Semi-automatic response in a Mail Center

Romain Vinot[1] and François Yvon[2]

[1] Akio Solutions
174, Quai de Jemmapes - 75010 Paris
(e-mail: vinot@akio-solutions.com)
[2] ENST, École Nationale Supérieure des Télécommunications,
46, rue Barrault - 75634 Paris Cedex
(e-mail: francois.yvon@enst.fr)

Abstract. E-mails are a rapidly growing medium to establish communications between customers and companies. They are processed in a large centralised system termed Mail Center. One important feature of those systems is their capacity to suggest relevant answers to reply to an incoming e-mail. This paper describes a new approach, based on two separate document bases and text classification algorithms.

1 Introduction

E-mails are rapidly becoming a privileged medium in customer-companies relationships, calling for the need for specialised, centralised e-mail client softwares. These systems, termed *Mail Centers* by analogy with Call Centers, help companies to manage increasing volumes of electronic correspondence.

A key functionality of Mail Centers is the automation of answer generation. Pending improvements of text generation techniques, answer generation systems can take advantage of techniques used in Information Retrieval (IR) systems. Relevant responses to new e-mails are suggested by searching through a knowledge base consisting of answers written in the past for similar queries; human operators can then use a suggested reply unchanged, or re-edit it, or if no solution is available, compose a new one from scratch. Given appropriate human interfaces, such systems contribute to (Busemann et al (2000)):

- reduce the training efforts of operators
- increase their productivity: frequent questions are rapidly answered
- ensure inter-operator consistency

The approach taken in Akio[1] Mail Center® is to consider (and search in parallel) two separate knowledge sources: one contains a small set of carefully designed typical answers (identical to FAQs); the other one contains all the others, more ad-hoc, anwers. In both databases, answers are associated with their corresponding question. Operators can thus choose an existing response from two distinct lists, one from each repository.

[1] http://www.akio-solutions.com

1. j ai mis ce dimanche15/10/00 vingt anonces avec photo dans le theme collection insigne et médailles mes aucune d elles n apparais merci de remédié a ce petit probléme. cordialement
2. je n'arrive pas à saisir mon annonce.

Table 1. Examples of incoming e-mails

This paper describes more specifically the module designed for matching incoming e-mails with existing *typical answers*, and compares various possible approaches to this task. Section 2 describes important features of the input data, clearly one of the main specificity of the task. Section 3 presents the architecture of the system. Experiments and results are described in section 4, and we conclude in section 5 with various perspectives offered by this work.

2 Description of corpus

Our original e-mail database contains 5 560 e-mails (and their respective reply) accumulated during 3 consecutive months in a Mail Center. Both questions and answers are written in French. The "external" knowledge base contains 42 typical answers, *as they are available at the end of the three months.*

An important caracteristic of e-mails is their poor compliance to linguistic standards, as evidenced by the number of grammatical or spelling errors. This is illustrated in the two examples displayed in table 1: *anonce* should have two *n*s, many accents are missing, conjugation error in *remédié*, space forgotten between *dimanche* and *15/10/00*... Furthermore, the questions are often not self-explanatory and operators need external knowledge to be able to answer. The average e-mail length is 90 words (but a few messages are much longer, reaching more than a thousand words). Overall, the vocabulary contains a grand total of 120 000 words, of which almost 50 % appears only once.

As our experiments aimed at evaluating the typical answer retrieval system, a first task has been to match the replies found in our database with typical answers: due to possible rewritings of typical anwers between the three month timeframe, this association was no longer directly available, and had to be infered from the data. To this end, two different methods based on fairly different principles were tested:

- a Vector Space Model with TF-IDF weights (see section 3 for more details). This method simply compares word occurrence profiles in the typical answer and in the actual reply.
- an algorithm based on Longest Common Subsequence (Paterson et al (1994)) comparisons. This method is able to cope with sequentiality information but doesn't take into account the relative importance of words.

Both methods work well, even if they cannot accomodate some complex cases, where multiple questions appear in the same mail. After completion of this step, we were able to match 2 404 actual replies with typical answers. Amongst these, the 5 more frequent typical answers account for 1 037 replies (43.2 %), while 11 answers are used less than 10 times and 21 less than 30 times. The remaining lot of e-mails was deemed to contain *ad-hoc* answers.

3 Description of the system

3.1 Text representation

Because of their poor compliance to linguistic standards, it is difficult to automatically annotate e-mails using sophisticated NLP tools, and derive complex representations. Consequently, we have chosen a representation based only on lexical occurrence profiles, similar to the "bag-of-words" approach taken in the *'Vector Space Model'*(Salton (1971)). Both e-mails and typical answers are encoded as a p-dimensional vector $(t_1, \ldots, t_k, \ldots, t_p)$, where p is the number of terms[2]in the entire document base, and t_k represents the weight (or importance) of the term k in the document. Preliminary experiments with the stemming algorithm available in the Muscat project[3] were also conducted, without any significant improvement, confirming our intuition that linguistic processing tools were of little help for our purposes. We used *TF-IDF* weights, computed for a document i and a term k as:

$$t_k = tfidf(i,k) = \ln(1 + \mathrm{Occ}(i,k)) * \ln(\frac{N}{n(k)}) \quad (1)$$

$\mathrm{Occ}(i,k)$ represents the number of occurrences of term k in document i, N the number of documents in the database and $n(k)$ the number of documents containing at least one occurrence of term k.

3.2 The retrieval of typical answers

We have to date mainly experimented with the typical answer base, by comparing five different retrieval techniques. The first one is used only as a baseline and consists in always proposing the five most frequent answers. The second one is a basic IR system based on the TF-IDF weights, which will be called *basic IR* in the remainder. In this model, the similarity between a query q (body of an incoming mail) and a document i (body of typical answer) is the

[2] Terms here simply refers to "word forms", as they are extracted on the basis on a crude tokenization process. As is usual, a stopword list filters out frequently occurring, non-informative words such as articles or pronouns.

[3] http://open.muscat.com

based on the cosine measure (i.e. the dot product of the *normalised* vectors):

$$Sim(q,i) = \frac{\sum_{j=0}^{p} tfidf(i,j) * tfidf(q,j)}{\sqrt{\sum_{j=0}^{p} tfidf(i,j)^2}\sqrt{\sum_{j=0}^{p} tfidf(q,j)^2}} \quad (2)$$

The third system tries to take advantage of the additional information avalaible under the form of the previous queries for which a typical answer has already been supplied. New incoming e-mail are now matched against the typical answers *and all the corresponding previous queries*. If one (or more) of the five best relevant documents is an e-mail, we present to the agent the typical answer used in the reply. We call this method *Advanced IR*.

The last conceptual step is to represent the situation as a classification problem, where every incoming e-mail is viewed as an instance of a conceptual class of problems. Each problem class is identified with a typical answer. Given labelled examples, that is, queries associated with the corresponding typical answer, we can use standard machine learning algorithms to learn e-mails classifiers. We have experimented with two such classifiers, known to be well suited for text classification tasks (Yang (1999)): the Rocchio algorithm (Rocchio (1971)) and the SVM (Support Vector Machine) algorithm (Vapnik (1995)). In this setting, we have considered one supplementary instance for each class: the answer text itself. As explained above, this allows for achieving reasonable performance even when very few queries are available.

The Rocchio algorithm is based on the notion of a *prototype*: it assumes that the centroid of each class is a suitable representation of its content. For each class, a protypical term vector is thus computed by summing the weight of each document in this class. The implementation we use computes the weight of term j for class C_k as follows:

$$weight(C_k,j) = max(0, \frac{\alpha}{|Ck|}\sum_{i \in C_k} w(i,j) - \frac{\beta}{|N-Ck|}\sum_{i \notin C_k} w(i,j)) \quad (3)$$

where $|C_k|$ denotes the number of documents in class C_k, $|N - C_k|$ the number of documents not in class C_k and $w(i,j)$ the normalised version of $tfidf(i,j)$. α and β are parameters of the method, whose values have been fixed in our experiments as: $\alpha = 1$ and $\beta = 0$. New incoming e-mails are classified on the basis on their (cosine) similarity with the class prototypes. In our setting, the 5 "closest" classes are proposed as possible answers.

SVMs are linear classifiers: learning consists in finding for each class the best linear separation (hyperplane) between positive and negative instances of the class. This hyperplane is computed by optimizing margin in a high dimensional representational space. Classification of new instances is then based on the distance between the new instance to the hyperplane, which closely related to the probability of belonging to the class. SVMs have been introduced in Vapnik (1995) and were first applied to text classification tasks

in Joachims (1998). In our setting, we train one classifier for each class and output the 5 more probable typical answers.

Experiments were performed using the Bow toolkit (McCallum (1996)), which provides facilities for tokenizing running texts, storing a 'Vector Space Model' of the documents, retrieving relevant documents, and using various learning algorithms.

4 Experiments

The five systems presented above were evaluated using the 10-fold cross-validation test protocol: the database is split in 10 parts, one is taken as test set, the others as training set and 10 runs are performed, one for each test set. Performance were evaluated on the basis of the measure proposed in the Question-Answering Track of TREC 8 (Voorhees and Tice (1999)), which redefines the classical notions of precision and recall for tasks where one single document is to be retrieved:

$$recall = \begin{cases} 1 \text{ if the good answer is amongst the 5 best ones} \\ 0 \text{ otherwise} \end{cases} \quad (4)$$

$$precision = \begin{cases} \frac{1}{\text{rank of the good answer}} \text{ if the good answer is found} \\ \text{not defined otherwise} \end{cases} \quad (5)$$

Recall measures the proportion of e-mails that were correctly answered, and precision measures the average rank of the answer. Results are presented in Table 2, revealing that both learning approaches clearly outperform the basic IR setting. As the results of the advanced IR suggest, this is mainly due to the use of past e-mails, which actually contains relevant information.

Our first experiments use a static design: training databases are fixed once for all and do not change over time. This is quite unrealistic, since in the real application, e-mails are received in a continuous process. In this context, an important practical issue concerns the reliability of the anwers proposed by the system in the first stages of the database constitution, when (statistical) predictions are in fact based on very little evidence. The curve on figure 1 displays the evolution of recall as a function of the size of the learning set, for the Rocchio classifier, demonstrating that using actual anwers as positive instances allows one to effectively bootstrap the system in its early stages.

5 Conclusions and Future work

We have presented a new architecture for improving answer production in a Mail Center, which combines two different document bases and retrieval methods. We have reported experiments on the typical answer database which show that good recall performance can be obtained using machine learning

	Recall	Precision
baseline	43.2 %	0.52
Basic IR	46.4 %	0.56
Advanced IR	72.5 %	0.65
Rocchio	75.4 %	0.71
SVM	70.6 %	0.78

Table 2. Results

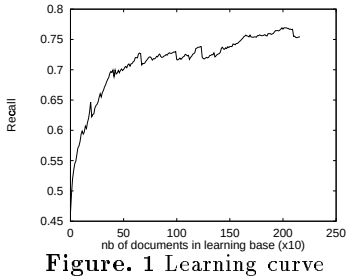

Figure. 1 Learning curve

techniques: 75% of "typical" queries could be answered in one single click. Typical queries though, in our database, only account for a half of incoming e-mails, and our current work focuses on the other kind of queries. Issues under investigation include:

- how to retrieve more efficiently non-typical answers, using weak domain information available in the typical ones;
- how to handle evolution in the Mail Centers: an accumulation of rare queries can well suggest the inclusion of new "typical answers"; conversely, typical answer can sometimes become useless: in both cases, tools for evolutively managing the knowledge base are needed.

References

S. Busemann, S. Schmeier and R.G. Arens (2000): Message Classification in the Call Center. In: *Proc. 6th Conference on Applied Natural Language Processing*, pp 159–165, Seattle, WA.

T. Joachims (1998): Text categorization with Support Vector Machines: Learning with many relevant features. In *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, pp. 137–142.

A.K. McCallum (1996): Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/~mccallum/bow.

M. Paterson and V. Danc'ik (1994): Longest common subsequences. In *19th International Symposium Mathematical Foundations of Computer Science*, pp 127–142. Lecture Notes in Computer Science 841, Springer-Verlag, 1994.

J. Rocchio (1971): Relevance feedback in Information Retrieval. In: *The SMART Retrieval System: Experiments in Automatic Document Processing* (Salton), chapter 14, 313-323, Prentice-Hall.

G. Salton (1971): The SMART Retrieval System: Experiments in Automatic Document Processing. Prentice-Hall, New Jersey.

Vladimir N. Vapnik (1995): The Nature of Statistical Learning Theory. Springer.

E. Voorhees and D. Tice (1999): The TREC-8 Question Answering Track Evaluation. In: *Proc. 8th Text REtrieval Conference*, 83-106.