Email Mining: Emerging Techniques for Email Management

Ioannis Katakis, Grigorios Tsoumakas, Ioannis Vlahavas

Aristotle University of Thessaloniki, Department of Informatics, Greece

ABSTRACT

Email has met tremendous popularity over the past few years. People are sending and receiving many messages per day, communicating with partners and friends, or exchanging files and information. Unfortunately, the phenomenon of email overload has grown over the past years becoming a personal headache for users and a financial issue for companies. In this chapter, we will discuss how disciplines like Machine Learning and Data Mining can contribute to the solution of the problem by constructing intelligent techniques which automate email managing tasks and what advantages they hold over other conventional solutions. We will also discuss the particularity of email data and what special treatment it requires. Some interesting email mining applications like mail categorization, summarization, automatic answering and spam filtering will be also presented.

Keywords: data mining, machine learning, email, text mining, feature selection, clustering, classification, summarization, automatic answering, spam filtering, intelligent systems

EMAIL MINING: EMERGING TECHNIQUES FOR EMAIL MANAGEMENT

The impact of electronic mail in our daily life is now more obvious than ever. Each minute, millions of plain text or enriched messages are being sent and received around the globe. Some of them are read with extra care and, at the same time, many of them are deleted with obvious disinterest. As the internet grows, electronic mail has not only turned into a vital tool for our work but also into an important means of interpersonal communication.

In professional life, email has invaded everywhere. Team organization, project management, information exchange (Ducheneaut & Belloti, 2001), decision making, client support are only a few of a company's daily processes where email has been vital. Email also made personal communication significantly easier as it offered instant messaging with minimum cost. People from all over the world can now exchange opinions and information with such ease that made email the second most popular channel of communications after voice (Miller, 2003).

Features that made email so popular are the rapidity of communication, the minimum cost and the fact that it is remarkably easy to use. An advantage over voice communication (e.g. phone) is that it is asynchronous, meaning that there is no need for both sides of communication to be on-line or in front of a computer at the same time.

Unfortunately, email could not escape the curse of Information Overload. Loads and loads of incoming messages (some extremely important, other simply junk) have turned handling of electronic mail into a tedious task. Today, an average email user may receive at about 100 or 200 messages per day and, in a recent research, IDC[1] predicts that by the year 2006 email traffic will be about 60 billion messages per day worldwide (Johnston, 2002). Nowadays, people struggle to separate important messages that demand immediate attention from the mound and large companies are investing money in order to maintain email centers with personnel dedicated to answer client requests and queries sent by emails. Additionally, the problem of spam messaging has grown at a level that it is now considered an industry problem. It costs billions of dollars (Rockbridge Associates 2004) as it takes up bandwidth, clutters inboxes and occupies employees who are receiving them. Moreover, the content of many spam messages is unsuitable for children (e.g. pornographic).

In this chapter, we will discuss what disciplines like Machine Learning and Data Mining have to offer to the solution of this email overload problem. How intelligent techniques, already used before for other text applications can apply to email data, what difficulties and obstacles have risen and what were the solutions proposed. Some interesting and novel applications like email answering, classification and summarization will be also presented.

HOW EMAIL WORKS (TERMINOLOGY)

Email does not work so differently than it used to when it first appeared. It relies on two basic communications protocols: SMTP (Simple Mail Transfer Protocol), which is used to send messages and POP3 (Post Office Protocol), which is used to receive messages. A simplified version of the email life cycle can be seen in Figure 1.

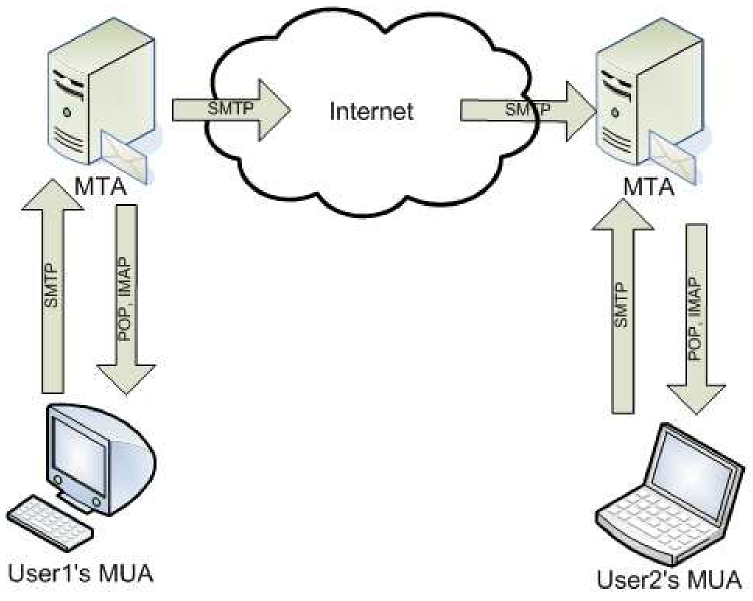


Figure 1. Life Cycle of an email

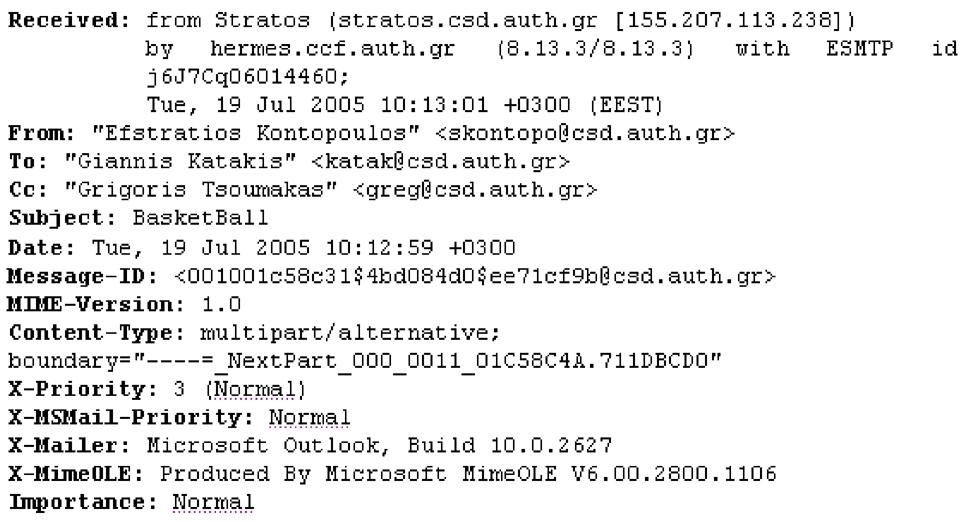
The most important *logical* elements of the Internet Mail System are:

1. Mail User Agent (MUA) – It is responsible for helping the user to read and write email messages. The MUA is usually implemented in software usually referred to as "email client". Two popular email clients are Microsoft Outlook[2] and Mozilla Thunderbird[3]. These programs transform a text message into the appropriate internet format in order for the message to reach its destination.
2. Mail Transfer Agent (MTA) –It accepts a message passed to it by either an MUA or another MTA and then decides for the appropriate delivery method and the route that the

mail should follow. It uses the SMTP protocol to send the message to another MTA or an MDA.

3. Mail Delivery Agent (MDA): It receives messages from MTAs and delivers them to the user’s mailbox in the user’s mail server
4. Mail Retrieval Agents (MRA): It fetches mail messages from the user’s mail server to the user’s local inbox. MRAs are often embedded in email clients.

An Email consists of two parts: Headers and Body. The Headers form a group of necessary information in order for the mail to reach its destination and be read properly by the recipient. Typical header fields are for example the “From:”, “To:”, or “Subject:” field. Full message headers of an email message can be seen in Figure 2.



```
Received: from Stratos (stratos.csd.auth.gr [155.207.113.238])
          by   hermes.ccf.auth.gr   (8.13.3/8.13.3)   with   ESMTP   id
          j6J7Cq06014460;
          Tue, 19 Jul 2005 10:13:01 +0300 (EEST)
From: "Efstratios Kontopoulos" <skontopo@csd.auth.gr>
To: "Giannis Katakis" <katak@csd.auth.gr>
Cc: "Grigoris Tsoumakas" <greg@csd.auth.gr>
Subject: BasketBall
Date: Tue, 19 Jul 2005 10:12:59 +0300
Message-ID: <001001c58c31$4bd084d0$ee71cf9b@csd.auth.gr>
MIME-Version: 1.0
Content-Type: multipart/alternative;
boundary="----=_NextPart_000_0011_01C58C4A.711DBCD0"
X-Priority: 3 (Normal)
X-MSMail-Priority: Normal
X-Mailer: Microsoft Outlook, Build 10.0.2627
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1106
Importance: Normal
```

Figure 2: Headers of a typical Email

EMAIL MINING

Email Mining can be considered as an application of the upcoming research area of Text Mining (TM or also known as Knowledge Discovery from Textual Data) on email data. Text Mining is an emerging field that has attracted the interest of researchers from areas like Machine Learning, Data Mining, Natural Language Processing and Computational Linguistics.

However, there are some specific characteristics of email data that set a distinctive separating line between Email and Text Mining:

1. Email includes additional information in the headers of email that can be exploited for various email mining tasks.
2. Text in email is significantly shorter and, therefore, some Text Mining techniques might be inefficient in email data.
3. Email is often cursorily written and, thus, linguistic well-formedness is not guarantied. Spelling and grammar mistakes also appear frequently.
4. In an email message, different topics may be discussed; a fact that makes e.g. mail classification more difficult.
5. Email is personal and therefore generic techniques are difficult to be effective to individuals.
6. Email is a data stream and concepts or distributions of target classes may change over time. Algorithms should be incremental in both ways: instance-wise and feature-wise, as new features (e.g. words) may appear.
7. Email will probably have noise. HTML tags and attachments must be removed in order to apply a text mining technique. In some other cases, noise is intensively inserted. In spam filtering for example, noisy words and phrases are inserted, in order to mislead machine learning algorithms.
8. It is rather difficult to have public email data for experiments, due to privacy issues. This is a drawback especially for research since comparative studies can not be conducted without public available datasets. An exception to the above statement is the Enron Corpus (Klimt & Yang, 2004), which was made public after a legal investigation concerning the Enron Corporation.

Email Pre-processing and Representation

The first step of almost every Knowledge Discovery task is the pre-processing step of the data which in the beginning is usually available in what is called "raw format". In order to mine into "raw" data and extract knowledge from it, it is necessary to transform it into a format that is more comprehensible to the machine learning algorithms.

In the last decade, people with the help of new and advanced email clients, have started to enter HTML code in order to enrich their plain text messages with different styles of text, different fonts, images, links etc. In fact, this is achieved by sending an HTML page as an attachment which every contemporary email client with a build-in web browser is able to present. In an email analysis procedure, HTML is most of the times not exploited to obtain knowledge and is removed using HTML parsers in order to keep the text contained in the HTML document. Some times though, HTML tags are used as characteristics (attributes) of the email (see section Automatic Mail Organization). For example in (Corney, Vel, Anderson, & Mohay, 2002) the HTML tag frequency distribution is defined as one of the attributes that describe the email.

In text and email mining the most prevalent model for representation is the vector space model (Salton, Wong, & Yang, 1975). In this approach, every message is represented by a single vector. Each element is associated to a token (or else "a feature" in Machine Learning terms). In textual data, tokens are usually words or phrases. The elements of the vectors have usually Boolean values (0 or 1) in order to denote presence or absence of the particular token in the document, or weights (usually numerical values between 0 and 1) to denote the importance of the token for the document (e.g. term frequency). In text classification the use of single words as tokens is much more common and it is usually referred to as "bag-of-words" representation. An alternative would be to pick phrases as tokens, but, maybe contrary to what someone would expect, using phrases instead of single words did not raise the effectiveness of the algorithms (Lewis, 1992). An interesting representation is discussed in (Boone, 1998), where additional so-called "concept" features are used. Those features are denoting the distance between the document and a "concept vector". These vectors are constructed after clustering all documents and finding a representative one for each cluster-concept.

So, more formally, we represent an email e_j as a vector $e_j = [w_{1,j},...,w_{n,j}]$, where weights $w_{1,j}$ to $w_{n,j}$ are the weights of tokens for the particular document j. In $w_{i,j}$, index i refers to token t_i in our collection of tokens. In case we use words as tokens, then t_i refers to the $i-th$ word of our vocabulary, which is basically a set of distinct words.

The vocabulary of each problem is built by analyzing some training documents if available and collecting distinct words, or incrementally constructed with every new message arrival, with the latter being the most suitable approach taking under consideration that email is a data stream. Of course, either a predefined application-specific or a generic dictionary can be used (local and global lexicon respectively).

Another decision that has to be made is whether to treat words with common stem as one. In that case a vocabulary of stems is built and words like “program”, “programming”, “programmer” are treated and counted as one. In bibliography there already exist stemming algorithms like the Porter Stemming Algorithm (Porter, 1997). Another common pre-processing step is the removal of commonly used words (like articles, prepositions etc) which we call “stop-words”. That could be effective in most applications because these are words that appear in natural language independently of topic. Hence, they lack discriminative power. On the other hand, in applications like email author identification, use of stop-words might be determinant since the frequency of those terms might reveal the identity of the author (Vel, Anderson, Corney, & Mohay, 2001).

Especially for the email domain, features – words from the email headers can be generated. A word appearing in the subject of the email can be more important than the same word appearing in the body. There is actually related work in the email mining literature that exploits only header information with respectable results (Brutlag & Meek, 2000; Zhang, Zhu, & Yao, 2004) .

Thus, in the case when mail headers are also exploited, w_i represents a feature if it appears in the subject and a different feature if it appears in the body. Therefore, the size of the vector is actually doubled, as it should be in the form shown below (Zhang et al., 2004):

$$E_j = [subject : w_1, body \ : w_1, \ ... \ , subject : w_n, body : w_n].$$

Different ways to define the weight of a word in a document have been proposed. Apart from the already mentioned Boolean weights (Androutsopoulos, Koutsias, & Chandrinos, 2000; Sahami, Dumais, Heckerman, & Horvitz, 1998), an alternative approach is to use TF-IDF

(Term Frequency – Invert Document Frequency) (Salton & Buckley, 1988) function for each word to calculate the weights. The TF-IDF function is defined as follows:

$$w_{ij} = TFIDF\,(t_i, e_j) = TF_{i,j} \log \frac{N}{DF_i}$$

where, $TF_{i,j}$ is the number of times token t_i occurs in document (email) e_j (Term Frequency), N is the total number of emails and DF_i is the number of emails t_i occurs (Document Frequency). The idea behind this metric is the intuition that a word is important for a document if it appears many times in it and at the same time it isn't a common word (a word that appears in many documents). Cosine normalization can be used if there is a need to map the values of the weights into the $[0,1]$ interval (Sebastiani, 2002).

$$w_{ij} = \frac{TFIDF(t_i, e_j)}{\sqrt{\sum_{s=1}^{|V|} (TFIDF(t_s, e_j))^2}}$$

where $|V|$ is the size of the vocabulary in use.

Feature Selection

A typical lexicon of words (e.g. for the English language) may consist of many thousands of words. Those words in a typical bag-of-words approach will constitute the application's feature space. A feature space that large is not only computationally inefficient but also misleading for many algorithms, as noisy or irrelevant features are taken under consideration and overfitting phenomena may occur (curse of dimensionality). Therefore, a considerable number of dimensionality reduction algorithms have been studied in the literature. We usually refer to these algorithms as *feature selection* methods, because dimensionality reduction is being achieved by trying to select the "best" features from the whole feature space. In text classification terms, this means to select words that distinguish one document category from another more efficiently. This is usually being accomplished by calculating a special quality measure for each word and then selecting to use only the top-N features of the rank.

Typical information retrieval measures like *TF-IDF* (see above) are still applicable, but now we use them to show the importance of a term for the whole corpus and not only for the specific document as when we used *TFIDF* function to calculate weights.

$$TFIDF_i = TF_i \log \frac{N}{DF_i}$$

where TF_i is the number of occurrences of the term i in the corpus and DF_i the number of different documents this term occurs. N is again the total number of documents.

Measures from the information theory field are more widely used. Most effective in text mining tasks as noted in (Yang & Pedersen, 1997) and therefore popular in email applications are the Information Gain and Chi-Squared Measure. Another widely used measure is Mutual Information (Yang & Pedersen, 1997). In general, a measure $M(t_i, c_j)$ indicates how much distinctive power term t_i has in order to distinguish c_j from other categories. See below the definition of Information Gain and Chi-Squared Measure.

Information Gain:

$$IG(t_i, c_j) = P(t_i, c_j) \log \frac{P(t_i, c_j)}{P(c_j)P(t_i)} + P(\bar{t}_i, c_i) \log \frac{P(\bar{t}_i, c_j)}{P(c_j)P(\bar{t}_i)}$$

Chi-Squared Measure:

$$x^2(t_i, c_j) = \frac{N[P(t_i, c_j)P(\bar{t}_i, \bar{c}_j) - P(t_i, \bar{c}_j)P(\bar{t}_i, c_j)]^2}{P(t_i)P(\bar{t}_i)P(c_j)P(\bar{c}_j)}$$

where N is the total number of documents, $P(t_i)$ is the number of documents where t_i occurs, $P(\bar{t}_i, c_j)$ is the number of c_j documents, where term t_i does not occur and so forth. Finally, some Machine Learning based methods for feature selection are described in (Montanes, Diaz, Ranilla, Combarro, & Fernandez, 2005) .

Email Classification

Most email mining tasks are being accomplished by using email classification at some point. In general, what email classification confronts is the assignment of an email message to one from a pre-defined set of categories. Automatic email classification aims at building a model (typically by using machine learning techniques), which will undertake this task on behalf of the user. Examples of applications are automatic mail categorization into folders, spam filtering and author identification. But there are also other applications, where we use classification in the process like automatic email summarization.

There are actually two kinds of classification. The first and simplest one is the flat classification when we have only one level of classes. The other category is known as hierarchical, where we have a hierarchy of classes and subclasses (Bekkerman, McCallum, & Huang, 2004; Itskevitch, 2001).

More formally, and in machine learning terms, if we have a set of predefined classes $C = \{c_1, ..., c_n\}$, we need to construct a function-like model, which assigns a mail message (e) to a class e.g. $M(e) \rightarrow C$. These models (also called *classifiers)* can be built with various machine learning techniques, such as Naïve Bayes (Sahami et al., 1998), Support Vector Machines (Klimt & Yang, 2004), Rule Learning (Pazzani, 2002), Decision Trees (Quinlan, 1986) based algorithms (Diao, Lu, & Wu, 2000) , Neural Networks (Clark, Koprinska, & Poon, 2003) and Inductive Logic Programming (Crawford, Kay, & McCreath, 2002). Most of the classification algorithms are compatible with the vector representation model. To build a classifier, a set of training examples is required. An example is a message that has already been categorized, usually by a user or a domain expert. An example is usually a vector $e = [w_1, w_2, ..., w_n, c_e]$ where c is the class that example e belongs to. This procedure of building the classifier is called *training.*

What is important in email classification is the fact that email is a dynamic environment and messages are constantly arriving. This means that although there might be a training set in availability, the classifier needs to be able to adapt new knowledge while new examples arrive. Therefore, it is of vital importance for classification algorithms to be characterized by the element of incrementality (P. Cunningham, N. Nowlan, S. Delany, & M. Haahr, 2003; Katakis, Tsoumakas, & Vlahavas, 2005; Richard Segal & Kephart, 2000).

The Naïve Bayes Classifier. The Naïve Bayes Classifier (John & Langley, 1995) has been used many times for email classification applications (Rennie, 2000; Sahami et al., 1998) . Its simplicity (not only it is computationally cost effective but also easy to implement), flexibility and considerable performance are the basic characteristics that made it so popular, not only in email applications, but in text classification in general (Lewis, 1998; Li & Jain, 1998; Mladenic, 1998). The Naïve Bayes Classifier is based on the simplifying assumption that the attribute values (e.g. the tf-idf values of an email vector) are conditionally independent. The Naïve

Bayesian classification of an email $e = [w_1, w_2 ..., w_n]$ into one category from $C = \{c_1, c_2, ..., c_n\}$ is calculated as follows (Mitchell, 1997):

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_i P(w_i \mid c_j)$$

where c_{NB} is the classification proposed by the Naïve Bayes algorithm. A proposal for probability estimation in text classification that can be found in (Mitchell, 1997) is displayed below:

$$P(w_i \mid c_j) = \frac{n_i + 1}{n_j + |V|}$$

where, n_j is the total number of words (not distinct) in all training examples belonging to c_j category, n_i is the number of times $word_i$ occurs among these n words and $|V|$ is the total number of distinct words found within the training data. Of course, there are other alternatives for calculating the probabilities (John & Langley, 1995).

Support Vector Machines. Since their significantly good performance was confirmed many times in the literature (Sebastiani, 2002) Support Vector Machines have gained popularity in text and email classification. In their initial and simplest form, Support Vector Machines are binary classifiers, meaning that they separate objects into only two pre-defined classes. In that case, the SVM classifier finds a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin (see Figure 3).

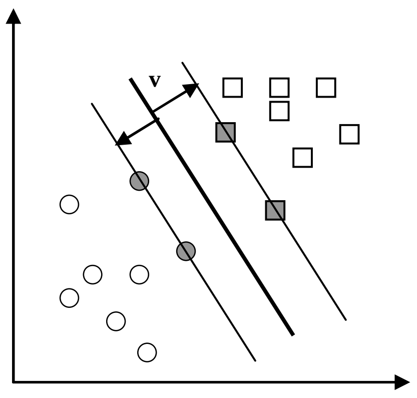


Figure 3: A Support Vector Machine classifier. The hyperplane separating the instances represented by the thick line. Items in grey represent the support vectors. V is the margin to be minimized.

The margin is defined as the distance of the hyperplane from the nearest of positive and negative examples (see Figure 3 – these examples are called “support vectors”). If $\vec{d}$ is the input document and $\vec{v}$ is the normal vector to the hyperplane (margin), then the output (proposed classification for the document $\vec{d}$) of the SVM classifier is calculated as follows:

$$C_{SVM} = sign\{\vec{v}\vec{d} + b\} = \begin{cases} +1, & if\ \ \vec{v} \cdot \vec{d} + b > 0 \\ -1, & else \end{cases}$$

The terminus is to maximize the margin which is represented by the $\vec{v}$ vector. The problem then could be redefined as an optimization problem:

Maximize: $|\vec{v}|$

Subject to: $y_i(\vec{v} \cdot \vec{d}_i + b) \geq 1$

where vectors $\vec{d}_i$ are all training examples (emails) and y_i is the real category (+1 or -1) that document d_i belongs to. Hence, the last constraint requires the correct classification of all training examples. Of course, this applies only if the problem is linearly separable. There are, however, already implemented modifications of this optimization problem which can be applied to non-linearly separable problems. Its solution usually requires a quadratic optimization method that will calculate the optimal hyperplane and margin. These methods are remarkably slow, especially in high-dimensional feature spaces like email classification. A fast training algorithm however, developed by Platt (1999) breaks the large QP problem down into a series of smaller ones (known as Sequential Minimal Optimization (SMO)) is already implemented in machine learning libraries like Weka (Witten & Frank, 2005) .

Another advantage of SVMs is that most of the times there is no need for term selection and no parameters require tuning. On the other hand, Support Vector Machines can still be computationally cost ineffective, especially for multiclass problems (Joachims, 1998).

Email Clustering

Email Clustering goes one step further. Subject-based folders can be automatically constructed starting from a set of incoming messages. In this case, the goal is to build automatic organization systems which will analyze an inbox recognize clusters of messages with the same concept, give an appropriate name to each cluster and then put all messages into their corresponding folders (Giuseppe Manco, Masciari, & Tagarelli, 2002; Surendran, Platt, & Renshaw, 2005).

The most widely used clustering algorithm in textual data is the K-Means algorithm. In order to group some points in K clusters, K-Means works in 4 basic steps:

1. Randomly choose K instances within the dataset and assign them as cluster centers
2. Assign the remaining instances to their closest cluster center
3. Find a new center for each cluster.
4. If the new cluster centers are identical to the previous ones, then the algorithm stops. Otherwise, repeat steps 2-4.

Calculation of distances between documents can be achieved by using a vector space model and a cosine similarity measure. Similarity between messages e_1 and e_2 is defined as:

$$Sim(\vec{e}_1,\vec{e}_2) = \cos(\vec{e}_1,\vec{e}_2) = \frac{\vec{e}_1 \cdot \vec{e}_2}{|\vec{e}_1||\vec{e}_2|}$$

Other similarity measures are proposed in (Baeza-Yates & Ribeiro-Neot, 1999) and (Strehl, Ghosh, & Mooney, 2000).

APPLICATIONS

While techniques for tasks like document summarization, organization, clustering and author identification are already explored in the literature, email data with its characteristics, as discussed in section 2, raises a new challenge to the community of Text Mining. Moreover, the domain itself offered the ground for new applications to be created. Such applications are: Automatic Answering, Thread Summarization and Spam Filtering.

Automatic Answering

Large companies usually maintain email centers (in conjunction with “call centers”) with employees committed to answer incoming messages. Those messages usually come from company clients and partners and many times address the same problems and queries. Automatic email answering is an effort to build mail centers or personalized software that will be able to analyse an incoming message and then propose or even send an applicable answer. Efforts towards this direction have been made recently in (Bickel & Scheffer, 2004; Busemann, Schmeier, & Arens, 2000; Scheffer, 2004).

Bickel and Scheffer (2004) describe methods for building an automatic answering system utilizing message pairs: questions and answers. Assuming that a respectable number of message pairs is available in a repository $R = \{(q_1, a_1),..,(q_n, a_n)\}$, they have proposed a number of approaches.

The simplest one is to find the question message q_i which is most similar to a new question message e and then propose the corresponding answer message a_i as the answer to e. Measures and methods to calculate document similarity are described in (Baeza-Yates & Ribeiro-Neot, 1999).

A more sophisticated approach treats the problem as a classification task. This is accomplished by grouping similar answers into categories (classes) $C_1,...,C_m \in C$, where $m \leq n$. The grouping of answers can be performed either by a human expert, or more efficiently by a clustering algorithm. For each cluster of answers $C_j = \{a_1,...,a_k\}$, where $k \leq n$, a default representative answer is automatically created by selecting the answer $a_i \in C_j$ which is most similar to the centroid of the cluster. Typically, if we have a group of vectors $C_j = \{a_1,...,a_k\}$ the weights of the centroid vector a_c are calculated as: $w_{i,c} = \frac{1}{k}\sum_{j=1}^{k} w_{i,j}$. Then, a classifier is trained on the dataset $D = \{(q_1, C_a),...,(q_n, C_b)\}$ consisting of $(question, classOfAnswer)$ pairs. When a new question message e arrives, the classifier outputs the predicted class and the representative answer of this class is proposed as the answer to e.

In another paper (Scheffer, 2004), Scheffer is discussing how automatic email answering could be enhanced by utilizing unlabeled data (a process called semi-supervised learning). He experiments with transductive Support Vector Machines (Joachims, 1999) and co-training (Blum & Mitchell, 1998). The approach is implemented in an integrated email manager for the Microsoft Outlook email client (Kockelkorn, Luneburg, & Scheffer, 2003).

Automatic Mail Organization into Folders

The growth of email usage has forced users to find ways to organize archive and manage their emails more efficiently. Many of them are organizing incoming messages into separate folders. Folders can be topic-oriented like “work”, ”personal” and ”funny”, people-specific like “John” and “Mary” or group-of-people-specific like “colleagues”, “family” and “friends”. Some users are archiving their messages according to importance and thus maintain folders like “urgent”, “for future reference”, “spam” etc. To achieve this, many users create some so-called *rules* to classify their mail. Those are heuristic rules searching for keywords in the message and then taking an action like moving or copying to a folder, deleting or forwarding the message etc.

```
if(sender=”John Smith” OR sender=”Mary Smith”)
then (moveInto FAMILY)

if(body contains “call for papers”)
then{(moveInto CFP)
     (forwardTo “COLLEAGUES”)}
```

where FAMILY, and CFP are folders and COLLEAGUES is a group of people (practically a list of addresses the user has created for mass emailing). Most Email clients today, support the creation of such rules.

What Machine Learning has to offer to this task is the automatic classification of incoming mail by observing past and current classifications made by the user (e.g. analyzing already existing folders or taking a current classification as an example). Thus, the user does not need to create the rules by himself. Furthermore, machine learning algorithms are able to classify a message, taking under consideration its content and not only by searching for specific keywords. This is usually achieved by combining statistical and linguistic techniques. It is extremely convenient for the user, since there are some concepts like “messages concerning my work” or “interesting messages” or “messages that I have to answer today” that cannot easily be described with a combination of keywords. In fact, a recent study (Ducheneaut & Belloti, 2001) notes that most users do not use conventional mail filters not only because they find them difficult to use, but also because they believe that two thirds of their mail volume would be impossible to filter automatically. Moreover, these concepts may change (e.g. the concept of "interesting message") from time to time. On the other hand a Machine Learning algorithm can learn to classify new messages just by silently observing past examples and can follow drift of concepts by accepting user feedback.

A lot of research has been recorded in the field (Clarck, Koprinska, & Poon, 2003; Cohen, 1996; Klimt & Yang, 2004; G. Manco, Masciari, Ruffolo, & Tagarelli, 2002) and lots of those ideas have been implemented into useful email tools (Graham-Cumming, 2002; Ho, 2003; Richard Segal & Kephard, 2000).

Popfile (Graham-Cumming, 2002) is a popular online tool for classifying email. It is written by John Graham-Cummings in Perl and it is inspired by ifile (Rennie, 2000). It stands between the mail server and client, retrieves messages from the server, classifies them and then sends them to the client. Popfile starts its training from point zero and normally all messages drop to the default inbox folder. If the user creates a bucket in popfile and then moves some message to it then popfile will start to learn how to classify similar messages. Users can create, destroy or merge buckets at any time. Naturally, popfile performs poorly in the beginning as it only has few examples to learn from, but reports are showing an average classification accuracy of 98,7% after 500 messages. Popfile implements a Naïve Bayesian classifier using a bag-of-words approach, but adding some extra handcrafted features (like the existence of html code, size of images, email addresses in the "to:" header field etc) mainly to enhance its spam filtering capabilities. This is because spammers use techniques like entering random text in spam messages in order to mislead the Naïve Bayes classifier.

Another email categorization tool, SwiftFile (formerly known as MailCat (R. Segal & Kephard, 1999)), an add-on to Lotus Notes, has been developed by IBM Research (Richard Segal & Kephard, 2000), and is emphasizing the need for incremental learning. With every new message arrival, SwiftFile predicts three destination folders. It places three buttons above the message, in order for the user to send them quickly to one of them. SwiftFile uses a TF-IDF type of classifier (basically a k Nearest Neighbour classifier (Cover & Hart, 1967) using TF-IDF weights (Sebastiani, 2002)), which is modified to support incremental learning and a bag-of-words representation, using word frequencies in message as weights. Each folder F is represented by its centroid vector, calculated from all messages in that folder. Similarity between centroid vectors and the new message is calculated and the system proposes the three folders with the highest similarity. The centroids of each folder are re-calculated after a new classification, in order to follow potential concept drift.

Other interesting software applications are EMMA (Ho, 2003), which uses multiple classification Ripple Down Rules (Kang, Compton, & Preston, 1995) and eMailSift (Aery & Chakravarthy, 2004), which uses a graph based mining approach for email classification.

There is although still some human effort involved in deciding and creating the subject-folders for the classification. This step could be avoided by using email clustering. For example in (Surendran et al., 2005) a personal email browser is built by discovering clusters of messages in the user's inbox. The most suitable noun phrase is then selected by analysing the messages in order to name the folder with a keyphrase that is representative of the folder's content. They use TF-IDF vectors to represent the messages and the K-Means algorithm for the clustering. Manco et al. (2002) are exploring the same problem, by taking under consideration similarity of structured and unstructured parts of an email message.

Mail and Thread Summarization

Text summarization has been previously explored in information technology literature (Hovy & Lin, 1997; Marcu, 1997; Zechner, 1996). The main motivation is again information overload. Large collections of text documents (news, articles, reviews, scientific documents, literature) are available in the world wide web or in digital libraries. It is impossible for a reader to find the time to read many documents and more importantly to adapt their content. Therefore, new techniques are being developed in order to automatically extract the gist of documents. In particular, some of those techniques embed machine learning algorithms, building *summarizers* that at some point of the summarization procedure are able to learn from examples (Neto, Freitas, & Kaestner, 2002).

The same need for summarization appears in electronic mail. There is a certain category of email users that receive hundreds of messages per day. Some of them are newsletters, others are business decision-making messages from colleagues, appointment arrangements etc. It would be extremely useful for them if they could avoid reading all of those messages and instead read only the most important and necessary parts and then decide if the messages demand immediate attention. From a summary, they could also find out if a newsletter for example is interesting for them or not and only then read the full text. Again, data mining techniques are explored in order to build trainable tools for summarization.

Muresan et al. (2001) describe an email summarization method that utilizes Natural Language Processing and Machine Learning techniques. The basic steps of their method are: a) extraction of candidate noun phrases (NPs) from the email message, b) linguistic filtering of the candidate NPs, such as removing common words and unimportant modifiers, and c) induction of a model that will classify the filtered NPs into those that are important and should be included in the summary and those that aren't.

The last step is accomplished with the aid of machine learning algorithms. Each candidate NP is described using a vector of features, such as the TF-IDF measure of the head of the candidate NP, the length of the NP (in words and characters) and the position of the NP. To construct a training set, a large number of NPs extracted from several emails was manually tagged as important or not. Experimental results with Decision Tree, Rule Induction and Decision Forest learning algorithms led to several interesting conclusions, including that NPs are better than n-grams (n consecutive words) for the phrase level representation of email messages and that linguistic filtering enhances the performance of Machine Learning algorithms.

The application of summarizing email threads (conversations among two or more people carried out by exchange of messages) has also gained interest. Email threads might consist of many email messages and it would be beneficial if a user could avoid reading all of them and instead read just the summary up to that point. In their technical report (Lam, Rohall, Schmandt, & Stern, 2002), Lam et al are trying to exploit email structure to improve thread summarization. The basic idea is to identify a message as a part of an email thread and then try to propose a summary by extracting knowledge from the parent (ancestor) messages as well. Wan and McKeown (2004) are exploring the same problem in business decision-making email threads. They assume that in threads like these there is always an issue-message (containing an issue-sentence) which sets the problem or a query to the others. The other messages are supposed to be response-messages, each one containing a response-sentence. Hence, in order to summarize the thread, all issue and response sentences have to be identified and collected. The method proposed to identify the issue sentence is to take all issue-message sentences and compare them with each response-message. Thus, the sentence that is more similar to the response-messages is considered the issue-sentence. Finally, to extract the response-sentences authors noticed that simply extracting the first sentence of every response-message is an effortless but effective approach.

A system for email summarization for the Spanish language is described in (Alonso, Casas, Castellon, & Padro, 2004) and is available for demonstration at http://www.lsi.upc.edu/~bcasas/carpanta/demo.html.

Spam Filtering

The main goal of spam filtering is to identify and sort out unsolicited commercial mails (spam) from a user's mail stream. Spam mail (also known as junk or bulk email) has begun as small annoyance in the early days of email to become a major industry problem in the last five years. The large amount of spam not only causes bandwidth (and therefore financial) problems, but also takes up valuable time from email users who try to separate and delete many unsolicited messages every day. Moreover, many spam messages include pornographic content inappropriate for children.

To defeat spam, the information technology community begun constructing spam filters in order to delete automatically such unwanted incoming messages. There are many different strategies proposed and implemented in the battle against spam. We could organize them into two general groups:

<u>*Technical or Non-Statistical Approaches*</u>*:* Which includes white and black lists, digital signatures and handcrafted rules.

<u>*Machine Learning or Statistical Approaches*</u>: Which includes statistical linguistic analysis and machine learning algorithms.

Spam filtering techniques could be additionally organized into two other categories:

<u>*Server based solutions*</u>: Where messages are filtered on the ISP's mail server.

<u>*Client based solutions*</u>*:* Where messages are categorized on the user's computer.

While the obvious advantage of the first category is that spam messages never reach the client's computer, server based filters are not always personalized and the user must check periodically the junk folder on the server, in order to see if there are any misclassified messages. Many different machine learning classifiers have been tested in the bibliography. A not-complete list can be seen below.

Naïve Bayes (Sahami et al., 1998)

Memory-Based Approaches (Sakkis et al., 2003)

Boosting Techniques (Carreras & Marquez, 2001)

Case-Based Reasoning (P. Cunningham, N. Nowlan, S. J. Delany, & M. Haahr, 2003)

Support Vector Machines (Drucker, Vapnik, & We, 1999)

Latent Semantic Indexing (Gee, 2003)

Stacking Classifiers (Sakkis et al., 2001)

Direct comparison of those methods is difficult because of the use of different corpora and optimization of the algorithms. Developers, however, have shown their preference to the Naïve Bayes classifier mainly because of its simplicity, flexibility, computational cost and decent performance[4,5] . Nevertheless, advanced high-accurate software, like SpamAssasin[6], usually combines techniques from both categories: Statistical and Non-Statistical.

To some extent, the task of spam filtering is similar to text and email classification. However, there are some characteristics that distinguish spam filtering from other classification tasks.

In spam filtering, classification mistakes are not all of equal importance. To classify a legitimate email as spam is a much more severe mistake than letting a non-legitimate message pass the filter. Thus, we should make sure that we mark a message as spam only if the classifier predicts so with high confidence. In Naïve Bayes for example classification of an email message e as spam should be made, if and only if $P(e = SPAM) > t$, where t is a threshold close to 1 (e.g. 0.98). The same applies in the evaluation of the algorithm. One false positive mistake should be counted as, for example, 100 false negative mistakes.

Another distinction is that in spam filtering we should probably use extra features, like information from headers, morphological characteristics or punctuation marks, which in fact are great indicators, as they exist in abundance in spam messages. Technical approaches like black and white lists are maybe unavoidable as spammers started to input some random “innocent” text in messages in order to mislead the classifier. Additionally, classes in spam filtering are far more heterogeneous in content than a typical text or email classification problem, meaning that there are a lot of different types of spam messages and a lot of different types of legitimate messages, a fact that makes the distinction even more difficult. Moreover, spammers tend to change the characteristics of spam messages and thus the need for incremental learning is more obvious in this case.

Although some of the above mentioned filters are remarkably accurate (some of them reaching 99% accuracy with only a few false positives) the problem is still remaining. Two conferences are taking place every year[7], gathering professionals and academics from all over the world introducing and discussing new ideas and trends.

Other Interesting Applications

Apart from the applications mentioned before, there are still some other worth mentioning. For example Lewis (1997) mentions that there might be a need to identify if a mail belongs to a thread without the use of structural information inserted by email clients (e.g. the well known “RE:” prefix). Thus, the use of a TF-IDF classifier, using similarity measures is proposed in order to find messages belonging to the same thread.

Author identification is another interesting application explored in (Vel et al., 2001). Email evidence can be central in cases of sexual harassment or racial vilification, threats, blackmailing and so on. In these cases, the sender will attempt to hide his/her true identity in order to avoid detection. For example, the sender's address can be hidden by using an anonymous mail server or the email's contents and header information can be modified in an attempt to hide the true identity of the user. Although author identification has been investigated before, email author identification displays again some unique characteristics. The text is significantly shorter, but we could at the same time exploit information from headers, like attachments, timestamps etc. Particularly the authors of this work use a set of Style Marker and Structural features like: Number of blank lines, average sentence length, average word length, vocabulary richness, existence of a greeting or a farewell acknowledgement, number of attachments etc. Finally, for the identification of the email author they use an SVM classifier. Same authors have experimented with the identification of the email author’s gender (Corney et al., 2002).

CONCLUSIONS AND FUTURE TRENDS

Email is now extremely important for interpersonal communication and professional life. Therefore its problems demand immediate attention and efficient solutions. What Data Mining and Machine Learning have to offer to the clarification of email overload is intelligent techniques for the automatization of many email management tasks. Email categorization into folders, email answering and summarization, spam filtering, are only a few representatives. All of these applications have been explored repeatedly in the literature with very promising results, but spam filtering seems to gather the greater attention of all, probably because of its negative financial impact. It is worth noticing though, that many of these applications are extremely demanding in terms of accuracy mainly because information in email data can be significantly important. Therefore there is still space and need for improvement in performance in many of the applications mentioned above.

Email Structure is very difficult to be generated from plain text, and therefore we treat email, almost always as unstructured text. On the other hand HTML, which until now is removed in the preprocessing step, could help to give an at-least semi-structured form to the email. Knowledge discovery from structured information is more convenient and maybe more effort should be made in this direction. For example we could transform HTML messages into XML using XSL-T patterns.

A new idea that seems promising is Semantic Email. In parallel with Semantic Web, Email could be enriched with meta-tags in order to describe better the information included in the message. As discussed in (McDowell, Etzioni, Halevy, & Levy, 2004), applications like Information Dissemination, Event Planning, Report Generation and Email Auctions / Giveaways could be achieved with the use of the Semantic Email.

Email mining raised new difficulties and challenges for the text mining community. New solutions had to be proposed in already discussed areas due to email data peculiarity. Additionally, domain-specific problems provoked the development of new applications like spam filtering, email answering and thread summarization.

While effective solutions have been proposed to most email problems, not all of them have been implemented in popular email clients. In fact, with the exception of spam filtering which is now integrated in most commercial clients, no other applications have been used widely from the average user. There is therefore an obvious need to implement those methods and integrate them into useful and accurate software which will let people take back control of their mailboxes.

REFERENCES

Aery, M., & Chakravarthy, S. (2004). *eMailSift: Adapting Graph Mining Techniques for Email Classification*.

Alonso, L., Casas, B., Castellon, I., & Padro, L. (2004). *Knowledge-intensive automatic e-mail summarization in CARPANTA.* Paper presented at the ACL '04, Barcelona, Spain.

Androutsopoulos, I., Koutsias, J., & Chandrinos, K. V. (2000). *An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages* Paper presented at the 23rd annual international ACM SIGIR conference on Research and development in information retrieval Athens, Greece.

Baeza-Yates, R., & Ribeiro-Neot, B. A. (1999). *Modern Information Retrieval*.

Bekkerman, R., McCallum, A., & Huang, G. (2004). *Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora*.

Bickel, S., & Scheffer, T. (2004). *Learning from Message Pairs for Automatic Email Answering.* Paper presented at the Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy.

Blum, A., & Mitchell, T. (1998). *Combining Labeled and Unlabelled Data with Co-training.* Paper presented at the Workshop on Computational Learning Theory.

Boone, G. (1998). *Concept features in Re:Agent, an intelligent email agent.* Paper presented at the International Conference on Autonomous Systems, Minneapolis, Minnesota, United States.

Brutlag, J. D., & Meek, C. (2000). *Challenges of the Email Domain for Text Classification.* Paper presented at the Seventeenth International Conference on Machine Learning.

Busemann, S., Schmeier, S., & Arens, R. G. (2000). *Message Classification in the call center*. Paper presented at the Sixth Conference on Applied Natural Language Processing, Seattle, Washington.

Carreras, X., & Marquez, L. (2001). *Boosting Trees for Anti-Spam Email Filtering.* Paper presented at the RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing.

Clarck, J., Koprinska, I., & Poon, J. (2003). *A Neural Network Based Approach to Automated E-Mail Classification.* Paper presented at the IEEE/WIC International Conference on Web Intelligence (WI '03).

Clark, J., Koprinska, I., & Poon, J. (2003). *A Neural Network Based Approach to Automated E-Mail Classification*. Paper presented at the IEEE/WIC International Conference on Web Intelligence (WI'03), Halifax, Canada.

Cohen, W. (1996). *Learning rules that classify e-mail*. Paper presented at the AAAI Spring Symposium on Machine Learning in Information Access.

Corney, M., Vel, O. d., Anderson, A., & Mohay, G. (2002). *Gender-Preferential Text Mining of E-mail Discourse*. Paper presented at the ACSAC '02: 18th Annual Computer Security Applications Conference

Cover, T., & Hart, P. (1967). Nearest Neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21-27.

Crawford, E., Kay, J., & McCreath, E. (2002). *IEMS - The Intelligent email Sorter*. Paper presented at the International Conference on Machine Learning.

Cunningham, P., Nowlan, N., Delany, S., & Haahr, M. (2003). *A Case-Based Approach to Spam Filtering that Can Track Concept Drift*. Paper presented at the ICCBR'03 Workshop on Long-Lived CBR Systems, Trondheim, Norway.

Cunningham, P., Nowlan, N., Delany, S. J., & Haahr, M. (2003). *A Case-Based Approach to Spam Filtering that Can Track Concept Drift*. Paper presented at the The ICCBR'03 Workshop on Long-Lived CBR Systems.

Diao, Y., Lu, H., & Wu, D. (2000). *A comparative study of classification-based personal e-mail filtering*. Paper presented at the PAKDD-00, 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Kyoto, Japan.

Drucker, H., Vapnik, V., & We, D. (1999). Support Vector Machines for Spam Categorization. *IEEE Transactions on Neural Networks, 10*(5), 1048--1054.

Ducheneaut, N., & Belloti, V. (2001). E-mail as habitat: an exploration of embedded personal information management. *Interactions, 8*(5), 30-38.

Gee, K. R. (2003). *Using Latent Semantic Indexing to Filter Spam*. Paper presented at the ACM Symposium on Applied Computing, Data Mining Track.

Graham-Cumming, J. (2002). PopFile: Automatic Email Classification (http://popfile.sourceforge.net).

Ho, V. (2003). *EMMA: An E-Mail Management Assistant.* Paper presented at the 2003 IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), 13-17 October 2003, Halifax, Canada.

Hovy, E., & Lin, C. (1997). Automated text summarization in SUMMARIST. In I. Mani & M. Maybury (Eds.), *Advances in Automatic Text Summarization* (pp. 18-24).

Itskevitch, J. (2001). *Automatic Hierarchical E-Mail Classification using association rules.* Simon Fraser University.

Joachims, T. (1998). *Text categorization with Support Vector Machines: Learning with many relevant features.* Paper presented at the ECML-98, 10th European Conference on Machine Learning.

Joachims, T. (1999). *Transductive Inference for Text Classification using Support Vector Machines.* Paper presented at the ICML-99, 16th International Conference on Machine Learning.

John, G. H., & Langley, P. (1995). *Estimating Continuous Distributions in Bayesian Classifiers.* Paper presented at the UAI '95: Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada.

Johnston, G. (2002). We've Got Mail: 60 Billion Daily. *PCWorld (http://www.pcworld.com/news/article/0,aid,105525,00.asp).*

Kang, B., Compton, P., & Preston, P. (1995). *Multiple Classification Riple Down Rules: Evaluation and Possibilities.* Paper presented at the 9th AAAI-Sponsored Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Canada.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2005). *On the utility of Incremental Feature Selection for the Classification of Textual Data Streams.* Paper presented at the PCI 2005: 10th Panhellenic Conference on Informatics, Volos, Greece.

Klimt, B., & Yang, Y. (2004, September 2004). *The Enron Corpus: A New Dataset for Email Classification Research.* Paper presented at the Machine Learning: ECML 2004: 15th European Conference on Machine Learning, Pisa, Italy.

Kockelkorn, M., Luneburg, A., & Scheffer, T. (2003). *Learning to answer emails.* Paper presented at the International Symposium on Intelligent Data Analysis.

Lam, D., Rohall, S. L., Schmandt, C., & Stern, M. K. (2002). *Exploiting E-Mail Structure to Improve Summarization.*

Lewis, D. D. (1992). *An evaluation of phrasal and clustered representations on a text categorization task.* Paper presented at the 15th annual international ACM SIGIR conference on Research and development in information retrieval, Copenhagen, Denmark.

Lewis, D. D. (1998). *Naive Bayes at forty: The independence assumption in information retrieval.* Paper presented at the ECML-98, 10th European Conference on Machine Learning, Chemnitz, DE.

Lewis, D. D., & Knowles, K. A. (1997). Threading Electronic Mail: A Preliminary Study. *Information Processing and Management, 33*(2), 209-217.

Li, Y. H., & Jain, A. K. (1998). Classification of text documents. *The Computer Journal, 41*(8), 537-546.

Manco, G., Masciari, E., Ruffolo, M., & Tagarelli, A. (2002). *Towards and Adaptive Mail Classifier.* Paper presented at the AIIA 2002.

Manco, G., Masciari, E., & Tagarelli, A. (2002). *A Framework for Adaptive Mail Classification.* Paper presented at the ICTAI '02: 14th IEEE International Conference on Tools with Artificial Intelligence, Washington, DV, USA.

Marcu, D. (1997). *From Discourse Structures to Text Summaries.* Paper presented at the 14th National Conference on Artificial Intelligence (AAAI-97), Providence, Rhode Island.

McDowell, L., Etzioni, O., Halevy, A. Y., & Levy, H. (2004). *Semantic email.* Paper presented at the 13th International World Wide Web Conference.

Miller, R. (2003). Email: The Other Content Management. *EContent Magazine (http://www.ecmag.net/?ArticleID=882)*.

Mitchell, T. M. (1997). *Machine Learning*: McGraw-Hill.

Mladenic, D. (1998). *Feature subset selection in text learning.* Paper presented at the ECML-98, 10th European Conference on Machine Learning, Chmnitz, DE.

Montanes, E., Diaz, I., Ranilla, J., Combarro, E. F., & Fernandez, J. (2005). Scoring and Selecting Terms for Text Categorization. *IEEE Intelligent Systems, 20*(3), 40-47.

Muresan, S., Tzoukerman, E., & Klavans, J. L. (2001). *Combining Linguistic and Machine Learning Techniques for Email Summarization.* Paper presented at the Proceedings of CoNLL-2001, Toulouse, France.

Neto, J. L., Freitas, A. A., & Kaestner, C. A. A. (2002). *Automatic Text Summarization Using a Machine Learning Approach*. Paper presented at the SBIA '02: 16th Brazilian Symposium on Artificial Intelligence

Pazzani, M. J. (2002). *Representation of electronic mail filtering profiles: a user study*. Paper presented at the Intelligent User Interfaces.

Platt, J. C. (1999). Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Scholkopf, C. Burges & A. Smola (Eds.), *Advances in Kernel Methods - Support Vector Learning* (pp. 185-207): MIT Press.

Porter, M. F. (1997). An algorithm for suffix stripping. In *Readings in information retrieval* (pp. 313-316): Morgan Kaufmann Publishers Inc.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning, 1*, 81-106.

Rennie, J. D. M. (2000). *ifile: An Application of Machine Learning to E-Mail Filtering*. Paper presented at the KDD-2000 Workshop on Text Mining.

Rockbridge Associates , I. (2004). *National Technology Readiness Survey, Summary Report, Prepared by: Rockbridge Associates, Inc. February 3, 2005. Available at: http://www.rockresearch.com/press_releases/NTRS_2004.pdf*.

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*. Madison, Wisconsin: AAAI Technical Report WS-98-05.

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. D., & Stamatopoulos, P. (2001). *Stacking Classifiers for Anti-Spam Filtering of E-Mail*. Paper presented at the EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing, Pittsburgh, US.

Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C. D., & Stamatopoulos, P. (2003). A Memory-Based Approach to Anti-Spam Filtering for Mailing Lists. *Information Retrieval, 6*(1), 49-73.

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

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM, 18*(11), 613-620.

Scheffer, T. (2004). Email Answering Assistance by Semi-Supervised Text Classification. *Intelligent Data Analysis, 8*(5), 481-493.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys, 34*(1), 1-47.

Segal, R., & Kephard, J. (1999). MailCat: An Intelligent Assistant for Organizing E-Mail. *Third International Conference on Autonomous Agents*.

Segal, R., & Kephard, J. O. (2000). *Incremental Learning in SwiftFile.* Paper presented at the Seventh International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, USA.

Segal, R., & Kephart, J. O. (2000). *Incremental Learning in SwiftFile.* Paper presented at the 7th International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA.

Strehl, A., Ghosh, J., & Mooney, R. (2000). *Impact of Similarity Measures on Web-page Clustering.* Paper presented at the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000), Austin, Texas, USA.

Surendran, A. C., Platt, J. C., & Renshaw, E. (2005). *Automatic Discovery of Personal Topics to Organize Email.* Paper presented at the 2nd Conference on Email ant Anti-Spam, Stanford University, USA.

Vel, O. d., Anderson, A., Corney, M., & Mohay, G. (2001). Mining e-mail content for author identification forensics. *SIGMOD Rec., 30*(4), 55-64.

Wan, S., & McKeown, K. (2004). *Generating Overview Summaries of Ongoing Email Thread Discussions.* Paper presented at the COLING 2004, the 20th International Conference on Computational Linguistics, Geneva, Switzerland.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques* (2nd ed.). San Francisco: Morgan Kaufmann.

Yang, Y., & Pedersen, J. O. (1997). *A comparative study on feature selection in text categorization.* Paper presented at the ICML-97, 14th International Conference on Machine Learning, Nashville, US.

Zechner, K. (1996). *Fast generation of abstracts from general domain text corpora by extracting relevant sentences.* Paper presented at the 16th conference on Computational Linguistics, Copenhagen, Denmark.

Zhang, L., Zhu, J., & Yao, T. (2004). An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, *3*(4), 243-269.

Endnotes

[1] http://www.idc.com/home.jhtml
[2] Microsoft Outlook: http://www.microsoft.com/outlook/
[3] Mozilla ThunderBird: http://www.mozilla.org/products/thunderbird/
[4] Spambayes (http://spambayes.sourceforge.net)
[5] Popfile (http://popfile.sourceforge.net)
[6] The Apache Spam Assassin Project (http://spamassasin.sourceforge.net
[7] Spam Conference: http://spamconference.org/, Conference on Email and Anti-Spam: http://www.ceas.cc/