Knowledge-Based Systems 71 (2014) 227–237

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Enhancing the experience of users regarding the email classification task using labels

Marcelo G. Armentano *, Analía A. Amandi

ISISTAN Research Institute, UNICEN-CONICET, Campus Universitario, Paraje Arroyo Seco, Tandil 7000, Argentina

ARTICLE INFO

Article history:
Received 4 February 2013
Received in revised form 18 July 2014
Accepted 5 August 2014
Available online 16 September 2014

Keywords:
Email classification
Recommender systems
Human–computer interaction
Personalization
User experience

ABSTRACT

Email is an indispensable tool for communication and users might have to deal with large volumes of information which they cannot always operate efficiently. For these users, the organization of emails is a tedious task. The use of automatic filters is not always possible or effective, because of difficulties regarding how to create a specific rule or because their use is impractical in some situations. In this article, we present an approach to enhance a webmail client with an interface agent that helps the user to label incoming email based on the knowledge of the user's preferences. We not only considered the label that can be applied to different emails but also how to better interact with the user to provide him/her with assistance in the labeling procedure. We performed a set of experiments using Google's webmail system, Gmail, obtaining a good rate of acceptance of the agent interactions.

1. Introduction

The electronic mail is nowadays one of the most used and effective communication mean and an indispensable working tool in many companies. The type of information that users received by email varies from user to user but also each user receives information from different projects, activities, interests, social networks, games, advertisements, etc. When the emails received by a user grows in number and diversity, grouping them becomes a necessary task to facilitate the order and the reading of the information received. The task of manually classifying incoming emails takes a considerable amount of time to users. Email clients commonly offer different tools to facilitate the management of incoming messages, for example, grouping related messages into *threads*, following the idea that there exist a conversation between a message and its successive replies. Another example is the use of folders or labels to manually classify incoming emails. In this latter case, email clients often provide the possibility of creating user-defined *filters* to apply a certain label to a message or to move it to a determined folder according to certain preset rules. However, filters are not always effective or possible to apply, either because the user does not have enough knowledge about how to create and update them or because there are too many filters to create and its use is impractical.

The exposed above suggest the utility of having an email client with the ability to personalize the task of incoming emails classification. Personalization aims to achieve the user's satisfaction by recognizing his/her preferences and needs. In contrast to *customization*, in which the user itself adapts the application to his/her specific needs, *personalization* refers to automatically adapting an application to the specific needs of a user, using the knowledge obtained by analyzing the user behavior and the data generated by him/her. The personalization process is initiated and conducted by the system, that continuously monitors the user behavior to automatically adapt itself. This adaptation is done without the need of the user to control how the system adjusts its behavior.

There are several mechanisms aiming to achieve the personalization of a system, particularly we are interested in the use of interface agents (also know as personal assistants). Interface agents are computer systems designed to provide personalized assistance to users who perform tasks using other software applications. An interface agent has the ability to learn the interests, preferences, priorities, objectives and needs of a user to provide proactive and reactive support in order to increase their productivity. They also serve as intermediaries between the user and a software application; they offer advice in real time, automate repetitive tasks, and hide the complexity of the system. A commonly used metaphor to understand the paradigm of interface agents is to compare them to a human assistant who works with the user in the same environment [23]. They have also been used, as assistants in electronic commerce [25,22], virtual teachers [21,3,32,33], web search assistants [5,6], etc.

* Corresponding author.
E-mail addresses: marcelo.armentano@isistan.unicen.edu.ar (M.G. Armentano), analia.amandi@isistan.unicen.edu.ar (A.A. Amandi).

http://dx.doi.org/10.1016/j.knosys.2014.08.007
0950-7051/

When developing agents that assist users we should pay special attention to two key issues: how to better interact with each individual user, and how to provide the right kind of assistance at the right time [30]. It is natural to think that every user interacts in a personal way with his/her interface agent. That is, the action expected from the agent, the kind of errors tolerated, and the type of assistance required, vary from one user to another. For example, a given user may not want to be interrupted with notifications or suggestions. Another user may probably disapprove certain types of assistance, which means that he/she will never tolerate a certain behavior of the agent.

To fulfill the user's expectations, the agent has to observe and analyze the user reactions concerning the various assistance types and find out what assistance type they prefer in different situations. Once the agent has learned the type of assistance that the user needs, it must learn how to provide it, that is interrupting the user or not. Most users tolerate interruptions by the agent if the situation is relevant to them. Thus, the agent has to analyze the relevance of the situation before interrupting the user, probably depending on the task that the user is performing. It is also important to analyze the user's tolerance to errors that the agent can commit, providing mechanisms to enable the user to provide a simple explicit feedback about the behavior of the agent.

The consequences of not meeting the user's expectations are usually highly negative for an agent interface. When an agent makes mistakes, especially in early stages of the learning process, it determines the user's trust regarding the use of the agent. In many cases, the user may choose to completely ignore or disable the agent [20].

In this article we present *Glabel*, an approach to enhance a webmail client with an interface agent that helps the user to label incoming email based on the knowledge of the user's preferences. Besides the prediction of the label that the user might apply to each incoming email, we put special attention on how to better interact with the user to assist him/her in the tagging procedure. *Glabel* was created to make it easier for webmail users to enhance their experience regarding the email classification task using labels.

The rest of this paper is organized as follows. Section 2 presents some related work in the area of email classification. Section 3 describes the classic confidence-based approach traditionally used by interface agents, along with its disadvantages. Section 4 presents our approach to personalizing the emails classification task and Section 5 the experiments carried out to validate our approach. Finally, in Section 6 we present our conclusions.

2. Related work

The definition of rules or filters is the most common tool provided to the user to automatically classify incoming emails. In this approach, the user is responsible for the definition of a set of conditions that, when fulfilled by an incoming email, triggers some action. This action can be, for example, to apply a label or to delete the message. This kind of approach, implemented by most email clients, can be considered semi-autonomous because the user has to manually detect different situations and design the corresponding rules. However, the definition of rules is not appropriate for big volumes of emails that can involve different and probably overlapping concepts. Furthermore, this approach does not adapt to changes in the user's habits for email classification or in adapting to new situations for which the user has to design new rules.

Regarding autonomous applications, several approaches aim at grouping messages in subject-based folders starting from a set of incoming messages (unsupervised approaches). Automatic Mail Category Organizer [24], for example, clusters messages sharing similar features into different folders using clustering and pattern discovery techniques for mining structured and unstructured information. Cutting et al. [8] uses a complex intermixing of iterative partitional clustering and an agglomerative scheme. Agrawal et al. [2] produce only cluster digests for topic discovery, and perform a message partitioning on the basis of such digests using a Bayesian classifier.

On the other hand, several learning techniques has been used to the task of classifying emails, specially to detect spam messages. Among the supervised approaches, we can mention the use of rule-based systems [7], Support-Vector Machines [12,17,35], Bayesian networks [28,4,29,14], memory-based reasoning [31,9], decision trees [36], linear logistic regression [1], neural networks [37] and semantic analysis methods [26].

Other approaches to categorize email messages include collaborative filtering techniques [15,27], social network analysis [34] and search operator suggestions [11].

There are other aspects of personalization that has been considered in other domains different to email classification, for example the device with which users access a given system. It is very common that users access an online system from different devices, such as mobile phones, notebooks, or tablets. Devices used by mobile users are diverse and heterogeneous, with different screen sizes, memory, connection speed, and computational power. Kao-Li et al. [16] developed a new social tag-based method for the recommendation of multimedia items which considered the user location, audience, mobile device, and network condition. These context descriptors were used to develop a set of rules to re-rank the recommendation list derived from the user preferences.

Differently to previous approaches that concentrate their efforts on maximizing the accuracy on the label (or folder) prediction, we focused on considering how to better interact with the user when the assistant detects an opportunity to apply a label to an incoming email. Therefore, to consider whether an interaction of the agent is correct or not, not only the content of the label suggested has to be correct but also the action taken by the agent has to be the action expected by the user. In this direction, some algorithms have been proposed to decide which action an agent should execute next. Most of these algorithms are based on confidence values attached to different actions [23,18]. However, these works do not consider a user's interaction preferences, the possibility of providing different types of assistance, or the particularities of the situation at hand.

In the following sections, we describe the classic confidence-based approach traditionally used by interface agents, along with its disadvantages. Next, in Section 4 we present our approach.

3. Confidence-based approach

The confidence-based approach traditionally used by interface agents, is based on the confidence on an agent action. The confidence on an action indicates how sure the agent is about executing that action, and it is computed according to the agent's experience in assisting the user. For example, in [23] the agent computes the confidence on the prediction of an action to the current situation taking into account how many similar situations the agent has memorized, whether or not all the nearest neighbors of the situation recommend the same action, and how close or distant these nearest neighbors are.

In the confidence-based approach, interface agents have generally three possibilities when they want to assist a user: executing a task autonomously, suggesting the user what to do, and doing nothing. These agents use two threshold values to take decisions, which are established by the user to control the agent's behavior: *do-it* threshold and *tell-me* threshold. If the confidence value asso-