# PEA – a Personal Email Assistant with Evolutionary Adaptation

WERNER WINIWARTER
Institute of Applied Computer Science and Information Systems,
University of Vienna, Austria

**Abstract**

In this paper we present PEA, a Personal Email Assistant, which filters incoming emails and ranks them according to their relevance. We provide tools for the acquisition of individual user models, which may consist of several profiles to map various interest domains of the user. In order to respond promptly to the shifts of interests of a user, we apply evolutionary algorithms to support an adaptive environment that constantly adjusts the user model to improve the quality of relevance assessment. As second adaptive component we make use of a monitoring module that records all activities of the user. By means of a classifier system we model the behavior of the user to predict future actions, which results first in suggestions to the user and later in automatically performed tasks. Additional features of the system include the segmentation of lengthy emails, efficient treatment of duplicate or new versions of messages, cross-language filtering, and the extraction of relevant information by using templates learned from examples.

## 1 Introduction

Users of electronic mail, the World Wide Web, or other on-line information systems are more and more confronted with the problem of selecting out of the huge information space just the information that is relevant to them. To solve this "information overload problem" [13] information filtering systems emerged within the last few years, with the aim of supporting the user with this difficult task [8].

However, many existing information filtering systems offer only static behavior, i.e. they cannot adjust to changes of the user's interests with time. Besides this, they often miss the required degree of personalization. In order to deal with the complex demands of this type of application, a flexible and adaptive architecture is needed to take the specific needs of a certain user into account. In this context the field of user modeling has developed many relevant answers and solutions to this problem within the last few years [1, 25].

A *cognitive user model* has to represent the cognitive style and personality factors of a user, his goals, plans, capabilities, and preferences. This is not an easy task because the users

- have difficulties in specifying their interests,
- change their interests in the course of time, and
- articulate subjective ratings representing their mental state.

We have concentrated our research on this topic first on a rather simple but important task, namely the filtering of email messages. Emails more and more replace phone and fax as internal communication medium in companies and organizations, a development which is promoted by the fast expansion of intranets. If you add to this the rapidly growing number of mailing lists, the user ends up with a large part of his working day that he spends with reading emails. Even the lucky user whose number of daily emails is still at a manageable rate is often confronted with the following annoying situations:

- the need to delete the same type of irrelevant emails again and again, e.g. printer errors, advertising material, or messages from unsubscribing people,
- the need to find important messages in a short time by the process of browsing through the email list and selecting messages for reading by trial-and-error based on the sparse information contained in the header,
- the need to verify often updated messages if they are simple duplicates or new versions containing relevant additional information, e.g. calls for participation,
- the need to browse through long digests of special interest groups to find some interesting message,
- the need to search through the same type of messages to gather relevant chunks of information, e.g. the deadline of a call for papers.

This quite desperate situation gets even worse if one considers environments where the user has to process emails in several languages the fluency of which is in most cases not the same as in his mother tongue. Therefore, the selection process of relevant emails becomes an even more time-consuming task.

Finally, a last example is the famous "the day after" the nice conference trip or any other business travel, which leaves you buried under a flood of several hundred emails. Even those ambitious people who fight to avoid this breakdown by spending their travel time in connecting via remote access to their computer at home get out of the frying-pan into the fire. This is because they have to deal with all the above-mentioned problems in slow motion, which is caused by the low transferring rates and the high network traffic.

To sum up the above-mentioned statements we felt the urgent need to address all these problems by developing PEA, a *Personal Email Assistant*. It assists the user in dealing more effectively with his daily load of emails so that valuable working time is saved for more productive and creative tasks. For that purpose we provide tools for the user to build his personal user model. It may consist of several profiles so that various interest domains can be mapped to them. In order to deal with the important requirement to be able to respond promptly to shifts of the user's interests, we make use of evolutionary algorithms to move the user model constantly closer to the current information need. In addition to this we also provide a monitoring module the task of which is to record all activities performed by the user. We model the behavior of the user by means of a classifier system so that future actions can be predicted on the basis of past experience. As result the system delivers suggestions to the user how to handle a specific email. With a growing number of affirmative reactions by the user, the system gains confidence in its decisions so that it is able to perform its tasks automatically.

After a short survey of related work we first give a brief overview of PEA before we provide a more detailed description of the individual components of the system. Finally, we address the problem of evaluation before we conclude the paper and give some prospects to future work.

## 2 Related Work

There already exist several quite successful information filtering systems, e.g. Information Lens System [29], EDS Template Filler [44], Iscreen [37], or SCISOR [24]. First prototypes of *adaptive information filtering systems* were developed at the University of Colorado [45], MIT [28], and University College Cork [36]; for other recent work on combining the fields of user modeling and information filtering see [30] or [34]. Regarding the applied techniques many different approaches have already been tried with quite mixed results; besides evolutionary algorithms [51] this also includes neural networks [31], methods from machine learning [45], and agent technology [35].

The subfield of *collaborative filtering* (often also called social filtering) represents another extension of information filtering systems in which the judgment of other people (or of their agents) is taken into account for determining the relevance of a document [18]. The most prominent recent system here is GroupLens [39], which is a Usenet Netnews extension that tries to find people with similar opinions based on the comparison of past rankings in order to reuse article ratings. Examples of similar approaches are Tapestry [17], MAXIMS [27], and WebFilter [14].

Finally, one important distinction has to be drawn between information filtering and *information retrieval*, a closely related research field from which information filtering has adapted many techniques. However, information filtering can be distinguished from the former by several important aspects, e.g. user behavior, user aims, representation of user interests, or system environment.

## 3 Overview of PEA

*Incoming emails* are first examined only according to their header information (see system architecture in Fig. 1). The *header analysis* works on the basis of the *pre-filter definition*, which specifies trigger words to assign relevance ratings to emails without any further processing. All emails that cannot be categorized based on the header information alone are in the next step analyzed whether they contain several distinct messages. This step is called *document segmentation*. We divide the emails into the individual parts according to the guidelines provided by the *segmentation definition*.

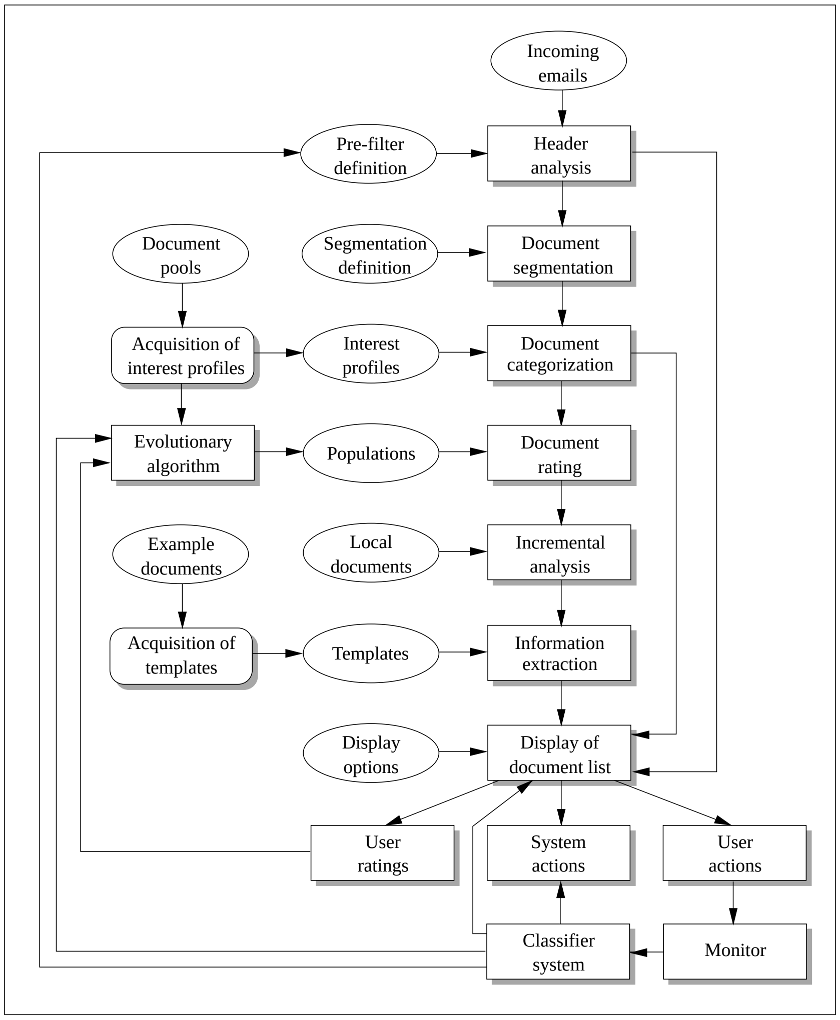


Figure 1: System architecture of PEA

The assignment to the correct interest domain is performed by the *document categorization*. It compares the content of the email with the *interest profiles* to select the one that reflects the content of the message best. If the document does not belong to any of the existing interest domains, no further processing occurs. The decision is then left to the user to either add the email to a document pool for creating a new interest domain or to adjust the pre-filter definition.

The actual *document rating* for successfully categorized emails is a result of taking the average relevance rating of the current *population* of the *evolutionary algorithm* for the interest domain. An additional feature represents the *incremental analysis* of emails by comparing them with a set of *local documents* stored automatically by the system for each interest domain. With this, we are able to eliminate duplicate emails and to determine the new information contained in updated versions of messages. Finally, we also support *information extraction* of relevant chunks of data, which are identified by means of user-defined *templates*.

The final result of this elaborate analysis is the *display of a document list*, which consists of ranked messages associated with system ratings. It can be customized to the specific needs of the user by defining *display options*, e.g. separate display of individual interest domains, restriction of the display to a certain relevance level, etc.

The displayed list provokes *user actions*, which are observed by the *monitor*. The monitoring component serves as input to a *classifier system* that predicts future user actions based on past recorded observations. By constantly verifying the correctness of its predictions, the classifier system gains confidence, which results first in the display of suggestions to the user and later in automatically performed *system actions*. Furthermore, detected inconsistencies lead to corresponding modifications of the pre-filter definition. The constant adaptation of the evolutionary algorithm is triggered either actively by explicit *user ratings* or passively by the classifier system.

In addition to the above-mentioned components for the processing of incoming emails, Fig. 1 also shows two acquisition modules. The *acquisition of interest profiles* is a prerequisite for the correct categorization and rating of documents. On the basis of a representative *document pool* for each interest domain provided by the user, we create a ranked list of descriptors that can be edited freely. The result of the acquisition process also serves as input to the evolutionary algorithm to create the initial populations for evolutionary adaptation. To enable cross-language filtering, the interest profiles can be extended to several languages. This definition process is facilitated in that we automatically present candidates of translated descriptors by utilizing machine-readable dictionaries. Finally, also the *acquisition of templates* for information extraction is supported based on *example documents* provided by the user.

# 4 System Description

## 4.1 Acquisition of Interest Profiles

Before the process of filtering incoming emails can be initiated, the user first has to define profiles for his interest domains. For that purpose we have developed a user-friendly acquisition module that constructs a separate interest profile for each of the user's several interest domains. It operates on the basis of a representative pool of documents collected by the user for each interest domain. This gathering of documents is performed by the display component (see Sect. 4.4) in which the user selects documents he has previously read and found useful. It is important that the pool is large enough to be representative for the interest domain. Therefore, we give the user a warning message if the size of the pool is smaller than a certain threshold (at the moment 50 documents). This is to make the user aware of the fact that the produced results could lack the necessary quality due to the small number of analyzed documents.

The basic components of the acquisition module are displayed in Fig. 2. The document texts included in the document pool of an interest domain are first transformed by the *tokenizer* into a sequential word list. This process of tokenization, i.e. the segmentation into individual words, is not always a trivial task; especially in languages like Japanese with no spaces between the individual words [50].

As next step we eliminate all meaningless stop words from the sequential word list to produce a reduced word list, which is then converted to a word index. For this purpose we make use of a *lemmatizing module* that replaces exact string match for the comparison of two words with a more sophisticated morphological analysis [43]. This feature is of particular importance for highly inflective languages like German. Of course, the use of the lemmatizing module is language-dependent; we provide for an easy extension to new languages in that we strictly separate language-independent techniques from language-specific features. The lemmatizing module deals with the following morphological phenomena (see also [49]):

- inflections,
- conjugations,
- suffixes,
- vowel-gradation.

Furthermore, we tolerate two common types of spelling errors, i.e. the insertion and deletion of one wrong character. For that purpose we perform a string comparison that allows one missing (or one additional) character in the two strings. On the basis of the resulting word index we apply some simple statistical and natural language processing techniques to extract candidates for *phrases* in a second run [46, 12].

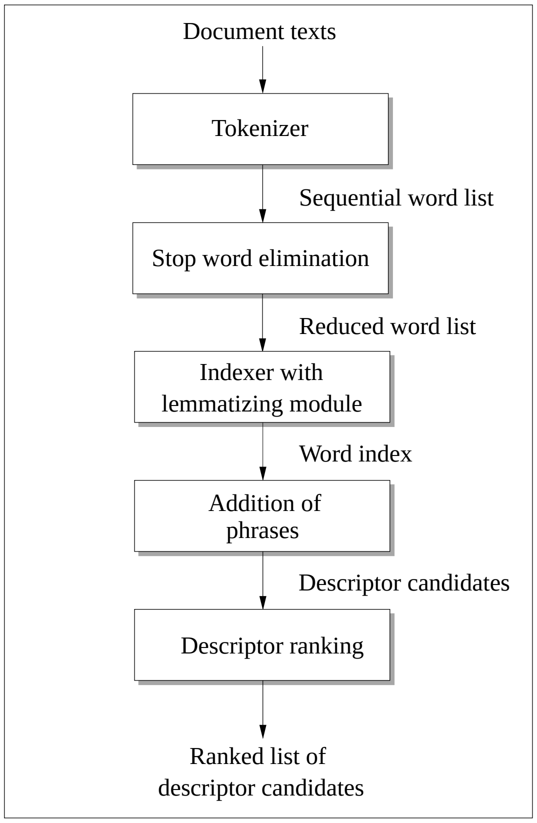


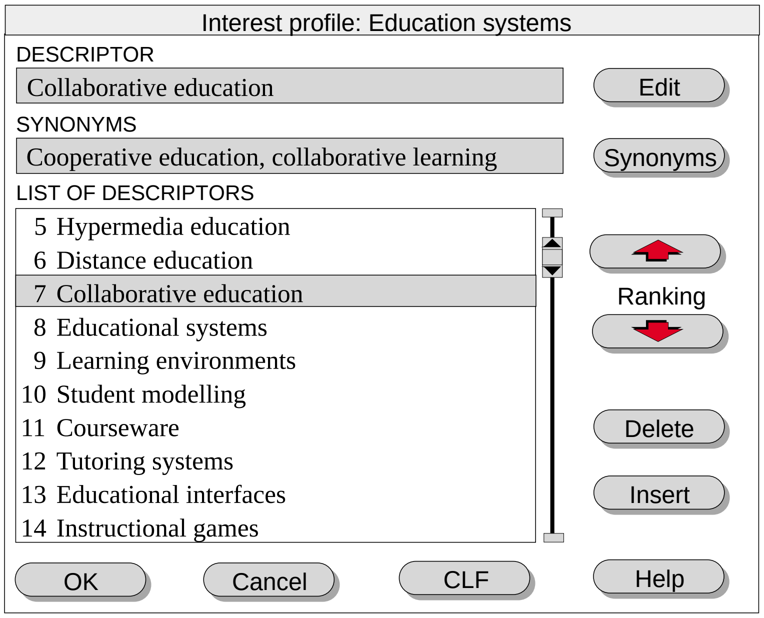
Figure 2: Acquisition of interest profiles

As final output we get a ranked list of *descriptor candidates*. For the ranking we have adapted the *weighted inverse document frequency* (WIDF, [47]) in that we compare the *term frequency* $\mathrm{TF}_{d,i}$ of descriptor $d$ in the document pool $i$ with the occurrences $\mathrm{TF}_{d,j}$ in the collection of document pools CDP for other interest domains $j$:

$$\mathrm{WIDF}_{d,i} = \frac{\mathrm{TF}_{d,i}}{\sum\limits_{j \in \mathrm{CDP}} \mathrm{TF}_{d,j}} \quad . \tag{1}$$

This measure determines the degree of selectivity of the individual descriptor candidates. The resulting ranked list is presented to the user who can freely edit it by modifying the ranking, deleting descriptors, adding additional descriptors, etc. (see Fig. 3 for an example). Another useful feature is that the user can define *synonyms* for descriptors either by adding own terms or by selecting other descriptors from the list of candidates, which are then automatically removed and added to the list of synonyms. In the same way abbreviations and acronyms can also be treated correctly.

The user can modify the ranking by simply selecting a descriptor and promoting or demoting it by use of the “arrow buttons”. This feedback from the user in the form of changes to the ranking order is essential for the quality of the information filtering system. As the acquisition component uses the selectivity of descriptors for an interest domain as approximation for the importance of this subtopic, the corrections by the user improve the representation of the interest profile. Therefore, the user is instructed to make adjustments to the descriptors to rank those subtopics higher in which he is interested more. With this, the interest profile can then really be used for determining the importance of a document for the user.


Interest profile: Education systems

DESCRIPTOR: Collaborative education

SYNONYMS: Cooperative education, collaborative learning

LIST OF DESCRIPTORS

5 Hypermedia education
6 Distance education
7 Collaborative education
8 Educational systems
9 Learning environments
10 Student modelling
11 Courseware
12 Tutoring systems
13 Educational interfaces
14 Instructional games

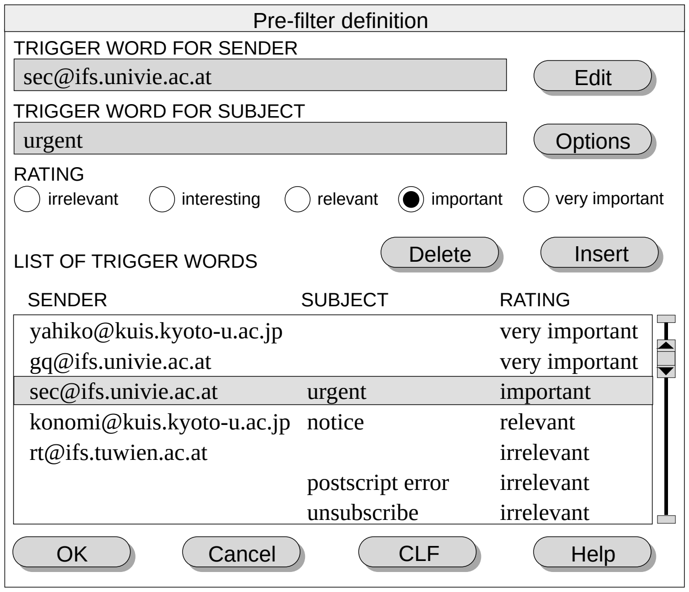
Figure 3: Example of interest profile

For the use of *cross-language filtering* (CLF), i.e. the filtering of emails in several languages, we allow the user to specify translations of the descriptors for the languages that PEA can handle (at the moment English, German, French, and Japanese). To facilitate this time-consuming process we make use of *machine-readable dictionaries* (e.g. EDICT [10] for Japanese-English) to automatically obtain candidates for the translated descriptors (see also [4]). Again, the user is free to edit the suggested translations.

## 4.2 Header Analysis

One of the main obstacles to the application of sophisticated information filtering techniques is the large amount of data which has to be analyzed in a short time. However, empirical tests have indicated [20] that almost about one third of incoming emails are deleted without reading because of header information contained in the sender or subject categories.

Therefore, we introduce a header analysis module that operates on the basis of a user-defined *pre-filter definition*. This definition specifies trigger words for the two header categories *sender* and *subject* and assigns *relevance ratings* to them. The names as well as the number of the categories for the relevance rating can be defined freely by the user (see also Sect. 4.4). It is also possible to specify trigger words on both categories for the same email, i.e. both trigger conditions must be satisfied to select the email. In this context more specific relevance ratings overwrite ratings for more general ones. Therefore, the user is equipped with a flexible tool to define his pre-filter as can be seen in the example displayed in Fig. 4. Finally, if several trigger conditions apply that cannot be resolved by the above-mentioned rule, we use the cautious approach of taking the maximum of the individual ratings.



Pre-filter definition

TRIGGER WORD FOR SENDER: sec@ifs.univie.ac.at

TRIGGER WORD FOR SUBJECT: urgent

RATING: irrelevant interesting relevant [✓ important] very important

LIST OF TRIGGER WORDS

| SENDER | SUBJECT | RATING |
|---|---|---|
| yahiko@kuis.kyoto-u.ac.jp | | very important |
| gq@ifs.univie.ac.at | | very important |
| sec@ifs.univie.ac.at | urgent | important |
| konomi@kuis.kyoto-u.ac.jp | notice | relevant |
| rt@ifs.tuwien.ac.at | | irrelevant |
| | postscript error | irrelevant |
| | unsubscribe | irrelevant |

Figure 4: Example of pre-filter definition

If a trigger condition is satisfied in an incoming email, the corresponding rating is assigned to the email and it is transferred directly to the display component. Depending on the chosen display options it is thus easy to delete all irrelevant emails automatically from the display. Another important use of the pre-filter are administrative emails such as messages from superiors that always have to be classified as very important irrespective of their content.

With this simple but efficient technique the total performance of the system is increased significantly without losing any important information. Again, the trigger words may be defined in several languages for cross-language retrieval with the assistance of machine-readable dictionaries.

## 4.3 Document Segmentation and Categorization

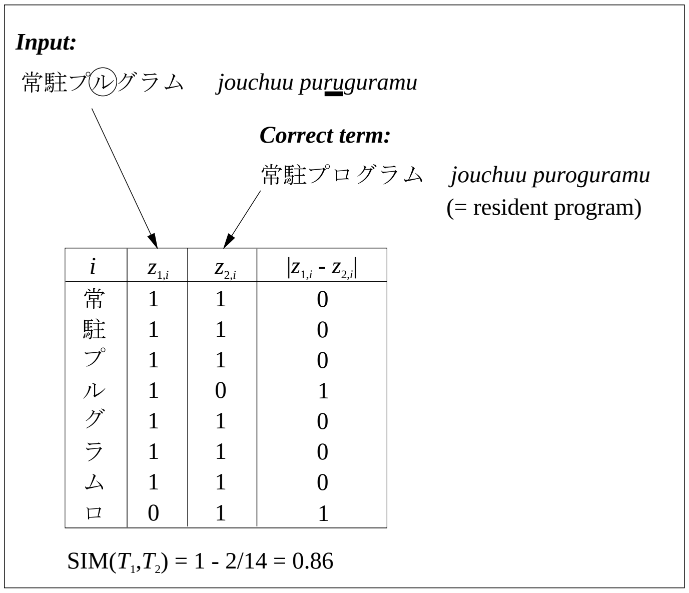
Emails often consist of several more or less unrelated messages, which is in particular true for periodical digests or newsletters. Any filtering system that tries to classify or determine the relevance of a complete email without taking into account this heterogeneous nature risks severe inaccuracies. Therefore, we support a powerful *segmentation language* that searches for text patterns including wildcards and formatting characters to segment the email into several messages [42]. User-defined strings and wild cards can be applied by the user to create title information for display.

For the subsequent process of *document categorization* we first create a *word index* (see Sect. 4.1), which is then searched for the *descriptors* of the individual interest domains according to the detected language of the email. Concerning the important issue of misspelled descriptors in the document text we make use of a refined *spelling error correction module* to increase the performance of the index algorithm. It enables a more powerful spelling error correction than the string comparison method used during the acquisition phase (see Sect. 4.1) because we now have available a set of valid existing descriptors for comparison. We apply the following *similarity measure*:

$$\mathrm{SIM}(T_1, T_2) = 1 - \frac{\sum_{i=1}^{k} |z_{1,i} - z_{2,i}|}{\sum_{i=1}^{k} z_{1,i} + \sum_{i=1}^{k} z_{2,i}} \ . \tag{2}$$

Here, $z_{1,i}$ ($z_{2,i}$) signifies the number of occurrences of character $i$ in the first (second) term $T_1$ ($T_2$). The formula defines similarity as 1 minus the distance between the two terms based on the number of divergent characters. The measure has as range the interval $[0, 1]$, i.e. as usual the similarity equals 1 if the two terms are identical; it equals 0 if they have no character in common. One advantage of the similarity measure is that it can also be used easily for non-alphabetic languages such as Japanese (see Fig. 5 for an example). As the performance of the measure decreases for smaller word lengths, we use only the string comparison method for very short descriptors.

To select the correct interest domain we apply statistical similarity measures adopted from the vector space model of information retrieval. To be more specific, we transform the word index of the analyzed email to a *document vector* $D_{\mathrm{DOC}}$ with the components $D_{\mathrm{DOC},i}$:


Input:
常駐プルグラム jouchuu puruguramu
Correct term:
常駐プログラム jouchuu puroguramu
(= resident program)

| $i$ | $z_{1,i}$ | $z_{2,i}$ | $\lvert z_{1,i} - z_{2,i}\rvert$ |
|---|---|---|---|
| 常 | 1 | 1 | 0 |
| 駐 | 1 | 1 | 0 |
| プ | 1 | 1 | 0 |
| ル | 1 | 0 | 1 |
| グ | 1 | 1 | 0 |
| ラ | 1 | 1 | 0 |
| ム | 1 | 1 | 0 |
| ロ | 0 | 1 | 1 |

$\mathrm{SIM}(T_1,T_2) = 1 - 2/14 = 0.86$

Figure 5: Example of spelling error correction

$$D_{\mathrm{DOC},i} = \begin{cases} W_i & \text{if descriptor } i \in \mathrm{DOC} \\ 0 & \text{otherwise} \end{cases} , \tag{3}$$

where $W_i$ represents the *weight* of the descriptor $i$ derived from the ranking in the interest profile provided by the user during the acquisition process (see Sect. 4.1). The weights $W_i$ are calculated by applying the following transformation to the rank $\mathrm{RANK}_i$ of descriptor $i$ ($n$ indicates the number of descriptors in the interest profile):

$$W_i = \frac{n + 1 - \mathrm{RANK}_i}{n+1} \quad . \tag{4}$$

This normalization maps the ranks to values in the range $(0, 1)$. The document vectors are then compared with the corresponding vector representation of the interest profile IP by adapting the *similarity coefficient of Dice* [40], which was originally used only with *binary indexing*, i.e. $W_i = 1$ for all descriptors:

$$\mathrm{SIM}(\mathrm{DOC}, \mathrm{IP}) = \frac{2 \cdot \sum_{i=1}^{n} \sqrt{D_{\mathrm{DOC},i} \cdot D_{\mathrm{IP},i}}}{\sum_{i=1}^{n} D_{\mathrm{DOC},i} + \sum_{i=1}^{n} D_{\mathrm{IP},i}} \quad . \tag{5}$$

In this formula, the numerator calculates the sum of the weights for those descriptors which are identical for the two vectors. It is multiplied by 2 and divided by the term in the denominator to produce a normalized value range in the interval $[0, 1]$. It equals 0 if the two vectors have no components in common. The other extreme is that the two vectors are identical, which results in a value of 1. Figure 6 gives an example of the application of the formula for the interest domain in Fig. 3.

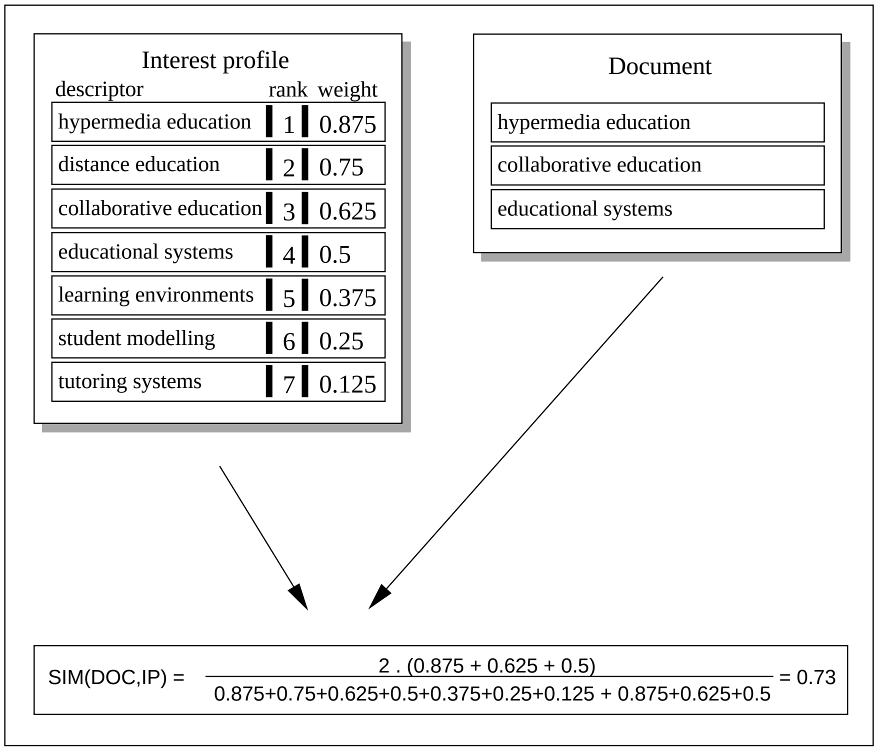


Figure 6: Example of similarity calculation

As final result we assign the document to the interest domain with the highest similarity. For other recent work on document categorization see [3], [7], or [33]. If an email cannot be categorized, i.e. there is no interest profile with sufficient similarity, it is directly transferred to the display component without further processing. As criterion for the minimum similarity a document needs to be considered as belonging to an interest domain, we use a threshold for the similarity coefficient in (5). In first empirical tests we have set this value to 0.03. The treatment of emails with insufficient similarity to any interest profile depends on the display options specified by the user who can either suppress their display or mark them for special treatment. This might be the manual assignment to one of the existing interest domains, the insertion into a document pool for the acquisition of a new interest domain, or a corresponding update of the pre-filter definition.

## 4.4 Evolutionary Adaptation and Document Rating

For the individual interest domains, the relevance of incoming documents is determined by the process of document rating. It relies on the output of an evolutionary algorithm that represents the current interest profile of the user.

Evolutionary algorithms provide methods for simulating biological evolution on a computer. Among others they include as subdisciplines genetic algorithms [22], evolution strategies [41], genetic programming [26], and artificial life [38]. All have in common that they are based on the biological principle of Darwin's "natural selection and survival of the fittest" in that they use computational models of evolutionary processes as key elements in the design and implementation of problem solving systems [16].

Evolutionary algorithms maintain a *population* of data structures (*chromosomes*) that evolve according to rules of selection and genetic operators such as recombination (crossover) or mutation. A chromosome consists of a string of task parameters (*genes*) that may be represented by a binary bitstring, an array of integers, etc. [32]. The particular values a certain gene can take are called its *alleles*. *Selection* implements the principle of the survival of the fittest in nature by evaluating the *fitness* of the individual chromosomes and determining the parent chromosomes for the creation of the next generation of *offspring*.

In PEA we create the initial population of chromosomes for each interest domain on the basis of random variations of the interest profile obtained from the corresponding acquisition component (see Sect. 4.1). This means that the descriptors correspond to the genes of the chromosomes and the individual weights of the descriptors to their alleles. As data structure for the internal representation of the genes we use arrays of real numbers.

In that the user corrects faulty relevance ratings for new incoming messages, we obtain important feedback that is used for activating the evolutionary algorithm after each user session. This feedback is provided by the user while displaying the document list by simply clicking on a rating category different from that suggested by the system (see also Fig. 8 for an example).

The algorithm is run separately for each interest domain for all new documents rated by the user. The *relevance rating* of the user $R_{\mathrm{U},i}$ for message $i$ then directly influences the fitness of the chromosomes for the corresponding interest domain. The *evaluation function* for the fitness of chromosome $x$ is an accuracy measure according to the following formula:

$$F_x = 1 - \frac{\sum_{i=1}^{n} |R_{x,i} - R_{\mathrm{U},i}|}{n \cdot (c-1)} \quad . \tag{6}$$

In this formula $|R_{x,i} - R_{\mathrm{U},i}|$ stands for the deviation of the $i$th rating of the chromosome $x$ for message $i$ from the $i$th rating of the user. A difference of one category counts as 1. We take the average over all messages $n$ and standardize the result by the number of rating categories $c$. Table 1 shows a simple example, the resulting fitness is calculated as:

$$F_x = 1 - \frac{\sum_{i=1}^{10} |R_{x,i} - R_{\mathrm{U},i}|}{10 \cdot (5-1)} = 1 - \frac{4}{40} = \frac{9}{10} = 0.9 \ . \tag{7}$$

Table 1: Example of rating differences

| Email no. | $R_{x,i}$ | $R_{\mathrm{U},i}$ | $R_{x,i} - R_{\mathrm{U},i}$ |
|---|---|---|---|
| 1 | important | very important | -1 |
| 2 | relevant | relevant | 0 |
| 3 | important | important | 0 |
| 4 | interesting | interesting | 0 |
| 5 | relevant | important | -1 |
| 6 | interesting | irrelevant | 1 |
| 7 | very important | very important | 0 |
| 8 | relevant | interesting | 1 |
| 9 | irrelevant | irrelevant | 0 |
| 10 | relevant | relevant | 0 |

The values for $R_{x,i}$ are calculated by applying again the similarity coefficient in (5). The resulting similarity represents the *score* $S_{x,i}$ for message $i$; it is transformed into a corresponding rating category $R_{x,i}$ according to a mapping function based on a uniform partition of the domain $[0, 1]$. Once again we would like to stress that the names and number of categories for the relevance rating can be freely defined by the user.

As procedure for selection we use *roulette wheel selection*. It assigns slots on a roulette wheel to the chromosomes of the population where the size of the individual slots is a function of the fitness of the individual chromosomes (see [6]). For that purpose we first sum up the fitness values of all chromosomes in the current population, which corresponds to the total area of the roulette wheel. Next, we generate a random number between 0 and 1 and multiply it by the sum of the fitness values. The resulting value represents the distance that the imaginary roulette ball travels before it falls into a slot. Finally, we sum up the fitness values of the chromosomes until we reach a chromosome that makes this partial sum greater or equal to the calculated distance. This chromosome is then selected for reproduction. The number of repetitions of the selection process equals the number of chromosomes to keep the size of the population constant.

New offspring are created by the application of the *crossover operator*. It involves two parent chromosomes that are combined to form a new child chromosome. We apply the most usual form of crossover, the *single-point crossover* operator. It selects a random position in the chromosome called *cut point*. All genes from the left of the cut point on the first parent are combined with the genes from the right of the cut point on the second parent to form the first offspring. The process is then repeated with opposite segments to create a second offspring.

To take the situation into account that the offspring may be of less fitness than the parents, the additional parameter of *probability of crossover* is introduced. Before we perform the crossover, we simulate the flipping of a coin that is biased to come up heads with this probability. Only if it does, we perform the crossover, otherwise we pass the parents to the next generation unchanged. Since this probability of crossover is a serious restriction to the advancement of the population, one usually uses values between 0.5 and 1.

Finally, the *mutation operator* is used for performing small alterations to new offspring. Unlike crossover, mutation is a unary operator, i.e. it only acts on one individual at a time. As genes are copied from a parent to a child, again a weighted coin is flipped. If it comes up heads, the value of the gene is altered by a small random amount. The weight for the coin is called *probability of mutation* and is usually below 0.1 to prevent the degeneration of the random search. Figure 7 summarizes the application of the crossover and mutation operator by providing a simple example of two chromosomes with 5 descriptors as genes.

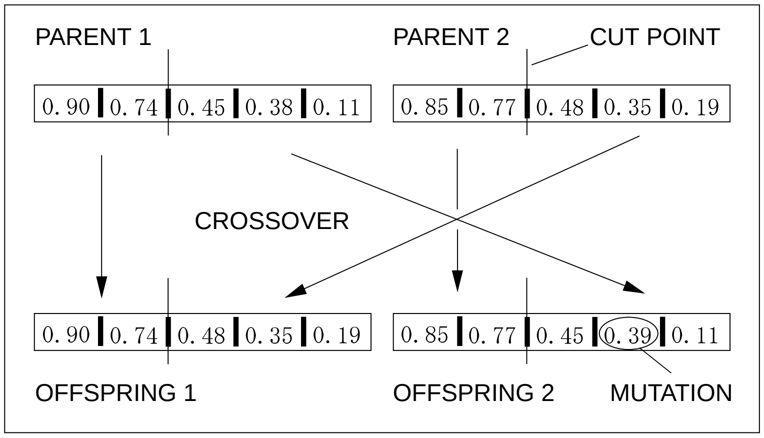


Figure 7: Example of genetic operators

In first empirical tests we have set the probability of crossover to 0.8 and the probability of mutation to 0.05. As size of the initial population we use 20 chromosomes and evolve them over 20 generations, which proved sufficient to converge to an acceptable solution. However, during later evaluation of our system (see Sect. 5) we will also experiment with different values for those parameters to see how this influences the performance of the evolutionary algorithm for this specific learning task, e.g. by plotting charts on the fitness of chromosomes as a function of the probability of crossover. The satisfactory behavior of the algorithm is certainly also caused by the fact that we do not perform a completely random generation of the values of genes in the initial population. Instead of this we use, as mentioned before, the weights of the interest profiles provided by the user so that we introduce a search bias which leads to an earlier convergence of the algorithm.

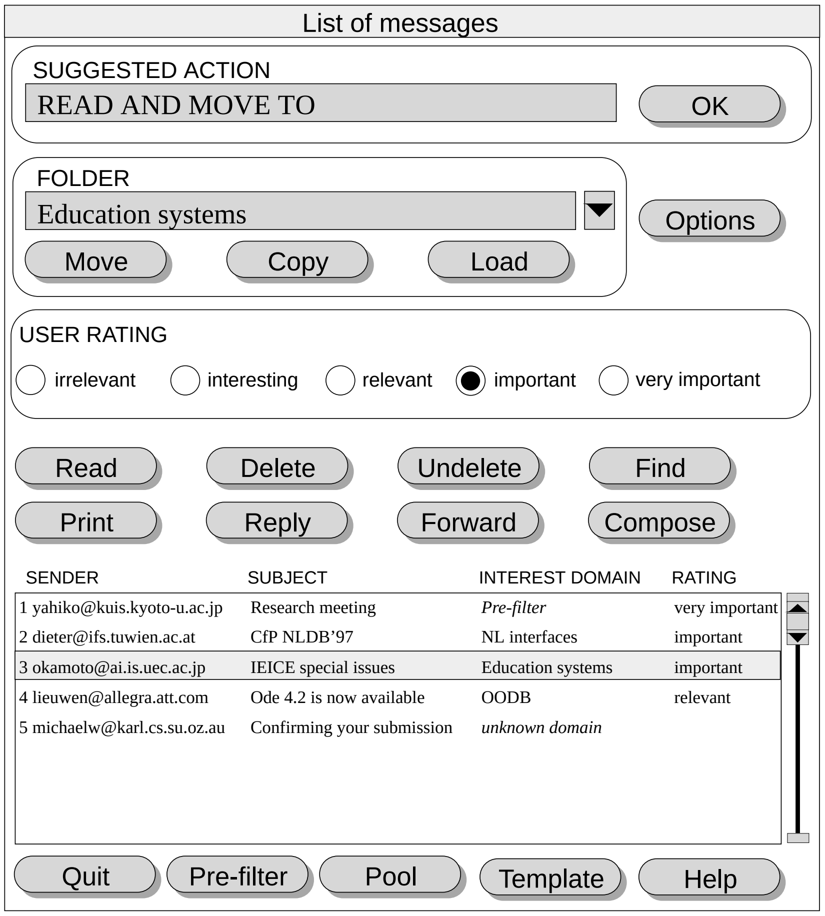
The whole evolutionary process controls the adaptive behavior of the information filtering system so that it is able to respond to changes in interests and adapt to such situations effectively. The system can also deal with stronger shifts of interests that introduce new descriptors in the document descriptions not covered by existing chromosomes. For that purpose we insert on the one hand additional genes into the chromosomes representing new strong descriptors of relevant messages; on the other hand we remove genes corresponding to weak descriptors that no longer reflect the focus of interest. In the same way the interest profiles for document categorization have to be updated by an incremental run of the acquisition component (see Sect. 4.1).

For the application of the results of the evolutionary algorithm to the rating of incoming documents, we determine the relevance of a document in that each chromosome in the current population for the interest profile scores the document. This scoring is performed according to the similarity coefficient in (5) in the same way as for the calculation of the fitness. Based on the scores $S_{x,\mathrm{DOC}}$ of the individual chromosomes $x$ in the population P the system calculates the *relevance score* $S_{\mathrm{S,DOC}}$ of the document DOC by taking the average of the scores of the $k$ chromosomes:

$$S_{\mathrm{S,DOC}} = \frac{1}{k} \cdot \sum_{x \in \mathrm{P}} S_{x,\mathrm{DOC}} \quad . \tag{8}$$

This internal relevance score is then mapped to a corresponding category of the displayed *relevance rating* $R_{\mathrm{S,DOC}}$ as described before. As final result we obtain ranked lists of documents for each interest domain, which represents the input to the display module. The display module can be customized to the individual preferences of the user by a rich set of display options. These options include the choice between a mixed or separate display of individual interest domains, sorting criteria, displayed header information, suppression or automatic deletion of certain relevance categories, and information about the treatment of emails with unknown domain.

Figure 8 shows an example of a message list with mixed display of the interest domains, automatic deletion of irrelevant emails, and the display of emails with unknown domain at the end of the list. On top of the window the user sees the suggested action for the selected email, which is computed by means of the classifier system (see Sect. 4.6). Therefore, in most of the cases the user can simply click on the "OK button" to acknowledge the suggestion of the system. However, the user also has the opportunity to correct faulty relevance ratings, which is essential for the evolutionary adaptation of the filtering system. Finally, the user can insert the document into a document pool to create a new interest profile or he can change to the window for the generation of an extraction template (see Sect. 4.5).



USER RATING: irrelevant interesting relevant [✓ important] very important

| SENDER | SUBJECT | INTEREST DOMAIN | RATING |
|---|---|---|---|
| 1 yahiko@kuis.kyoto-u.ac.jp | Research meeting | *Pre-filter* | very important |
| 2 dieter@ifs.tuwien.ac.at | CfP NLDB'97 | NL interfaces | important |
| 3 okamoto@ai.is.uec.ac.jp | IEICE special issues | Education systems | important |
| 4 lieuwen@allegra.att.com | Ode 4.2 is now available | OODB | relevant |
| 5 michaelw@karl.cs.su.oz.au | Confirming your submission | *unknown domain* | |

Figure 8: Example of message list

## 4.5 Incremental Analysis and Information Extraction

A very important feature of PEA is the *incremental analysis*. It saves the user from the annoying task of checking duplicate or new versions of emails whether they contain any additional relevant information. Since a duplicate of a relevant email would again be evaluated as relevant, there is no way for the other components of the filtering system to eliminate such messages. Therefore, we added an additional module, which compares new messages that were successfully assigned to an interest domain with previously received messages for that domain. For that purpose we have to maintain a set of *local documents* that are considered for comparison up to a certain user-defined past period. According to the preferences of the user this may be restricted to messages in the corresponding folder or it may also include deleted messages.

On the basis of this set of local documents we are now able to recognize identical documents and mark them as duplicates or delete them automatically. Another related problem are often updated messages such as calls for participation. Here the incremental analysis too can assist the user in that it identifies highly similar document texts and extracts the differences. The divergent information is then displayed on top of the message window in the same way as for information extraction (see below).

In many situations the user first wants to see only the relevant data of an email before he reads the complete message, a typical example of which are call for papers for conferences where the user might decide the relevance of a conference first on the basis of the conference title, date, deadline, and location. This observation is especially true for remote logins from abroad with slow access to the computer at the home office. Even after reading a message, an informative summary of the relevant information is very useful to make decisions on further actions.

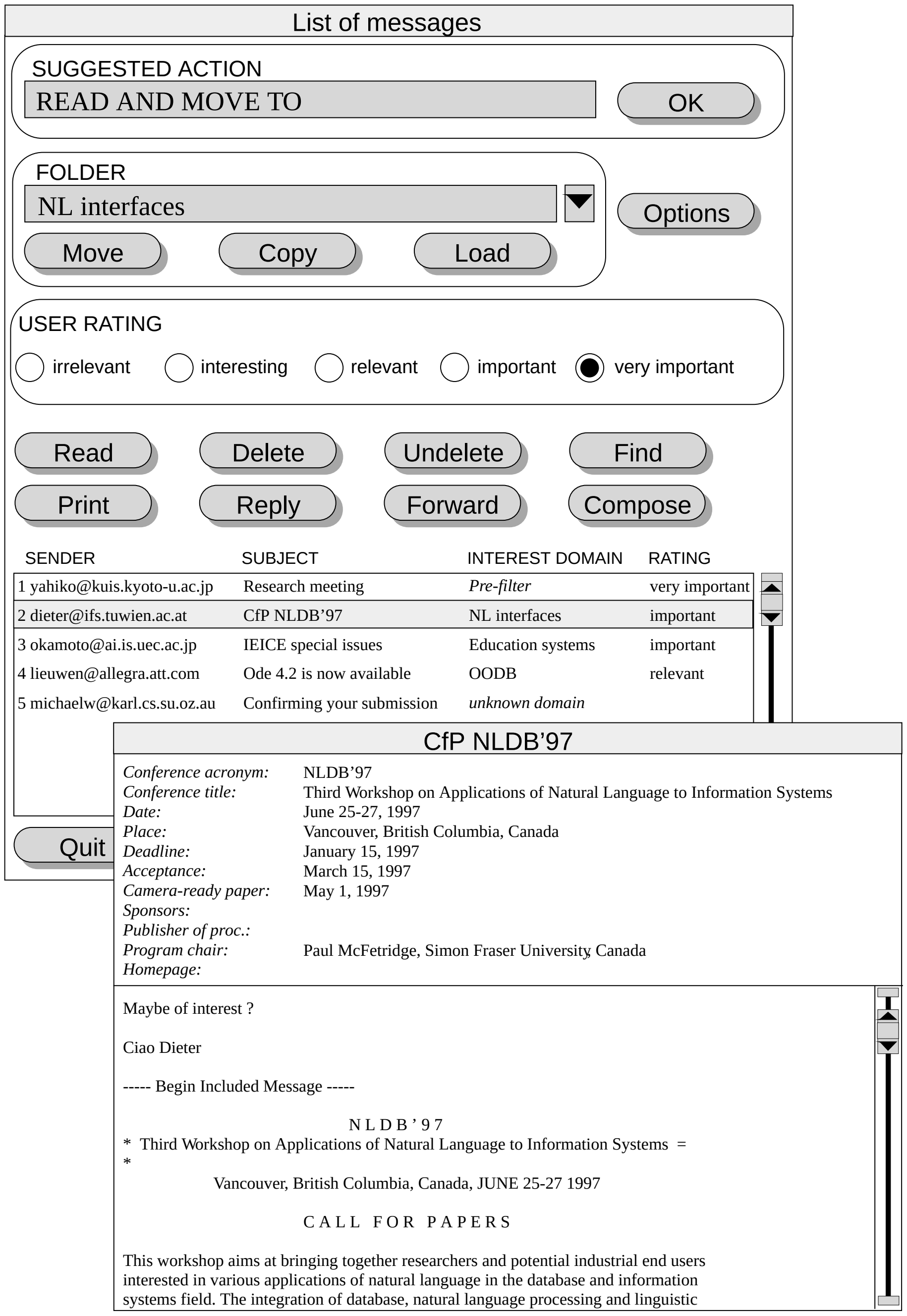
Here we apply techniques from *information extraction.* This approach represents an interesting compromise between in-depth text understanding of documents and standard information retrieval techniques in that it provides a more tractable and robust method. Its aim is to extract specific types of information from a document by effectively ignoring non-relevant text portions [11].

We make use of a *cascaded architecture* that narrows the scope of analysis by first retrieving text segments of special relevance, which are then further examined. The information of interest is extracted in two steps:

1. At *sentence level processing* we use a *pattern matcher* to identify the concepts represented by words and phrases in the text. First, the relevant contexts are triggered by key words and then the required information is filled into slots of a *concept template* that models the connotation of the concept. The activation of a template can be restricted by enabling conditions on certain linguistic constructs, e.g. passive, imperative, etc. The search for the information patterns to fill the slots requires an efficient pattern matcher such as Tomita's generalized LR parser that skips unnecessary words during parsing [5].

2. *Discourse processing* merges coreferential information scattered throughout the text by unifying the concerned templates. This also includes the difficult task of resolving references introduced by anaphora or ellipsis [48].

As result of the extraction process the user gets a separate display on top of the message that shows the information contained in the slots of the unified template (see Fig. 9 for an example).



USER RATING: irrelevant interesting relevant important [✓ very important]

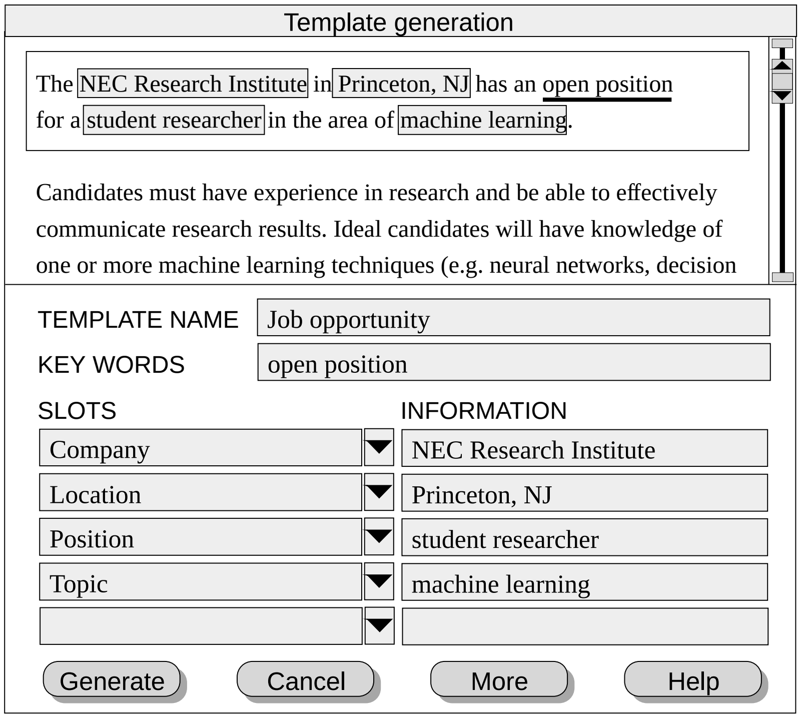
| SENDER | SUBJECT | INTEREST DOMAIN | RATING |
|---|---|---|---|
| 1 yahiko@kuis.kyoto-u.ac.jp | Research meeting | *Pre-filter* | very important |
| 2 dieter@ifs.tuwien.ac.at | CfP NLDB'97 | NL interfaces | important |
| 3 okamoto@ai.is.uec.ac.jp | IEICE special issues | Education systems | important |
| 4 lieuwen@allegra.att.com | Ode 4.2 is now available | OODB | relevant |
| 5 michaelw@karl.cs.su.oz.au | Confirming your submission | *unknown domain* | |

| | |
|---|---|
| *Conference acronym:* | NLDB'97 |
| *Conference title:* | Third Workshop on Applications of Natural Language to Information Systems |
| *Date:* | June 25-27, 1997 |
| *Place:* | Vancouver, British Columbia, Canada |
| *Deadline:* | January 15, 1997 |
| *Acceptance:* | March 15, 1997 |
| *Camera-ready paper:* | May 1, 1997 |
| *Sponsors:* | |
| *Publisher of proc.:* | |
| *Program chair:* | Paul McFetridge, Simon Fraser University, Canada |
| *Homepage:* | |

```
Maybe of interest ?

Ciao Dieter

----- Begin Included Message -----

                              N L D B ' 9 7
*  Third Workshop on Applications of Natural Language to Information Systems  =
*
            Vancouver, British Columbia, Canada, JUNE 25-27 1997

                        C A L L   F O R   P A P E R S

This workshop aims at bringing together researchers and potential industrial end users
interested in various applications of natural language in the database and information
systems field. The integration of database, natural language processing and linguistic
```

Figure 9: Example of information extraction

In contrast to information extraction systems that rely on the availability of large text corpora, we followed a different approach in that we learn extraction templates from examples provided by the user. For that purpose we apply a similar technique to the one used in LIEP [23]. Figure 10 shows an example of the generation of a template for job opportunities. The user reaches this window by clicking on the "Template button" in the message list. He first has to select the relevant context by marking the corresponding area in the message. As next step the user chooses a template name and provides key words by clicking on the corresponding words or phrases in the message window. To define the individual slots, the user can either select slot categories from the list of existing categories or create new ones. The information to be extracted is then identified in the same way as the key words, namely by simply marking the relevant position in the text.



Template generation

The NEC Research Institute in Princeton, NJ has an open position for a student researcher in the area of machine learning.

Candidates must have experience in research and be able to effectively communicate research results. Ideal candidates will have knowledge of one or more machine learning techniques (e.g. neural networks, decision

| | |
|---|---|
| TEMPLATE NAME | Job opportunity |
| KEY WORDS | open position |

| SLOTS | INFORMATION |
|---|---|
| Company | NEC Research Institute |
| Location | Princeton, NJ |
| Position | student researcher |
| Topic | machine learning |
| | |

Figure 10: Example of template generation

After the complete definition of the extraction example, the user starts the generation of the template by clicking on the "Generate button". The generation process first creates constituents for the slots and then searches for syntactic relationships which relate all of those constituents. These relationships can either be direct or indirect; for the second case we use a recursive search algorithm that attempts to find intermediate constituents.

Figure 11 displays the template created from the input in Fig. 10. The learned template is very detailed in that it requires specific properties and head words for intermediate constituents. Therefore, its application is very restricted, which leads to the need for a *generalization* of templates. This generalization is performed by means of later examples provided by the user. If a new extraction template is created, the system searches for similar existing templates that can be merged with the new template to create one generalized template. As a simple example consider a new template that only differs from the one in Fig. 11 in that it uses "at" as preposition for the location. The two templates can be merged by creating a *disjunctive value set* `dvs = (in, at)` and by rewriting the definition of the preposition as `preposition(PREP1, head(member(dvs))`.

```
Gen_template:
   noun-group(CNG, head(isa(company-name))),
   noun-group(LNG, head(isa(location-name))),
   noun-group(PNG, head(isa(position-name))),
   noun-group(TNG, head(isa(topic-name))),

   preposition(PREP1, head(in)),
   verb-group(VG, type(active), head(has)),
   noun-group(NG1, head(position)),
   preposition(PREP2, head(for)),
   preposition(PREP3, head(in)),
   noun-group(NG2, head(area)),
   preposition(PREP4, head(of)),

   subject(CNG, VG),
   post_nominal_prep(CNG, PREP1),
   prep_object(PREP1, LNG),
   object(VG, NG1),
   post_verbal_post_object_prep(VG, PREP2),
   prep_object(PREP2, PNG),
   post_verbal_post_object_prep(VG, PREP3),
   prep_object(PREP3, NG2),
   post_nominal_prep(NG2, PREP4),
   prep_object(PREP4, TNG)

==> job_opportunity(company(CNG), location(LNG),
        position(PNG), topic(TNG)).
```

Figure 11: Example of extraction template

## 4.6 Monitor and Classifier System

Given the diversity of the users of information filtering systems, the fact that they will not have the same problems or information needs, and that the level of expertise and interests of a user are likely to change in the course of time, it is essential that the user models are able to adapt to and support the requirements of individual users. Monitoring data collection techniques, think-aloud protocols, tape recording of interaction, interviews, and questionnaires are useful instruments to better understand the filtering process of an individual user

[2] so that the behavior of the user can be mapped correctly to the behavior of the filtering system.

Besides the active feedback from the user in the form of relevance ratings for emails we also gather passive feedback from the user. We have implemented this feature as *monitor component* that “watches” the behavior of the user, i.e. his reaction to incoming emails, e.g. deleting, forwarding, saving, replying, printing, etc. By monitoring these actions we try to measure how effective the recorded usage patterns predict future user behavior.

We apply for that purpose a *classifier system* [15, 9], i.e. a genetic-based machine learning system that combines syntactically simple rules called *classifiers*, parallel rule activation, rule rating, and conflict resolution. The classifier system defines *strategies* and *substrategies* that trigger the *action* part of the classifier if the *condition* part is satisfied. Figure 12 shows an example of a simple strategy for the processing of new emails.

```
PROCESS NEW EMAIL:
  (SS1) Process new email  AND Read:
    IF (Doc = unknown relevance) OR (Doc = relevant) THEN Read Doc
  (SS2) Process new email AND Move:
    IF (Doc = relevant) THEN Move Doc to folder Name
  (SS3) Process new email AND Delete:
    IF (Doc = not relevant) THEN Delete
  (SS4) Process new email AND Autoforward:
    IF (Doc = relevant) AND (User = absent)  AND (cc: <> New_email_addr)
      THEN Autoforward Doc to New_email_addr
```

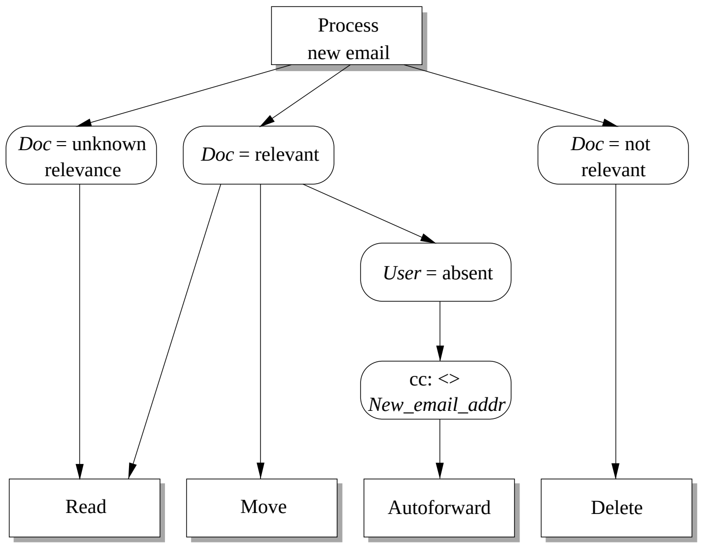


Figure 12: Example of strategy

In Fig. 13 we display the process model of the implemented classifier system. All actions by the user are transformed by the *detectors* into *messages* that represent the input to the classifier system. The messages are placed on a *message board* and *compared* with the condition part of the classifiers in the *classifier list*. If more than one classifier matches a given message, the *selector* selects a winning classifier by applying roulette wheel selection (see Sect. 4.4). This results in the posting of messages to the message board and the consequent generation of system actions by the *effectors*.

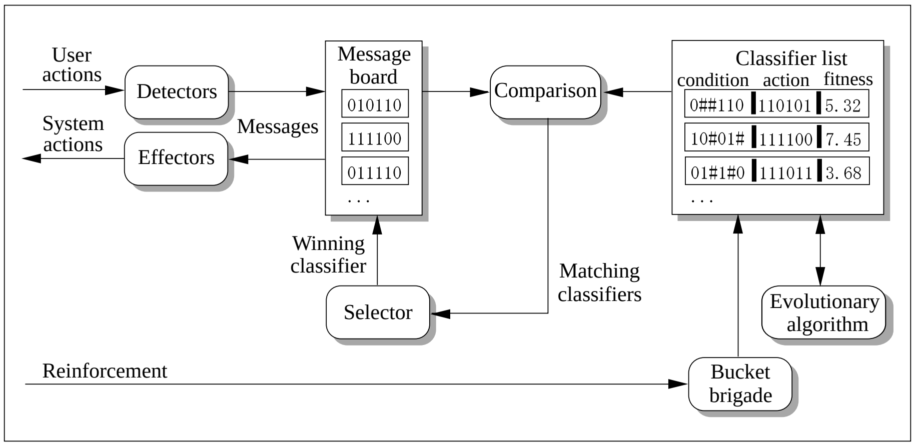


Figure 13: Classifier system

For the purpose of selection, a certain fitness is assigned to each classifier. At first, all classifiers start with uniform fitness. They acquire additional fitness by directly causing actions with positive *reinforcement* from the environment. A second way to gain fitness is to get paid by other classifiers. The applied *bucket brigade* algorithm requires that if a classifier is selected based on a message produced by another classifier, it has to pay a certain percentage of its fitness to that classifier (for more details see [21]). This ensures that not only those classifiers which actually produce positive reinforcement are strengthened but also the stage setting classifiers which allow other classifiers to win by posting appropriate messages to the message board.

The condition and action part of classifiers are represented internally as bitstrings. The action part has a special prefix (for example `11` as in Fig. 13) to distinguish it from incoming messages in the message board. As additional symbol we use `#` to indicate that we do not care about the value of the message at this point. This is useful because the classifier can cover multiple messages, which reduces the number of required classifiers. The binary encoding also facilitates the genetic adaptation of the classifiers. It is performed by applying an *evolutionary algorithm* with roulette wheel selecection, single-point crossover, and mutation (see Sect. 4.4).

The task of the classifier system is to learn new substrategies or to refine existing ones. By monitoring the actions of the user, the system derives predictions for future behavior, which it then compares with the actual outcome. At first the system gives only suggestions what to do with a new email. After perceiving several agreements, the system gains confidence in its decisions so that it can perform the proposed action automatically. This feature is implemented by a *confidence level* which is increased with each agreement until a threshold for performing automatic actions is reached. However, we still keep a log of all performed actions so that a user can check and undo past activities. Besides this, each disagreement also results in a decrease of the confidence level. Therefore, we can guarantee that only those actions which represent a consistent behavior of the user really are executed.

This cautious model is essential to assist the user in an optimal way but also to keep him in command so that he does not feel suspicious that the information filtering system “is doing things behind his back”; a problem that has to be addressed by all agent-based systems if they want to achieve the required user acceptance.

Another important use of the monitor is to verify the relevance ratings of the system in a passive manner. Whereas the presented evolutionary approach to filter adaptation (see Sect. 4.4) has the big disadvantage that it relies on the active participation of the user in the form of corrections to faulty relevance ratings, the monitor enables to detect inconsistent behavior automatically, e.g. deleting an important long message after reading the first page. The detected anomalies are used as input to the classifier system, which generates a corresponding action to correct the relevance rating of the email. This automatic correction of system ratings results then in the adaptation of the concerned population of chromosomes during the next run of the evolutionary algorithm for the interest domain.

Finally, we also deal with inconsistencies between the user model and the actual user actions concerning the header information of those messages which were directly transferred to the display component by the header analysis. An example of such an inconsistent behavior is the deletion of an email from the list without reading, which was before rated by the header analysis as very important. If such an action is repeated several times, it leads to a corresponding automatic change in the pre-filter definition. This is accomplished by looking for the responsible trigger words in the pre-filter definition and correcting the associated relevance rating.

## 5 Evaluation

Our goal for the evaluation of PEA is to develop an evaluation methodology that considers adequacy aspects, guarantees a certain degree of generalization, and includes subjective criteria such as user acceptance as well as objective criteria, e.g. processing time, correctness, etc. Besides the evaluation of the complete system we will also conduct experiments to examine the actual impact of the various subcomponents to the overall system performance. This will include different test runs with parts of the system disabled as well as the search for optimal values of the numerous parameters for the individual algorithms.

Whereas there are many well-established techniques available for gathering subjective data from users (questionnaires, interviews, etc.), the definition of quantitative measures is a more difficult problem for information filtering systems. For the field of information retrieval the measures *recall*, *precision*, and *fallout* are pretty standard [19]. However, they can be applied to only two relevance categories, usually called "relevant" and "not relevant". Because of the flexible definition of relevance categories in our system, which allows any number of categories, these measures are of only limited use. Therefore, we propose an extended accuracy measure according to (6), which we have used for calculating the fitness of chromosomes during evolutionary adaptation (see Sect. 4.4):

$$\text{Accuracy} = 1 - \frac{\sum_{i=1}^{n} |R_{\mathrm{S},i} - R_{\mathrm{U},i}|}{n \cdot (c-1)} \quad . \tag{9}$$

In this formula we have replaced the rating of a single chromosome by the overall relevance rating of the system $R_{\mathrm{S},i}$. Therefore, $|R_{\mathrm{S},i} - R_{\mathrm{U},i}|$ stands for the deviation of the $i$th rating of the system from the $i$th rating of the user.

We have also developed a *graphical representation* of the *rating differences* in that we first transform them into a *matrix* $(r_{ik})$ where the individual $r_{ik}$ indicate the absolute frequency of the event that the system chooses rating category $i$ and the user the category $k$. The matrix is then displayed by making use of 3-dimensional visualization techniques (see Fig. 14 for an example of the graphical representation of the data from Table 1).

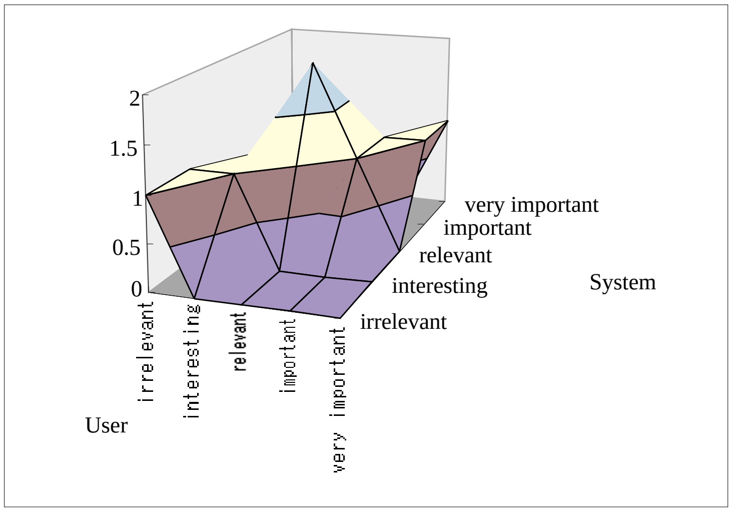
This gives a clearer and more detailed picture of the rating performance of the system and also assists in finding exact locations and tendencies of rating errors in a more precise and direct way.

## 6 Conclusion

In this paper we have presented a Personal Email Assistant, which aims at providing an adaptive environment that goes beyond the limitations of existing static approaches. In particular we address the problem of adjusting the user model in response to shifts of interests and behavior by making use of evolutionary algorithms and monitoring techniques.

We have finished the implementation of all components of PEA by using as programming language C/C++ under UNIX at a SUN SPARC 10 workstation. As next step we have just started an extensive evaluation study with a test group of 30 students from our department. Therefore, it is too early to make any final remark about the success of our system. However, in first test sessions we have received quite positive comments and feedback from the users.

$$(r_{ik}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$



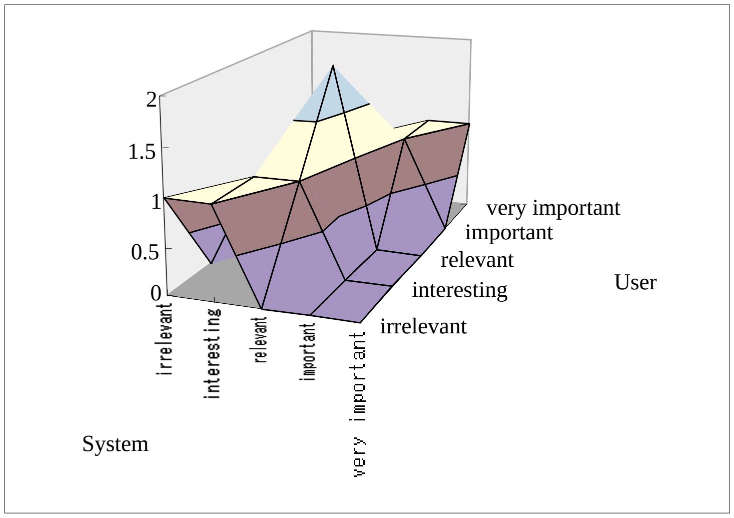


Figure 14: Example of graphical representation

Future work will concentrate on an extension of our approach to the use with the World Wide Web. By applying agent technology we want to create intelligent agents that browse the Web on behalf of the user to gather useful information. Furthermore, we plan to display the search result as HTML document by utilizing hypertext facilities and to also support the collaborative filtering of agents.

We believe that PEA will become a valuable tool for cognitive information filtering with a high potential for solving the urgent "information overload problem" and for helping the user to effectively administrate the most valuable good of the coming information society.

## References

[1] R. B. Allen. User models: Theory, method, and practice. *Intl. Journal Man-Machine Studies*, 32:511–543, 1990.

[2] J. Anderson. *Cognitive Psychology and Its Implications.* Freeman, New York, 1985.

[3] C. Apté, F. Damerau, and S. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 1994.

[4] L. Ballesteros and W. B. Croft. Dictionary methods for cross-lingual information retrieval. In *Intl. Conf. on Database and Expert Systems Applications*, pages 791–801, 1996.

[5] J. Bates and A. Lavie. Recognizing substrings of LR(k) languages in linear time. Technical Report CMU-CS-91-188, Carnegie Mellon University, 1991.

[6] R. J. Bauer. *Genetic Algorithms and Investment Strategies.* John Wiley & Sons, New York, 1994.

[7] T. Bayer et al. Domain and language independent feature extraction for statistical text categorization. In *Workshop on Language Engineering for Document Analysis and Recognition*, pages 21–32, 1996.

[8] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[9] L. B. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40:235–282, 1989.

[10] J. W. Breen. Building an electronic Japanese-English dictionary. In *Japanese Studies Association of Australia Conf.*, 1995.

[11] J. Cowie and W. Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, 1996.

[12] D. A. Evans and C. Zhai. Noun-phrase analysis in unrestricted text for information retrieval. In *Annual Meeting of the Association for Computational Linguistics*, 1996.

[13] G. Fischer and C. Stevens. Information access in complex, poorly structured information spaces. In *CHI Conf.*, pages 63–70, 1991.

[14] S. P. Gant. Information filtering agents. In J. Williams, editor, *Bots and Other Internet Beasties*, pages 173–214. Sams.Net, Indianapolis, 1996.

[15] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, 1989.

[16] D. E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, 37(3):113–119, 1994.

[17] D. E. Goldberg et al. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[18] T. Gross and R. Traunmüller. Computer-supported cooperative work and the internet. In *Intl. Workshop on Database and Expert Systems Applications*, pages 425–430, 1996.

[19] D. Harman. Overview of the third Text REtrieval Conference (TREC-3). In *Text REtrieval Conference*, pages 1–20, 1994.

[20] M. Höfferer, B. Knaus, and W. Winiwarter. Adaptive information extraction from online messages. In *RIAO Conf., Intelligent Multimedia Information Systems and Management*, 1994.

[21] M. Höfferer, B. Knaus, and W. Winiwarter. Cognitive filtering of information by evolutionary algorithms. In *KI-94 Workshop on Information Filtering*, 1995.

[22] J. H. Holland. *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, Cambridge, Mass., 1992.

[23] S. B. Huffman. Learning information extraction patterns from examples. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 246–260. Springer-Verlag, Berlin, 1996.

[24] P. S. Jacobs and L. F. Rau. SCISOR: Extracting information from on-line news. *Communications of the ACM*, 33(1):88–97, 1990.

[25] J. Kay and B. Kummerfeld. User model based filtering and customisation of web pages. In *Workshop on User Modelling for Information Filtering on the World Wide Web*, 1996.

[26] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, Mass., 1992.

[27] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In *Annual Meeting of the American Association for Artificial Intelligence*, 1994.

[28] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7), 1994.

[29] T. Malone et al. Intelligent information sharing systems. *Communications of the ACM*, 30(5):390–402, 1987.

[30] G. McCalla et al. Analogical user modelling: A case study in individualized information filtering. In *Intl. Conf. on User Modeling*, pages 13–20, 1996.

[31] M. McElligott and H. Sorensen. An evolutionary connectionist approach to personal information filtering. In *Irish Neural Network Conf.*, pages 141–146, 1994.

[32] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolutionary Programs.* Springer-Verlag, Berlin, 1992.

[33] I. Moulinier and J.-G. Ganascia. Applying an existing machine learning algorithm to text categorization. In S. Wermter, E. Riloff, and G. Scheler, editors, *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 343–354. Springer-Verlag, Berlin, 1996.

[34] S. Mukhopadhyay et al. An adaptive multi-level information filtering system. In *Intl. Conf. on User Modeling*, pages 21–28, 1996.

[35] S. C. Newell. User models and filtering agents for improved internet information retrieval. *User Modeling and User-Adapted Interaction*, 7(4):223–237, 1997.

[36] A. O'Riordan and H. Sorensen. An intelligent agent for high-precision text filtering. In *Intl. Conf. on Information and Knowledge Management*, pages 205–211, 1995.

[37] S. Pollock. A rule-based filtering system. *ACM Transactions on Information Systems*, 6(3):232–254, 1988.

[38] T. S. Ray. Is it alive, or is it a GA? In *Intl. Conf. on Genetic Algorithms*, pages 527–543, 1991.

[39] P. Resnick et al. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM Conf. on Computer Supported Cooperative Work*, pages 175–186, 1994.

[40] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[41] H. P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser Verlag, Basel, 1977.

[42] E. Schweighofer and D. Scheithauer. The automatic generation of hypertext links in legal documents. In *Intl. Conf. on Database and Expert Systems Applications*, pages 889–898, 1996.

[43] E. Schweighofer and W. Winiwarter. Refining the selectivity of thesauri by means of statistical analysis. In *Intl. Congress on Terminology and Knowledge Engineering*, 1993.

[44] H. K. Shuldberg et al. Distilling information from text: The EDS template filler system. *Journal of the American Society for Information Systems*, 44(9):493–507, 1993.

[45] C. Stevens. *Knowledge-Based Assistance for Accessing Large, Poorly Structured Information Spaces*. PhD thesis, University of Colorado, 1992.

[46] T. Strzalkowski, J. P. Carballo, and M. Marinescu. Natural language information retrieval: TREC-3 report. In *Text REtrieval Conference*, pages 39–54, 1994.

[47] T. Tokunaga and M. Iwayama. Text categorization based on weighted inverse document frequency. Technical Report 94-TR0001, Tokyo Institute of Technology, 1994.

[48] T. Wakao. Reference resolution using semantic patterns in Japanese newspapers. In *Intl. Conf. on Computational Linguistics*, pages 1133–1137, 1994.

[49] W. Winiwarter. MIDAS — the morphological component of the IDA system for efficient natural language interface design. In *Intl. Conf. on Database and Expert Systems Applications*, pages 584–593, 1995.

[50] W. Winiwarter, O. Kagawa, and Y. Kambayashi. Applying language engineering techniques to the question support facilities in VIENA Classroom. In *Intl. Conf. on Database and Expert Systems Applications*, pages 613–622, 1996.

[51] B.-T. Zhang, J.-H. Kwak, and C.-H. Lee. Building software agents for information filtering on the internet: A genetic programming approach. In *Genetic Programming Conf.*, 1996.