P@rty: A Personal Email Agent

Yu-hsiung Deng Thong-han Tsai Jane Yung-jen Hsu
Dept. of Computer Science Information Engineering
National of Taiwan University
1 Sec4 Roosevelt Rd., Taipei 106, Taiwan
URL: http://mobile.csie.ntu.edu.tw/~deng
{deng@robot.csie.ntu.edu.tw}

Abstract

Electronic mail provides an essential communication media for people all over the world. This paper presents a personal email agent, named P@rty, which helps its user manage the growing volume of messages. P@rty is designed to 1) *classify* the user's incoming messages into folders automatically, and 2) *prioritize* each incoming message based on the user's preferences. The classification algorithm is a hybrid of statistical keyword-based method and case-based reasoning. The agent prioritizes messages based on a *user profile*, which is constructed from both positive and negative feedback from the user over time. Preliminary results from our experiments showed that the proposed approach is able to classify mail messages with good accuracy after training.

Introduction

As the most popular application on the Internet, electronic mail (email) has become an increasingly critical tool to running a business and/or one's daily life. Many people are overloaded with a large number of emails on a regular basis, and important messages can often get buried under piles of other emails by mistake. Existing email software typically offers functions that help users manage multiple email accounts, organize their emails into folders, specify filters to sort emails automatically by subject, sender, etc. and to block unwanted emails. However, having to perform such organizational tasks manually is both time-consuming and tedious for most users.

Instead of simply offering tools that are manipulated by a human user, an email agent attempts to organize email messages automatically based on its knowledge about each individual user. In particular, the agent should be able to classify incoming email messages into folders, and to prioritize them so that the user can focus on more important emails first. This research has developed a personal email agent, named P@rty, which provides such functionality.

Email classification belongs to the general problem of automatic document classification [5]. However, the choices of folders depend on the subjective view of each individual user. The folders are usually not defined a priori, and additional folders can be created as a user's messages accumulate. As a result, email classification involves handling a fair amount of exceptions, and creating additional folders when necessary. On the other hand, while prioritization of emails can be viewed as a classification problem as well, it is much more dynamic in nature. As a person carries out her daily job requirements, her interests shift with time and context. For example, messages related to a specific event may lose their significance when the event is over; messages advertising a sale may be more important during holiday seasons when one has a stronger motivation for making purchases.

Standard classification techniques are not adequate for processing emails. In the following sections, we will start by describing the representation of an email message for classification. A similar structure is used to represent an email folder. A hybrid classification algorithm, SEC (**S**tatistical method with **E**xception handling using **C**ase-based reasoning) is introduced. The method allows the agent to deal with exceptions in classifying emails. Finally, a time-dependent scheme for prioritizing emails is proposed to accommodate the dynamics in user preferences.

Representation of Emails and Folders

There are three kinds of text documents. The first kind has no specific structure, such as plain-text documents. The second kind has a well-defined structure, such as SGML documents. The structure of a document can provide clues for classification. The last kind of documents has a partial structure. For example, email messages and news articles are semi-structured, each with a structured header part and an unstructured body of text. The structured part of the document can help us identify specific attributes useful for classification.

The proposed representation is an extension of the standard *Vector Space Model* for text documents [6]. Given an email, there are some informative features in its header part, such as ¡§From¡¨, ¡§To¡¨, and ¡§Subject¡¨ fields (RFC 822). A user often sorts mails into different folders according to the sender or subject of each message. Instead of extracting all the keywords and put them in one bag, the keywords from individual fields are kept and used separately.

As there is no explicit field in the body of an email message, which could be a plain text or an encoded binary file, a strategy to extract information is required. The strategy presented in this paper is straightforward and simple. Our approach is parsing the content of the body part and retrieving the tokens that are Email addresses, nouns, and special strings that are not nouns, verbs, adjectives, or adverbs. The special string above could be a human name or an abbreviation of a project, etc. It is probably informative although it might be just a misspelling word. The same strategy is performed in the ¡§subject¡ field of the header part. There are 6 fields extracted from a mail eventually. They are showed below respectively.

Subject: nouns in the subject field of the header part.
Sender: sender address in the header part.
Receiver: receiver addresses in the header part.
Address: E-mail addresses occurred in the body part.
Specific: special strings in the subject and body part.
Keyword: nouns in the body part.

is introduced in this paper. Mails are represented as vectors, which are made up of these six fields. Elements, which are called terms in the following sections, in the vectors are weighted according to their term frequencies and ***centroid*** values. The word "centroid" refers to the concentration of a term among classified classes. It is defined by equation (1).

where N_{jk}^{f} refers to the number of the term k occurred in the field j of the folder f, and N_{jk} refers to the total number of the term k occurred in the field j of all folders.

A mail is represented as following.

$$M = (m_1, m_2, m_3, m_4, m_5, m_6), \quad (2)$$

where m_i represents one of the six fields mentioned above, and

$$m_i = (tf_{i1} ¡ £_{i1}, tf_{i2} ¡ £_{i2}, tf_{i3} ¡ £_{i3}, ¡ , tf_{in} ¡ £_{in}), \quad (3)$$

where tf_{ij} is a normalized term frequency, a value from 0 to 1, of the j-th term in corpus, and $£_{ij}$ is the centroid value of the same term.

In order to simplify calculation, folders into which mails are classified are also represented as the same structure as mails. Calculating similarities between mails and folders helps make decisions to classify mails. High centroid value of a term means the term partially appearing in some certain folders. By contrast, low centroid value shows that it has a more average distribution. A folder is represented as following.

$$f_{jk} \quad \sqrt{f \quad \frac{N_{jk}^{f}}{N_{jk}}^{2}} \quad , \quad (1)$$

$$F = (f_1, f_2, f_3, f_4, f_5, f_6), \quad (4)$$

where f_i represents one of six fields.

$$fi = (df_{i1} ¡ £_{i1}, df_{i2} ¡ £_{i2}, df_{i3} ¡ £_{i3}, ¡ , df_{in} ¡ £_{in}), \quad (5)$$

where df_{ij} is a normalized document frequency, a value from 0 to 1, of the j-th term in corpus, and $£_{ij}$ is the centroid value of the same term.

Classification

In recent years, several people have tried to apply machine learning and information retrieval techniques to text classification. The NewT [7] system is a personalized news filtering system. It uses genetic algorithms to decide what document should be filtered. William Cohen also has worked on email classification. He compared two kinds of text classification algorithms. One is a traditional IR method, TFIDF. The other method is developed by himself, a rule-based method named RIPPER [1]. In his experiments, he confirmed that the RIPPER¡¦s performance is better than TFIDF [5]. CMU¡¦s machine learning group has been very active in the field. They applied the naive Bayesian classifier to the problem of email classification. Their system ifile [4] has a correctness of over 70%. The classification scheme used in this research combines both the rule-based and case-based methods in order to handle the special characteristics of emails. The rule-based approach performs classification in the large, while the case-based approach takes care of exception handling.

Our focus in this section is on classifying semi-structured texts like emails. The classification heuristic provided here is the distribution status of terms. A mail having more terms with high centroid values over a folder is considered more "similar" to that folder than others. The similarity function between a mail and a folder is defined as following.

$$Similarity ¡ M , F ¡ = _i ¡ m_i ¡ f_i ¡ ^ _i , \quad (6)$$

where $_i$ is the weights of the field i.

By sorting similarity scores, a mail could be first approximately classified into a folder that is the most "similar" to it. To get a more appropriate answer, exceptional cases must be considered.

Exception

After the approximate classification, users still have chances to change the decisions made by the agent. In one case, a mail is manually filed into a folder F_d which is totally dissimilar to it in the vector space, but it perhaps makes sense to users. This mail becomes the ¡§negative exception¡¨ of F_d. In the second case, a mail is filed into a folder, which is similar to it, but not the most similar one. It means that there is at least one another folder, named F_a, more similar to this mail. As a result, this mail becomes the ¡§positive exception¡ of F_a. An intelligent agent should be able to learn from these. When a new mail is similar to some previous

exceptions, the solutions of the exceptions will be more appropriate than the approximate decisions for it.

In this module, each folder keeps both these two exceptional cases in databases to remind the agent. In the second case, there are mappings between exceptional mails and their finally filed folders in the database as shown in Table 1.

Exceptions	Filed Folders
M_a	F_9
M_g	F_2
M_l	F_5
...	¡
M_w	F_2

Table 1. The database of the folder F_n in the 2nd case is a mapping of exceptions and their finally filed folders.

A similarity function between two mails is required to calculate the similarity between a new mail and an exception, and is defined below.

Similarity ¡ *M* , M_i ¡ = $_i$ ¡ m_i ¡ m'_i ¡ ^ $_i$, (7)

where $_i$ is the weights of the field *i*.

Algorithm

The heuristic of the classifying algorithm is first getting an approximate answer, searching for the potential similar exceptions, and applying the most suitable solution of one similar exception, if it exists. When searching for the suitable exceptions, the algorithm of **case-based reasoning** is used[3]. The following is the simplified classifying algorithm.

1. For each new mail M_{new}, find a set ***SimSet*** = {F_k |*Similarity*(M_{new}, F_k) ƒ },where ƒ is the similarity threshold.

2. If *SimSet* = , go to setp 6.

3. Choose the folder F_{approx} whose similarity value is the highest as the approximate solution.

4. Find a set ***OuterSet*** = {M_{outer} | *Similarity*(M_{new}, M_{outer}) ƒ }, where M_{outer} is the exception mail in the second case of the folder F_{approx}.

5. If *OuterSet* = , return F_{approx},
   else return $F_{confidence}$ selected from ***Confidence Rating***[3], which is described later.

6. Search each folder for exceptions in the first case, and collect a set ***InnerSet*** = {M_{inner} | *Similarity*(M_{new}, M_{inner}) ƒ }, where M_{inner} is the exception mail in the first case of all folders.

7. If *InnerSet* = , return NULL, which means no recommended folders,
   else return $F_{confidence}$ selected from *Confidence Rating*.

Simplified classification algorithm

Confidence Rating

As described in above algorithm, OuterSet and InnerSet are made up of pairs of the exception and its filed folder. In the folder part, folders could possibly be the same with, or differ from each other in the extreme. Confidence Rating is a confident level to a specific folder.

$$Confidence(M, F) \quad \frac{\text{Similarity}(M, M_{exc})}{\sum_{all\ M_{exc}} \text{Similarity}(M, M_{exc})} \quad 100\% \qquad (8)$$

Prioritization

Prioritizing mails by their importance to different users is another functionality of *P@rty*. The importance of a mail to a single user could be mapped to a real number. The value exceeding a positive threshold indicates the email is important to the user. When the value is less than a negative threshold, it implies that the email might be refused by the user. Otherwise, the mails with values between the positive threshold and negative threshold are considered as unimportant mails.

Prioritization starts from creating users' profiles first. Every user has an empty profile in the beginning. After P@rty receives the feedback about likes or dislikes of read mails from the user, it keeps track of "terms" appeared in those read ones. The word "term" has the same definition with one described in the classification module. A personalized profile consists of ***two*** sets of terms. These sets recorded terms appeared in favorite mails and in refused mails, and are called the **favorite set** and **refused set**, respectively. There is still some information about each single term provided in a profile, such as appearance periods, and the latest date at which it appeared.

Weighting each term in that mail is feasible. The situation that terms partially appears in the favorite set rather than the refused set imply that new mails with these terms are probably important or interesting to the user. On the contrary, mails with terms mostly occurring in the refused set are probably disliked ones. ***Term-relevant weighting***[1] is introduced as equation (9).

$$f_{jk} = log \quad \frac{\frac{r_k}{R \quad r_k}}{\frac{s_k}{I \quad s_k}} \qquad , \qquad (9)$$

where *j* refers to field *j*, *R* is the total number of relevant mails, r_k is the number of occurrences of term *k* in the relevant mails, *I* refers to the total number of irrelevant mails, and s_k is the number of occurrences of term *k* in the irrelevant mails.

People may change their preferences or priorities about mails. If weighting terms only by term-relevance mentioned above, it will never reflect the changes of a

user's preference because terms getting high weights before still get high values, even the user doesn't like them any more. By declining weights of terms until the user gives a positive or negative feedback[2], P@rty is able to adapt users' preferences. While the weights of some other terms decrease to zero gradually without getting any feedback, the terms getting feedback will have the highest weight(value 1 in this module) immediately. In order to get the declining period of each term, the same period is initially set to all terms. After the user's feedback, the periods of terms in a mail are updated separately by their own time elapsed between two occurrences. The declining function is defined below:

$$d_{jk} \quad \frac{1}{\frac{time_{jk}}{h_{jk}} \quad 1} \qquad (10)$$

where h_{jk} is the declining peroid, and $time_{jk}$ is the time elapsed between the last two occurrences.

Via combining the equations (9) and (10), following is the weighting function of term *k* in field *j*.

$w_{jk} = tf_{jk} ¡ \ f_{jk} ¡ \ d_{jk}$,
(11)

The summation of the weights of the terms occurring in the favorite set in a mail is an index showing how this mail matches the favorite `preference.

$$Match_{favorite}(M) = \ _j (\ _{term\ k \quad favorite\ set} w_{jk}) ¡ \ _j, \qquad (12)$$

where $_j$ is the weights of the field *j*.

The summation of the weights of the terms occurring in the refused set in a mail is an index showing how this mail matches the refused preference.

$$Match_{refused}(M) = \ _j (\ _{term\ k \quad refused\ set} w_{jk}) ¡ \ _j,$$
(13)

where $_j$ is the weights of the field *j*.

An index, defined as the equation (14), combines these two matching indices which maches the opposite preferences, and is used to prioritize the mail it indicates.

$$PrefIndex(M) = [Match_f(M) - Match_r(M)] + Match_f(M) * \{ Match_r(M) / [Match_f(M) + Match_r(M)] \},$$
(14)

where $Match_f(M)$ refers to $Match_{favorite}(M)$, and $Match_r(M)$ refers to $Match_{refused}(M)$.

Conclusion

This document introduces an adaptive email agent that prioritizes and classifies mails by combining exception handling. A text classification algorithm **SEC** and a novel prioritization approach using a time-decayed method are described. To improve the prioritization performance, handling more than three priority classes and tuning the periods of decay call for further investigation.

References

[1] W. W. Cohen, Learning Trees and Rules with Set-valued Features. In *Proceedings of The Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, pp. 709-716, 1996.

[2] Y. Deng, *A Personalized E-mail Agent*. Master¡¦s thesis, Department of Computer Science and Information Engineering, National Taiwan University, June 1998.

[3] T. Payne and P. Edwards, Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface. *Applied Artificial Intelligence*, 11, pp. 1-32, 1997.

[4] J. Rennie, ifile: An Application of Machine Learning to Mail Filtering. http://www.cs.cmu.edu/~jr6b/papers/ifile98.ps.gz

[5] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, 1989.

[6] G. Salton, A. Wong, C.S. Yang, Vector Space Model for Automatic Indexing. In K.S. Jones and P. Willett, editors, *Readings in Information Retrieval*, pp. 273-280, San Francisco, Morgan Kaufmann, 1997.

[7] B. Sheth, *A Learning Approach to Personalized Information Filtering.* Master¡¦s thesis, Department of Electrical Engineering and Computer Science, MIT, 1994.