Intelligent Information Filtering

The IntFilter Project
Fredrik Kilander
Eva Fåhræus
Jacob Palme

Dpt of Computer and Systems Sciences
Stockholm University

February 17, 1997

Contents

Chapter 1

Introduction

1.1 The need for intelligent information filtering

Group communication systems are spreading increasingly fast as they become part of the infrastructure in educational, commercial and governmental organizations. Modern information technology provides efficient means for fast coordination, dissemination and retrieval of information. As with all media they have specific advantages, but with increased popularity a number of disadvantages also emerge, such as information overload and a low ratio of quality to quantity.

When users must spend more time on the processing of incoming messages they are usually faced with two alternatives. Some may be able to cut down the flow by deselecting a source, eg by cancelling subscriptions on Internet mailing-lists and newsgroups. Others who do not have this option need help. This help can be an additional worker or a computer program. Even if the message processing task is out of reach of the most modern artificial intelligence[1], the choice may still be in favour of the computer because software cost less than people.

The user of a filtering program is, inially at least, under pressure to both continue normal processing and provide instructions for the filter. For this reason the interface between the user and the program should be as simple as possible. Current products for the filtering of electronic mail and Usenet News employ everything from explicit programming to a simple selection of characteristic examples.

A filtering program is not required to go all the way and perform at level of the human user. Usually it will be sufficient if the filter is able to recognize what is routine and do a gross selection. As with most tools, a specialized skill is more useful than just being able to nibble at this and that.

When electronic messages such as email and Usenet News are to be filtered by a computer program many users and participants hesitate at the concept of

[1] AI systems can be, and are being built that perform quite well in narrow domains ([7, 3, 20]). Outside these domains the systems perform poorly, if at all.

allowing the computer to remove potentially important information or otherwise censor the information directed at them. But there is no need to reject or hide anything; a simple rearrangement of the order in which new items are presented is usually adequate.

Human beings adapt very well to their environment, be it an arctic tundra or an information super-highway. We form our own concepts and make decisions on the basis of the information available. We create inferential rules that govern our behaviour based on small cues. Most of the time we are not aware of these cues and form our decisions unconsciously. Therefore the task of an information filtering program is not only to do basic categorization, but to provide cues for the human eyes and ears which empowers the mind and allows fast, correct and intuitive decisions with a minimum of cognitive effort.

The preferred properties of a message filtering system outlined so far can be summarized as:

- small but competent
- easy to configure
- messages are sorted, not rejected
- it provides the user with clear perceptual cues

1.2 The purpose of the IntFilter project

The purpose of the IntFilter project was to investigate techniques applicable to the filtering of electronic messages.

While the goal includes all form of computer-mediated communication, it was early decided to restrict the project's attention to messages from Internet Usenet News, a global bulletin board system. It was expected that many subscribers of Usenet News share their opinions on a considerable subset of the messages, something which appears unlikely for email. Usenet News is also a formidable source of electronic messages (approximately 20 Mbytes each day) and it is trivial to create a situation where the amount of input exceeds the capacity of any human. As opposed to email, most subscribers feel that they can afford to miss News. A lot of people desist from reading News at all, simply because there is too much of it.

1.3 Background

This section presents a number of areas and techniqes relevant for message filtering. One particular interesting distinction is that between information retrieval and message management. The two concepts overlap in several places but are distinctly different in some important properties.

1.3.1 Retrieval

The nature of computer-mediated communication is that messages are created, copied to their destination and stored there until the receiver explicitly deletes them. This abundant storing of copies creates an accumulation of messages in whatever message system is used. For example articles in Usenet News are stored in thousands of copies across all News servers; email is sometimes stored by the sender and by all recipients; systems for group communication maintain central or distributed databases of the messages.

With all these messages being stored in a multitude of places it appears possible to view daily message processing as performing a set of operations on the result of a database query. In its simplest form the query can be informally specified as

> *'Select all messages not marked as read from my subscription set, order by arrival time.'*

The qualifier *from my subscription set* is, in the context of Usenet News, the set of newsgroups the user subscribes to. Without this restriction the hypothetical query would be directed at all unread messages in the News system, an unwieldy amount. The qualifier can henceforth be assumed.

Actually, the whole issue of message filtering can be captured in the expressiveness and semantics of the two verbs *'select'* and *'order by'*. We would like to be able to say:

> *'Select all* interesting *messages not marked as read, order by* importance.*'*

Naturally we are forced to yield a definition of *interesting* and *importance* in terms which the computer can understand. The most common technique to define *interesting* today is in the form of a list of keywords or phrases, while the term *importance* is defined as the number of matching keywords in a message[2].

Information retrieval is among other things also associated with a purposeful search for information rather than the routine dissemination of recent developments. The difference is for all practical purposes the same as the separate uses one has for a morning paper and an encyclopedia. In the former case, articles have structure in the form of threads of discussion, immediate and important information or regular briefings. The articles in an encyclopedia, however, are reference material. The content of each article is carefully prepared and relations between articles are not those of debate but of topical proximity and historical succession. This distinction is in a way inherited from the work in the information retrieval community which aim at finding encyclopedically prepared articles

[2] Keywords are usually weighed by discriminative power to avoid words present in all or most texts.

(scientific papers, books, WWW-pages) in huge databases. The legacy is sometimes a burden, when systems are applied to retrieve news articles or mail and are unable to properly preserve the temporal and threaded relations between the articles.

While the selection mechanism (eg keywords) works just as well for news clippings as for books, the impediment of the information retrieval view is that it prefers to see the result of a query simply as a set of documents. This view does not work well for messages which are part of on-going discussions, already arranged in newsgroups and often short (25–50 lines).

1.3.2 Incremental Retrieval

Anyone who is the receiver of a message flow will have to deal with the messages periodically or they will overflow the receptable. Returning to the imaginary query in the previous section, an information retrieval system could be configured to continously match all submissions to the database against the standing query. Matching articles are added to a temporary result which is fetched by or sent to the user at a convenient time.

Incremental queries allows the user to have an immediate response at the expense of extra storage in the database. Systems like SIFT [30] and BORGES[3] perform in this way; SIFT sends a regular result to its subscriber via email and BORGES prepares a WWW-page.

1.3.3 Selecting Rejected Messages

Some simpler filtering systems work by identifying messages to delete. Rules, keywords or whatever matching mechanism is used, serve only to pick out unwanted material. The rest is assumed to be readable. While censoring the user's messages this technique has been common in News clients for a long time under the name of *'killfiles'*. The point is of course that the user in one way or another specifies how the filter can recognize unwanted messages.

Since the filter prevents messages from reaching the user it can be hard to detect classification errors. On the other hand, messages that are not covered by the undesirability definition will be presented and this allows for serendipitious discoveries, ie finding out about something you didn't know you were interested in.

1.3.4 Selecting Preferred Messages

In this filtering technique, rules, keywords, or whatever is used to pick out articles which are predicted to be of interest to the user. Messages not selected are

[3]http://borges.compapp.dcu.ie

deleted.

This strategy may be useful if one is monitoring a specific set of topics. Since non-matching articles are deleted serendipitous discoveries are less likely to occur. Misclassifications may also be difficult to identify.

1.3.5 Order by Importance

The next step up from systems that perform a binary classification, ie if the message is allowed *'to be or not to be'* read by the user, is to place each message on a scale. The user expresses an opinion about the article in the form of a rating. The rating is often a point on a discrete scale ranging from bad to good with a few intermediate steps. Working from the examples rated by the user, the filter then tries to predict which rating the user will give to the unread articles. The prediction is usually a real number which makes it easy to sort the unread articles in order of predicted interest. Some systems allow the user to place a threshold against the list and hide the articles that fall below the threshold.

While this mechanism has been pursued in some systems (GroupLens [25], Browse [12], NewsWeeder [18, 19], and Newt [27]) a strong argument against it is that there is no general meaning attached to a point on a scale. Using a single dimension of interest may be simplifying things too far; what if the topic of an article is fascinating, but the text is technical, difficult or poorly written? How to value a text on a subject which lacks all appeal but is written by one of the favourite authors? Using a single dimension forces a collapse of properties which may be in conflict. Once a single point has been choosen it cannot reveal if this was an all mediocre text or a case where interests clashed. Information is lost.

Although most proponents feel that there is a noticable distinction between *'good'* and *'bad'*, using a single scale for entering a user's rating makes the filter's job much harder. The filter is given no clue to the features in the message that makes it good or bad in the eyes of the user. The features must be induced from the examples, a process which, if it is to be even moderately successful, requires a substantial collection of carefully choosen examples.

In favour of the single dimension approach is that ordinary people reads what they read one text at a time. So, if the goal of the filtering system is to arrange the texts in the order believed to be the most preferred reading order, then there is no escape from resolving conflicts between properties.

1.3.6 Order by Category

Rather than working with discrete or continous points on a scale, categorization of messages follows closely the way we normally archive information. A label is affixed to the message making it a member of a class of messages. The classes are weakly disjoint; a message may be a member of several classes but should in general be in only one class. This is the approach taken in the PEFNA News filtering system developed by us.

The PEFNA system allows the user to create a list of classnames. If the user comes across a particularly good example of a class the user assigns a class label to the article and continues with the reading. The article is stored on disk.

Before the next Usenet News session, a filtering agent is run to assign distances between each unread article and each user defined class. This is done using a standard information retrieval function, the Cosine measure [26]. The measurements are stored in the user's personal database.

As the user reads News once again, the newsreader (also developed by us) sorts the unread threads in each newsgroup according to three values:

- The major class label of the thread.
- The average proximity between the thread and the major class.
- The estimated uniqeness of the major class.

Note that the reader sorts *article threads* rather than articles. The reason is of course that threads represent ongoing discussions and these should be followed in temporal order.

The order of the unread threads follows the order in which the user has arranged the class labels. There is no inherent assumption of good or bad in the system, the user must make this distinction (if so inclined) by creating the appropriate classes.

Our approach to the information filtering problem is similar to the information retrieval process, where the user has a number of long-term queries. Each query corresponds to a class the user has created. The document collection at any time, is the stream of unread messages provided by the subscribed newsgroups. This view has earlier been presented by Belkin and Croft [1].

Our work is similar to that of Sheth [27], Morita and Shinoda [24], and Yan and Garcia-Molina [29]. Like them, we have decided to use a simple statistical similarity measure for queries and documents.

1.3.7 Collaborative Filtering

Studies by Malone et al [23] present three aspects of filtering performed by humans, *cognitive, economic* and *social*.

Cognitive filtering is content filtering, ie the human reader understands the contents of the document and makes that understanding the basis for decisions of further reading.

Economic filtering evaluates the document in terms of the resources it will consume and occupy when read. For instance, the user's time will be consumed, attention will be occupied and disk space may be allocated to store the document for future reference.

Social, or collaborative filtering is when the user values or is influenced by the opinions of other people. These other people may be editors and moderators, (un)popular authors or simply fellow readers. The idea behind computer-mediated social filtering is that a group of users submit their opinions about articles to a database. The computer, in turn, perform what in its simplest form can be described as compu-dating, ie matching people with similar interests in the assumption that people who have agreed on articles in the past will agree in the future.

The computer's role in collaborative filtering is to collect opinions from many readers, allow users to take advantage of these opinions and to use the correlations to provide ratings for unread articles.

For the cynic, efficient collaborative filtering resides upon two assumptions; that others have rated the message already and that it works better the more you conform to the majority.

Obviously, someone must be the first person to rate a message. Subsequent readers may add ratings of their own, but as the message gets older the incentive to rate it decreases. Rating the message for the benefit of others is meaningless for the last reader. In general, fast and early readers will have to do a lot of rating while receiving little or no support themselves. It follows that these persons either will refrain from submitting ratings or that their input will guide the selection of the majority of the subsequent readers.

The reason why collaborative filtering is feasible in spite of the cynic's view is that with a *sufficiently large group* there will be enough variation in who reads a particular message first to give everyone support over time. There will also be enough participants to enable almost everyone to recognize a subgroup matching their own taste.

A reasonable compromise between pure collaborative filtering and personal filter systems is to allow for the exchange of user profiles. That way the individual still receives filtering support from a personalized service, but is given the opportunity to import queries or profiles from other users. This removes the frailty of the first assumption (someone else has already rated the message) since the first reader will always be the computer. It also moves the altruistic nature implicit in collaborative filtering from articles to generalizations of articles. Conformity is also less of an issue, since each profile is shaped after its user. Software can perform the matching and brokering of profiles without having the user feel guilty for not providing enough feedback to the community.

1.3.8 User Input Stress

Work on filtering systems invariably runs into the requirement that the user provides some form of feedback to the filter. This feedback demands extra cognitive effort from the user and since the goal of the effort is to reduce workload, this feedback should be kept as simple as possible. The problem influences the way that user interfaces are created; when the user needs to communicate with the filter the process should be expedient and intuitive.

Kristina Höök at the Swedish Institute of Computer Science has addressed the problem of how to present an intuitive and workable user interface to the complex interiors of a software system [13, 11]. Metaphorically, the glass box is the transparent user interface which presents and allows the manipulation of simple concepts and structures. The system's internal complexity and actual machinery is hidden from the user in a black box.

The need for simplicity explains in part why the discrete rating approach is so popular: the user only has to press one of five buttons. Other workers (eg Morita et al [24]) have attempted to infer the user's interest from the time spent examining an article. While it may be possible to detect that some articles suffered an early rejection (the user did not scroll through the entire article, only 25 % of the lines were on the screen long enough to be read) this is a risky business. The filter can never be sure that an exposed article is actually being read since there is an infinite number of reasons why the user *is not* currently reading the article.

Chapter 2

The history of the IntFilter project

2.1 Formation of the project

A number of workers has been involved in, or associated with the IntFilter project. A rough sketch of their involvement is shown in figure 2.1 on page 12.

Several people took part from the outset, and a core group consisting of Daniel Pargman, Jussi Karlgren, Ann Lantz, Olle Palmgren and Kia Höök set about gathering related research and discussing the problems involved in filtering.

2.2 Ghosts

Background research continued and during the fall of 1993 Palmgren and Pargman started to build a prototype system they called GHOSTS. This was a system based on the newsreader NN and the design featured several components: a filtering news client, an email filter, a rule editor and a newsgroup editor. It was implemented in C and C++ and slowly grew larger and larger.

GHOSTS was taken to the level of carefully guided demonstration in the first months of 1994. At this time both Palmgren and Pargman were compelled to direct their attention elsewhere in order to complete their exams before the end of the spring semester. Pargman continued in close association with the IntFilter project but lacked the resources necessary to continue coding. The development of GHOSTS ceased.

2.3 Discussion

While Kilander completed his dissertation in May 1994, debate ensued within the core group. Conflicting interests drew in different directions while there was still hope that Palmgren would be able to complete GHOSTS to a level sufficient for user testing.

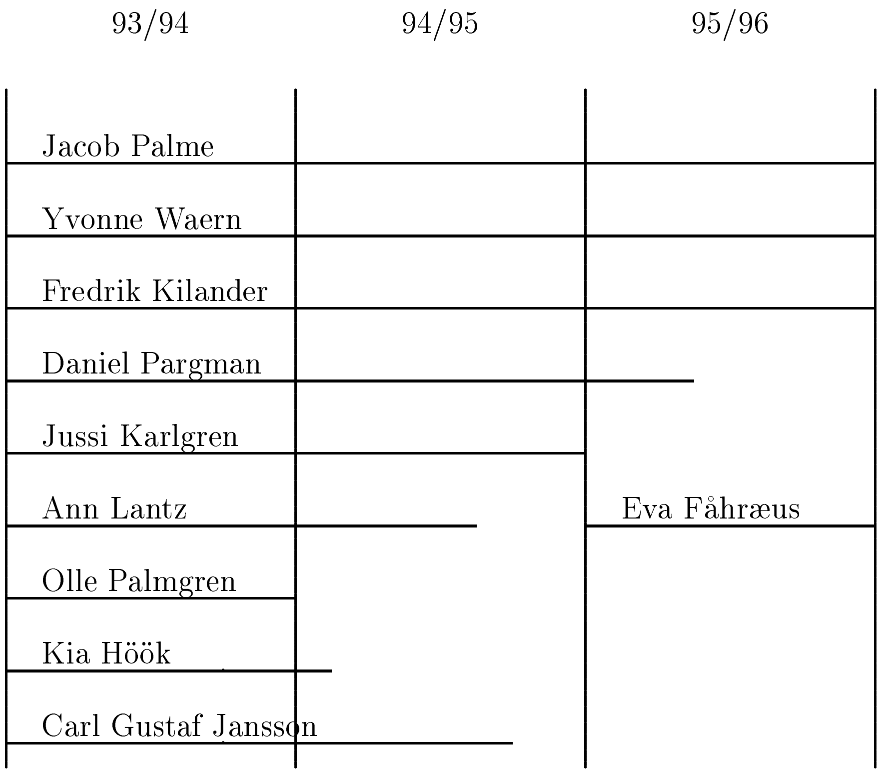

Figure 2.1: Workers in the IntFilter project 1993–1996.

The group agreed upon a number of key factors of the filtering problem and a filtering software tool. The tool would have to operate in close cooperation with the user, raising a requirement for a well developed user interface. The user should also be given enough information from the filter to be able to understand how the filter had operated on a particular message.

However, the working group had not resolved the issue of how articles were to be measured by the filter, ie which machine-readable features that should be the atomic functions on the filter's decisive expressions. Everyone understood clearly that there were many possible measurements, eg the number of words, the proportion of cited text, genre classification or the presence of selected keywords, but there was no good understanding how these related to human judgement. Lantz was already underway to tackle this problem from a psychological and behavioural view but the results so far were inconclusive.

2.4 Ghosts reevaluated

By August 1994 Kilander began to take active part in the project. His first task was to examine the GHOSTS system in the expectation that he would be able to continue development. As it turned out GHOSTS was a remarkable piece of software, but not very well suited for research purposes. Documentation was scarce and mostly missing. The package required non-standard libraries and tools (read *hacks*) to compile and relied on the presence of a complex newsreading system (NN) which was not installed on the available host computer.

Furthermore, the basic filtering mechanism in GHOSTS was a forward chaining production system with dubious properties. The rules would fire indeterministically, ie the conflict resolution strategy was random selection. Rule antecedents could contain both classification operations and destructive message handling. The combination made it very difficult to ensure that a rule set did not accidentally remove a message. To create and maintain a developing set of rules under these circumstances was unappealing and the decision was made to abandon GHOSTS.

2.5 Scripts

In order to examine the efficiency of naive filtering, Kilander wrote a small system called SCRIPTS. It used unix shell-scripts, standard unix utilities *(sed, grep, wc, comm, awk, sort)* to fetch articles from a news server and compare the number of words common to a set of stored example texts. Only words five characters or longer were counted and the number of lines of original text was weighed into the final result. The output consisted of a HTML-page listing article references in order of descending weight.

From this simple system a number of important points were learned. First, it worked! It was not terribly accurate but for the only newsgroup and topic selected in the short trials the most relevant articles were usually found in the top ten percent of the output list. Another point which was established was that the five character limit was too narrow. It did not allow for acronyms which on most occasions are very related to the target topic. Finally, the system was unwieldy and inefficient; there would be no cheap victories won by slow programs[1].

2.6 Ellemtell

In late 1994 Lantz and Kilander approached Hans Öhrvall on Ellemtell to inquire into the possibilities to do a field study at their site of a filtering system yet to be developed. While the needs of Ellemtell did not exactly correspond to the interests of the IntFilter project it was agreed that techniques and concepts could be tested, when available.

The members of the core group, now consisting of Lantz, Kilander, Karlgren and Pargman, continued to debate the possible features and mechanisms in a filter. Kilander's SCRIPTS system had demonstrated that simple systems could provide a benefit and as frustration over the lack of a proper prototype system accumulated, the group reached the firm understanding that it was not necessary to have a totally competent filter; a little help would be a lot more than none at all.

2.7 Lantz's user studies

Lantz undertook two studies in which she interviewed subjects on how they decided to read or not to read a news article. The first study was performed in late 1993 [21], the second, complemented by a questionnaire, in 1994 [22].

Essentially, she found that the main features of the articles used by the subjects to form their decision was

- keywords in the header's subject line
- keywords in the body
- genre
- author
- length

[1] Applying system programs like awk, sed etc from a shell script requires that separate processes are created for each. To measure a single article usually required 10–15 processes run and the delays quickly added up.

- thread position, and
- a body inconsistent with the subject line.

In addition, almost all subjects felt that filtering would be useful in Usenet News.

2.8 Algorithmic studies

Meanwhile (late 1994) Kilander had started a search for algorithms which could be used to compare News articles against keywords or prototypical texts. The results are presented in [14] and the conclusions were that the established Cosine measure (Salton and McGill [26]) was preferred over any ad hoc solution. Lang reported [18, 19] slightly better results using *Minimum Description Length* (MDL), an information theoretic measure, but MDL was not choosen because it is complicated and difficult to thoroughly understand.

2.9 A new prototype

As the pressure to produce a working prototype increased, Kilander pursued the search for algorithms no further and began designing a filter system. Two major issues influenced the design. The filter should have a modern graphical user interface and the processing of articles should be efficient. Other considerations involved how the filter tied in to the normal reading of Usenet News. Proxy news servers (as had been the case in GHOSTS), modified readers and even servers were contemplated.

In the end, the design settled on a two-component system. A News client was to be written using the Tcl/Tk scripting environment. The client would offer simple news reading operations and provide the user interface to the filter component. The filter was implemented as a standalone (batch) program, written in C. The communication between the filter and the News client was in the form of a database where the filter stored information about measured articles. When the user read News, the client consulted the database to order the articles in a presentation sequence.

The separation of the filter from the client relieved the client from the burden and delay of filter processing. It also relieved the filter of an unneccesary portion of user interface routines, which in the case of the X Windows System is quite substantial. As it turned out, this was the correct choice, since the filter needed that runtime memory to store many lists of words.

Writing the client in Tcl/Tk enabled rapid prototyping of a modern GUI. In addition, the lowlevel NNTP routines could be borrowed from the arTCLs News client written by Mike Hoegeman[2].

[2] mh@awds.imsd.contel.com

2.10 Dissipation

In early 1995 the core group of IntFilter workers had already begun to dissolve. Pargman had been accepted to Linköping University in August 1994 and while attending meetings his association with the IntFilter project gradually weakened. In Febrary 1995, Lantz was given a position at the department of Psychology at Stockholm University and took the opportunity to pursue her dissertation. Karlgren was involved in various other projects and finally left for New York in the summer of 1995.

Discussion continued in the administrative group consisting of Jacob Palme, Yvonne Waern and Kilander. Lantz and Pargman attended these meetings during the first half of 1995.

2.11 The prototype emerges

The prototype filtering system was given the name PEFNA *(Private Filtering News Agent)*. Originally scheduled for July 1995, the prototype was not ready to be given to users until December that year. The reason was that changes in the way user feedback was to be given required portions of the code to be rewritten several times.

The first design specified that the user classified articles with a point on scale of attractiveness. Five buttons labelled -2, -1, 0, +1 and +2 were placed beneath the text. If the user felt inclined to rate the article a button was pushed using the terminal's pointing device. The article was then saved under a corresponding label where the filter component could get at it.

The filter was then supposed to compare unread articles to the rated examples using the Cosine measure. It would then weigh the distance with the label and the client could use the weight to sort the presentation sequence accordingly.

However, this did not work very well. The lack of rigid semantics in the numeric scale made it difficult to judge why a certain article was given a specific rating. Since rating was performed by the filter agent, a separate program, the client could not easily motivate the decision. The filter could possibly store motivations with each article but that would require massive amounts of disk space, something which could not be accepted for regular users. While a Cosine distance metric is a simple number, the filter was designed to be able to include more complex measurements.

2.12 A new member

By September 1995 Eva Fåhræus joined the project to pick up the work on psychological and behavioural research previously done by Lantz. As Fåhræus familiarized herself with the ongoing work she suggested that the discrete rating numbers in PEFNA should be replaced by arbitrary labels. In the following discussion Kilander and Fåhræus realized that there was no need even to distinguish these labels in terms of weight. Articles were simply assigned to their closest label and the user could arrange and rearrange the order of the labels at any time.

As a result, the discrete number buttons were removed and replaced by a tear-off menu. The user reads an article and if the article is a good example of a category of texts the user assigns a label to the article. Labels are essentially names for user-defined categories. The categories are defined by the example texts placed in them. The filter does not weigh the categories, but simply measures the distance between each unread article and the categories. The measurements are stored in the database and the News client sorts the presentation sequence in the same way as the user has arranged the labels.

2.13 First user trial

In December 1995 PEFNA was placed into the hands of three users. The subjects were asked to consistenly create categories (labels) and categorize articles. The subjects did not at this time use the filtering component.

The purpose of the study was:

1. To collect a large number of texts categorized by real users in a real situation. Close to 1000 articles were collected in this way.

2. To augment the understanding of how users spontaneous filter their information.

Three subjects used the PEFNA prototype to read and classifie texts in categories of their own choice. After the reading the subjects were interviewed about how they choose what to read and how to classifie the texts. For a detailed description of this part of the study, see [9].

2.14 Prototype refinements

Working against the collections provided by four users of the PEFNA system, Kilander examined various techniques to improve the classification of unread articles. The idea was that certain words were unique in almost every category and therefore should be given higher weight than words present in several categories. Using the information entropy measure

$$\sum_{i=1}^{i=N} P(w|C_i)log_2(P(w|C_i))$$

the discriminative power of each word was established. Several combinations of reweighing the words with this information before applying the Cosine measure were tried, but the results were initially contradictory since no single variant gave a consistently higher accuracy for all users. Further examination [16] revealed that term weighing based on information entropy provides more accuracate classification than just term frequency counts.

The experiments reinforced the confidence in the Cosine measure since with it, the probability of an unread article being assigned to the correct category were several times higher [3] than if a category had been choosen randomly.

2.15 Second user trial

In june 1996 the subjects in the first trial were approached again and asked to use the PEFNA system once more. This time they were to recreate categories important to them but only store relevant and typical examples. They were also to use the filter component.

At this point PEFNA had seen further revisions and refinements. The categories to be used for article labelling could now be selected for each newsgroup. This enabled a more focussed presentation and avoided the confusion that arose from articles being assigned to a category only relevant in another newsgroup.

[3] In this particular case, 5–7 times higher.

Chapter 3

Contributions

3.1 Comparisons of two document similarity metrics

This section is a summary of [14].

In order to estimate the similarity between a query and a collection of documents in a database, a number of metrics have been applied and tried in the area of information retrieval. The metrics are easily adapted to provide a numerical distance between an unread message and a set of long-term queries.

For the purpose of developing a message filter application, two particular metrics were empirically evaluated. They were the the cosine measure by Salton et al [26] and sub-string indexing by Morita and Shinoda [24].

The cosine measure has received its name from a trigonometrical analogy: the angle between two vectors in N-dimensional space is always measurable across a 2-D plane. When the vectors point in the same direction the angle between them is $\phi = 0$. When they are perpendicular to each other the angle between them is $\phi = \pi/2$. Taking the cosine of the angle normalizes the range between the extremes into

$$\{\cos(\phi = \pi/2) = 0 \ldots \cos(\phi = 0) = 1\}.$$

In this *vector-space* model, the desired result is directly available by taking the cross product of the vectors. For texts, each document or query is represented by a vector of weighted terms:

$$V = \{wt_0, wt_1, \ldots, wt_n\}$$

where each wt_i consists of two parts, the numerical weight and the term. The term is some component feature of the document, usually a string containing a word or a phrase, e g ‘computer’ or ‘concept graphs’. In addition, the term may be any boolean feature of the document.

Given a candidate document d and a stored document q, the cosine similarity was computed in the following way:

$$sim_{q,d} = \frac{\sum_t w_{q,t} \cdot w_{d,t}}{\sqrt{\sum_t w_{q,t}^2} \cdot \sqrt{\sum_t w_{d,t}^2}}$$

where $w_{x,t}$ was the number of times term t occurred in document x. The resulting term, $sim_{q,d}$ would be 1.0 for identical documents and 0.0 for documents with no common terms.

When the frequencies of isolated words are used to describe a document, no memory of ordering remains. For example, a document like

'A horse is better than a car.'

is regarded as identical to

'A car is better than a horse.'

although they say different things. Additionally, the sentences

'Take off your shoes.'

and

'Remove thy footwear.'

are regarded as completely dissimilar. In summary, the cosine measure is a *statistical* comparison which draws its strength from the style and vocabulary commonly shared by texts on a particular topic.

Using phrases instead of single words reduce the chance that words from different contexts support a match but also requires in a higher degree that similar documents actually share wording. The first example above will still yield a high similarity since both sentences share the two-word phrases

'a horse', 'is better', 'better than', 'than a', 'a car'.

Comparing phrases rather than single words will do nothing for the third and fourth examples which requires semantic knowledge to judge similarity. The analysis is quite complex. Not only must the single word 'remove' be matched against the phrase 'take off', but it must also be inferred that 'take off' is used in a context where it means exactly that, rather than 'depart' or 'aeronautic manouver'. The words 'shoes' and 'footwear' do not have any simple equality either; while the imperative in the two sentences collapses the generalization to the particular situation of the utterance, 'shoes' can be seen as an instance of 'footwear' and may be equally replacable by 'sandals', 'boots' or 'ankle chains'.

The sub-string indexing similarity measure is simply the proportion of terms shared by two documents:

$$sim = \frac{2 \mid V_q \cap V_d \mid}{\mid V_q \mid + \mid V_d \mid}.$$

	Words	Case	No case
Cosine	1	0.264948 (0.115154)	0.307219 (0.117363)
	2	0.073103 (0.114970)	0.073977 (0.115743)
Substring	1	0.259642 (0.086204)	0.282888 (0.084398)
	2	0.073875 (0.098067)	0.077426 (0.097920)

Table 3.1: Average similarities (and standard deviation).

3.1.1 Experiments

A sample of articles consisting of 101 articles was taken in non-sequential order from a newsgroup with a number of ongoing threads and heterogeneous postings (alt.games.doom). One article was selected to act as the query. All articles in the sample (including the query itself) were then compared against the query and their similarity registered. The message headers were included in the comparison.

Two dimensions were tested for each similarity measure: document feature and letter case significance. Document features were terms, constructed from single words or two-word phrases created by combining adjacent pairs of words. Table 3.1 summarizes the average measurements for the eight test combinations. The cosine measure on single word features gave the highest average similarity while the sub-string indexing metric had a steeper similarity slope than the cosine measure as documents separated.

3.2 A survey of filtering systems

This is a summary of [15].

A number of workers are providing or researching mechanisms to filter Internet messages. The most common target of these efforts are directed at Usenet News. Inspired by a compilation of references to message filtering systems provided by Doug Oard[1] eleven systems differing in approach and ambition were compared.

3.2.1 Summary of examined systems

SIFT

The Stanford Information Filtering Tool (SIFT) [30] is a service provided over the World Wide Web (WWW). Users access the filter by submitting one or more *subscription profiles* to the service using a WWW-browser with forms capability. Each profile specifies a number of keywords using two alternative matching strategies; a boolean conjunction of (optionally negated) terms, or, a set of keywords and a weight threshold. The profiles are matched against the entire news-feed received at Stanford. Matching articles are sent to the user by email.

[1] http://www.glue.umd.edu/enee/medlab/filter/filter.html, oard@glue.umd.edu

NewsWeeder

NewsWeeder [18, 19] is a system in experimental use at Carnegie Mellon University. A special WWW-browser serves as user interface. The user visits each newsgroup of interest in turn. For each group, the user rates interesting articles using a number in the range 1–5.

The system collects the rating information and presents the overall rating averaged across all other NewsWeeder users. As such the systems provides collaborative filtering. NewsWeeder can also provide a *prediction* of the score, computed with machine learning techniques from the user's profile of past ratings.

Browse

Browse [12] is a reader for Usenet News with a neural network model of the user's interests. Each message presented to the user is classified as either being accepted (read) or rejected (not read). The first 300 words of the message (filtered by stop list of common words) are used to represent the features of the message.

Browser rates articles and can be used in one of three modes: *all articles, selected ranked articles* and *augmented keyword search*. The *all* mode simply shows all articles in chronological order. The *search* mode employs the neural network model to rank each article and filter them against a threshold value (settable on the GUI). In the augmented keyword search mode the user can type in keywords that increase the acceptance or rejection of an article.

NewsClip

NewsClip[2] is a programming language focussed on Usenet News filtering. The compiled filter operates offline by comparing the user's *.newsrc* file with the contents of the news server. It applies its rules to all unread messages and rewrites the *.newsrc* file in such a way that messages that are filtered out are marked as already read. As a result, the user is free to employ any conventional newsreader.

Lurker

Lurker is a personal retrieval program written in Perl by Vince Taluskie[3]. In short, it can be seen as a way to periodically perform *grep* on a Usenet News server and have the selected articles stored on disk or sent to the user as email. Lurker is similar to SIFT in spirit.

[2] http://www.clarinet.com/newsclip.html
[3] taluskie@utpapa.ph.utexas.edu

SMART

SMART [4] is an old and ambitious software package. It is basically an information retrieval system, with its functions being open to specialized applications. The general system can be used with the newsfeed as the message store, or its routines can be built into a special-purpose browser.

GroupLens

GroupLens by Resnick et al [25] is a system for distributed, collaborative filtering of Usenet News. Unlike most other systems who regards the News systems as an opaque message store, GroupLens ties deeply into the infrastructure of News and exploits its mechanisms for its own purpose. Briefly, GroupLens allows users to rate articles as they are read. The ratings from each user is then stored in a rating server, a *Better Bit Bureau.* Just as in the News system, there can be many servers across the Internet and users subscribe to the server nearest to them. The rating servers use a distinguished newsgroup in the ordinary news system to exchange ratings.

The synergy achieved by having articles rated by many readers is that the rating servers can correlate users with similar preferences. This allows the rating server to hypothesize the rating a user will give to an unread article.

Borges

The BORGES project[4] is a European effort to build a News and WWW filtering system using a WWW-browser as the client and the SMART system as the retrieval engine.

BORGES consists of a WWW-server interfacing a SMART database. The database stores (or can access) user profiles and News articles. The user submits standing queries (as with SIFT) and then retrieves the result in the form a WWW-page updated each day.

The interface for query submission allows the interactive refinement of keywords and phrases based on polysemous words. That simply means that the interface highlights words and phrases that may have multiple meanings, like the word *'bank'* which may refer to a financial institution, a subset of a computer's memory or the sloping sides of a riverbed.

InfoScan

InfoScan[5] is a commercial system which in essence presents a two-dimensional view of how a number of locally stored texts relate to a particular query.

[4] http://borges.compapp.dcu.ie

[5] http://www.machinasapiens.qc.ca/english/infoscanang.html

The texts (which must be files on the local file system) are arranged into what InfoScan calls *sources*. These sources are then further aggregated into *collections*. The user maintains a set of *filters* where each filter is a set of keywords. When a filter is combined with a collection the result is a query, and infoscan graphically visualizes the relation between the filter and the texts found in the collection.

RAMA

RAMA [2] is a retrieval system where a query server accepts queries via email. It then searches through a collection of News, FTP and WWW sources it knows about and returns the result via email. The user may have standing queries. The server keeps track of users, queries and the time answers were reported to avoid serving the same answer twice. The matching algorithm uses the Cosine measure and simple pattern matching (agrep).

Pefna

PEFNA is a rating system, developed by Kilander, Fåhræus, Lantz and Palme in the IntFilter project[6] at the department of computer and systems sciences (DSV) at Stockholm University.

PEFNA consists of a Usenet News client and a classifier agent. The user runs the client to read Usenet News from a standard news server. The client allows the user to create and maintain an ordered list of freely named categories. An article which is judged by the user to be a good example of a particular category is saved under the name of the category. Over time, the user accumulates a number of categories and one or more example articles in each.

The agent compares the user's *.newsrc* file with the contents of the news server and retrieves any new articles. Each article is then compared against the examples in each category and a list of Cosine distances is built and saved.

When the user launches the client the next time, it examines the article ratings provided by the agent. In each thread, one category is selected as the most representative for the thread. The list of threads presented to the user is then sorted to follow the sequence in which the user has arranged the categories.

3.2.2 Classification

The purpose of the classification mechanism is to assign each article to a class, or more generally to a point on a scale of attractiveness. Methods that use boolean expressions commonly reach a point where a binary decision is made; the article is either interesting enough to be recommended or it is not. Other classification schemes allow a softer gradation, allowing the user to modify the cutoff threshold at will.

[6] http://www.dsv.su.se/~fk/if_Doc/IntFilter.html

Boolean expression	Cosine measure	MDL	Neural network	User correlation
NewsClip	NewsWeeder	NewsWeeder	Browse	GroupLens
SIFT	SIFT			
Lurker	SMART			
	RAMA			
	InfoScan			
	BORGES			
	PEFNA			

Table 3.2: Filter systems categorized by classification method.

Percentage	Alternative
80%	Subject line (what I thought the article was about)
40%	The genre the text belongs to
30%	Where the article is placed in the thread
25%	(Articles are read) in the presented order

Table 3.3: How subjects choosed an article to read.

Five different types of classification schemes are represented among the examined filters:

- Boolean expressions.
- The Cosine measure.
- Minimum description length (MDL).
- Neural network modelling.
- Correlation of user profiles.

Table 3.2 shows the 11 systems and their method of classification.

3.3 Criterias for filtering

Ann Lantz performed two studies [22] to investigate how readers of Usenet News evaluate and classify an article. The first study used four subjects who were subjected to monitored reading sessions and interviews. The work resulted in a number of message properties which were submitted to a larger audience in the second study in the form of a questionnaire. Twenty subjects responded.

When asked how they choosed an article to read, 80 percent answered that they looked at the subject line (table 3.3). Many also regarded the genre, or style, of the text as a factor in the decision.

Percentage	Alternative
80%	Subject line
35%	The length of the article
30%	The amount of quoted text in the article
25%	The author of the article
25%	If the article was well written and structured
25%	The genre the text belonged to

Table 3.4: How subjects choosed an article to avoid.

To avoid an article, more factors came into play (table 3.4). Again the subject line was rated as most important by 80 percent, followed by a fairly even distribution (25–35 percent) between such properties as the length, the genre, the amount of cited text, the author, and the structure or the article.

Nearly all subjects (95 percent) felt that there was a need to filter information in Usenet News. The subjects were asked what they could accept in order to accomplish mechanical filtering. The responses were that

- 65% accepted selection rules
- 45% accepted keyword search in the subject line
- 45% accepted that the system had access to their user profile
- 35% were willing to supply a binary rating
- 15% were willing to supply a rating on a scale of ten steps.

Only 5 percent preferred that an automatic filtering systems worked without intervention, while 55 percent wanted to have an active dialogue with the filter.

3.4 Message properties as perceived by users

This is a summary of [17].

Students at the Department of Computer and Systems Sciencs (DSV) were given the task of writing a short essay on the problem of using machine learning to classify electronic texts and messages such as electronic mail (email), Usenet News postings or World Wide Web (WWW) pages. The purpose was to collect information from real users on what they perceive to be important properties in electronic messages. For the more than one-hundred students the essay was a small part of a wider examination. Fifty-four of them elected to adress the issue.

The authors of the essays were all students at DSV, at the time halfway through their second year of a four year curriculum. With all likelyhood, all of them used WWW and email at DSV, some of them also Usenet News. A non-neglible fraction

Property	Count
Keywords	40
Sender	37
Subject	30
Length	21
Date and time	21
Categories of senders	13
Recipients	11
Relations to other texts	9
Semantic text analysis	6
Readability analysis	6
Keyword synonyms	5
Attachments	5
Language	4
Priority	3
Newsgroup	3
Layout	2
Negative keywords	2
Author's keywords	1
Nof readers	1
History of sender	1

Table 3.5: Message properties suggested as important

indicated in their essays experiences from working in other environments where email played an important role. The majority of the authors were 20–30 years old. Female authors provided nine of the essays.

Table 3.5 shows a summary of the properties suggested by the students. Most of them had approached the task with email in mind and listed the sender first, followed by the subject. This likely reflects the way most people peruse their mail. Many also noted keywords in the body of the message as an important property, but as an afterthought or after having spent time thinking further about the problem. The appreciation of length appeared to be a very individual issue: some authors apparently used length in conjunction with the sender to hypothesize the contents of the message (again, mostly for mail). Time was also an important factor for many; essentially the discussion revolved around deadlines, events and the time a message was allowed to be stored but not read.

3.4.1 Sender

3.4.2 Senders and recipients

Several authors indicated that senders should be categorized for easier processing and developed the concept in several ways:

- Senders can be persons, organizations or mechanisms.
- Senders have properties.
- Senders can be categorized.

Senders that are persons are usually known to the user. If not known personally, the social and organizational relation between the sender and the user is often already established: supervisor, co-worker, employee, client, customer etc. Some authors recognized that categories could overlap (your boss can also be your friend) in which case other mechanisms would have to be employed in order to determine which *role* the sender is having in the message.

A number of authors felt that knowing the group of recipients of a message was important. Knowing that others have recieved a message may influence decisions related to the message.

3.4.3 Subject

Many authors felt that the subject line was very important and argued that it would be searched for keywords or phrases. Some authors recognized that subject lines can be useful, but also devastating if their import is overly emphasized. Writing a good subject or title is difficult and sometimes willfully abused.

3.4.4 Length

Length was quoted by many authors as important for categorization but the opinions on its interpretation varied, apparently from their personal experiences. The basic evaluation of a message's length was that a longer message would take more time to read (economic filtering). Some authors argued that short messages usually were the result of errors or test and could therefore mostly be ignored, while others felt that the sender and subject determined the importance of short messages.

None of the authors suggested that the proportion between cited and original text was an important property, but neither did any one specify explicitly that length should be counted only as original content.

3.4.5 Date and time

Information about dates and times were considered to be important by many authors. Several suggested that important time information could be embedded within the message and should be identified and extracted. Surprisingly enough, some were apparently ready to discard messages about events past without reading them. At least one author suggested that unread email should be subject to aging, ie receive higher priority with time.

3.4.6 Comments

A number of authors suggested that a combination of techniques and properties should be used for classification. Considering that many of the authors wrote with their personal experience in mind, it should be safe to say that people use whatever information they have at hand and form heuristics around that.

On a more general level, the attention given to email by the authors appears guided by two things, the commitment expected of them and crucial time information, ie deadlines. Commitment is not easily quantified, but many cite a letter from their 'boss' as an example of something which is given the highest priority.

3.5 User studies

3.5.1 Think-aloud-study

In 1993 and 1994 Ann Lantz conducted two studies with the aim to se how experienced users of the Usenet News choose the information to read [21, 22]. In the first study she let the users think aloud while they were searching for the contributions they wanted to read. In the other study, Lantz sent an electronic inquiry to members in Swedish speaking news groups. The results show that the users usually made their choice on keywords in the subject line.

3.5.2 Field investigation

During 1995 and 1996 Eva Fåhræus conducted a user study [9] to find out the users opinion about the PEFNA filtering system and to search for possible improvements. At the same time the study produced a sample of classified texts to use as test material in the experiments with the filtering system.

The study was conducted as a field investigation. Three users used the PEFNA prototype to read Usenet News and placed the contributions in different categories or 'baskets', depending on content and user's interest.

The behaviour of the users and the result of their manual filtering were studied through interviews and then analyzed by inductive methods, based on Grounded Theory [28].

The goal was not to formulate a theory as such, but to try to get new ideas about how to support users in search of interesting texts.

The analysis lead to the following formulation of the core of the studied task:

The main story seems to be about how a user, by just glancing on a part of a text, surveys it and decides what it is about and if it is worth reading, now or later and if so under what circumstances. The decision is also whether to read more before deciding what to do.

This formulation can be translated into a more abstract level:

Perception of elements in a contribution give, if they are recognized, an association that results in a text classification which decides whether the text is saved for later use.

3.5.3 Generated ideas

Grounded on these core categories and inspired of ideas born during the interviews and meetings we formulate how the computer could support the reader.

The perception can be supported by the system presenting the elements in the contribution more clearly. The subject line and the keywords can be displayed brighter, in bold font or in different colours. Questions and answers can be shown together and perhaps in a different colour or font. Connections between contributions in a chain can also be shown graphically. By showing the complete contribution in a smaller scale, the system can quickly give an impression of the contribution's outline, format and size. These measures will make it easier for the reader to observe recognisable elements valuable for the classification.

To support the association the system could show the readers' baskets next to the contribution. The readers shall also be able to look into a basket and see the contributions already put there, reminding of the chosen strategy of classification.

Saving the contribution can be made simpler by direct manipulation, i.e. the user can click and drag the contribution over the screen and drop it in the basket.

3.5.4 Conclusions

The results from the field investigation confirm the conclusions drawn by Ann Lantz [22], i.e. that keywords in the subject line are very important as to how the users judge the contributions, and completes it with a more detailed image of how the users act.

Through this study we have received support for the assumption that the computer should be used to present the contributions in a clear way and that different readers have different wishes and needs. There are differences between different readers and the same reader on different occasions.

We have not got any support for the theory that even a *rather* good sorting of the contributions can be of great help. The study lasted too short a time to make the users really able to use the prototype efficiently.

The study does not give any reason to reject the filtering technique chosen but the users say they questioned the possibility to 'teach' the computer what is interesting or not interesting. Instead they suggested some criteria to use.

Filters can very well be used by groups in co-operation, by letting different categories of contributions be guided to different individuals watching different areas for one another. The idea of letting readers put a judgement on contributions, that others can benefit from (so called social filtering) is not supported in this study.

3.6 The PEFNA prototype

The Private Filtering News Agent (PEFNA) is a prototype system for filtering Usenet News. PEFNA consists of two pieces of software, a Usenet News browser (PFCLIENT) and a classifier (PFAGENT). The user runs the client to read news from a standard news server. The client allows the user to create and maintain an ordered list of freely named categories. An article which is judged by the user to be a good example of a particular category is saved under the name of the category. Over time, the user accumulates a number of categories and one or more example articles in each.

The agent runs as a batch program, preferrably in the morning hours. It compares the user's *.newsrc*-file with the contents of the news server and retrieves any new articles. Each article is then compared against the examples in each category and a list of distances is built using Salton and McGill's cosine measure [26]. The list is stored in a database in the user's home directory.

When the user launches the client the next time and visits a newsgroup, the client builds a list of threads for unread articles in the group. It also examines the user's database for the ratings provided by the agent. In each thread, a single category is selected and used to label the thread. The list of threads presented to the user is then sorted to correspond to the sequence in which the user has arranged the list of categories.

For example, if the user has placed the category 'Call for papers' at the top, then all threads closest to that category are sorted to the top of the list.

3.6.1 The article database

Practically all information needed by PEFNA is stored in a subdirectory in the user's home directory. This directory contains the article classification database, a log over the classification runs, the names and sequence of the user's preferred categories and a subdirectory tree holding articles which were selected by the user as good examples of each category.

The database is a formatted ASCII file. For each article information pertaining to the identity of the article is stored, along with other properties detected in the message. Of particular interest to PFCLIENT, the Usenet News browser, is the list if distances between the article and the user's categories.

3.6.2 The classification agent

The classification agent PFAGENT is written in C and covers approximately 7600 lines of code. It opens the user's *.newsrc*-file[7] and compares the read articles in the subscribed groups with the contents of the NNTP-server. Each unread and unmeasured article is fetched, and its similarity to each of the existing categories is examined.

The conceptual architecture of PFAGENT is that of a central message fetching engine and a collection of measurement routines. The message engine visits each of the newgroups to which the user is a subscriber and retrieves the article number (not the article itself) of each available article. It then passes the newsgroup name and article number to each of the measurement routines. Each routine may independently decide if it needs to retrieve the text of the article or not.

3.6.3 The presentation sequence

When the user commands PFCLIENT to visit a newsgroup, the subject, references and message-id fields are fetched from the server. The message-id is used to lookup the article's entry in the article database prepared by PFAGENT. The references and subject fields are used to tie together the articles in threads.

Within each thread the articles are ordered temporally, there's really no other convenient way if one wants to follow a discussion over time. Therefore, the filtering task boils down to arranging the threads. Since the user's categories are viewed and available from top to bottom in a menu, the threads are sorted to conform to that order.

Since many threads consists of a single article, these are very simple to assign. The thread's category is simply the category the article is most similar to. For longer threads the method of choosing a winning category is not that obvious so a simple voting scheme was used. Each article in the thread adds its similarity counts to each category and the category with the highest sum is choosen to represent the thread.

3.6.4 Experiments with term weights

While the cosine measure works quite well, it does lead to misclassifications. In order to evaluate the accuracy of the similarity measure, the core of PFAGENT was modified to select a number of testcases from a user's set of examples, and assign them to the most similar category.

[7] Actually, the file is called `.xnewsrc` to separate it from the test subject's ordinary news reader.

Three classes of term weighing measures were evaluated, as represented by a total of 12 different weighing methods. The base case was to assign the weight of a word from its occurrence count, ie the more it appears in a category the more important it must be. The first alternate case was the inverse document frequency, which estimates that the word frequency should be divided by the number of categories in which the word appears, but without further regard for how often it is represented in each category. The final, and most successful weighing measure was discriminative informativity, a measure adapted from the machine learning community. This measure penalizes words that tend to be evenly distributed across several catagories while favouring words which are good indicators of category membership.

In quantitative terms, the best case achieved when using the informativity measure was an average classification accuracy 7.6 times better than assigning the article to a category randomly.

Chapter 4

Summary and future work

The IntFilter project started out boldly, with many members and a promising future. Many of the initial ambitions, however, were abandoned for trivial reasons; as the number of active workers waned the wide selections of ideas was cut down by the necessity to choose.

After an abandoned software project and initial experiments, the PEFNA News filtering prototype was designed and developed. It allowed the user to create arbitrary categories of texts, and performed filtering by estimating the distances between the user's unread texts and the categories. Field tests of the PEFNA system revealed that it was able to perform as intended, but also that the structure of natural language made classification more intricate than was expected of a keyword search.

At the time of this writing, late 1996, the project is not scheduled for any continued activity. However, as a note to fellow researchers in the field we would like to ventilate a few ideas below, ideas which emerged when it was too late to reconsider the choices already made.

Work on the visualization of retrieved information is already being pursued in several places. The interested reader may consult [5, 6, 8, 10] for a comprehensive rundown on this fascinating topic.

4.1 Epilogue

A lot of things can be measured in a text but what is easily computed appears to have only a weak relation to how human readers estimate a text. The span between computer analysis and human judgement and intuition is still very wide.

What can be done is what almost everybody else in filtering are doing; to look at the words in the texts and make the assumption that two interesting texts on the same topic will share a significantly large vocabulary. This is a superficial comparison, which does not analyse or understand the text, and without the ability to perform analogy, inference or paraphrasing. Searching for keywords only covers a part of the human interest space, since we can't easily model such concepts as 'a letter from the editor', 'a funny thing happened on the way to Rome' or 'one more of these virus hoaxes today and I'll scream'.

What appears to be underestimated in the problem area of text classification and filtering, is the remarkable ability of ordinary people to make intuitive decisions based on whatever information is at hand. But why not use the computer to perform tasks it is good at, like visualization of a data space; to hide and expose information and place this magnifying lens in the hands of user?

To be specific, rather than having the computer make the decision if the message is worth reading, let the reader continue to make the decision, but have the computer supply the relevant information on which to ground this decision in a swift and timely manner. Replace the tedious list of subject lines with a threedimensional view of the entire newsgroup. Let the user select long messages, short messages, heavily cited articles at the twitch of a finger. As soon as a message has been read, allow the user to immediately select other texts close to the one just read.

This study has evolved to support the idea of graphical presentation as a means to help readers find interesting texts. The combination of the computer's possibility to sort and display information graphically and the human ability to recognize patterns and discriminate graphic properties will open new possibilities to help us take advantage of the information now sunk in the ocean of Internet.

Future work on this topic ought to focus on different ways to display the properties of a text that carry meaning to the reader and on the human ability to visually discriminate them.

It is however important to take in consideration that readers are different. Physiologically they have different ability to discriminate different kinds of perceptions and intellectually they find different properties in a text more or less important for judging the interest.

To test our conclusions it would be interesting to conduct a longer user test with a further developed prototype containing graphical presentation of different properties in texts. The prototype should be flexible so that the user can adjust it to his or her actual situation and preferences. If the study goes on for at least two months, preferably more, there is a chance that the user really adapts his/her behaviour to the tool and finds out the best way to use it. If different users, working under different conditions, take part in the test the flexibility of the prototype could provide a good picture of useful support functions.

The prototype could for instance have the following functions:

- The user can give keywords that are preferred or not preferred in texts. Preferred words are shown green, not preferred are shown red.
- The user can give maximum and minimum length of texts.
- The user can give maximum (or minimum) ratio of cited text.

- The texts are shown graphically as circles or cylinders with their size illustrating length, the colour telling if there are preferred or not preferred keywords.
- The user can choose to look closer to one part of the presentation, e.g. one or two categories.
- When pointing at a text symbol, the closeness to a category is shown by arrows and the layout can be examined by displaying the text in "stamp size" where the subject line is readable and the keywords shown in colour.

From any of these presentations it should be possible to click and read the text, to drag a text to a folder for later reading, to drop it in the waste paper basket or in a letter box to some other person, adding a comment.

Appendix A

Screenshots

The last pages of this document contains a number of screenshots from the PEFNA system.

A.1 The group selection page

The first picture shows a list of the newsgroup subscribed by the user. This is the initial display when PEFNA is launched. The user selects a newsgroup to read by clicking on the name.

A.2 The thread selection page

When a newsgroup has been selected and loaded, the thread selection page shows the list of available discussion threads. The display has five columns; the number of articles in the thread, the winning category label, the average distance to the winning category, three-character representation of the uniqueness of the winning category and the subject line of the thread.

The user selects an article to read by clicking on the line in the display, or one of the buttons in the control bar at the bottom of the window.

A.3 The article page

An article is displayed using different fonts for selected parts of the header, cited text and contributed text.

A.4 The category distance plot

This is a graphical representation of the distances between the articles in a single thread and the user's categories. In this particular example, there are four articles in the thread (shown as white discs). Although not captured in the picture, the mouse pointer is placed over the leftmost article, and the arrows emanating from the center of the plot show the relative distances between that article and the categories.

A.5 The example bin sequence editor

This is a popup dialogue which allows the user to arrange the sequence of the categories. This affects the order in which the threads are sorted in the thread selection page.

A.6 The bin membership editor

This popup dialogue allows the user to assign only a subset of the available categories to a newsgroup. Only the assigned categories are used when building the thread display for the newsgroup.

Bibliography

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

[2] Jim Binkley and Leslie Young. Rama: An architecture for internet information filtering. ftp: // ftp.cs.pdx.edu / pub / faculty / jrb / rama / rama.k1.ps, 1996. Computer Science Department, Portland State University, Portland, OR 97207.

[3] Ivan Bratko and Stephen Muggleton. Applications of inductive logic programming. *Communications of the ACM*, 38(11):65–70, November 1995.

[4] Chris Buckley. Implementation of the smart information retrieval system. Technical report, Dpt of Computer Science, Cornell University, Ithaca, New York, May 1985. http://cs-tr.cs.cornell.edu:80/TR/CORNELLCS:TR85-686?abstract=.

[5] K. Stuart Card. Visualizing retrieved information: A survey. *IEEE Computer Graphics and Applications*, pages 63–67, March 1996.

[6] Tiziana Catarci. Interaction with databases. *IEEE Computer Graphics and Applications*, pages 67–69, March 1996.

[7] W. Church, Kenneth and Lisa F. Rau. Commercial applications of natural language processing. *Communications of the ACM*, 38(11):71–79, November 1995.

[8] Stephen G. Eick. Aspects of network visulization. *IEEE Computer Graphics and Applications*, pages 69–72, March 1996.

[9] Eva Fåhræus. Intelligent filtering on usenet news a user study. Technical Report 97-003, Department of Computer and Systems Sciences, Stockholm University, 1996.

[10] Nahum Gershon. Moving happily through the world wide web. *IEEE Computer Graphics and Applications*, pages 72–75, March 1996.

[11] Kristina Höök. *A Glass Box Approach to Adaptive Hypermedia*. PhD thesis, Department of Computer and Systems Sciences, Stockholm University, 1996.

[12] Andrew Jennings and Hideyuki Higuchi. A personal news service based on a user model neural network. In *IEICE Transactions on Information and Systems*, March 1992. http://www.cse.rmit.edu.au/˜rdsajj/brow.ps.

[13] Jussi Karlgren, Kristina Höök, Ann Lantz, Jacob Palme, and Daniel Pargman. The glass box user model for filtering. In *Fourth International Conference on User Modeling*, 1994. Hyannis, Massachusetts, the MITRE corporation.

[14] Fredrik Kilander. Comparisons of the cosine measure and substring indexing on usenet news articles. Technical report, Department of Computer and Systems Sciences, Stockholm University, 1994. http://www.dsv.su.se/˜fk/if_Doc/T.ps.Z.

[15] Fredrik Kilander. A brief comparison of news filtering software. Technical report, Department of Computer and Systems Sciences, Stockholm University, 1996. http://www.dsv.su.se/˜fk/if_Doc/Comparison.ps.Z.

[16] Fredrik Kilander. Pefna the private filtering news agent. Technical Report 97-004, Department of Computer and Systems Sciences, Stockholm University, 1996.

[17] Fredrik Kilander. Properties of electronic texts for classification purposes as suggested by users. Technical report, Department of Computer and Systems Sciences, Stockholm University, 1996.

[18] Ken Lang. Newsweeder: An adaptive multi-user text filter. Technical report, School of Computer Science, Carnegie Mellon University, 1994. http://anther.learning.cs.cmu.edu/res_sum.ps.

[19] Ken Lang. Newsweeder: Learning to filter netnews. Technical report, School of Computer Science, Carnegie Mellon University, 1995. http://anther.learning.cs.cmu.edu/ml95.ps.

[20] Pat Langley and A. Simin, Herbert. Applications of machine learning and rule induction. *Communications of the ACM*, 38(11):55–64, November 1995.

[21] A. Lantz. How do experienced users of the system usenet news select their information? Technical report, Department of Computer and Systems Sciences, Stockholm University, 1993.

[22] A. Lantz. Useful criteria for intelligent filtering? Technical Report 95-042, Department of Computer and Systems Sciences, Stockholm University, 1995.

[23] T. W. Malone, K. R. Grant, F.A. Turbak, S.A. Brobst, and Cohen D. Intelligent information.sharing systems. *Communications of the ACM*, 30(5):390–402, 1987.

[24] M. Morita and Y. Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 272–281. Springer-Verlag, 1994.

[25] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186. Chapel Hill, NC, 1994.

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

[27] B. Sheth. A learning approach to personalized information filtering. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1994.

[28] A. Strauss and J. Corbin. *Basics of Qualitative Research: Grounded Theory, Procedure and Techniques*. Newbury Park, SAGE Publications, 1990.

[29] T. W. Yan and H. Garcia-Molina. Index structures for information filtering under the vector space model. Technical report, Dpt. of Computer Science, Stanford University, Stanford CA 94305, 1993.

[30] Tak Y. Yan and Hector Garcia-Molina. Sift — a tool for wide-area information dissemination. In *Proceedings of the 1995 USENIX Technical Conference*, pages 177–186, 1995. ftp://db.stanford.edu/pub/sift/sift.ps.