.

We consider a three dimensional representation [12] 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, similarly to the work in [13], the generated features are embedded into a metric distance in density and hierarchical clustering algorithms [14, 15] which cluster messages in threads. In order to en-

hance our approach to efficiently identify threads in any type of dataset, we train a classification model with a set of messages previously organised in threads. The classifiers exploit the same features used in the clustering phase and they return 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 identify the threads. We observe that the classifiers' output can help the clustering process to achieve higher accuracy by identifying the threads correctly. In this paper we extend our aforementioned approach by comparing classical Machine Learning supervised algorithms with a Deep Learning Multi-Layer Perceptron. Deep learning algorithms are proven to achieve good results in several mining domains [16], expecially in big-data contexts [17]; thus considering that the identification of conversational thread in social networks could be a problem with an huge amount of data to analyze, a deep learning approach could fit well into this task.

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.

To summarize, the main contributions of this work are:

- a three dimensional message representation based on textual semantic content, social interactions and time to generate features for each message;
- clustering algorithms to identify threads, on top of the features generated from the three dimensional representation;
- combination of the generated features to build classifiers that identify 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 identify threads;
- extension of the combined classification techniques with clustering algorithms that achieves a higher accuracy than using clustering alone;
- comparison of performances obtained by several machine learning algorithms and the deep learning Multi-Layer Perceptron.

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 identification problem. In Section 4, we introduce our model and our algorithms for thread identification. Section 5 presents the experimental results on real datasets and Section 6 concludes the paper.

2 Method

In this section, we outline a generic algorithm for identifying 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 identify threads based on this similarity measure in Section 2.3.

2.1 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 [12, 18] 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 identify 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.

2.2 Messages Features

Social text messages, like emails or posts, can be summarized by three main components: text content, temporal information, and social relations [12]. 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 [19] is used to group words with a common stems. To estimate the importance to each word, there exist several different weighting schemes [20], here we make use of the commonly used *tf.idf* scheme [21].

Using *BoW* representation, the similarity between two vectors m_i, m_j can be measured by means of the commonly used *cosine similarity* [22]:

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

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 concepts extracted using AlchemyAPI [1]. 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.

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

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:

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

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.

2.3 Clustering

In this section, we present the clustering methods used to identify 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:

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

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.

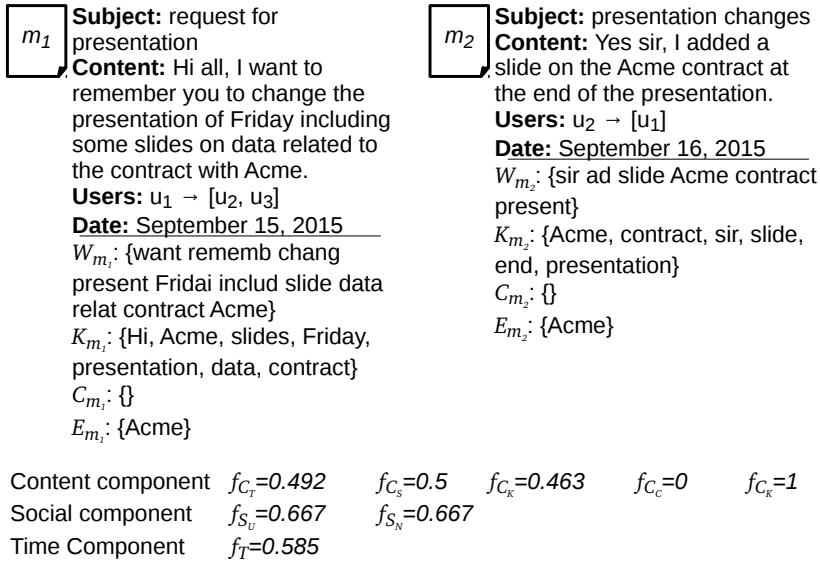

Fig. 2. Example of features calculation for a pair of messages. Message components: Subject, Content, Users (sender → recipients) and creation date. $\mathcal{W}(m_i)$ refers to the bag of words of a message obtained after the tokenization, stopwords removal and stemming. The vectors of keywords ($\mathcal{K}(m_i)$), 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.

Density-based clustering We use the DBSCAN [14] 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.

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

Hierarchical clustering This approach uses the Agglomerative hierarchical clustering method [15] 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* [23]. 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.

2.4 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 2.2 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 [24] 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 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 3.3, the Agglomerative hierarchical clustering achieves better results with respect to the DBSCAN, thus, we use this clustering algorithm in the *SVC* approach.

2.5 Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) is a deep learning algorithm [16, 25] that can be viewed as a logistic regression classifier where the input is first transformed using a learnt non-linear transformation Φ. This transformation maps the input data into a space where they expect to be better linearly separable. This intermediate layer is referred to as a hidden layer.

Formally, a one-hidden-layer MLP is a function $f : R^D \rightarrow R^L$, where D is the size of input vector x and L is the size of the output vector $f(x)$, such that, in matrix notation:

$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))),$$

with x that is the input vector, i.e. the set of feature values $\mathcal{F}$ computed for the coupled messages; $b^{(1)}$, $b^{(2)}$ are bias vectors; $W^{(1)}$, $W^{(2)}$ are weight matrices and G and s activation functions. The vector $h(x) = \Phi(x) = s(b^{(1)} + W^{(1)}x)$ constitutes the hidden layer. $W^{(1)} \in R^{D \times D_h}$ is the weight matrix connecting the input vector to the hidden layer. Each column $W^{(1)}_{\cdot i}$ represents the weights from the input units to the i-th hidden unit. After a pre-tuning phase, we chose the *hyperbolic tangent* activation funtion:

$$s = tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$$

for its faster and higher results with respects to other functions, like for instance the *sigmoid*. The output vector is then obtained as:

$$o(x) = G(b^{(2)} + W^{(2)}h(x))$$

.

By considering that we need a binary classification, namely either the coupled messages belong or not to the same thread, the output is a couple of probabilities for each class achieved by choosing G as the *softmax* function. To run our experiments, we make use of Yusuke java implementation Sugomori[2] [26].

3 Evaluation

In this section, we compare the accuracy of the clustering methods described in Section 2 in terms of identifying the actual threads.

3.1 Datasets

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

- The *BC3* dataset [27], which is a special preparation of a portion W3C corpus [28] 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[3] 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[4] and it contains the discussions that took place in the first six months of 2009.
- Two Facebook pages datasets, namely *Healthy Choice*[5] and *World Health Organizations*[6], crawled using the Facebook API[7]. 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 identify the language).
- Two Facebook public groups datasets, namely *Healthcare Advice*[8] and *Ireland Support Android*[9], 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.

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

We use the first three datasets that consist of emails in order to compare our approach with existing related work [29, 30, 31] 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.

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.

3.2 Evaluation Metrics

The *precision*, *recall* and F_1*-measure* [23] are used to evaluate the effectiveness of the conversation threads identification. Here, we explain these metrics in the context of the conversational identification 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:

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

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

$$purity(\Omega, \mathcal{C}) = \frac{1}{|\mathcal{M}|} \sum_{k} \max_{j} |\omega_k \in c_j|$$

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

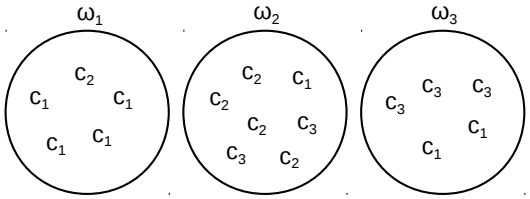

Fig. 3. Conversation extraction example. Each ω_k refers to an extracted thread and each c_j corresponds to the real conversation of the message.

To better understand the purity metric, we refer to the example of thread identification 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 $purity = (4 + 4 + 3)/17 = 0.647$.

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

3.3 Results

Table 2 and 3 report the results obtained with the seven datasets using the Weka [32] implementation of *Random Forest* algorithm. We applied a 2×2 cost matrix with a weight of 100 for the instances labelled with one and the *Sugomori* Java implementation of the *Multi-Layer Perceptron* as described above. 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 are 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 [31], Erera and Carmel [30] and the lastest one of Dehghani et al [29]. The first two approaches [31, 30] are unsupervised, as the two clustering baselines, while the approach in [29] is supervised, like our proposed SVC; both this supervised

Table 2. Conversational identification results on email datasets. [31, 30], DBSCAN and Agglom. are unsupervised methods, while [29] and SVC are supervised, both using the *Random Forest* (RF) and the *Multi-Layer Perceptron* (MLP) algorithms. 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 [31]	0.601	0.625	0.613	0.406	0.459	0.430	0.498	0.526	0.512
Erera and Carmel [30]	0.891	0.903	0.897	0.771	0.705	0.736	0.808	0.832	0.82
Dehghani et al. [29]	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_{RF} (+s)	**1.000**	**0.986**	**0.993**	0.998	**1.000**	**0.999**	0.995	**0.984**	**0.989**
SVC_{MLP} (+s)	**1.000**	0.982	0.991	0.969	**1.000**	0.984	**0.99**	0.981	0.985
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_{RF} (-s)	**1.000**	0.952	0.975	**0.916**	0.972	**0.943**	**0.966**	**0.914**	**0.939**
SVC_{RF} (-s -a)	0.967	**0.979**	0.973	0.892	**0.994**	0.940	0.815	0.699	0.753
SVC_{MLP} (-s)	**1.000**	0.973	**0.986**	0.654	0.671	0.663	0.703	0.541	0.612
SVC_{MLP} (-s -a)	**1.000**	0.972	0.986	0.633	0.616	0.624	0.469	0.579	0.518
SVC_{RF} (content)	**1.000**	**0.919**	**0.958**	**0.954**	**0.974**	**0.964**	**0.988**	**0.984**	**0.986**
SVC_{RF} (content -s)	0.964	0.902	0.932	0.604	0.706	0.651	0.899	0.872	0.885
SVC_{RF} (content -s -a)	**1.000**	0.828	0.905	0.539	0.565	0.552	0.68	0.558	0.613
SVC_{RF} (social)	0.939	0.717	0.813	0.345	0.361	0.353	0.360	0.045	0.08
SVC_{RF} (time)	0.971	0.897	0.933	0.656	0.938	0.772	0.376	0.795	0.511

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 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 2.3. 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

Table 3. Conversation identification 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, both using the *Random Forest* (RF) and the *Multi-Layer Perceptron* (MLP) algorithms; 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_{RF}	**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_{RF} (-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_{MLP}	0.657	0.722	0.688	**0.615**	0.698	**0.654**	0.665	0.762	0.71	0.68	**0.739**	0.709
SVC_{MLP} (-a)	0.65	**0.726**	0.686	0.61	0.696	0.65	0.664	**0.764**	0.71	0.68	0.737	0.708
SVC_{RF} (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_{RF} (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_{RF} (social)	0	0	0	0	0	0	0.548	0.188	0.280	0.155	0.234	0.186
SVC_{RF} (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**

lead to an improvement of results especially in *Redhat*, which is the largest and more challenging dataset.

Table 2 also compares the performances of the SVCmethod using both the Random Forest algorithm (SVC_{RF}) and the Multi-Layer Perceptron (SVC_{MLP}). Noteworthy is the drop of precision and recall of the MLP algorithm for the apache and redhat dataset when the subject are not considered, in this case the Random Forest clearly outperforms the Deep Learning approach.

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 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 identify the importance of the three dimensions and to achieve a method that can deal with different datasets without requiring ad-hoc tuning or interventions.

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 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*), SVM (*LibSVM*), *Logistic Regression* and Deep Learning *Multi-Layer Perceptron*. For all the standard Machine Learning algorithms we used the default parameter values provided by the Weka implementation. For the MLP we done a tuning - we omit for space reasons - and we defined and fixed in all the experiments the following parameters: i) *hyperbolic tangent* activation function: $s = tanh(a)$, ii) number of training iterations: $n_e = 500$, iii) number of hidden layers: $n_h = 50$, and iv) learning rate: $lr = 0.5$.

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.

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
RF	**1.000**	**1.000**	0.961	0.980	45
RF:100	**1.000**	**1.000**	0.952	0.975	46
MLP	**1.000**	**1.000**	0.973	0.986	44
Apache ($\|\mathcal{C}\| = 334$)					
LibSVM	0.785	0.584	0.583	0.584	500
Logistic	0.883	0.904	0.883	0.893	275
RF	0.862	0.885	**0.979**	0.930	255
RF:100	**0.920**	**0.916**	0.972	**0.943**	286
MLP	0.821	0.654	0.671	0.663	431
Redhat ($\|\mathcal{C}\| = 802$)					
LibSVM	0.575	0.473	0.674	0.556	450
Logistic	0.709	0.619	0.697	0.656	572
RF	0.89	0.888	0.900	0.894	762
RF:100	**0.954**	**0.966**	**0.914**	**0.939**	818
MLP	0.773	0.703	0.541	0.612	820
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
RF	0.771	**0.682**	0.656	0.668	218
RF:100	0.787	0.670	0.712	0.690	214
MLP	**0.792**	0.657	0.722	0.688	220
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
RF	0.731	0.536	0.718	0.614	186
RF:100	0.747	0.552	0.714	0.623	222
MLP	**0.784**	**0.615**	0.698	**0.654**	220
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
RF	0.766	0.596	**0.773**	0.673	467
RF:100	**0.909**	**0.809**	0.721	**0.763**	714
MLP	0.822	0.665	0.762	0.71	531
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
RF	0.786	0.646	0.641	0.644	627
RF:100	0.821	**0.685**	0.655	0.670	663
MLP	**0.837**	0.68	**0.739**	**0.709**	583

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

An interesting outcome deduced by analysing Table 4 is that, as already reported in [33], the Random Forest algorithm obtains really good results, particularly when one is able to dive it into the right space of features opportunely weighted with respect to the classes. The deep learning method should overcome this problem, in fact MLP obtains higher performance in 6 out 7 datasets with respect to the simple unweighted Random Forest, but only in 1 out of 7 datasets are instead better of the cost-sensitive RF. For this reason, the choice of one or other algorithm highly depends on the previous knowledge available about the dataset.

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.

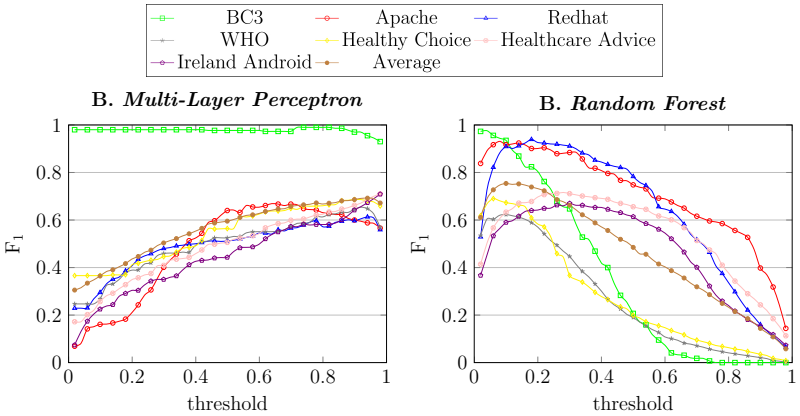

Fig. 4. F_1 measure for varying number of threshold, using (**A**) the MLP-based supervised algorithm and (**B**) the Random Forest algorithm.

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 4 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 4. This is also confirmed by the average trend, that has a peak with a threshold equal to 0.1.

4 Related Work

In last years, thread identification 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 identifying threads in different types of datasets, not only in email sets like the most of related work [31, 10, 34, 30, 35]. The authors of [31] focus on identifying conversational 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 dif-

ferent subject. Similarly, in [34] the authors identify threads in emails using the extracted header information. They first try to identify the parent/child relationships using Zawinski algorithm [10] and then they use a topic-based heuristic to merge or decompose threads to conversations. Another approach for identifying threads in emails is proposed in [30], 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 [10] 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 [35] where the authors use a segmentation and identification of duplicate emails and they group them together based on reply and forwarding relationships.

The work most closely related to ours is that of [29], that studies the conversation tree reconstruction, by first identifying the threads from a set of emails. Specifically, they map the thread identification 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 identify threads even when there are missing features. Although, it is not clear which graph clustering algorithm is used and how it identifies the clusters. We conduct an extensive comparison between our approach and the study of [29] in Section 5.

Another line of research addresses mining threads from online chats [4, 9, 8, 7]. Specifically, the study of [4] focuses on identifying threads of conversation by using pattern recognition techniques in multi-topic and multi-person chat-rooms. In [9] they focus on conversation topic thread identification 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[11] 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 [7], 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 [8] which employs the contextual correlation between short text streams. Similar to [9], they use the concept of correlative degree, which describes the probability of the contextual correlation between

[10] https://www.jwz.org/doc/threading.html
[11] http://wordnet.princeton.edu/

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 [36], a probabilistic model in conditional random fields framework is used to predict the replying structure for online forum discussions. The study in [24] 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 identification 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 identification performance.

5 Conclusions

This paper has studied the problem of identifying threads from a pool of messages that may correspond to social network chats, mailing list, email boxes, chats, forums etc. We have addressed the problem by introducing a novel method which given a pool of messages, it leverages the textual semantic content, the social interactions and the creation time in order to group the messages into threads. The work contains an analysis of features extracted from messages and it presents a similarity measure between messages, which is used in clustering algorithms that map messages to threads. Moreover the paper introduces a supervised model that combines the extracted features together with the probability of couples of messages to belong to the same thread, which is interpreted as a distance measure between two messages. Experiments show that this method leads to higher accuracy in thread identification, outperforming all earlier approaches.

Furthermore we investigated two differents main supervised approaches to create the classification model: standard machine learning algorithms, such as Random Forest, SVM and Logistic regression, and the deep learning Multi-Layer Perceptron. The results highlight that the cost-sensitive Random Forest approach achieves higher accuracy, whereas the Multi-Layer Perceptron seems a good choice with huge amount of data or when features are unknown and hidden relationships need to be found. Hence with cases with a good and thoughtful features set, a standard machine learning approach can provide better results.

There are many directions for future works. An interesting variation is the reconstruction of conversational trees, where the issue is to identify the reply structure of the conversations inside a thread. Another more general development is studying the streaming version of the problem where identifying temporal thread discussions from a stream of messages rather than from a static pool of texts.

Bibliography

[1] Pawel Jurczyk and Eugene Agichtein. Discovering authorities in question answer communities by using link analysis. In *CIKM, Lisbon, Portugal, November 6-10, 2007*, pages 919–922, 2007.

[2] Kristof Coussement and Dirk Van den Poel. Improving customer complaint management by automatic email classification using linguistic style features as predictors. *Decision Support Systems*, 44(4):870–882, 2008.

[3] Kristin Glass and Richard Colbaugh. Toward emerging topic detection for business intelligence: Predictive analysis of meme' dynamics. *CoRR*, abs/1012.5994, 2010.

[4] L. Shuler T. Wu F. M. Khan, T. A. Fisher and W. M. Pottenger. Mining chatroom conversations for social and semantic interactions. In *Technical Report LU-CSE-02-011, Lehigh University*, 2002.

[5] Thomas Hofmann. Probabilistic latent semantic indexing. In *ACM SIGIR*, pages 50–57. ACM, 1999.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[7] Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. Thread detection in dynamic text message streams. In *SIGIR, Washington, USA, August 6-11, 2006*, pages 35–42, 2006.

[8] Jiuming Huang, Bin Zhou, Quanyuan Wu, Xiaowei Wang, and Yan Jia. Contextual correlation based thread detection in short text message streams. *J. Intell. Inf. Syst.*, 38(2):449–464, 2012.

[9] Paige H. Adams and Craig H. Martell. Topic detection and extraction in chat. In *ICSC 2008*, pages 581–588, 2008.

[10] Jen-Yuan Yeh. Email thread reassembly using similarity matching. In *CEAS, July 27-28, 2006, Mountain View, California, USA*, 2006.

[11] Giacomo Domeniconi, Konstantinos Semertzidis, Vanessa Lopez, Elizabeth M. Daly, Spyros Kotoulas, and Gianluca Moro. A novel method for unsupervised and supervised conversational message thread detection. In *Proceedings of the 5th International Conference on Data Management Technologies and Applications - Volume 1: DATA,*, pages 43–54, 2016.

[12] Qiankun Zhao and Prasenjit Mitra. Event detection and visualization for social text streams. In *ICWSM, Boulder, Colorado, USA, March 26-28, 2007*, 2007.

[13] Pietro Lena, Giacomo Domeniconi, Luciano Margara, and Gianluca Moro. Gota: Go term annotation of biomedical literature. *BMC Bioinformatics*, 16(1):346, 2015.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *(KDD-96), Portland, Oregon, USA*, pages 226–231, 1996.

[15] Athman Bouguettaya, Qi Yu, Xumin Liu, Xiangmin Zhou, and Andy Song. Efficient agglomerative hierarchical clustering. *Expert Syst. Appl.*, 42(5):2785–2797, 2015.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[17] Maryam M Najafabadi, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2(1):1, 2015.

[18] Qiankun Zhao, Prasenjit Mitra, and Bi Chen. Temporal and information flow based event detection from social text streams. In *AAAI, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1501–1506, 2007.

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

[20] Giacomo Domeniconi, Gianluca Moro, Roberto Pasolini, and Claudio Sartori. 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, 2016.

[21] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

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

[23] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[24] Erik Aumayr, Jeffrey Chan, and Conor Hayes. Reconstruction of threaded conversations in online discussion forums. In *Weblogs and Social Media*, 2011.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[26] Yusuke Sugomori. *Java Deep Learning Essentials*. Packt Publishing Ltd, 2016.

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

[28] Ian Soboroff, Arjen P. de Vries, and Nick Craswell. Overview of the TREC 2006 enterprise track. In *TREC, Gaithersburg, Maryland, USA, November 14-17, 2006*, 2006.

[29] Mostafa Dehghani, Azadeh Shakery, Masoud Asadpour, and Arash Koushkestani. A learning approach for email conversation thread reconstruction. *J. Information Science*, 39(6):846–863, 2013.

[30] Shai Erera and David Carmel. Conversation detection in email systems. In *ECIR, Glasgow, UK, March 30-April 3, 2008.*, pages 498–505, 2008.

[31] Yejun Wu and Douglas W. Oard. Indexing emails and email threads for retrieval. In *SIGIR*, pages 665–666, 2005.

[32] Mark A. Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[33] Sebastian Raschka. *Python Machine Learning*. Packt Publishing, Birmingham, UK, 2015.

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

[35] Sachindra Joshi, Danish Contractor, Kenney Ng, Prasad M. Deshpande, and Thomas Hampp. Auto-grouping emails for faster e-discovery. *PVLDB*, 4(12):1284–1294, 2011.

[36] Hongning Wang, Chi Wang, ChengXiang Zhai, and Jiawei Han. Learning online discussion structures by conditional random fields. In *In SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 435–444, 2011.