J Intell Inf Syst (2008) 30:153–181
DOI 10.1007/s10844-006-0024-x

Mining categories for emails via clustering and pattern discovery

Giuseppe Manco · Elio Masciari · Andrea Tagarelli

Received: 20 February 2004 / Revised: 30 November 2005 /
Accepted: 2 June 2006 / Published online: 25 January 2007

Abstract The continuous exchange of information by means of the popular email service has raised the problem of managing the huge amounts of messages received from users in an effective and efficient way. We deal with the problem of email classification by conceiving suitable strategies for: (1) organizing messages into homogeneous groups, (2) redirecting further incoming messages according to an initial organization, and (3) building reliable descriptions of the message groups discovered. We propose a unified framework for handling and classifying email messages. In our framework, messages sharing similar features are clustered in a folder organization. Clustering and pattern discovery techniques for mining structured and unstructured information from email messages are the basis of an overall process of folder creation/maintenance and email redirection. Pattern discovery is also exploited for generating suitable cluster descriptions that play a leading role in cluster updating. Experimental evaluation performed on several personal mailboxes shows the effectiveness of our approach.

Keywords Email classification · Text mining · Clustering · Pattern discovery

G. Manco · E. Masciari
ICAR-CNR, 87036 Rende (CS), Italy

G. Manco
e-mail: manco@icar.cnr.it

E. Masciari
e-mail: masciari@icar.cnr.it

A. Tagarelli (✉)
DEIS, University of Calabria, 87036 Rende (CS), Italy
e-mail: tagarelli@si.deis.unical.it

Abbreviations

H.2.8 (Database Management)	Database Applications–*Data Mining*
I.5.3 (Pattern Recognition)	Clustering–*Algorithms, Similarity measures*
I.5.4 (Pattern Recognition)	Applications–*Text processing*
H.4.3 (Information Systems Applications)	Communications Applications–*electronic mail*

1 Introduction

The ever increasing popularity of the Internet has generated a huge traffic of different kinds of email messages. Email is a powerful means originally conceived for exchanging information across present-day networks. Also, it is rapidly evolving for handling knowledge management activities, such as personal filing, work scheduling and reminding. Huge amounts of emails received from users need to be effectively and efficiently managed to meet the users' expectations and requirements.

The problem of handling personal messages, newsletters and mailing-list messages by distinguishing important messages from unsolicited ones has become pressing. The activity of managing emails keeps users busy for a significant amount of their time, thus raising the so-called problem of *email overload* (Whittaker & Sidner, 1996).

Many tools have been designed for supporting users in the organization of their mailboxes: a typical example is represented by ad-hoc filters, i.e., rules that allow a user to manage a message under specific constraints. However, such tools are mainly *human-centered*, since users have to manually describe rules and keyword lists that could be used to recognize the relevant features of messages. This task has two main drawbacks: (1) inadequacy for a massive volume of emails, which may contain too many distinct and possibly overlapping concepts; (2) lack of adaptation to requirement changes that usually occur when users need to periodically reorganize their mailboxes and filters.

In this scenario, handling email messages and classifying them is a challenging application for Data Mining techniques. In particular, message classification can be stated as a Text Mining problem: given a set of textual documents (i.e., message contents), assign each document to a suitable cluster (i.e., mail folder) according to their contents.

However, in contrast to plain texts, email messages exhibit some features that characterize them: important structural properties (such as, sender, recipients, date/time of delivery or reception) and two unstructured elements, namely subject and content. The interaction between structured and unstructured components can play a crucial role in the message characterization. Also, email messages may contain further features (e.g., attachments) that in principle can provide significant information for classification. However, such features need to be managed with ad-hoc information retrieval techniques.

Moreover, the incoming messages from a mail server may be seen as a continuous flow of documents. Different topics in such documents may appear (or renew) over different periods of time. As a consequence, the management of incoming messages has to face the task of topic detection and tracking (Allan, Carbonell, Doddington, Yamron, & Yang, 1998a; Allan, Papka, & Lavrenko, 1998b; Lavrenko et al., 2002; Swan & Allan, 2000; Yang, Pierce, & Carbonell, 1998). In this context, the problem of automatic message categorization can be seen as a problem of mining huge, possibly infinite, streams of messages, thus triggering the need for efficient algorithms (i.e., algorithms exhibiting a complexity which is proportional to the size of the models themselves and poorly influenced by the size of the data).

Related work. Current research has focused mainly on the detection of *spam* messages. In this context, the problem of characterizing and filtering junk (unsolicited) information has been addressed by several classification techniques, which range from Bayesian techniques (Androutsopoulos, Koutsias, Chandrinos, Paliouras, & Spyropoulos, 2000; Drucker, Wu, & Vapnik, 1999) to support vector machines and rule-based classifiers (Cohen, 1996; Hidalgo, López, & Sanz, 2000; Pantel & Lin, 1998).

A different target consists in assisting users by performing the more general task of organizing her/his incoming messages into a set of predefined folders. Several approaches, mainly based on machine learning techniques, have been developed (Boone, 1998; Crawford, Kay, & McCreath, 2001; Payne & Edwards, 1997; Segal & Kephart, 1999). For example, in Segal and Kephart (1999) instance-based text classification techniques are exploited to predict the most likely destination folder for each incoming message. A similar approach is adopted in Mock (1999): starting from a set of existing categories, messages are dynamically assigned to each category on the basis of the relevance of their contents. In particular, messages are assigned to a given category according to their ranking with respect to an automatically generated query associated with that category.

The above techniques and systems perform a *supervised classification* task: message folders preexist and the goal is to detect the most suitable folder for an incoming message. To the best of our knowledge, a few efforts have been devoted to *unsupervised classification*, i.e., the automatic construction of category-based folders starting from a set of messages. Among them, we mention *Scatter/Gather* (Cutting, David, Pedersen, & Tukey, 1992) which uses a complex intermixing of hierarchical and partitional clustering. Also, the *Athena* system (Agrawal, Bayardo, & Srikant, 2000) provides a clustering algorithm based on the previously mentioned approach to produce only cluster digests for topic discovery, and performs a message partitioning on the basis of such digests using a Bayesian classifier.

A major problem in the above approaches is concerned with the maintenance of the classification schemes. Indeed, the dynamic nature of email messages eventually makes the discovered categories obsolete, thus triggering the need for constant retraining to keep the schemes up-to-date.

Contribution. Computer-mediated communication systems are composed of two main modules: *User Agents* (UA) and *Message Transfer Agents* (MTA). The main purpose of UA tools is to provide assistance to users in writing and delivering messages, while MTAs cope mainly with message transfer, according to transfer protocols. The problem of message classification can hence be formally stated as follows. Given a MTA, define a UA which is able to integrate strategies suitably conceived for handling email messages according to a number of dimensions sufficient to fulfill the expectations and requirements of a specific user. In particular, such strategies should accomplish the following functionalities:

1. Organize messages coming from MTA and directed to a given user into homogeneous groups / hierarchies;
2. Redirect further incoming messages from MTA according to such an organization;
3. Build and maintain reliable descriptions of the discovered message groups.

The above problems are strictly related and provide different perspectives in the message classification problem. More precisely, points 1 and 3 can be referred to as the problems of (1) automatically identifying folders similar by content, and (2) assigning understandable labels which capture all the content specifics of such folders. Problem 2 consists mainly in finding relevance criteria for each incoming message with respect to a preexisting collection of message folders.

In this paper we propose a message classification system capable of automatically organizing email messages stored in a mail server. We provide a unified framework for dealing with all the above described aspects, by formalizing them substantially in terms of *clustering* and *pattern discovery* techniques. A major novelty of our work is a novel algorithm for the problem of incremental cluster maintenance. The algorithm inserts a set of incoming messages into a predefined partition, and performs the required refinements to the partition so that the resulting folders reflect the most appropriate topics underlying the messages. Differently from traditional approaches to incremental clustering (e.g., Fisher, 1987; Gennari, Langley, & Fisher, 1989), the algorithm is specifically designed to exploit mainly summary statistics, without considering the messages available within clusters. In addition, the algorithm has a specific bias in trying to smooth the effects of the order-dependence of the data, via a suitable combination of multiple clustering results.

We can summarize our approach to email classification into the three following activities:

– Given an initial set $\mathcal{M} = \{m_1, \ldots, m_N\}$ of messages in a MTA, a partition $\mathcal{P} = \{C_1, \ldots, C_k\}$ of $\mathcal{M}$ is provided. The feature set to be used for representing email messages is built around both structured and unstructured fields of the messages.
– We propose a technique for discovering cluster labels which is based on the novel notion of *discriminative cluster patterns*. Discriminative cluster patterns highlight all the characteristics of a given cluster, since they are expected to lie in a specific cluster and at the same time not to lie in any other cluster.
– The initial partition is incrementally updated according to a (possibly infinite) stream $\{m_{N+1}, \ldots, m_n, \ldots\}$ of incoming messages. Viewed in this respect, each message m_i induces a new partition $\mathcal{P}_i$ that could contain a different number of categories (i.e., folders).

Plan of the paper. The paper is organized as follows. Section 2 presents the basic text preprocessing steps required to suitably model the message collection for the mining phase. Section 3 formally introduces the mail classification problem, providing a suitable definition of message similarity and illustrating a clustering methodology for addressing the problem. Section 4 describes a pattern discovery strategy conceived for extracting cluster descriptions reliably. The problem of incrementally updating a clustering scheme is addressed in Section 5. Section 6 reports on the experimental evaluation stating the effectiveness of our email classification system. Finally, Section 7 contains concluding remarks and some pointers to open problems.

2 Preliminaries

In this section, we briefly review how to extract relevant information from a collection of messages. This requires a study of the representation of both structured and unstructured features of a message.

2.1 Text preprocessing

When dealing with data containing textual information, a major issue is the selection of the set of relevant terms, or *index terms*, i.e., the terms capable of best representing the topics associated with a given textual content. In order to achieve this, some standard text processing operations are used (Baeza-Yates & Ribeiro-Neto, 1999), such as *lexical analysis*, *removal of stopwords*, *stemming*, *lemmatization*.

Terms have different discriminating power, i.e., their relevance in the context where they are used. To weight term relevance, a common approach is to assign high significance to terms occurring frequently within a document, but rarely with respect to the remaining documents of the collection. The weight of a term is hence computed as a combination of its frequency within a document (term frequency-TF) and its rarity across the whole collection (inverse document frequency-IDF). We denote by $tf(w_j, m_i)$ the number of occurrences of term w_j within message m_i, and by $df(w_j, \mathcal{M})$ the number of messages within a given message collection $\mathcal{M}$ that contain w_j. A term w_j is denoted as an *index term* for $\mathcal{M}$ if $l \leq df(w_j, \mathcal{M}) \leq u$, where l and u represent default threshold values. The ratio here is that terms appearing in a few documents, as well as terms appearing in most documents, are less significant, and hence they should be discarded.

A widely used representation model is the vector-space model (Baeza-Yates & Ribeiro-Neto, 1999). Each message m_i is represented as an m-dimensional vector $\mathbf{w}_i$, where m is the number of index terms and each component $\mathbf{w}_i[j]$ is the (normalized) *TF.IDF* weight associated with a term w_j:

$$\mathbf{w}_i[j] = \frac{tf(w_j, m_i)\log(N/df(w_j, \mathcal{M}))}{\sqrt{\sum_{p=1}^{m}\left[tf(w_p, m_i)\log(N/df(w_p, \mathcal{M}))\right]^2}}$$

2.2 Message features

For each message we extract and store information concerning *Sender*, *Recipients*, *Date/Time*, *Subject*, *Attachment filename*, and *Content* of the message. From the above fields we can derive a feature vector $\mathbf{x} = (\mathbf{y}\ \mathbf{w})$ that is composed of structured and unstructured information.

Structured information (denoted by **y**) is obtained from four fields, and comprises the following features:

- **Categorical**: sender domain (e.g., `yahoo.com`), root domain of the most frequent recipient (e.g., `gov`), weekday (binary), time period (e.g., early morning, evening), attachment file extensions;
- **Numerical**: message length, number of recipients, number of attachments, number of messages received from the same sender.

Unstructured information (denoted by **w**) is mainly obtained from the Subject and Content fields. However, such two fields contain information that, in principle, may have different importance (as discussed in Kilander, Fahraeus, & Palme, 1997, Sect. 3.4): usually, a lot of redundant information that the Subject field does not contain can be found in the Content field (e.g., it is likely that the subject contains keywords of the message). For this purpose, it is more convenient to assign a parametric higher importance to the subject of a message. The frequency of a term w_j in a message m_i is hence computed as

$$tf(w_j, m_i) = \lambda s_{ij} + c_{ij},$$

where s_{ij} and c_{ij} denote the number of occurrences of w_j within the subject and the content of m_i, respectively, and λ represents the importance factor of the subject.

The idea in the above formula is to weight a given term in the subject of the message by "virtually" increasing the number of its occurrences proportionally to a factor λ. Various definitions can be provided for λ. For instance, by setting $\lambda = \lceil \sigma n_i \rceil$, the importance factor is assumed to be proportional to the total number n_i of occurrences of the index terms appearing in the content of m_i, i.e., proportional to the content size. Alternatively, we can set $\lambda = \lceil \sigma c_{ij} \rceil + 1$, thus binding the importance factor to the occurrences of the same term within the content. In both cases $\lambda \propto \sigma$, so we subsequently refer to σ as the ratio of importance of the subject.

3 Choosing an initial cluster partition

Clustering of messages aims at identifying homogeneous groups that will represent folders in a reorganized mailbox. Formally, the problem can be stated as follows: given a set $\mathcal{M} = \{m_1, \ldots, m_N\}$ of messages coming from a MTA, find a suitable partition $\mathcal{P} = \{C_1, \ldots, C_k\}$ of $\mathcal{M}$ in k groups (where k is a parameter to be determined), such that each group refers to a homogeneous subset of messages, with an associated label. The detection of homogeneous groups of messages relies on the capability of:

- Defining matching criteria for messages based on message contents;
- Exploiting suitable clustering and categorization schemes;
- Detecting suitable descriptions for each cluster.

We now elaborate each issue in turn.

3.1 Message similarity

The notion of homogeneity can be conceived by considering the similarities between the message feature vectors defined in Section 2.1.

Given two messages m_i, m_j, and their respective feature vectors $\mathbf{x}_i, \mathbf{x}_j$, the similarity measure $s(\mathbf{x}_i, \mathbf{x}_j)$ is defined as

$$s(\mathbf{x}_i, \mathbf{x}_j) = \alpha s_1(\mathbf{y}_i^n, \mathbf{y}_j^n) + \eta s_2(\mathbf{y}_i^c, \mathbf{y}_j^c) + \gamma s_3(\mathbf{w}_i, \mathbf{w}_j).$$

In the above definition, s_1 (resp. s_2) denotes the similarity among the structured parts composed by numerical (resp. categorical) features in the messages, and s_3 takes into account the unstructured part of the message. More precisely:

– $s_1(\mathbf{y}_i^n, \mathbf{y}_j^n)$ and $s_2(\mathbf{y}_i^c, \mathbf{y}_j^c)$ are defined in terms of dissimilarity among objects. For each $\mathbf{y}$, we can separate the categorical features $\mathbf{y}^c$ from the numerical features $\mathbf{y}^n$, and define a dissimilarity measure for each of them as follows:

  - $d_1(\mathbf{y}_i^n, \mathbf{y}_j^n)$ is the standard Euclidean distance;
  - $d_2(\mathbf{y}_i^c, \mathbf{y}_j^c) = \sum_{p=1}^{q} w_p \cdot \delta(\mathbf{y}_i^c[p], \mathbf{y}_j^c[p])$.

  In the latter formula, δ denotes the Dirichlet function (Huang, 1998), q is the number of categorical features and w_p is the weight associated with the pth attribute.[1] By exploiting the above dissimilarities, similarity functions can be defined straightforwardly. For our purposes, we adopt a normalized inverse exponential function, since intuitively it suffices to find a few correspondences in the structured components to consider two messages as being quite similar.

– s_3 can be chosen among the similarity measures particularly suitable for documents (Baeza-Yates & Ribeiro-Neto, 1999; Strehl, Ghosh, & Mooney, 2000). We adopt the *cosine similarity* measure, since it captures an intuition of scale invariant similarity which guarantees the same treatment of documents with different size but similar content.

The values α, η and γ (falling within the range [0..1], and such that $\alpha + \eta + \gamma = 1$) are used to tune the influence of each part of the feature vectors to the overall similarity. The assessment of convenient values for such parameters is particularly important and is studied in more detail later in this paper.

3.2 Clustering algorithms

In our domain, a clustering approach particularly convenient is the hierarchical one (Jain & Dubes, 1988; Jain, Murthy, & Flynn, 1999), since it allows the generation of a hierarchy of topics (dendrogram), and hence of message folders. Figure 1 shows an adaptation of the *agglomerative hierarchical clustering* algorithm to our problem (AHC). Initially, each feature vector $\mathbf{x}$ is considered as a unique cluster; then, at each iteration, the algorithm selects and merges the pair of most similar clusters. Similarity among clusters is evaluated on the basis of their cluster representatives: it is worth

[1] The weight can be defined in order to guarantee that a greater similarity is obtained for objects that differ in the most frequently used feature values, whereas a smaller similarity is obtained for objects that differ in the most rarely used feature values. An example is $w_p = \frac{n_{\mathbf{y}_i^c[p]} + n_{\mathbf{y}_j^c[p]}}{n_{\mathbf{y}_i^c[p]} \cdot n_{\mathbf{y}_j^c[p]}}$, where $n_{\mathbf{y}_i^c[p]}$ is the number of messages for which the p-th attribute has a value equal to $\mathbf{y}_i^c[p]$.

noting that a cluster representative is built on three centroid vectors corresponding to textual, categorical and numerical feature vectors, respectively. The algorithm exploits a measure (Q) of the validity of a clustering scheme to yield the "best" partition corresponding to a level of the previously built dendrogram.

Formally, given a cluster partition $\mathcal{P} = \{C_1, \ldots, C_k\}$ of N messages, the quality $Q_{\mathcal{P}}$ of the partition is computed as

$$Q_{\mathcal{P}} = \mathit{IntraSim}_{\mathcal{P}} - \mathit{InterSim}_{\mathcal{P}},$$

where

- $\mathit{IntraSim}_{\mathcal{P}} = \frac{1}{k} \sum_{p=1}^{k} \frac{n_p}{N} \sum_{\mathbf{x} \in C_p} s(\mathbf{x}, \mathbf{c}_p)$ denotes the *intra-cluster similarity*,
- $\mathit{InterSim}_{\mathcal{P}} = \frac{\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} s(\mathbf{c}_i, \mathbf{c}_j)}{\frac{1}{2} k(k-1)}$ denotes the *inter-cluster similarity*,
- n_p is the number of messages assigned to cluster C_p, and $\mathbf{c}_p$ is the associated cluster centroid.

The n_p/N factor is used here to avoid the proliferation of singleton clusters. Also, the inter-cluster similarity acts as a correction function to account for the over-estimation of the intra-cluster similarity.

The algorithm of Fig. 1 has complexity $O(N^2)$, due to the cost of computing the pair-wise similarities. To avoid such a drawback, we combine the AHC algorithm with a partitional clustering technique in an integrated scheme. Essentially, the scheme exploits the agglomerative hierarchical algorithm over a small arbitrary subset $\mathcal{S}$ of messages to find initial cluster centroids. Such cluster centroids are then used to build an initial cluster partition which can be further refined using a centroid-based clustering algorithm. In particular, we can exploit the well-known *k-Means* algorithm, that has the main advantage of requiring linear time and space while guaranteeing a good quality of clusters (Steinbach, Karypis, & Kumar, 2000). An adapted version of the k-Means algorithm is shown in Fig. 2. Here, unstructured feature vectors $\mathbf{w}$ are assumed to be points inside the unit sphere. In such cases, the computation

Fig. 1 The AHC algorithm

```
Input:
    A set $\mathcal{M} = \{m_1, \ldots, m_N\}$ of messages.
Output:
    A set of cluster centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$.
Method:
    compute the feature vector $\mathbf{x}$ for each $m \in \mathcal{M}$;
    place each $\mathbf{x}_j \in \mathcal{M}$ in its own cluster, with $\mathbf{c}_j := \mathbf{x}_j$;
    $\mathcal{P}_{(0)} := \{C_1, \ldots, C_N\}$;   $t := 0$;
    repeat
      $t$++;
      compute pair-wise similarities $s(C_i, C_j) = s(\mathbf{c}_i, \mathbf{c}_j), \forall C_i, C_j \in \mathcal{P}$;
      choose $C_i$ and $C_j$ such that $s(C_i, C_j)$ is maximized;
      $C := C_i \cup C_j$;   $\mathcal{P}_{(t)} := \left(\mathcal{P}_{(t-1)} - \{C_i, C_j\}\right) \cup C$;
      recompute the cluster centroid $\mathbf{c} = (\mathbf{c}^u\ \mathbf{c}^c\ \mathbf{c}^n)$ of $C$ as follows:
        $\boldsymbol{\mu} := 1/|C| \sum_{\mathbf{x} \in C} \mathbf{w}$;   $\mathbf{c}^u := \boldsymbol{\mu}/\|\boldsymbol{\mu}\|$;
        $\mathbf{c}^c$ is the mode vector of $\{\mathbf{y}^c | \mathbf{x} \in C\}$;
        $\mathbf{c}^n := 1/|C| \sum_{\mathbf{x} \in C} \mathbf{y}^n$;
    until $|\mathcal{P}_{(t)}| = 1$;
    choose $\mathcal{P}_{(j)}$ where $j = \mathrm{argmax}_{1 \le i \le t} Q_{\mathcal{P}_{(i)}}$;
    return $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ where each $\mathbf{c}_i$ is the cluster centroid of $C_i$ in $\mathcal{P}_{(j)}$;
```

Fig. 2 The `k-Means` algorithm

```
Input:
        A set $\mathcal{M} = \{m_1, \ldots, m_N\}$ of messages;
        A set of $k$ initial points $\{\mathbf{c}_1, \ldots \mathbf{c}_k\}$.
Output:
        A partition $\mathcal{P} = \{C_1, \ldots, C_k\}$ of $\mathcal{M}$.
Method:
        compute $\mathbf{x}$ for each $m \in \mathcal{M}$;
        repeat
          for $1 \leq j \leq k$ do
            build $C_j := \{\mathbf{x} | s(\mathbf{x}, \mathbf{c}_j) > s(\mathbf{x}, \mathbf{c}_l), 1 \leq l \leq k, j \neq l\}$;
            recompute the cluster centroid $\mathbf{c}_j = (\mathbf{c}_j^u\ \mathbf{c}_j^c\ \mathbf{c}_j^n)$ as follows:
              $\mathbf{c}_j^u := \boldsymbol{\mu}_j / \|\boldsymbol{\mu}_j\|$, where $\boldsymbol{\mu}_j := 1/|C_j| \sum_{\mathbf{x} \in C_j} \mathbf{w}$;
              $\mathbf{c}_j^c$ is the mode vector of $\{\mathbf{y}^c | \mathbf{x} \in C_j\}$;
              $\mathbf{c}_j^n := 1/|C_j| \sum_{\mathbf{x} \in C_j} \mathbf{y}^n$;
          compute $cost(\mathcal{P}) = \sum_{C_j \in \mathcal{P}} \sum_{\mathbf{x} \in C_j} s(\mathbf{x}, \mathbf{c}_j)$;
        until $cost(\mathcal{P})$ is maximized;
        return $\mathcal{P}$;
```

of the cosine similarity is reduced to the computation of the inner product among vectors (Dhillon & Modha, 2001).

The integration of the above two algorithms results in a scheme we called `Hybrid` algorithm, which can be summarized in the following steps:

1. Sample a small random subset $\mathcal{S} = \{m_1, \ldots, m_h\} \subset \mathcal{M}$, and compute $\mathbf{x}$ for each $m \in \mathcal{S}$;
2. Call `AHC` on $\{\mathbf{x}_1, \ldots, \mathbf{x}_h\}$, and return $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ as output;
3. Call `k-Means` on $\mathcal{M}$ exploiting $\mathbf{c}_1, \ldots, \mathbf{c}_k$ as initial centroids.

The result $\mathcal{P} = \{C_1, \ldots, C_k\}$ of step 3 is the desired cluster partition.

The convergence of the algorithm is essentially related to the convergence of the partitional scheme. We combine the theoretical results provided in Dhillon and Modha (2001); Huang (1998); Selim and Ismail (1984) to show the convergence of the proposed scheme. The starting points are the following properties, which allow for an optimal choice of the cluster centroids in the presence of numerical and categorical features, respectively.

Lemma 1 (Selim & Ismail, 1984) Given a cluster partition $\{C_1, \ldots, C_k\}$ of objects in $\mathbb{R}^n$, the sum of the Euclidean distances of each object from the center $\mathbf{c}_i$ of its cluster C_i is minimized when $\mathbf{c}_i$ is the mean value of the objects in C_i, $1 \leq i \leq k$.

Lemma 2 (Huang, 1998) Let $\mathcal{U} = \mathcal{U}_1 \times \mathcal{U}_2 \times \ldots \times \mathcal{U}_q$ be a set of categorical domains, where $\mathcal{U}_p = \{c_{p1}, \ldots c_{pn_p}\}$ is the set of possible values for the category c_p. Given a set $\mathcal{S} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\} \subseteq \mathcal{U}$ of categorical vectors, the representative $\mathbf{r} \in \mathcal{U}$ of $\mathcal{S}$ is such that $freq(\mathbf{r}[p]|\mathcal{S}) \geq freq(c_{ph}|\mathcal{S})$ for each $1 \leq p \leq q$ and $1 \leq h \leq n_p$, where $freq(c_{ph}|\mathcal{S}) = \frac{1}{m}|\{\mathbf{y} \in \mathcal{S} \mid \mathbf{y}[p] = c_{ph}\}|$.

According to Lemma 1 and Lemma 2 we are now able to prove the soundness of the stopping condition used in the `k-Means` algorithm.

Lemma 3 The function $cost(\mathcal{P}^{(t)})$, associated with a partition obtained at a generic iteration t of the `k-Means` algorithm, is a non-decreasing function.

Proof Consider the following three functions at a generic iteration t:

$$f_u^{(t)} = \sum_{i=1}^{k} \sum_{\mathbf{x}\in C_i^{(t)}} s_3\left(\mathbf{w}, \mathbf{c}_i^{(t)}\right),$$

$$f_n^{(t)} = \sum_{i=1}^{k} \sum_{\mathbf{x}\in C_i^{(t)}} d_1\left(\mathbf{y}^n, \mathbf{c}_i^{n^{(t)}}\right),$$

$$f_c^{(t)} = \sum_{i=1}^{k} \sum_{\mathbf{x}\in C_i^{(t)}} d_2\left(\mathbf{y}^c, \mathbf{c}_i^{c^{(t)}}\right).$$

Function $f_u^{(t)}$ is a non-decreasing function, as proved in Dhillon and Modha (2001). Now, we firstly show that $f_n^{(t)}$ and $f_c^{(t)}$ are non-increasing. The claim then trivially holds by simple algebraic manipulations. Concerning $f_c^{(t)}$ we have:

$$\begin{aligned}
f_c^{(t)} &= \sum_{i=1}^{k} \sum_{\mathbf{x}\in C_i^{(t)}} d_2\left(\mathbf{y}^c, \mathbf{c}_i^{c^{(t)}}\right) \\
&= \sum_{i=1}^{k}\left(\sum_{j=1}^{k} \sum_{\mathbf{x}\in C_i^{(t)}\cap C_j^{(t+1)}} d_2\left(\mathbf{y}^c, \mathbf{c}_i^{c^{(t)}}\right)\right) \\
&\geq \sum_{i=1}^{k}\left(\sum_{j=1}^{k} \sum_{\mathbf{x}\in C_i^{(t)}\cap C_j^{(t+1)}} d_2\left(\mathbf{y}^c, \mathbf{c}_j^{c^{(t)}}\right)\right) \\
&= \sum_{j=1}^{k}\left(\sum_{i=1}^{k} \sum_{\mathbf{x}\in C_i^{(t)}\cap C_j^{(t+1)}} d_2\left(\mathbf{y}^c, \mathbf{c}_j^{c^{(t)}}\right)\right) \\
&= \sum_{j=1}^{k} \sum_{\mathbf{x}\in C_j^{(t+1)}} d_2\left(\mathbf{y}^c, \mathbf{c}_j^{c^{(t)}}\right) \\
&\geq \sum_{j=1}^{k} \sum_{\mathbf{x}\in C_j^{(t+1)}} d_2\left(\mathbf{y}^c, \mathbf{c}_j^{c^{(t+1)}}\right) = f_c^{(t+1)}
\end{aligned}$$

where the first inequality follows from the definition of cluster $C_j^{(t+1)} = \left\{\mathbf{x}_i | d_2\left(\mathbf{y}^c, \mathbf{c}_j^{c^{(t)}}\right) < d_2\left(\mathbf{y}^c, \mathbf{c}_l^{c^{(t)}}\right), 1 \leq l \leq k, l \neq j\right\}$, and the last inequality follows from the definition of $\mathbf{c}_j^{c^{(t+1)}}$ (see Lemma 2). Analogously, by considering the definitions of d_1 and $\mathbf{c}^n$ (see Lemma 1), we can show that $f_n^{(t)} \geq f_n^{(t+1)}$.

Then, according to the definition of s_1 and s_2 and the overall similarity s, it can be straightforwardly derived that

$$cost\left(\mathcal{P}^{(t)}\right) = \sum_{C_i\in\mathcal{P}^{(t)}} \sum_{\mathbf{x}\in C_i^{(t)}} s\left(\mathbf{x}, \mathbf{c}_i^{(t)}\right) \leq \sum_{C_i\in\mathcal{P}^{(t+1)}} \sum_{\mathbf{x}\in C_i^{(t+1)}} s\left(\mathbf{x}, \mathbf{c}_i^{(t+1)}\right) = cost\left(\mathcal{P}^{(t+1)}\right)$$

□

Theorem 1 (Selim & Ismail, 1984) *The* `k-Means` *algorithm terminates for any input data.*

Proof The algorithm essentially explores the set of all possible partitions. Since such a set is finite, we only need to show that, as soon as a partition is considered twice, the stopping condition is met. For this purpose, let us assume that there exist two iterations t_1 and t_2 such that $t_1 < t_2$ but $\mathcal{P}^{(t_1)} = \mathcal{P}^{(t_2)}$. According to the algorithm, the sets $\left\{\mathbf{c}_1^{(t_1)}, \ldots, \mathbf{c}_k^{(t_1)}\right\}$ and $\left\{\mathbf{c}_1^{(t_2)}, \ldots, \mathbf{c}_k^{(t_2)}\right\}$ coincide, and hence $cost(\mathcal{P}^{(t_1)}) = cost(\mathcal{P}^{(t_2)})$. From Lemma 3, for each t such that $t_1 \leq t \leq t_2$, the value $cost(\mathcal{P}^{(t)})$ is constant, thus it satisfies the stopping condition. □

4 Extracting cluster descriptions

In the email classification context an important task is the assignment of meaningful descriptions to message clusters. Intuitively, this can be accomplished by selecting the most "informative" terms which appear in the messages of a given cluster. In particular, a good cluster description should explain the content of the messages in a cluster while maintaining a strong distinction with respect to the other cluster labels.

A trivial solution consists in ranking the terms of a cluster by decreasing TF. IDF values, and then choosing the top-n terms. The main drawback of such an approach is due to the informative content of the TF. IDF weighting function. Indeed, each TF. IDF value takes into account the frequency of a term across the whole message collection but it does not deal with the frequency of the same term within different clusters.

The relationship between a cluster C and a label L can be modeled to as an implication rule $L \Rightarrow C$ having the following properties: (1) $L = \langle w_1 w_2 \ldots w_h \rangle$ is a sequence of terms which must be *discriminative* for cluster C; (2) the rule must be *strong*. In the following we discuss these properties.

A term w is a discriminative term with respect to a cluster C if w appears with high frequency in C but with low frequency in any cluster $C' \neq C$. Let $cf_C(w)$ denote the *cluster frequency*, i.e., the number of messages of C in which w appears.

Definition 1 Given a partition $\mathcal{P} = \{C_1, \ldots, C_k\}$ and a minimum-support threshold *min_sup*, term w is a *discriminative term* for cluster $C_i \in \mathcal{P}$ if the following conditions hold:

1. $cf_{C_i}(w) \geq minsup_{C_i}$, and
2. $cf_{C_j}(w) < minsup_{C_j}, \forall C_j \in \mathcal{P}, C_j \neq C_i$.

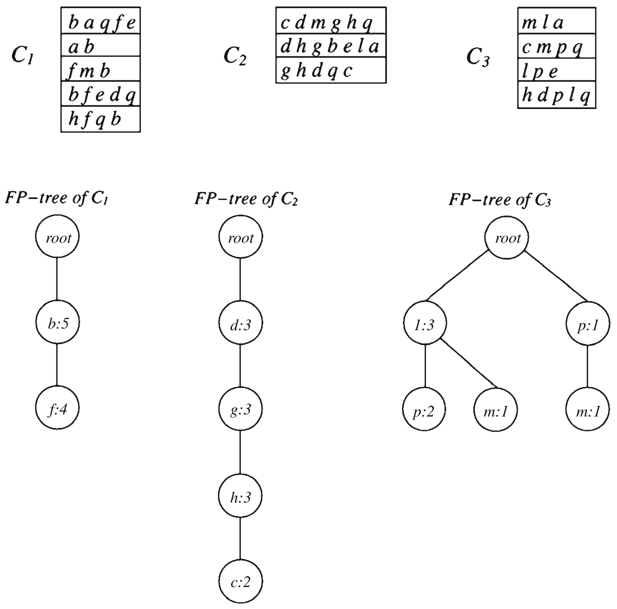
where $minsup_{C_j} = min\_sup \times |C_j|$ denotes the minimum number of messages supporting terms with respect to cluster C_j.

The above definition states a strong requirement: a term cannot be discriminative in any cluster other than the one under consideration. Intuitively, this requirement is specified in order to obtain well separated clusters. However, other definitions which relax the previous one can be provided. For instance, we could replace condition 2 with the following: $\frac{cf_{C_j}(w)}{|C_j|} < \frac{cf_{C_i}(w)}{|C_i|}, \forall C_j \in \mathcal{P}, C_j \neq C_i$ (i.e., a term cannot be more discriminative in any other cluster than the cluster C_i under consideration).

The notion of discriminative term can be extended to a sequence of terms, i.e., a *pattern* $p = \langle w_1 w_2 \ldots w_p \rangle$, by requiring that p is frequent in the cluster and that each w_i in p is discriminative. This guarantees that the anti-monotonicity property—subsets of discriminative itemsets are discriminative as well—holds, and hence allows the adoption of any standard algorithm for pattern mining. Notice that a weaker notion could be adopted, requiring that the conditions of Definition 1 hold for the frequency of p. The weakness in this definition is that the anti-monotonicity property does not hold anymore.

We compute frequent discriminative patterns by exploiting the *FP-growth* algorithm (Han, Pei, & Yin, 2000). FP-growth exploits a trie structure, called *FP-tree*, to represent all the necessary information for mining frequent itemsets in a highly condensed way. This representation has two main advantages. First, it allows to overcome the difficulty of selecting a good minimum-support threshold for the dataset under consideration; indeed, itemsets satisfying different support thresholds can be extracted from subtrees of an original tree, built over the data using a relatively low support threshold. Second, expensive candidate generation and test phases can be avoided, since all the frequent patterns can be directly extracted from the trie. Moreover, the adoption of the FP-tree is particularly suitable (e.g., compared to the prefix-tree of the Apriori algorithm) when the set of messages is dense (i.e., when messages share several features) as it likely holds with clustered data.

Example 1 As a running example, consider the following scenario in which twelve messages (represented as bags-of-words) are partitioned in three clusters. Assume that `min_sup` = 50%, thus $minsup_{C_1} = 3$, $minsup_{C_2} = 2$, $minsup_{C_3} = 2$. The algorithm for detecting discriminative terms works as follows. Initially, discriminative terms are detected: $C_1 : \{b, f\}$, $C_2 : \{d, g, h, c\}$, $C_3 : \{l, p, m\}$. Next, for each cluster, an FP-tree is built over the corresponding discriminative terms.



Then, discriminative patterns are generated for each cluster. In the following, the representation of each pattern also denotes its support inside brackets. For cluster C_1, we have:

$$f : \{f(4), bf(4)\}, \quad b : \{b(5)\}.$$

The discriminative patterns for cluster C_2 are:

$$\begin{aligned} &C : \{c(2), hc(2), gc(2), dc(2), ghc(2), dhc(2), dgc(2), dghc(2)\}, \\ &h : \{h(3), gh(3), dh(3), dgh(3)\}, \\ &g : \{g(3), dg(3)\}, \quad d : \{d(3)\}. \end{aligned}$$

Finally, for cluster C_3 we have:

$$m : \{m(2)\}, \quad p : \{p(3), lp(2)\}, \quad l : \{l(3)\}.$$

Note that patterns containing term $\{q\}$ are not generated, although they might be frequent. Indeed, term $\{q\}$ is not discriminative, since it is frequent in all the clusters under consideration.

The above technique enables the detection of a set of candidate labels, as a set of discriminative patterns that characterizes each cluster. Since we are interested in characterizing a cluster C with a rule $L \Rightarrow C$, where L represents a discriminative pattern, we need a strategy for selecting the strongest discriminative pattern among those discovered. A traditional way of measuring the strength of a rule is given by its *confidence*, which can be adapted to our context as follows.

Definition 2 Given a discriminative pattern p for a cluster $C \in \mathcal{P}$, the *strength* of the rule $p \Rightarrow C$ is defined as

$$R(p \Rightarrow C) = \frac{cf_C(p)}{df(p)},$$

where $df(p)$ counts the number of messages in partition $\mathcal{P}$ in which pattern p occurs.

The measure is high when cluster frequency is comparable to document frequency: as a consequence, labels with higher confidence are more likely to characterize the cluster.

Even if confidence is taken into account, still there are situations where multiple choices can be made, e.g., when two labels exhibit the same confidence. More specifically, given two discriminative patterns p_i, p_j computed for cluster C, the following *strength conditions* can be devised, which make p_i preferable to p_j (denoted by $p_i \prec p_j$):

1. $R(p_i \Rightarrow C) > R(p_j \Rightarrow C)$;
2. $R(p_i \Rightarrow C) = R(p_j \Rightarrow C)$, but $p_i \supset p_j$;
3. $R(p_i \Rightarrow C) = R(p_j \Rightarrow C)$ and no inclusion holds between p_i and p_j, but $cf_C(p_i) > cf_C(p_j)$;

```
Input:
    A cluster $C$ and a cluster partition $\mathcal{P}$ in which $C$ appears;
    A minimum-support threshold $min\_sup$.
Output:
    A discriminative pattern $L$ as the label of $C$.
Method:
    compute $\mathcal{F}_0 = \{w_1, \ldots, w_h\}$ := gen_discriminative_terms($C, min\_sup$);
    call FP-tree construction algorithm on $(C, \mathcal{F}_0)$ to build $FP\text{-}tree_C$;
    call FP-growth algorithm on $FP\text{-}tree_C$;
    let $\mathcal{F} = \{p_1, \ldots, p_l\}$ be the set of frequent discriminative patterns found;
    return $L$ = findStrongest($\mathcal{F}$);

Function gen_discriminative_terms($C, min\_sup$) : $\mathcal{F} = \{w_1, \ldots, w_h\}$;
    $\mathcal{F} := \emptyset$;
    for all $w \in C$ do
      if ($discriminative(w, C, min\_sup)$)
        $\mathcal{F} := \mathcal{F} \cup \{w\}$;
    return $\mathcal{F}$;

Function findStrongest($\mathcal{F}$) : $L = \{p\} \in \mathcal{F}$;
    choose pattern $p$ s.t. $\nexists p' \prec p$, $p' \in \mathcal{F}$;
    return $p$;
```

Fig. 3 The `cluster_labeler` algorithm

4. If none of the above conditions is satisfied, then the choice is made on the basis of the lexicographic order.

The scheme of our approach to cluster labeling is reported in Fig. 3.

Example 2 In our running example, we measure the strength of each discriminative pattern with respect to the corresponding cluster. For instance, in C_1 patterns $\langle f \rangle$ and $\langle b f \rangle$ exhibit the maximum strength. Now, according to condition 2, we can note that $\langle b f \rangle \prec \langle f \rangle$: thus, we set the label for C_1 as $L_{C_1} = \langle b f \rangle$. Similarly we have $L_{C_2} = \langle dghc \rangle$, whereas $L_{C_3} = \langle lp \rangle$.

5 Updating clusters

Another interesting aspect in email management is related to the periodic updating of clusters mainly due to the assignment of new incoming messages according to a current organization. Since new messages may concern topics not covered by preexisting clusters, the *cluster maintenance* problem cannot be considered as a traditional problem of supervised classification: rather, it refers to the problem of deciding when a new message exhibits substantially different features from the ones in the current clusters, and whether an overall reorganization of such clusters has to be planned.

Periodically (e.g., daily or weekly) reclustering the whole collection of messages could apparently represent a feasible solution to address the above problem. However, a number of practical considerations reveals the limitations of such a solution. Reorganizing a set of messages according to a preexisting cluster partition and a set of new incoming messages has to be faster than reclustering them from scratch. Intuitively, if a mailbox contains thousands of documents, a reorganization should benefit as much as possible from the preexisting organization. Therefore, discarding previous knowledge results in a waste of performance and higher maintenance costs. Also, obsolete messages may not be available at the time reorganization is required.

Nevertheless, an overall reclustering needs to take into account the contribution of such messages.

In this context, the problem of *incrementally updating* a clustering scheme is particularly challenging. Here it is stated as follows: given an initial partition $\mathcal{P}_0$ and a (possibly infinite) stream $\mathcal{S} = \{\mathcal{B}_1, \ldots, \mathcal{B}_h, \ldots, \mathcal{B}_k, \ldots, \mathcal{B}_i\}$ where $\mathcal{B}_j = \{m_{j_1}, \ldots, m_{j_{n_j}}\}$ is a collection (burst) of messages that are available at timestamp j, the goal is the computation of new partitions $\mathcal{P}_i$ such that

1. $\mathcal{P}_i$ is computed according to $\mathcal{P}_{i-1}$ and $\mathcal{B}_i$;
2. $\mathcal{P}_i$ does not rely on $\mathcal{B}_j$, for each $j < i$;
3. $\mathcal{P}_i$ is order-independent: that is, considering a different stream $\mathcal{S}' = \{\mathcal{B}_1, \ldots, \mathcal{B}_k, \ldots, \mathcal{B}_h, \ldots, \mathcal{B}_i\}$ in which two bursts hold in a different timestamp than in $\mathcal{S}$, then $\mathcal{P}_i$ is the same both in $\mathcal{S}$ and in $\mathcal{S}'$.

Conditions 1 and 2 state that the computational complexity of an incremental clustering algorithm should not depend on the size of the input data. On the other hand, condition 3 states that the result is deterministic, i.e., it does not depend from a particular order in the data. In practice, it is difficult that conditions 2 and 3 are both satisfied. Indeed, since messages can only be processed once, their comparison with new incoming messages is only possible by exploiting summaries and statistics. Indeed, the analysis of the main features of the preexisting organization are the basis of our approach.

Figure 4 sketches an algorithm for incrementally updating a clustering scheme according to the following strategy. For each message $m \in \mathcal{B}$, the best fitting of m in $\mathcal{P}$ is computed. More precisely, each message is processed by tentatively placing it into each of the existing folders and evaluating the quality of the resulting candidate partition; if no folder is evaluated as suitable for the given message, then a new folder containing the message is placed into the partition. Observe that the $\mathit{InterSim}_{\mathcal{P}}$ component in the quality criterion $Q_{\mathcal{P}}$ (see Section 3.2) is conceived to avoid the proliferation of new small or even singleton clusters. Notice also that $Q_{\mathcal{P}}$ can be incrementally maintained, thus making the cluster assignment procedure fast and effective for each new incoming message.

Each message m_i leads to a variant $\mathcal{P}_i^*$ of the original partition. Hence, a final candidate partition $\mathcal{P}'$ is generated by merging all the variants. In Fig. 4, $\mathcal{P}'$ is computed by means of the $\sqcup$ operator, which is defined as follows. Let $\mathcal{P}_1 = \{C_1^1, C_2^1, \ldots, C_{k_1}^1\}$ and $\mathcal{P}_2 = \{C_1^2, C_2^2, \ldots, C_{k_2}^1\}$ be two partitions computed starting from two incoming messages. By construction, either $k_1 = k$ or $k_1 = k + 1$, and the same happens for k_2. Define $C^i = C_{k+1}^i$ if $k_i = k + 1$, and $C^i = \{\}$ otherwise. Then, $\mathcal{P}_1 \sqcup \mathcal{P}_2 = \{C_1^1 \cup C_1^2, C_2^1 \cup C_2^2, \ldots, C_k^1 \cup C_k^2\} \cup C^1 \cup C^2$. In practice, we merge the changes made to the original partition. The $\sqcup$ operator is associative, as it can be easily verified.

As a final step, a refinement of the updated cluster partition is performed by detecting both clusters that can be merged and clusters that need splitting. The `refine_clusters` procedure is thus aimed at overcoming problems which are related to cluster instability. The basic idea is to exploit knowledge extracted from cluster labels in order to identify those clusters whose labels suggest splitting or merging operations. Two clusters are merged when their labels share common terms, in particular when they exhibit the maximum overlapping with respect to the average label length. On the other side, a cluster with a high

```
Input:
    A partition $\mathcal{P} = \{C_1, \ldots, C_k\}$ of $\mathcal{M}$ in $k$ clusters;
    A set $\mathcal{B} = \{m_1, \ldots, m_b\}$ of incoming messages;
    A minimum-support threshold $min\_sup$.
Output:
    A partition $\mathcal{P}' = \{C_1, \ldots, C_{k'}\}$ of $\mathcal{M} \cup \mathcal{B}$ in $k'$ clusters, where $k' \geq k$.
Method:
    for all $m_i \in \mathcal{B}$ do
      compute $\mathcal{P}_i^* = \texttt{best\_quality\_partition}_{1 \leq j \leq k+1}\{\texttt{add\_member}(\mathcal{P}, m_i, j)\}$;
    $\mathcal{P}' := \bigsqcup_{i=1}^{b} \mathcal{P}_i^*$;
    $\mathcal{P}' := \texttt{refine\_clusters}(\mathcal{P}', min\_sup)$;
    return $\mathcal{P}'$;
```

```
Function add_member($\mathcal{P} = \{C_1, \ldots, C_h\}, m, j$) : $\mathcal{P}'$;
    $\mathcal{P}' := \mathcal{P}$;
    if $j \leq h$
      $C'_j := C_j \cup \{m\}$;
      $\mathcal{P}' := \mathcal{P} - \{C_j\} \cup \{C'_j\}$;
    else
      build a new cluster $C_{h+1} := \{m\}$;
      $\mathcal{P}' := \mathcal{P} \cup \{C_{h+1}\}$;
    return $\mathcal{P}'$;
```

```
Function best_quality_partition($\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$) : $\mathcal{P}_u$;
    for all $Q_{\mathcal{P}_i}$, $1 \leq i \leq n$, do
      compute the partition quality $Q_{\mathcal{P}_i}$;
    return $\mathcal{P}_u$, where $u = \arg\max_{1 \leq i \leq n}\{Q_{\mathcal{P}_i}\}$;
```

```
Function refine_clusters($\mathcal{P}, min\_sup$) : $\mathcal{P}$;
    let $\mathcal{P}_u$ be the set of clusters involved in the last update, and
    let $\mathcal{P}_n \subseteq \mathcal{P}_u$ be the set of new (singleton) clusters;
    for all $C \in \mathcal{P}_u - \mathcal{P}_n$ do call cluster_labeler($C, \mathcal{P} - \mathcal{P}_n, min\_sup$);
    for all $C \in \mathcal{P}_n$ do call cluster_labeler($C, \mathcal{P} - \mathcal{P}_n \cup \{C\}, min\_sup$);
    compute $\delta$ (the average relative support of the label of a cluster in $\mathcal{P} - \mathcal{P}_n$);
    let $\mathcal{P}_s := \left\{C \in \mathcal{P}_u - \mathcal{P}_n \mid \frac{|cover(L_C)|}{|C|} < \delta\right\}$;
    /* splitting */
    while ($\mathcal{P}_s \neq \emptyset$) do
      $\mathcal{S} := \emptyset$;   $\mathcal{S}' := \emptyset$;
      for all $C \in \mathcal{P}_s$ do
        $C_1 := cover(L_C)$;   $C_2 := C - C_1$;
        if (cluster_labeler($C_2$) $\neq \emptyset$)
          $\mathcal{P}_u := (\mathcal{P}_u - \{C\}) \cup \{C_1, C_2\}$;
      $\mathcal{P}_s := \mathcal{P}_s - \{C\}$;
    $\mathcal{P} := \mathcal{P} \cup \mathcal{P}_u$;
    /* merging */
    compute $\varepsilon$ (the average size of the label of a cluster in $\mathcal{P}$);
    while (find_merge($\mathcal{P}, \varepsilon$) $\neq \emptyset$) do
      let $\{C_i^*, C_j^*\}$ be the clusters to be merged;
      set $L_C := L_{C_i^*} \cap L_{C_j^*}$;
      $\mathcal{P} := \mathcal{P} - \{C_i^*, C_j^*\} - \{C\}$;
    return $\mathcal{P}$;
```

```
Function find_merge($\mathcal{P}, \varepsilon$) : $\mathcal{S} \subseteq \mathcal{P}$;
    $\{C_i^*, C_j^*\} := \arg\max_{C_i, C_j \in \mathcal{P}} \left\{\frac{|L_{C_i} \cap L_{C_j}|}{|L_{C_i} \cup L_{C_j}|} \geq \varepsilon\right\}$;
    return $\mathcal{S} = \{C_i^*, C_j^*\}$;
```

Fig. 4 The incremental `update` algorithm

heterogeneity is typically unable to exhibit a label representing the contents of most of the messages. The latter feature is formalized by the concept of *anti-cover* of a cluster label.

Formally, the cover of a cluster C (denoted as $cover(L_C)$) is the set of messages supporting the cluster label. The anti-cover of a cluster then represents the portion of cluster which does not share that cluster label. In principle, the insertion of new messages can degrade the representativity of the cluster label, by increasing the size of the anti-cover. The splitting steps in the `refine_clusters` procedure progressively selects a cluster exhibiting a large degree of degradation and, if its anti-cover has a suitable label, then splits such a cluster into two sub-clusters. The first sub-cluster contains all the messages covering the cluster label, whereas the second one contains the anti-cover and is labeled by the new computed label.

We relate the degree of worsening of a cluster with the support of the cluster label. Let us denote by δ the average relative support of a cluster:

$$\delta := \frac{1}{|\mathcal{P}|} \sum_{C_i \in \mathcal{P}} \frac{|cover(L_{C_i})|}{|C|}.$$

The insertion of new messages may cause a higher variability with respect to δ in the clusters involved by the update. Thus, a suitable measure of the degradation of a cluster is the comparison of the cover of a cluster with such an average value.

The splitting phase only affects those clusters for which a suitable label can be extracted from the anti-cover. If the anti-cover is too heterogeneous, then the clustering is not split. On the other hand, the procedure for assigning the cluster membership of each newly arrived message in general prevents a high heterogeneity in the anti-cover. Indeed, if a message happens to be quite different from the other messages within the cluster, then the `add_member` procedure will more likely assign the message to a new singleton cluster.

While splitting only acts on the updated clusters, the general updating procedure may produce too many singleton clusters. In order to prevent the proliferation of several unnecessary singleton clusters, we further add a merging step, which recursively detects two clusters exhibiting a high overlap in the cluster label, and merges such clusters. The whole merging step is governed by the ε parameter, defined as

$$\varepsilon := \frac{1}{|\mathcal{P}|} \frac{\sum_{C_i \in \mathcal{P}} |L_{C_i}|}{\max_{C_i \in \mathcal{P}} |L_{C_i}|}.$$

and representing the average (normalized) size of a cluster label.

Note that in `refine_clusters` any call of the `cluster_labeler` algorithm involves only those clusters that need (re)labeling at a specific point in time. In particular, when a singleton cluster needs labeling then the remaining singleton clusters have to be ignored. This is suggested by the concrete possibility that two new singleton clusters (i.e., clusters which are created during the last message burst processing) have similar contents and potentially overlapping labels.[2]

It is worth noting that the incremental update algorithm terminates for any input burst of messages. Indeed, the proof relies on the observation that the `refine_clusters` procedure explores only a finite number of possible configurations during the splitting and merging steps.

[2]In this case, each of such clusters should be labeled at different times since, according to the definition of discriminative term, two clusters cannot be associated to the same label.

6 Experimental results

Experiments were mainly aimed at evaluating the conformance of the results of clustering to the most appropriate folder organization, and at evaluating the dependence of the proposed framework from the set of parameters which are needed to tune the system. In particular, for the latter point it is important to determine the optimal choice of the parameters which guarantee the highest accuracy in describing the correspondence between clusters and appropriate folders.

Our email classification system was tested against distinct private collections of messages for which a predefined "ideal" classification was devised. Table 1 describes the datasets we considered in the evaluation. It is worth noticing that publicly available email data is not easy to obtain as other data sources, for obvious privacy concerns. Also, traditional text datasets (e.g., the standard Reuters collection) do not properly reflect both the nature of typical user mailboxes and the modalities of managing the messages by individual users.

In the datasets under consideration, the total number of terms contained within the body of all messages was 773,862 with an average size of 488 terms per message. The shortest message was composed of 15 terms, whereas 4,757 was the count of terms in the longest message. After removal of stopwords and stemming, term selection was accomplished by trying different combinations of the preprocessing parameters. Table 2 shows how the features of the Mailbox no. 2 collection change according to such parameters. Clustering experiments (described in detail in the next subsection) revealed that optimal preprocessing is accomplished by setting $l = 0.02$ and $u \in [0.3, 0.4]$.

The experimental evaluation was structured in three parts. In the first part (Section 6.1), we evaluated the impact of the parameters to the quality of clustering. The second part (Section 6.2) was concerned with the effectiveness of cluster labeling, finally we aimed at evaluating the effectiveness of the `update` algorithm in Section 6.3.

6.1 Evaluating clustering results

The experiments described in this section were carried out on the Mailbox no. 2 collection, whose messages are related to different topics. Example topics are *Clustering Algorithms*, *Decision Trees*, *Neural Networks*, *Genetic Programming*, *Web Data Classification*, *Multi Agents Systems*, *W3C News*. From this collection, an ideal partition composed of 67 folders was inferred. We evaluated the quality of the clusterings obtained by adopting the standard *F-measure* (Baeza-Yates & Ribeiro-Neto, 1999) (*FM* in the following).

FM is defined starting from a contingency table in which each column represents a cluster C_k in a discovered partition $\mathcal{P} = \{C_1, \ldots, C_r\}$, and each row represents a

Table 1 Mail collections as test datasets

Dataset	No. of folders	No. of messages
Mailbox no. 1	34	251
Mailbox no. 2	67	1,585
Mailbox no. 3	30	1,000

Table 2 Statistics about index terms for Mailbox no. 2

index terms	$\sigma = 0$	$\sigma = 0.15$	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.4$	$\sigma = 0.5$
Total	362,258 / 285,101	586,269 / 430,816	660,295 / 478,756	807,160 / 573,707	954,658 / 669,002	1,102,602 / 764,881
(Distinct) total	2,550 / 1,472	2,558 / 1,483	2,558 / 1,483	2,558 / 1,483	2,558 / 1,483	2,558 / 1,483
Avg freq. Per message	228.55 / 179.87	369.89 / 271.81	416.59 / 302.05	509.25 / 361.96	602.31 / 422.08	695.65 / 482.57
(Distinct) avg Freq. per message	115.84 / 93.19	116.52 / 93.92	116.52 / 93.92	116.52 / 93.92	116.52 / 93.92	116.52 / 93.92
Freq. in the Shortest msg	4 / 3	6 / 4	6 / 4	8 / 4	8 / 4	10 / 4
Freq. in the Longest msg	2,302 / 1,545	4,032 / 2,712	4,607 / 3,097	5,757 / 3,884	6,907 / 4,829	8,062 / 5,774
(Distinct) freq. in The shortest msg	4 / 3	5 / 4	5 / 4	5 / 4	5 / 4	5 / 4
(Distinct) freq. in The longest msg	659 / 483	661 / 486	661 / 486	661 / 486	661 / 486	661 / 486

Each element reports two values (separated by symbol '/') referring to setting ($l = 0.01, u = 0.5$) and setting ($l = 0.02, u = 0.3$), respectively

class $\mathcal{C}_h$ in an ideal partition $\mathcal{I} = \{\mathcal{C}_1, \ldots, \mathcal{C}_i\}$. Each element (h, k) in the contingency table corresponds to the number of messages that were associated with cluster C_k and actually belong to the class $\mathcal{C}_h$. Formally, given a collection $\mathcal{M}$, *FM* is defined as:

$$FM = \sum_{h=1}^{i} \frac{|\mathcal{C}_h|}{|\mathcal{M}|} \max_{k=1\ldots r} F(h, k)$$

where

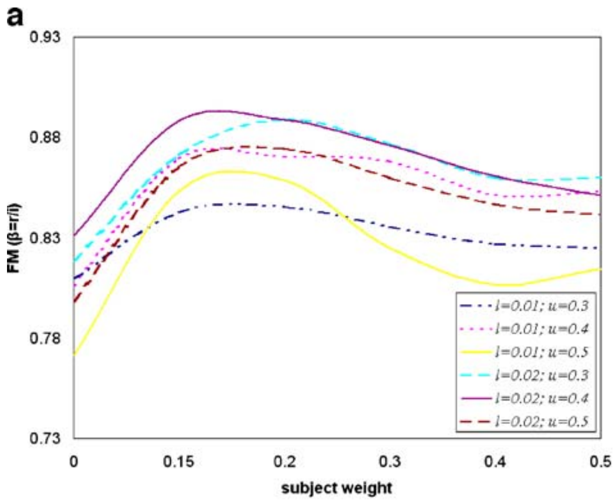
$$F(h, k) = \frac{(1 + \beta^2) P(h, k) R(h, k)}{\beta^2 P(h, k) + R(h, k)}$$

and $R(h, k)$ and $P(h, k)$ are the usual definitions of *precision* and *recall* measures relative to cluster C_k and class $\mathcal{C}_h$:

$$R(h, k) = \frac{|C_k \cap \mathcal{C}_h|}{|\mathcal{C}_h|} \qquad P(h, k) = \frac{|C_k \cap \mathcal{C}_h|}{|C_k|}$$

In the above formula, the parameter β weights the relative importance of *precision* with respect to *recall*. In our experiments, it was set to r/i. Indeed, if $r > i$, (that is, the number of clusters is higher than the number of ideal folders), precision has a bias toward high values. Hence, it is reasonable to assign it a lower weight in the formula. By contrast, if $r < i$, precision tends to be low (indeed, each cluster is likely to contain messages from more than a class), and hence the overall quality should better account for it.

To summarize, $F(h, k)$ measures the correspondence between class $\mathcal{C}_h$ and cluster C_k. The correspondence is low when $F(h, k) = 0$ and is high when $F(h, k) = 1$. The *FM* measure associates each cluster k to the class h with highest correspondence, and measures the weighted sum of such correspondences. Again, $FM \approx 0$ results in a low



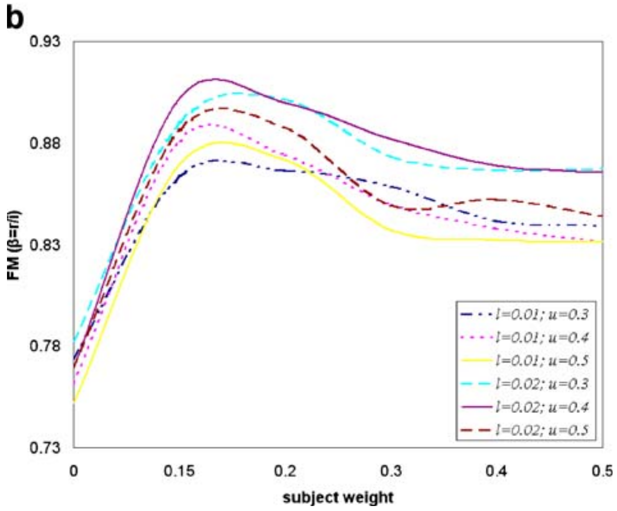


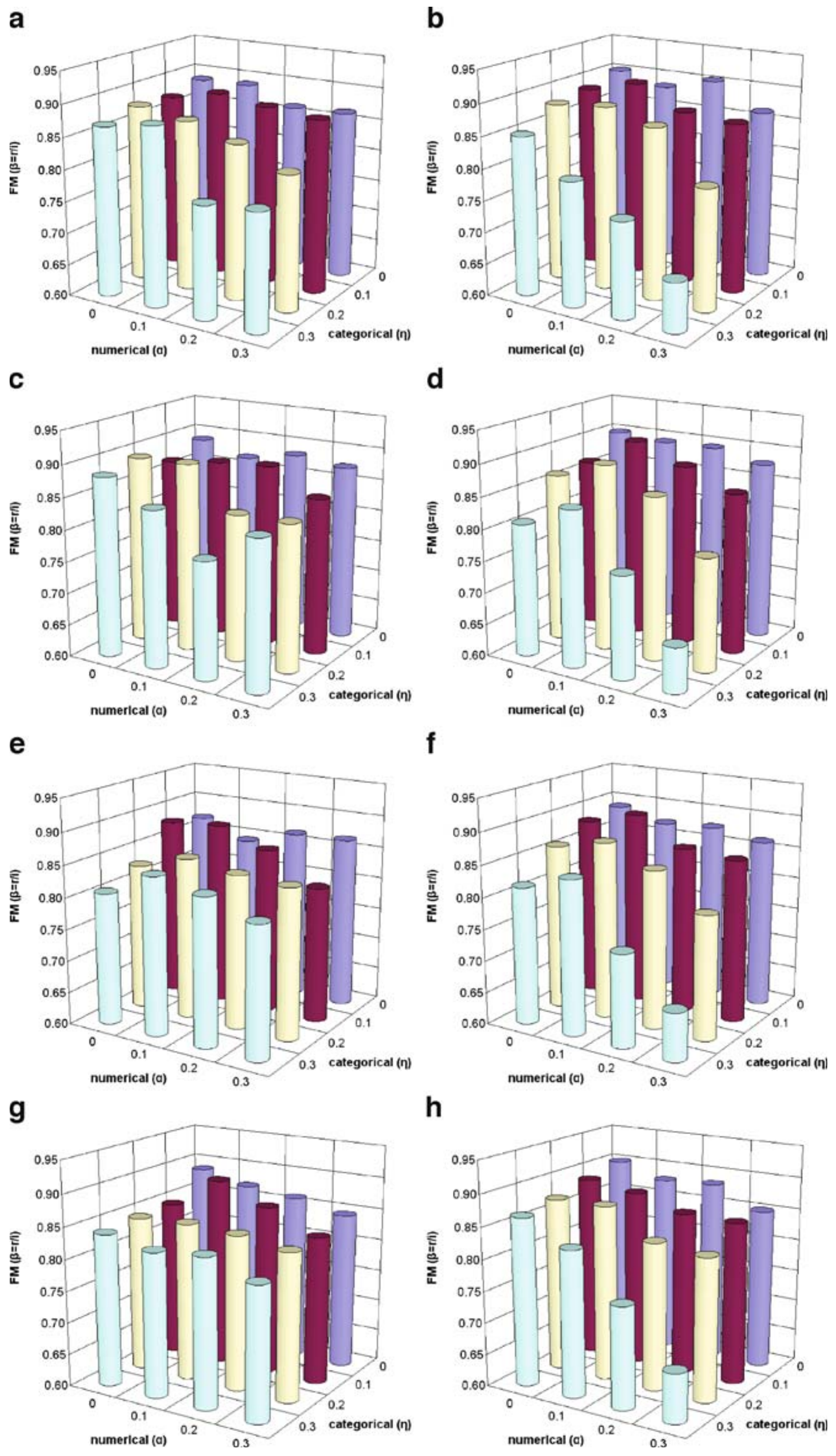
Fig. 5 Accuracy vs. subject weight: **a** `AHC`, **b** `Hybrid`

correspondence between the clustering and the partition, whereas $FM \approx 1$ results in a high correspondence.

In an initial set of experiments, we analyzed the impact of the subject weight to the clustering results. Recall that the importance factor σ denotes the weight to be assigned to the Subject field, as described in Section 3. Figure 5 shows that the quality of a clustering is affected by σ, and is maximized when $0.15 \leq \sigma \leq 0.20$. Values outside these bounds negatively affect the quality of clustering.

We also compared the performances of the `Hybrid` algorithm with the results achieved by means of a traditional hierarchical clustering. Clearly, applying `Hybrid`

Fig. 6 Clustering quality exploiting structured features. Each row compares `AHC` (*left-side*) with `Hybrid` (*right-side*), for each of the best settings of preprocessing parameters: **a**,**b** $u = 0.4$, $\sigma = 0.15$; **c**,**d** $u = 0.4$, $\sigma = 0.2$; **e**,**f** $u = 0.3$, $\sigma = 0.15$; **g**,**h** $u = 0.3$, $\sigma = 0.2$ ▶


a
b
c
d
e
f
g
h
FM (β=r/l)
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
numerical (α)
0
0.1
0.2
0.3
categorical (η)
0
0.1
0.2
0.3

leads to a significant improvement in the clustering quality, provided that similarity parameters are properly tuned. Indeed, the best results were obtained by giving higher importance to the unstructured part of the message (as somehow expected), and lower importance to the structured part of the message. Figure 6 reports clustering quality results by significantly varying the parameters controlling the structured features. We observed that the usage of numerical features provides no significant benefit. On the other hand, categorical features can positively affect the clustering quality, provided that a suitable weight ($10 \div 20\%$) is assigned to such features. In general, the advantage of applying the hierarchical algorithm over a portion of the original dataset for computing the initial centroids yields a significant improvement in the quality of results.

The proposed message clustering approach is sensitive to several parameters, so in principle it may be hard to tune it in practice. However, the experimental evaluation clearly fixes the margins for the similarity parameters. To summarize, significant values for the structured features were found in $\alpha \cong 0.1$, $0.1 \leq \eta \leq 0.2$. Unstructured features are instead governed by the bounds $l \cong 0.02$, $0.3 \leq u \leq 0.4$ and $0.15 \leq \sigma \leq 0.20$.

Obviously, there is no guarantee that such values are effective in every scenario: in particular, different users could exhibit different attitudes which are reflected by different values of the various parameters. Under this perspective, the experimentation described in this section can be conceived as a tuning methodology. Moreover, it is worth emphasizing that a clustering task is preliminarily performed to build an initial partition which is next incrementally refined as soon as new incoming messages are available. Hence, by studying the appropriateness of the initial clustering to a predefined set of folders, one can tune the various parameters for the next stages of classification.

6.2 Evaluating cluster descriptions

Orthogonally to the evaluation of clustering results discussed above, it is worth making some remarks on how `cluster_labeler` works. In a sense, the intention should be that of evaluating how our cluster labeling approach is capable of capturing the actual specifics of any cluster within a given collection. Unfortunately, a rigid evaluation of cluster labels is intrinsically difficult, as it is eventually conditioned by subjective factors when choosing proper descriptions for clusters. Therefore, we conducted an evaluation which is limited to highlighting some behavioral aspects of our cluster labeling technique.

The benefits of using `cluster_labeler` can be experienced by a user as he/she acquires specific knowledge of the topics covered within a given folder. In principle, TF. IDF ranking can help in characterizing topics. Nevertheless, according to the proposed pattern discovery technique, such topics should be better reflected by the labeling rule which is both the most discriminative and interesting for the associated cluster. Indeed, a labeling rule may involve not only terms with high TF. IDF values, but also emerging terms which are not characteristic of any cluster than the one under consideration.

As an example, Table 3 reports the descriptions associated to (some) clusters extracted from Mailbox no. 2 collection. For each cluster C, we compared the discriminative pattern L, which forms the labeling rule $L \Rightarrow C$, with a label

Table 3 An example cluster descriptions

Cluster	(Stemmed) term	TF. IDF value	Discriminative pattern
C_1	kdnugget	0.66	kdnugget
	kdd	0.16	
	tm	0.15	
	html	0.10	
	mine	0.09	
...	...	...	...
C_3	wekalist	0.55	wekalist
	list	0.28	
	weka	0.15	
	waikato	0.13	
	nz	0.13	
...	...	...	...
C_{10}	mine	0.50	mine, text, market, student
	text	0.18	
	web	0.16	
	market	0.13	
	test	0.11	
...	...	...	...
C_{17}	mine	0.43	support, decis
	ml	0.41	
	support	0.37	
	decis	0.36	
	cfp	0.07	
...	...	...	...
C_{22}	cfp	0.49	cfp
	issu	0.18	
	special	0.18	
	workshop	0.15	
	comput	0.12	
C_{23}	ml	0.50	ml, geneticalgorithm, geneticprogram
	geneticalgorithm	0.33	
	geneticprogram	0.33	
	rule	0.14	
	associ	0.12	
...	...	...	...

composed as a list of terms with the highest TF. IDF values.[3] As we can observe, the sequence of terms in a discriminative pattern is included in the related TF. IDF top list. However, labels computed by `cluster_labeler` exhibit some specifics. For example, although clusters C_{17} and C_{10} share the first term in their respective TF. IDF top list, the term "mine" appears only in the discriminative pattern of C_{10} (indeed, it has a higher frequency in C_{10} than C_{17}). A similar behavior can be observed in clusters C_{17} and C_{23} concerning the term "ml".

Also, the term "student" appears in the discriminative pattern of C_{10}, but it has not a high TF. IDF rank. This makes evident an additional advantage in using the

[3]We set the length of TF. IDF top list to 5, and the minimum-support threshold to 30%.

proposed cluster labeling procedure: the independence from the size of the labels. Indeed, the length of a label does not need to be user-specified, but is determined by the number of terms which compose the discriminative pattern provided as a result of the algorithm. This avoids any user effort to size the description of each cluster. As a consequence, cluster labeling turns out to be the outcome of an automatic task of pattern discovery rather than the effect of a subjective tagging which may either underestimate or overestimate the identification of cluster topics.

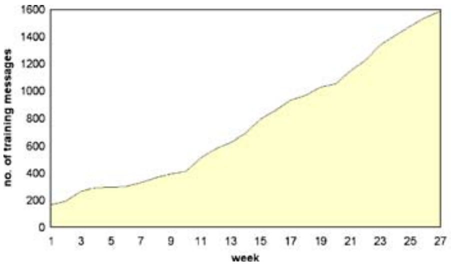
6.3 Evaluating cluster maintenance

In this section, we evaluate the effectiveness of the message classifier across a temporal window. More precisely, learning from an initial (training) message partition $\mathcal{P}$, a classifier is tested on a stream of incoming messages $\mathcal{S}$.

The effectiveness of the `update` algorithm is evaluated by properly adapting the classic notions of precision and recall. When considering continuous streams of messages, the set of categories is not fixed, since new clusters may be formed according to new messages whose topics may have not yet been covered in the current partition. Thus, the above notions need to be tuned to such a dynamic context. In particular, at a given timestamp, we consider the set $\{C_1, \ldots, C_k\}$ of preexisting categories, and an emerging category C_{new} which can be envisaged from the new burst of messages.

Given an incoming message m, we denote with $class(m)$ the assignment of m to a class in the ideal partition, and with $cluster(m)$ the assignment of m within the partition computed by `update`. The following contingencies can be computed:

- $TP_{ex,i} = \{m | class(m) = C_i, cluster(m) = C_i\}$, for each $C_i \in \mathcal{P}$, is the set of messages correctly classified into the existing folder C_i;
  $TP_{new} = \{m | class(m) = C_{new}, cluster(m) = C_{new}\}$ is the set of messages correctly assigned to a new folder.
- $FP_{ex,i} = \{m | class(m) = C_j, cluster(m) = C_i\}$, for each $C_i \in \mathcal{P}$, is the set of messages incorrectly classified into the existing folder C_i;
  $FP_{new} = \{m | class(m) = C_j, cluster(m) = C_{new}\}$ is the set of messages to be classified into existing folders but incorrectly assigned to a new folder.
- $FN_{ex,i} = \{m | class(m) = C_i, cluster(m) = C_j\}$, for each $C_i \in \mathcal{P}$, is the set of messages to be classified into C_i but incorrectly assigned to another existing folder $C_j \neq C_i$;



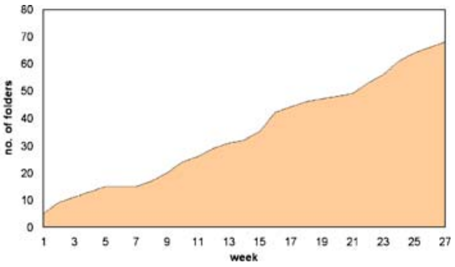


Fig. 7 Temporal distribution of message stream based on the Mailbox no. 2 collection

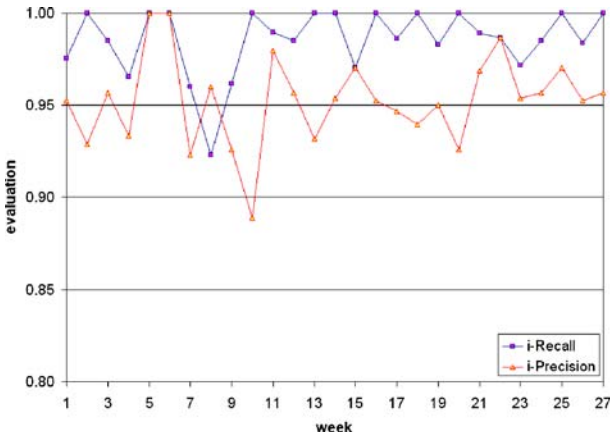


Fig. 8 Performance of `update` on the Mailbox no. 2 collection

$FN_{new} = \{m | class(m) = C_{new}, cluster(m) = C_i\}$ is the set of messages to be classified into new folders but incorrectly assigned to existing folders.

We can now define the measures which capture respectively the notions of "soundness" and "completeness" of an incremental categorizer:

$$i\text{-}Precision = \frac{|TP_{ex}| + |TP_{new}|}{|TP_{ex}| + |TP_{new}| + |FP_{ex}| + |FP_{new}|}$$

$$i\text{-}Recall = \frac{|TP_{ex}| + |TP_{new}|}{|TP_{ex}| + |TP_{new}| + |FN_{ex}| + |FN_{new}|}$$

Intuitively, *i-Precision* (resp. *i-Recall*) is the proportion of messages correctly classified with respect to the total count of assignments decided by the system (resp. expert).

In the following, we show the experiments performed on the Mailbox no. 2 collection. For this collection, the temporal window was divided in *bursts*, each of which corresponding to a week. Figure 7 shows the distribution of messages over 27

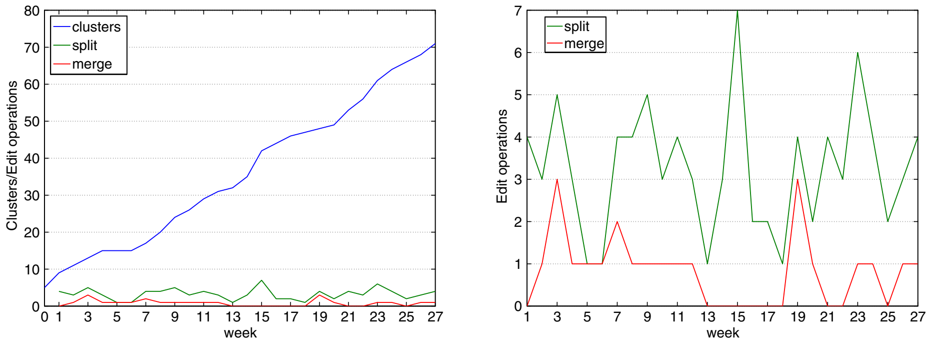


Fig. 9 Number of *split*/*merge* operations (compared to the total number of clusters) on the Mailbox no. 2 collection

weeks, with an average of 54 messages per week. The performance of the `update` algorithm, across a 27-week period, is reported in Fig. 8. As we can observe, except for the low peak corresponding to week ten, both *i-Precision* and *i-Recall* are always above 90%, with average values equal to 95.3 and 98.5%, respectively.

The efficiency of the `update` algorithm is mainly affected by the number of restructuring (split/merge) operations. As mentioned in Section 5, in principle such operations could involve the entire set of messages. However, Fig. 9 shows that the number of restructuring operations is limited in the bursts under evaluation. In particular, the left-side graph compares the number of restructuring operations with the number of clusters generated. As we can see, the latter is poorly affected by such operations. The right-side graph details the restructuring operations, and shows the influence of the merge operations is quite limited, since their number is always less than the number of split operations.

We also compared the `update` algorithm with the *naïve Bayesian* classifier, which has been widely used for text classification and has been shown to give very good performance (Domingos & Pazzani, 1997; Lewis, 1998; Lewis & Gale, 1994; Lewis & Ringuette, 1994; McCallum & Nigam, 1998; Mitchell , 1997). Such a classifier does not

Random set	*Training size*	*Test size*
1	985	15
2	922	78
3	904	96
4	877	123
5	855	145
6	812	188
7	754	246
8	698	302
9	655	345
10	568	432
11	550	450
12	520	480

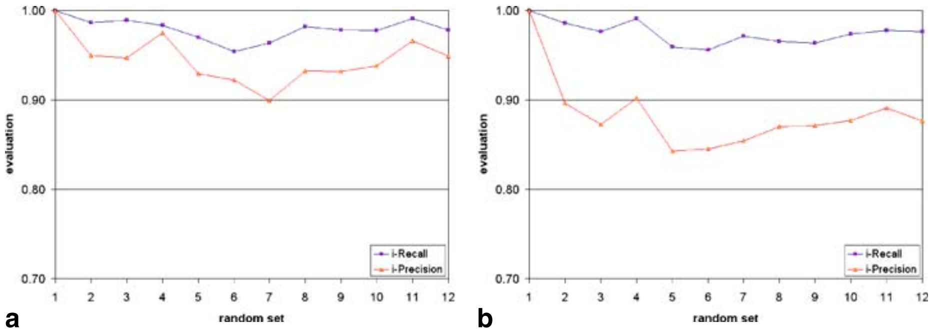


Fig. 10 Prediction performance of **a** `update` vs. **b** naïve Bayesian classifier on the Mailbox no. 3 collection

support incremental classification. Hence, a fair comparison has to be accomplished by fixing a predefined set $\mathcal{P}$ of categories (clusters), and by analyzing the behavior of such algorithms solely with respect to such a set. In practice, the assignment contingencies related to new folders (i.e., TP_{new}, FP_{new}, and FN_{new}) have to be ignored in the computation of *i-Precision* and *i-Recall* measures.

Figure 10 shows how the incremental `update` algorithm provides a substantial improvement with respect to the naïve Bayesian classifier, especially from the i-Precision point of view. In these experiments we used several message sets randomly chosen from the Mailbox no. 3 collection.

The motivation behind the performance improvement is due to a different strategy for deciding the class assignment for each new message. The naïve Bayesian classifier evaluates the conditional probability that a message belongs to different folders and assigns it to the folder with the highest conditional probability. Such a probability relies on the posterior probability of a message conditioned on a folder, which is computed by assuming the *class conditional independence*. The latter means here that the occurrence of a term, when referred locally to a given folder, is statistically unrelated to the occurrences of any other term. In practice, this assumption implies that there are no dependence relationships among the terms contained in the messages. Therefore, the naïve Bayesian classifier does not take into account the co-occurring dependence between terms. By contrast, the partition quality criterion used by the `update` algorithm allows for the computation of intra- and inter-cluster similarities which both depend on the degree of term co-occurrence among messages.

7 Conclusions

We presented a unified framework for addressing the problem of email categorization. The contribution was threefold and covered the following issues: classifying messages similar from a content point of view, updating a current message organization according to new incoming messages, and extracting significant descriptions for the message folders.

We studied how to suitably represent email messages by means of unstructured and structured features, and how to exploit them in a mixed hierarchical–partitional clustering scheme for building a mailbox partitioning.

To maintain a preexisting message organization, we conceived an incremental update strategy which benefits from the exploitation of both internal partition–quality criteria and reliable syntheses of the message folders. Such synthesized information is based on the extraction of discriminative labels obtained by means of pattern discovery techniques. The novel concept of discriminative term allows the identification of those terms which reflect the content of messages within a given cluster, and are at the same time a suitable characterization of the cluster.

We showed how the proposed data mining techniques were effective in accomplishing the overall process of folder creation/description/maintenance. Several experiments on different private mail collections were performed to assess the quality of the results provided by our techniques.

Nevertheless, some issues have still not been addressed. For instance, an interesting open problem is concerned with investigating the possibility of managing the organization of messages according to both their contents and the preferences of the corresponding recipient. Actually, when users interact with a mail service, they provide a substantial amount of information about their preferences and requirements: how they react when receiving new messages, how they would group messages belonging to the same conversational thread, which messages they consider as priority based on their current workload, and any advantage they experience in using the service. Therefore, the organization of a collection of email messages could be improved by tracking and analyzing user behavior.

References

Agrawal, R., Bayardo, R., & Srikant, R. (2000). ATHENA: Mining-based interactive management of text databases. In *Proceedings of the International Conference on Extending Database Technology (EDBT)* (pp. 365–379). Konstanz, Germany.

Allan, J., Carbonell, J., Doddington, G., Yamron, J., & Yang, Y. (1998a). Topic detection and tracking pilot study: Final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop* (pp. 194–218).

Allan, J., Papka, R., & Lavrenko, V. (1998b). On-line new event detection and tracking. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)*(pp. 37–45). Melbourne, Australia.

Androutsopoulos, I., Koutsias, J., Chandrinos, K., Paliouras, G., & Spyropoulos, C. (2000). An Evaluation of naive Bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age* (pp. 9–17). Barcelona, Spain.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*, ISBN-0-201-39829-X. New York: ACM.

Boone, G. (1998). Concept features in re: Agent, an intelligent e-mail agent. In *Proceedings of the International Conference on Autonomous Agents* (pp. 141–148). Minneapolis: ACM.

Cohen, W. (1996). Learning rules that classify e-mail. In *Proceedings of the AAAI Spring Symposium in Information Access*. Stanford, California.

Crawford, E., Kay, J., & McCreath, E. (2001). Automatic induction of rules for e-mail classification. In *Proceedings of the Australasian Document Computing Symposium* (pp. 13–20). Coffs Harbour, NSW Australia.

Cutting, D., David, K., Pedersen, J., & Tukey, J. (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)* (pp. 318–329). Copenhagen, Denmark.

Dhillon, I., & Modha, D. (2001). Concept decompositions for large sparse data using clustering. *Machine Learning, 42*, 143–175.

Domingos, P., & Pazzani, M. J. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning, 29*(2/3), 103–130.

Drucker, H., Wu, D., & Vapnik, V. (1999). Support vector machines for spam categorization. *IEEE Transactions on Neural Networks, 10*(5), 1048–1054.

Fisher, D. (1987). Concept acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

Gennari, J., Langley, P., & Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence, 40*, 11–61.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*. Dallas, Texas (pp. 1–12). New York: ACM.

Hidalgo, J., López, M., & Sanz, E. (2000). Combining text and heuristics for cost-sensitive spam filtering. In *Proceedings of the Computational Natural Language Learning Workshop (CoNLL)* (pp. 99–102). Lisbon, Portugal.

Huang, Z. (1998). Extensions to the *k*-Means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery, 2*(3), 283–304.

Jain, A., & Dubes, R. (1988). *Algorithms for clustering data*, Prentice-Hall advanced reference series. Englewood Cliffs, New Jersey: Prentice-Hall.

Jain, A., Murthy, M., & Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys, 31*(3), 264–323.

Kilander, F., Fahraeus, E., & Palme, J. (1997). Intelligent information filtering. Technical report, Department of Computer and Systems Sciences, Stockholm University. Available at http://www.dsv.su.se/~fk/if_Doc/IntFilter.html.

Lavrenko, V., Allan, J., DeGuzman, E., LaFlamme, D., Pollard, V., & Thomas, S. (2002). Relevance models for topic detection and tracking. In *Proceedings of the Conference on Human Language Technology*. San Diego, California.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the European Conf. on Machine Learning (ECML)* (pp. 4–15). Berlin Heidelberg New York: Springer.

Lewis, D. D., & Gale, W. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)* (pp. 3–12). Berlin Heidelberg New York: Springer.

Lewis, D. D., & Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. In *Proceedings of the Symposium on Document Analysis and Information Retrieval (SDAIR)* (pp. 81–93).

McCallum, A., & Nigam, K. (1998). A Comparison of event models for naive Bayes text classification. In *Proceedings of the AAAI Workshop on Learning for Text Categorization* (pp. 41–48). Madison, Wisconsin.

Mitchell, T. (1997). *Machine Learning*, Computer Sciences Series. New York: McGraw-Hill.

Mock, K. (1999). Dynamic email organization via relevance categories. In *Proceedings of the IEEE International Conference on Tools With Artificial Intelligence (ICTAI)* (pp. 399–405). Chicago, Illinois.

Pantel, P., & Lin, D. (1998). SpamCop: A spam classification and organization program. In *Proceedings of the AAAI Workshop on Learning For Text Categorization* (pp. 95–98). Madison, Wisconsin.

Payne, T. R., & Edwards, P. (1997). Interface agents that learn: An investigation of learning issues in a mail agent interface. *Applied Artificial Intelligence, 11*(1), 1–32.

Segal, R., & Kephart, J. (1999). MailCat: An intelligent assistant for organizing e-mail. In *Proceedings of the International Conference on Autonomous Agents*. Seattle, Washington (pp. 276–282). New York: ACM.

Selim, S. Z., & Ismail, M. A. (1984). K-Means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 6*, 81–87.

Steinbach, M., Karypis, G., & Kumar, V. (2000). A comparison of document clustering techniques. *In Proceedings of the ACM SIGKDD International Workshop on Text Mining*. Boston, Massachusetts.

Strehl, A., Ghosh, J., & Mooney, R. (2000). Impact of similarity measures on web-page clustering. In *Proceedings of the AAAI workshop on artificial intelligence for web search*, Austin, Texas (pp. 58–64). California: AAAI.

Swan, R., & Allan, J. (2000). Automatic generation of overview timelines. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)*. Athens, Greece (pp. 49–56). New York: ACM.

Whittaker, S., & Sidner, C. (1996). Email overload: exploring personal information management of email. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)* (pp. 276–283). New York: ACM.

Yang, Y., Pierce, T., & Carbonell, J. (1998). A study on retrospective and on-line event detection. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR)* (pp. 28–36). Melbourne, Australia.