Message Classification and Filtering

Fredrik Kilander
fk@dsv.su.se
http://www.dsv.su.se/~fk

January 4, 1995

IntFilter
K2LAB — Laboratory for Knowledge Engineering and Communication
Department of Computer and Systems Sciences
Stockholm University and the Royal Institute of technology
Electrum 230, S-164 40 KISTA
Sweden

1 Introduction

This document is a proposal of how to implement classification and filtering of messages in the computer conference system KOM. The KOM system consists of two basic components, a *client* and a *server*. The client is the user interface through which the user exchanges messages with the server and updates personal information. The server acts as a central repository and database for both messages and information about users. In addition to its communication with the clients, the server also exchanges information with other servers. These other servers may be other KOM servers or servers for Usenet News or electronic mail.

From the user's point of view, the KOM system collects messages from several sources (KOM, Usenet News, mailing lists and email) and presents them in a coherent form. The KOM client enables the user to process messages from or to these sources through a single interface.

2 Messages Sequenced by Multiple Sources

In a traditional setup where the user has to execute email, Usenet News and computer conferencing as separate tasks, each such task is a discrete and easily identified piece of work. It appears intuitive to expect that people schedules these tasks in order of importance; say, email as most important and done first,

computer conferencing as second most important and performed as the second task, while Usenet News is done last if time so allows.

Within each of these discrete messages sources (email, Usenet News and computer conferences) messages have internal priority. For instance, mailing-lists are commonly recieved through email, and their contents is probably not as highly valued as a letter addressed directly to the user. Likewise, a computer conference system usually contains its own mechanism for personal messages. Such messages are equal in status to the email letter, while most users probably rate the discussion groups mediated by the conference system with different levels of interestingness. Finally, Usenet News may in turn contain newsgroups which the user pays undivided attention, while others receive only cursory interest.

The use of a single interface such as the KOM client has both advantadges as well as caveats. While the user only has to learn a single program, the amount of information made available through that program is increased, as everything is funneled in to the user through the client. It is technically possible, but pointless, to artificially recreate in the KOM system the division between email, KOM and Usenet News. It is probably more meaningful to the user to process a sequence of messages where the most important ones comes first and the least important messages arrive last. This implies that (given the preferences already assumed) the sequence begins with personally addressed letters, originated either within the KOM system or from an outside email source. Next follows a mixture of mailing lists and computer conferences generic to KOM or imported from Usenet News.

The KOM system visibly distinguishes between two types of message origins, personal mail and conferences. External sources such as mailing lists and Usenet News newsgroups are coerced into the form of conferences. The default message sequencing in KOM consists of personal mail followed by a list of conferences with unread submissions. The sequence of conference presentations depends on the order in which the user subscribes to the conferences. The user may alter the sequence of conference presentations by moving a conference to the top of the list, but if left intact the sequence will remain the same between sessions. This has the effect of offering the user a sequence of conferences similar in appearence, but where messages has originated in mailing lists, (Usenet News) newsgroups and generic KOM conferences.

3 Message Sequencing within a Conference

Within a particular conference or newsgroup there are basically two types of messages; *original* messages and *followups*, although the distinction at times can be hard to make. Original messages are commonly presented to the readers of the conference as questions, information, advertisements or opinions. Followups usually refer to the text of an original message or other comments. Together,

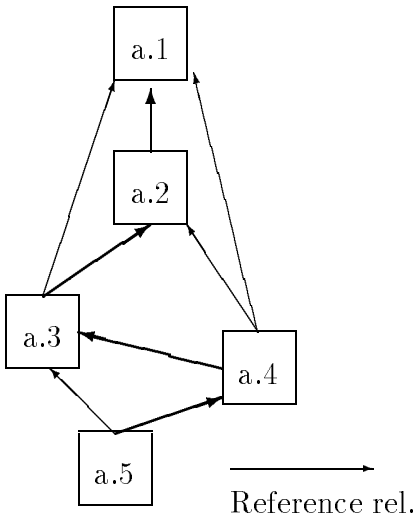

Figure 1: An example of a thread in a newsgroup.

the original message and the comments form a *thread* of discussion on some topic. It is not uncommon that threads only contain the original message.

Threads in Usenet News are maintained by two fields in the header of a Usenet News message:

- `Message-ID` and
- `References`.

The `Message-ID` field gives the message a uniqe identity. When a message is a followup to another message, the last item of the `References` field is the message ID of the followup'ed message. The field is not restricted to a single message ID, it may be a list of several message ID's, copied from the `References` field of the followup'ed message. However, the standard allows for this list to be truncated if the list becomes too long for local software to handle. As a consequence, the last item on the list is the only required link to the referenced message, and a message can only be a followup to one message, not several.

Figure 1 shows the structure of a chain of followups. Bold arrows indicate the main reference (the last item in the `References` field) while thin arrows represent references inherited from the followup'ed messages. Message $a.1$ is the original message, $a.2$ is a followup to $a.1$, $a.3$ is a followup to $a.2$, $a.4$ is a followup to $a.3$ and $a.5$ is a followup to $a.4$. Message $a.5$ has truncated its list of message ID's and has only one prior reference.

Figure 2 illustrates the structure that emerges when only the last item of the `References` field is regarded and there are several followups to the same

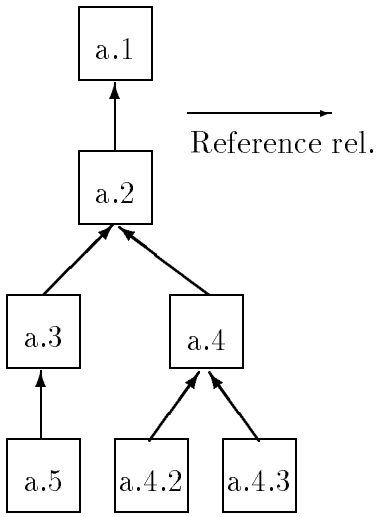

Figure 2: A example of a logical thread tree.

message. In the example, messages $a.3$ and $a.4$ both are subroots of independent followup chains.

Figure 3 shows a conference with three threads. The threads are labelled a, b and c. The structure is a combination of temporal ordering (thin arrows, pointing towards the future) and reference relations (thick arrows, pointing towards the message referred to). In essence, the contents of a conference can be viewed as a temporally ordered lists of threads.

The presentation sequence offered by most news readers and the KOM system is the result of a scan over the threads (major ordering) and within each thread a depth-first traversal of the graph (minor ordering). In addition, the interface ensures that each message is shown to the user only once.

Most news readers allow random access to messages but usually the sequence governed by newsgroup, thread and depth-first traversal of the thread prevails. This is technically very convenient and usually compatible with the user's needs. Since text is inherently a sequential form of communication, it is up to the interface to convey to the user why a certain message is shown at a particular time.

4 Message Classification and Filtering

The purpose of classification and filtering of messages is to *increase the amount of relevant information* that reaches the user. This is done by increasing the weight of interesting messages and decreasing the weight of uninteresting messages. The criteria for what is regarded as interesting, lies entirely with the user.

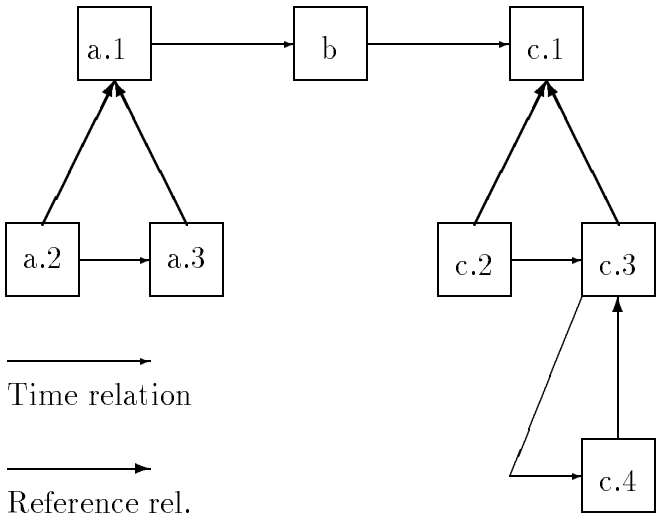

Figure 3: An example of a conference with three threads.

A computerized message classificiation process must address two fundamental issues:

- modelling the user's interest in the computer
- classifying each message according to the user model.

Once the classification process is complete, the filtering, or weighting of each message takes place. The simplest result of the filtering is to rearrange the message sequence so that the most interesting messages are presented before the ones classified as less, or not interesting. Indeed, it may be the case that the user trusts the system to completely delete the most uninteresting messages.

The main reason to separate classification from filtering is that the user's classification profile is relatively constant between sessions, while the filtering requirement is subject to daily variation. Consider for example a user who one day spends thirty minutes with the conference system and only five the next. Alternatively, a user who wants at most ten messages each day, regardless of the inflow. The filtering criteria is different each day.

Figure 4 shows a schematic view of the classification and filtering process.

4.1 Human Message Classification

Categories of messages are in their simplest form only two, interesting and not interesting. Elaboration may provide more classes, or a continuous range between extremes.

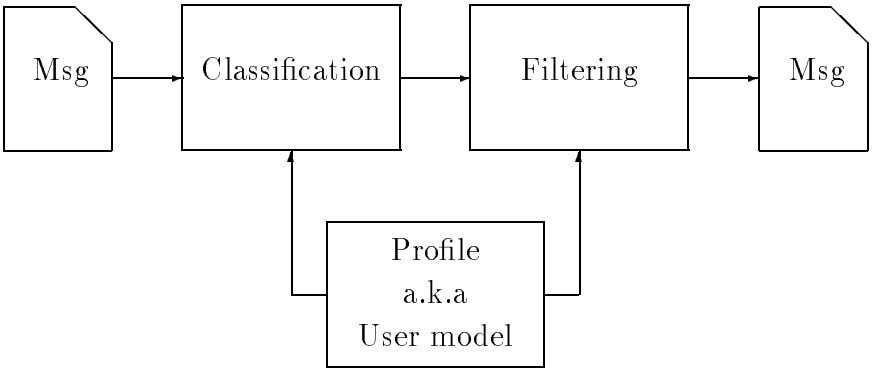

Figure 4: Schematic view of the classification and filtering process.

However, in the assimilation of a message stream the user (with varying degrees of conciousness) assigns each message with a set of *perceived properties*. Examples of such properties may be *subject*, *origin* or *comprehensibility*.

For each message, the collection of perceived properties is balanced to form the final judgement of interestingness. The properties may be in conflict with each other.

While reading, the property assignment and classification process continues as more and more information is obtained from the message. As a particular newsgroup is visited, the nature of the newsgroup itself creates a set of expectations, or a context in the mind of the reader. Messages will be judged in relation to this context. As a message appear, the first pieces of information (the contents of the subject, the layout of the text, the first lines of the message) adds to the set of expectations. Depending on which expectations that are denied or fulfilled, the reader may switch between reading and scanning until the reader abandones the message (see figure 5). A switch from reading (which is a cognitive effort) to scanning (which is not) signifies that the reader is uncertain and hopes to be able to satisfy the expectation that the message contains something of interest.

The message is always abandoned in some stage. It may be abandoned before the reader has seen or read all text. It can be saved to a file or a printer for future reference or the reader reaches the end of the message. At this point the user either knows that the message indeed was interesting and the reading a meaningful experience. The remaining case is a disappointment in that the whole message had to be exposed or read before the reader knew that the reading was effort wasted. This last case is common for short messages simply because they fit on a computer terminal and contain a rather small amount of information. In order to know if this information is interesting or not one has to read it. So, if the human reader can't decide upon the subject and context

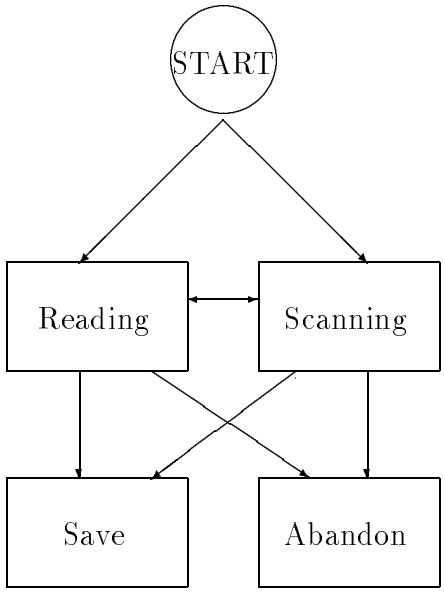

Figure 5: Message reading state diagram.

of the conference, there appears to be some minimum amount of text (e.g. 5-15 lines) necessary in order to form a definite opinion.

The presence of key words or phrases in the subject and text greatly aid the classification process since the keywords release specific sets of expectations in the reader.

The identification of perceived properties in the user and their mapping to machine-readable properties in messages appears central to the success of an automatic message classification process.

4.2 Mapping the User Model

The coupling between the user's intermediate categories and a particular message can be viewed as the projection of properties onto the message in the mind of the user. This is a very difficult task, for two reasons:

- The set of perceived properties a message invokes in a human reader varies between different individuals. The set of properties is difficult to capture and model accurately in a general way.
- The mapping from the machine-readable and physical properties of a message correspond very loosely to concepts that are familiar and important to the reader.

In order to perceive the same properties as the user, the computer must be able to read and understand the message in the same way as the human. This is currently not within reach. To even get close, and establish at least a fundemental understanding requires expensive computations for natural language processing, domain knowledge and psychological competence.

The prevailing strategy is to force the user to provide the mapping by restricting the user model to those message properties that are immediately available or inferrable from unexpensive computation. By restricting the language, the user is forced to formulate the preferences in terms that the machine can understand and manipulate: the length of a message, the author, the conference, an answer to a question etc. This is probably the reason why keywords so often are choosen as a simple and fairly effective classification aid: the keywords are characteristic symbols for well-defined concepts in the user's mind. It is not necessary to understand the text in order to scan for keywords and it is a task well suited to a computer.

4.2.1 Machine-readable Message Properties

Machine-readable properties are computed from the message itself. Some of the properties are absolute, like the number of lines or the contents of the header fields. Other properties are inferred by computation. Inferred properties are guided by a model. In some cases the model is considered universal and in other cases provided by the user. An example of a universal model is genre classification, while a keyword search is an example of a user-specific inference model.

Obviously, absolute and universally inferred properties only has to be established once for each message. Properties inferred from user models must be done for each user and message[1].

Table 1 lists a number of message properties suitable for computer processing. For each property, the table indicates if the property is absolute, inferred according to a universal model or a model which must be complemented for each user. In each case, the value of each property is compared to the user model if the model requests it.

4.2.2 The Message Header

Some of the fields in the message header have already been discussed. Their description is repeated in this section. The most common header fields are shown in table 2. In addition to the required and optional fields, any valid mail-header field is allowed. Furthermore, unrecognized fields are to be left unmolested [?].

The `From` field contains the name and email address of the person or entity posting the message. The `Sender` field should also be specified when the user

[1] Each message in the user's default presentation sequence.

Property	Abs.	Univ.	User	Cost
Message header	X			low
Position in subject thread	X			low
Wordcount		X		low
Other reader's evaluations	X			low
Percentage of quoted text		X		low
Quoting depth		X		low–medium
Language		X		medium–high
Keyword search			X	medium–high
Pattern search			X	medium–high
Layout analysis		X		medium–high
Spelling		X		high
Genre analysis		X		high

Table 1: Machine-readable message properties.

Field	Required
From	yes
Date	yes
Newsgroups	yes
Subject	yes
Message-ID	yes
Path	yes
Reply-To	no
Sender	depends on the From field
Followup-To	no
Expires	no
References	iff the message is a followup
Control	no
Distribution	no
Organization	no
Keywords	no
Summary	no
Approved	in a moderated newsgroup
Lines	no
Xref	no

Table 2: Standard header fields in a Usenet message.

edits the `From` field rather than using the supplied default. This may happen when someone is sending a message from a username used by several people.

The `Date` field contains the date and time when the message entered the news system (was posted). If a finer resolution than 24 hours is required time-zones must be understood. The standard recommends that universal time (GMT) is used and translated to local time whenever necessary. Unfortunately, the `Date` field does not prevent messages from occasionally being presented in the wrong order; clocks are not globally synchronized and transportation delays may cause an answer to reach the local news server *before* the question.

The `Newsgroups` field is a list of newsgroups to which the message is posted. The storage and presentation software uses this to arrange the messages in newsgroups. In particular, messages directed to groups not subscribed to can be erased at once by the news server.

The `Subject` field is for humans. According to [?]:

> "It should be suggestive enough of the contents of the message to enable a reader to make a decision whether to read the message based on the subject alone."

Messages with identical `Subject` fields usually belong to the same thread. However, it is common practice for the software (and sometimes the users themselves) to prepend the string *Re:* to the original subject string when submitting a followup.

The `Message-ID` field gives the message a uniqe identity. The standard recommends that a message ID is uniqe for at least two years. With some restrictions, it is up to the sending software to compose the ID, using a counter or some transformation of the local date and time.

The `Path` field is for the benefit of the distribution software. As the message is passed from one host to the next, each one prepends its own name to the contents of this field. The information is used for diagnostics and to avoid sending messages to hosts who already has received them.

The `Reply-To` field is specified if mailed replies to the author of the message should go to some other address than the one specified in the `From` field.

The `Followup-To` field is used to redirect followups to the message to other newsgroups than those specified by the `Newsgroups` field.

The `Expires` field suggests an expiration date for the message. This field should be used with deliberation since practically all news servers have their own expiration policy, based on storage resources and volume.

The `References` field is a list of message ID's, if the message is a followup to those messages. The field is not allowed for original messages. The structure that is created by the list of messages is slightly peculiar in that each followup inherits the list of referenced messages from the message it is a followup to. It then adds the message ID of the referenced message to the end of the list. In theory, the `References` field of each message which is a followup to another

message, contains the entire list of messages preceeding it. However, the standard allows for this list to be truncated if the list becomes too long for local software to handle. As a consequence, the last item on the list is the only required link to the referenced message, and any other references before the last link are bonus information.

The `Control` field is for internal administration of the News system and not aimed at any human user (with the possible exception of the system administrators).

The `Distribution` field controls how far the message is spread. The intended use of this field is to *reduce* distribution; i.e. limit distribution to a smaller area. The contents of the field form a conjunction with the `Newsgroups` field which must be satisfied for the message to be transmitted and received.

The `Organization` field is for humans. It is intended to contain a short phrase which helps to identify the person or machine which sent the message.

The `Keywords` field is intended for

> "a few well-selected keywords identifying the message... This is used as an aid in determining if this message is interesting to the reader."

One wonders how many news-readers that encourage their users to submit keywords when posting a message.

The `Summary` field

> "should contain a brief summary of the message. It is usually used as part of a follow-up to another message. Again, it is very useful to the reader in determining whether to read the message."

Since almost all users of Usenet News appears top cite other messages in their own text, it appears that this field is largely unused.

The `Approved` field is used in moderated newsgroups. It could be seen as the moderator's signature.

The `Lines` field simply is the number of lines in body of the message.

The `Xref` field is provided by the local system. The information links the message to the files that store it.

4.2.3 Position in Subject Thread

This information is extracted from the `References` field in the header. Its usefulness is related to such concepts as *answers to questions*, *original* messages and, of course, followups.

4.2.4 Wordcount

The number of words in the body of the message provides complimentary information to the number of lines obtainable from the header. Although a limit of 80 characters per line often is assumed, both longer and shorter lines are

common. The algorithm performing the wordcount is quite affected by which language the message is written in. For instance, the '{' character is not regarded as part of a word in an english message, but is very likely to represent 'ä' in a swedish language message.

4.2.5 Other Reader's Evaluations

This is an interesting topic and it assumes that the conference system is able to collect and store reader's assessments of a message. In its simplest form, the assesments is represented as a point in an N-dimensional space, for example in terms of *accuracy, quality, novelty* and *contribution*.

There are two major problems with reader assessments. The first is that user's may be unwilling to provide good assessments simply because it takes too much of their time and adds a cognitive strain to the regular reading process. This problem has no obvious solution—the users must be motivated, either by altruism or recognition of the synergetic benefit.

The second problem is that an assessment always is made in some context; for instance personal satisfaction or recognition of the needs of a working group. The recorded assessment must somehow be able to reflect this context or the stored value will be meaningless.

Perhaps the most general solution to both problems is that each user so inclined reviews messages for personal reasons only, and that other users *prefer* the assessments made by people they feel share their own views.

4.2.6 Percentage of Quoted Text

Many messages include large portions of text cut and pasted from other documents. Usually, these quoted sections are indicated by indentation and prefixation by a special character, for instance '>', ':', ']' or '!'. A simple, yet quite effective heuristic, is to regard lines that begin with a printing, non-alphabetic character as cited text (excluding blank lines and initial white-space).

4.2.7 Quoting depth

The analysis of quoted text can be extended to indicate different levels of quotation and assigning probability estimates to the status of each line. The percentage of quoted text and quoting depth is related to questions and answers, followups short in original content and layout analysis.

4.2.8 Language

With the dominant languge being english, readers may be expected to pay little attention to deviant languages other than those in national groups. Other classification routines, for instance the wordcount, work better if the language of the message (or paragraph, or line) is known.

4.2.9 Keyword Search

Keywords are very effective (if choosen correctly) to cull those messages that are of a particular class. The search and matching process is subject to a great deal of variation and sophistication. Keywords found in a message have different status depending of where they are found; in the *Keywords* or *Subject* fields, in original or quoted text.

4.2.10 Pattern Search

Pattern searches in a message aim at finding special constructs that occur in specific contexts. Examples of such constructs are email addresses, URL's for World Wide Web, portions of source-code from programming languages or text-encoded binaries.

4.2.11 Layout Analysis

The idea behind layout analysis is to model or mimick the impression the reader gets when scanning the message. It reflects the structure of the message, not the content. Typical patterns that reflect different messages are long, original documents with no quoted text and justified text (call for papers, short stories); followups with much quoted text and few lines of original content at the end ("me too"); snippets of source code with regular text between (tutorials, examples). The layout analysis needs to classify each line according to some criteria, such as quoted text or programming language.

4.2.12 Spelling

Checking the spelling of the text gives information if the text was written in haste or carefully prepared. Spelling control of course relies on language classification.

4.2.13 Genre Analysis

The genre analysis attempts to classify the message in general terms, for instance *religion, skills and hobbies, general fiction* or *humor*. While this has proven to be possible, these general genres appears to be subsumed by the specialization of the conferences. It may be a valuable tool to classify conferences and newsgroups.

4.3 Classification Mechanisms

Message properties and the their relation to the classification model must be encoded in some data structure convenient for processing. The classification

mechanism applies the data structure to each message and arrives at a classification. Conceptually, the model maps a specified pattern of properties to a classification. Data structures and classification engines are available in several guises: rule systems, decision trees, neural network models and probabilistic classifiers.

4.3.1 Rule Systems

A rule system consists of a set of rules and a production engine. Each rule is of the format:

`IF` *condition* `THEN` *assertion.*

The *condition* of the rule is a boolean expression. For each message it will be either true or false. The basic terms of the condition can be almost anything which can be decided to be either true or false. For instance *keyword* '`computerization`' *appears more than three times* or *quoted text more than 50 percent.*

The *assertion* of the rule specifies what happens when the condition is true. In the simplest case, the assertion is a classification. Alternatively, the assertion is either a classification or a new property, inferred by the rule. The decision to allow rules to infer properties which other rules may act upon is one of implementation. The idea with inferred properties is that recurring combinations of conditions doesn't have to be repeated in each rule that use that condition:

$$\begin{aligned} P \wedge Q \wedge R &\rightarrow Z \\ Z \wedge S &\rightarrow W \\ Z \wedge T &\rightarrow X \\ Z \wedge U &\rightarrow Y \end{aligned}$$

is much more convenient than

$$\begin{aligned} P \wedge Q \wedge R \wedge S &\rightarrow W \\ P \wedge Q \wedge R \wedge T &\rightarrow X \\ P \wedge Q \wedge R \wedge U &\rightarrow Y\,. \end{aligned}$$

Problems that must be addressed with rule systems can be summarized like this:

- who writes the rules?
- which basic terms are to be used for the conditions?
- what happens when more than one rule has a true condition and the assertions are mutually exclusive?
- should one allow the condition of one rule to subsume another?

Among the good points with a rule system are that they are fairly easy to read and understand, and allows for a compact and powerful representation.

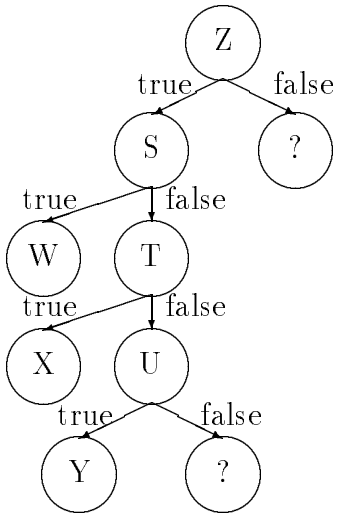

Figure 6: A simple decision tree.

4.3.2 Decision Trees

Decision trees are similar to rules in that a decision tree can be rewritten as a rule system. A decision tree typically consists of a recursive structure in which each node contains a test on a basic term. Depending on the outcome of the test, a branch leading to a subtree is selected where another test is found. Eventually a leaf is reached and instead of a test, the leaf contains the assertion. The path from the root to the leaf can be seen as a rule, with a condition formed by the conjunction of the tests on the basic terms.

Figure 6 shows the symbolic rules presented above as a binary decision tree. Note that two leafs are marked *'?'*, indicating that there is no known assertion at that leaf.

Problems with decision trees are that they handle disjunctions uneconomically, in that each disjunction generates extra subtrees. They can also become quite large and therefore a bit unwieldy to inspect. A naive translation into rules presents them with long and complicated conditions.

Positive aspects of decision trees are that there are good techniques to generate them automatically from pre-classified examples and that they are efficient in use.

4.3.3 Neural Network Models

Neural network models, or NN for short, has been studied and used for many years in classification. From an operational point of view they can be viewed as black boxes that after training perform a certain classification task. In the

classification of messages, the user is required to supply the NN machinery with a set of accurately classified training examples.

The NN models do not easily reveal their internal structure. It is distributed and is for all practical reasons, opaque. This makes it difficult to provide the user with motivations for a particular classification. Erroneous classifications can only be removed by retraining, and then it may be difficult to pinpoint the training examples that should be removed from the training set.

4.3.4 Probabilistic Classifiers

Probabilistic classifiers use various statistical techniques, mostly based on Bayesian probability. From a set of given features, they calculate the set of probable classes and their respective probability. Their are quite robust and not that complicated in their basic appearence. They have problems with the assumed independence between selected features and certain pathological sitations involving mututally exclusive features. These problems have solutions, but the solutions typically are research issues.

Just as with neural network models, it may be difficult to structure an explanation for a classification in intuitive terms; the mapping from features to class is a matter of frequency and likelihood.

4.4 Machine Learning Techniques

Three machine learning techniques are considered. They are not mutually exclusive and hybrid solutions are possible. With one exception, the learning technique relates very closely to the classifier it creates.

4.4.1 Induction of Decision Trees

This technique constructs a decision tree by recursive subdivision of a set of training examples. The result can be used either as is, or reformulated as rules for a rule system. Originating from the CART algorithm by Breiman et al and ID3 by Quinlan, modern decision tree builders are more robust and capable than their predecessors: C4.5 by Quinlan; Assistant by Bratko et al; ID5 by ??? and IDL are some of the modern algorithms.

4.4.2 Neural Network Models

Many different variants exists here both in architecture and algorithm. Among the commonly used variants are found a three-layered net trained by backpropagation; Kohonen networks and Hopfield nets.

4.4.3 Probabilistic Classifiers

The Recursive Bayesian Classifier by Langley, LMDT by Utgoff and Brodley.

4.4.4 Genetic Programming

Also known as genetic algorithms, and the slightly confusing term *"classifier systems"* this technique is applicable to each of the above machine learning techiques. It is a strategy for parallel hill-climbing in the space of solutions. To be used, it requires a function that provides a numerical distance between two alternate solutions. The solutions themselves, may be almost anything: rules, decision trees, networks or probabilities.

One particular filtering system has been built with the aid of genetic programming; the Newt system by Sheth. It uses a vector-based approach to model messages and profiles.

4.5 Message Filtering

The conference sequence structure presupposes that users prefer to assimilate all messages in each conference before moving on to the next. While this appears perfectly reasonable both from a cognitive and technical point of view, it conflicts with the possibility of reading a message classified as important ahead of the conference sequence. One viable solution is to allow the classification process to *rearrange* the conference sequence, based on some measure of importance for the conference as a whole.

Within each conference, a similar substructure is displayed by the thread abstraction, and again the solution appears to be to form a measure of the importance of the thread as a whole and rearrange the sequence of threads to reflect this importance.

In essence, the filter assigns each message a weight. The weight of individual messages propagate to their thread, and the weight of the threads propagate to the conference. This retains the the conference and thread structure, but does not rearrange interesting messages within a thread. Weights can of course be both positive and negative.

There are two problems with this approach. Unless signalled for each message, the user has little chance of knowing when an important message (in the opinion of the automatic system) is presented. The interestingness of messages will be similar to the curve in figure 7. The valleys in the curve are created by messages with less weight but presented with the more important ones because they are bound together by the same thread.

The second problem is how to formulate a proper summation function for a thread or a conference. Is a one-message thread with a message of weight 1.0 equally important to multi-message thread where only one message has the same weight?

In conjunction with the regular presentation sequence it may be advantageous to offer a separate list of interesting messages, sorted by weight.

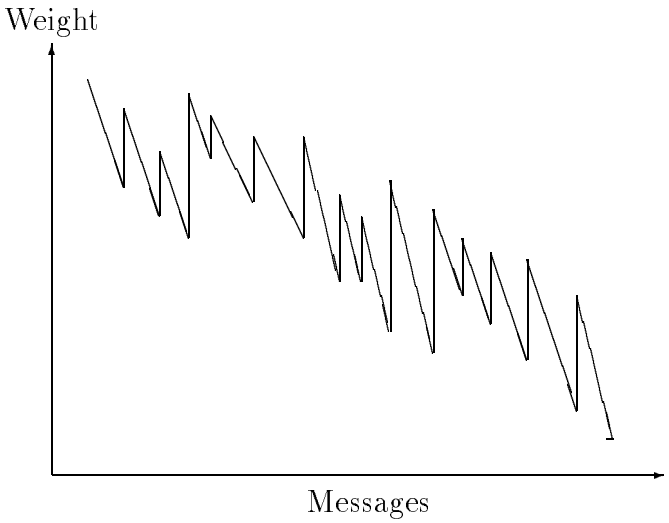

Figure 7: Hypothetical message weight over the standard presentation sequence.

5 Requirements on the Classification and Filtering Module

The requirements are divided into three groups:

- user interface
- application interface
- operation.

5.1 User Interface

1. The user must be able to see and understand the classification and filtering condition.
   (a) The most recent classification and filtering decision made for a user must be stored.
   (b) The decision must be time-stamped as it may become outdated by learning or editing of the classifier and filter.
   (c) The decision can be removed when the message is erased from the conference system (implies erasure notification).
2. The user must be able to modify the filtering conditions.

(a) The user must be able to label messages, or part of messages, as positive or negative instances of a class, and to maintain these as a current set of interests. This implies deletions and additions to the set.

(b) The user must be able to confirm or disapprove of the automatic classification. (What bias can be expected from messages lost to erroneously assigned low weights? The user will see and judge most of the messages classified as interesting, but runs the risk of not being able to correct hidden messages.)

(c) The ability to specify priority conditions (rules that override the automatic classification) is, while desirable, not critical to the automatic classification task per se.

5.2 Application Interface

In the context of the KOM system, the classification module is perceived as a piece of software cooperating and communicating with the KOM server.

Messages in the direction KOM $\rightarrow$ classifier:

1. Message M may be presented to users $\{U_1 \ldots U_n\}$.

2. Message M has been deleted.

5.3 Operation

5.4 User Profile Maintenance

The computer's model of a user's interests is kept in a user *profile* or *model*. A factor that tends to complicate the maintenance of the profile is that the importance of the intermedaite categories change over time and sometimes are dependant on every-day context.

In order to capture long-term change in the user's interests, it is necessary to provide continuous feedback from the user to the computer on its performance. For this several mechanisms are plausible. In the simplest case, the user reformulates, or edits the interest profile as if it was issued for the first time. More elaborate methods rely on confirmation from the user as a signal for incremental change.

Every-day contexts appears difficult to capture and formalize without the user explicitly communication the context to the computer. One solution may be to arrange a set of pre-arranged names, to which the user has associated recurring contexts, like *extremely busy*, *calling remote*, *cracker attack* or *feeling lazy*. In each of these contexts, the relative importance of the intermediate categories is altered, to better reflect the needs of the current context.