PEFNA
The Private Filtering News Agent

The IntFilter Project
Fredrik Kilander
Eva Fåhræus
Jacob Palme

Dpt of Computer and Systems Sciences
Stockholm University

February 17, 1997

Contents

1 Introduction **3**
- 1.1 Overview . . . 3
- 1.2 The Agent . . . 4
  - 1.2.1 Category distance . . . 4
- 1.3 The Client . . . 4
  - 1.3.1 The Thread Distance Plot . . . 5
- 1.4 Related work . . . 5

2 System description **7**
- 2.1 User data storage . . . 7
  - 2.1.1 The article database . . . 7
  - 2.1.2 The example directory tree . . . 8
- 2.2 The classification agent . . . 8
  - 2.2.1 The citation analyzer . . . 9
  - 2.2.2 The category similarity evaluator . . . 10
- 2.3 The news browser . . . 10
  - 2.3.1 The presentation sequence . . . 11
  - 2.3.2 The category editor . . . 12
- 2.4 Experiments with term weights . . . 13
  - 2.4.1 Word classes . . . 13
  - 2.4.2 Analysis of the cosine measure . . . 14
  - 2.4.3 Sources of misclassifications . . . 14
  - 2.4.4 Term frequency . . . 15
  - 2.4.5 Inverse document frequency . . . 15
  - 2.4.6 The probability of the word given the category . . . 16
  - 2.4.7 Term informativity . . . 16
- 2.5 Configuration of the test situation . . . 17
- 2.6 Running the tests . . . 17
  - 2.6.1 Inverse document frequency . . . 18
  - 2.6.2 Informativity weighing . . . 18
  - 2.6.3 Analysis . . . 21
  - 2.6.4 Removing term distribution from the article . . . 22
- 2.7 Conclusions . . . 25

Chapter 1

Introduction

The Private Filtering News Agent (PEFNA) is a prototype system for filtering Usenet News. PEFNA was developed by the Intelligent Filtering project (IntFilter). This document describes the PEFNA system.

1.1 Overview

The PEFNA system consists of two pieces of software, a Usenet News browser (the client) and a classifier (the agent). 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. 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.

1.2 The Agent

PFAGENT is the program which performs the analysis and classification of unread articles. The agent does not take any destructive action, it only rates each unread article and stores the information in the user's personal database. The agent runs independently and interaction with the user is indirect, via the categories and examples supplied by the user and the distance estimates stored in the database.

The agent is modularized and can be seen as an assembly of separate parts where each part can be added or disabled without disturbing the operation of other components. Currently only two such modules are implemented; a heuristic for measuring the ratio between cited and original text in the article and a category distance metric.

The major work in the agent consists of establishing the message id of each unread article and calling each module. The module then decides if the article is to be fetched and examined. Some modules may choose to ignore articles already analyzed, while others may have to update the article's database entry in response to a changed collection of examples and categories.

1.2.1 Category distance

When the user instructs PFCLIENT (the newsreader) to label an article as an example of a category, the article is saved to a file under the user's home directory. The label is used as the name of the directory and while this invites the use of a concept hierarchy the list of categories is a flat, disjoint set. Technically, the user can arrange for the text to be a member of several categories but the the article is unable to distinguish between the categories of which it is a member.

When the agent is run it examines the files in each category (directory) and creates a word frequency list for each file. This file is cached along with the example to avoid redundant reconstruction on each run. For each category, the agent then collapses all the word lists in the category into a concept master. This word list contains all word frequencies in the category and is counted as single document.

When an article is to be examined a temporary word list is created for it. Salton's cosine measure [9] is then used to calculate a distance measure to each of the concept masters. The list of distances is stored in the database under the article's message id.

1.3 The Client

The PEFNA client is written in Tcl/Tk. Communication with the NNTP server is provided by code from arTCLs, another newsreader written in Tcl/Tk by Mike

Hoegeman (mh@awds.imsd.contel.com). The client provides a modern graphical user interface and all the functionality needed to read Usenet News.

The client has three displays: the list of subscribed groups, the list of threads in the current group and the text of the currently displayed article. Any of these can be placed into view at any time. However, making a selection in the group list will reload the contents of the thread list; making a selection in the thread list reloads the current article.

The filtering action of the PEFNA system becomes visible in the thread list, ie the list of threads in the currently selected group. The threads are constructed twice, first by traversing the pointers in the References: field in the article header, and secondly by collapsing articles with identical subject lines.

The client examines the category distances for all articles in the thread. The name of the winning category is then used to label the thread.

After labelling, the threads are sorted in the order given by the user's list of categories. The user can rearrange the order of the labels at any time. Within each thread, articles are kept in chronological order.

1.3.1 The Thread Distance Plot

From the thread selection list the user can command a separate window for any thread. This window contains a plot of the articles in the thread and their relation to their categories. The plot is in the shape of a 'radar' where the names of the categories are placed around the circumference of a circle and the articles are shown as dots within the circle. The position of an article is determined by the relative distances to the surrounding categories.

1.4 Related work

A few filtering systems exists which base their operation on that the user provides the system with examples instead of instructions, ie a specification of *what* rather than *how*.

NewsWeeder [6, 7] 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 with 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 [5] 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.

Many filtering and retrieval systems has been built using the cosine measure to estimate the similarity between documents or a query and a collection of documents.

The Stanford Information Filtering Tool (SIFT) [10] 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 cabability. 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.

SMART [3] 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 a newsfeed as the message store, or its routines can be built into a special-purpose browser.

RAMA [1] 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).

InfoScan[1] 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.

The texts (which must be files in 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 visualizes the relation between the filter and the texts found in the collection.

The BORGES project[2] 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.

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

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

Chapter 2

System description

The PEFNA system consists of two separate components: a Usenet news browser (PFCLIENT) and a classification agent (PFAGENT). The two interacts through a collection of labelled examples and a database of classified news articles.

2.1 User data storage

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.

2.1.1 The article database

The database is a formatted ASCII file. It is divided into two sections: a short symbol table mapping small integers to category labels and a list of article measurements.

Each line in an article measurement contains a record of information about an article. The following information is kept:

- Message-ID (as per RFC 822).
- Newsgroup. The name of the Usenet news group from which the article was fetched.
- Sequence. The sequence number of the article as maintained by the user's NNTP-server.
- Flags. A set of flags for the classication agent's internal use.
- Expiredate. Time and date when the article record can be deleted.

- Update. Time and date of the most recent classification.

The rest of the line consists of a list of category-similarity pairs where the categories are those created by the user. There are additional fields and flags not mentioned here but they are no longer used.

2.1.2 The example directory tree

Each article saved by the user as an example of a category is saved as a file in a subdirectory. The filename is its message id. The directoryname is the label the user has choosen for the category. In addition to the raw text of the article, additional files may be present: a list of words created and recreated by the classification agent and a text file containing any comments the user sees fit to provide.

Since word extraction can be a lengthy process (especially if checks for spelling errors, homonyms and synonyms is being performed) the extracted words are saved in an easily accessible format. Free text written by the user was added to allow the test subject to comment on the classification method, ie to note down why a particular article was interesting. In practice, however, this feature was used very little, even by experienced users.

2.2 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[1] and compares the read articles in the subscribed groups with the contents of the NNTP-server. Each unread and unmeasured article is fetched, its similarity to each of the existing categories is examined and the result stored in the database.

Each time the user changes the definition of a category by adding or removing example files, the agent will retrieve *all* unread articles and reexamine them. Articles both read and classified have their entries in the database deleted after an expiration period (the default is seven days).

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. For instance, the category similarity routine may need to reexamine all articles because the examples has changed while the citation count routine only needs to

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

examine each article once. The actual bytes of the article are of course cached as soon as a routine has requested them.

The modular architecture allows for additional modules to be easily integrated into the existing framework. At the time of this writing, only two such modules were in place, the category similarity evaluator and a citation analyzer.

2.2.1 The citation analyzer

Many articles in Usenet news consists of text copied from earlier articles. This helps the reader to see what exactly is being responded to and makes it possible to profit from an answer without making the full text of the original question or statement available. However, the amount of contribution in an article can not be estimated by inspecting the number of bytes alone. For markup reasons (color, font etc) it is desirable to be able to distinguish between original text and cited text.

Another features of cited text is that the proportion between cited and original text can be used to estimate certain properties of the article. For instance, if the article consists of 10 lines, half of which is cited text, it can be expected to be a succinct comment to a short statement or an excerpt from a longer text. On the other hand, if the article runs for 200 lines and 98 % of them are cited, its likely that the poster included the previous message in its entirety without making the effort of editing it.

Another common structure in cited vs original text in Usenet news is where the answer is carefully constructed and the comments are interleaved with the cited text. This indicates that the author is making several points at specific statements in the cited text and has made the effort to target those statements rather than making a general assertion at the end.

Thus, the citation analyzer should not only count the total number of lines in the article, it should also count the number of cited lines and detect how many times the text switches from cited to original text.

Detecting most cited lines is easy. Detecting all cited lines and not making any errors about it is very hard. In general, a few carefully selected heuristic rules make a good job. For instance, if a line has a non-letter character in its first column, like ´>´ or ´|´ it may be assumed that the line is cited. The assumption can be strenghtened if all or most of the suspected lines start with the same non-letter character. The existence of quoted material can also be supported by locating the string

> `In article` *message-ID author* `wrote:`

somewhere in the body of the article. This string is automatically inserted by many Usenet news browsers when editing a followup article.

2.2.2 The category similarity evaluator

The bulk of the work in PFAGENT is performed by the routine that compares each article against the collection of examples provided by the user. The user creates categories by naming them. When the user encounters an article which is a good example of a category the user wants to distinguish, the user may save the article as an example of that category.

The similarity measure used in PFAGENT is the cosine measure by Salton et al [9]. The general mechanism of the measure is adapted to the classification task in the following way. A word list is built for each category, consisting of the words in all articles in the category. The weight of each term (word) $w_{c,t}$ is defined as the occurence count. A similar word list is constructed for the article to be measured with a term (word) weight of $w_{a,t}$. The similarity between the article and the category is then measured with:

$$sim_{a,c} = \frac{\sum_t w_{a,t} \cdot w_{c,t}}{\sqrt{\sum_t w_{a,t}^2} \cdot \sqrt{\sum_t w_{c,t}^2}}$$

with a result ranging from 0.0 (no common terms) to 1.0 (equality). The result for each category is noted in the database entry for the article.

A lot of work went into efficiency considerations: for testing purposes it was necessary to keep the word lists for each example article even after they had been merged into a category word list. Word lists were implemented as hashed tables and the string for each word was only stored once. Still, the practical limit was reached around 50–90 example articles. With nine categories and six newsgroups a regular classification run occupied the better part of 15 minutes on a DEC 5500 Ultrix workstation. This time varies with many factors: the number of unread articles to examine, the load on the computer running PFAGENT, the bandwidth available to the Usenet news server and the number of categories and articles created by the user.

A number of alternative weighing methods have been evaluated, see section 2.4.

2.3 The news browser

The Usenet news browser is called PFCLIENT and consists of approximately 4000 lines of Tcl/Tk code. Tcl is an interpreted environment integrated with the Tk toolkit for the X Windows System. The combination allows for rapid and powerful prototyping of applications based on a modern graphical user interface.

PFCLIENT's basic purpose is to show Usenet news articles to the user. The user selects a newsgroup to read and is presented with a sorted list of the available and unread articles. The articles are organized into discussion threads and the user may go directly to a thread by clicking on the text of the thread's subject.

Alternatively, the user may simply advance to the next article from the current position and onwards.

The basic article filtering performed by PFCLIENT is the preparation of the thread list's presentation order. This is the sequence in which the articles will appear if they are read sequentially from the top of the list.

At any time the user may go to another article, thread or newsgroup, assign an article to a category, create a new category, rearrange the existing ones or leave the browser.

2.3.1 The presentation sequence

When the user commands PFCLIENT to visit a newsgroup, the first step is to contact the Usenet news server and obtain information about the unread articles. The subject, references and message-id fields are fetched form the server. The message-id is used to look up the article's entry in the article database prepared by PFAGENT. The references and subject fields are used to tie the articles together in threads.

Within each thread the articles are ordered temporally, there's really no other convenient way if one wants to follow the 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. But before this can be done, each thread must be assigned to a category.

Since many threads consist 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.

In addition to the winning category, two extra columns attempt to visualize how certain the winner is. The first piece of information is the similarity value of the winner, averaged over the number of articles in the thread. The second indicator shows the distance from the winner to its competitors. In truth, the second indication is difficult to interpret since it attempts to collapse a point in a multidimensional linear space down to a scalar.

However, the user can inspect the threads classification by ordering a graph over the articles and their ratings. This graph has the shape of a circle, with the category names evenly spaced around the circumference. Inside the circle the articles are represented as points, each located at the geometrical center determined by the similarities to the categories. An article singularly close to one of the categories is thus plotted close to that label. The points are mouse-sensitive and when the cursor is placed above a point its similarities are displayed as vectors from the center of the graph.

Though a bit cluttered, this graph display is a simple and effective way to show the relation between an article and the categories it has been compared against.

2.3.2 The category editor

The user manages the categories in the category editor, a collection of pop-up dialogs available from PFCLIENT. Three main operations are available through this editor, creating new categories, arranging the sequence from top to bottom of the categories and selecting which categories should be used to classify threads in a particular newsgroup.

Creating a new category is straight-forward. The user types the name in the appropriate place and the category appears at the bottom of the drop-down menu and can be immediately affixed to articles.

The need to have only a subset of all categories active in a particular newsgroup became apparent as the number of categories grew. For one user, one popular category accumulated so many examples that it exerted a significant pull on articles in other newsgroups, not at all relevant to the category.

2.4 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 test cases from a user's set of examples, and assign them to the most similar category. If an article was assigned to the category it originally came from this was counted as a correct classification, or a hit. If the article was classified as belonging to another category the classification was counted as incorrect, ie a miss.

2.4.1 Word classes

Texts in general and articles in Usenet News, has a distribution of words which follows that of a living, natural language. This distribution does not yield to any simple analysis as it is influenced by the writer's message, choice of style or genre, education, rhetorical skill and general literacy. Still, for the purpose of argument, we will use a hypothetical division of words into three frequency classes.

First, there is a class of very common words, like articles, pronouns, and determiners which are needed to build sentences and identify the grammar phrase structure. These words help to identify the language of the text, like German, English or Swedish but have no real impact on topical classification.

The second class of words are those which are not directly related to the subject but are reflections of how the author thinks or decides to formulate message and meaning. These words carry the style, manner and flavour of the text. Take for example these two sentences, the first one could be from a children's book and the second from a novel:

> 'Mary didn't want to ride any more.'

> 'Mary was disinclined to further horseback excercise.'

These words are also necessary, but the author has a greater choice and may select from a larger palette than in the first category.

The third class of words are those which are associated with a certain domain. These are the specific verbs, nouns, terms and idioms which identify a distinguished piece of the discourse to the reader. Words like *'sample', 'miking', 'pitot'* or abbreviations like *'DMA', 'NTSB', 'DAT', 'HTML'* are more indicative of their related topics than anything else, because they usually have very specific meanings.

It must be remembered that there exists no discrete boundaries between these three classes and that the matter is, from a linguistic point of view, grossly oversimplified.

2.4.2 Analysis of the cosine measure

$$sim_{a,c} = \frac{\sum_t w_{a,t} \cdot w_{c,t}}{\sqrt{\sum_t w_{a,t}^2} \cdot \sqrt{\sum_t w_{c,t}^2}} \tag{2.1}$$

The cosine measure (2.1) is actually the cross product between two vectors. Each vector represents a text. Each term (word) represents a dimension (a direction perpendicular to all other directions) in a multidimensional space. The weight of each term represents a distance from zero along the corresponding dimension.

Any two vectors defined in two or more dimensions, can be found in the same two-dimensional plane. In this plane the angle between the vectors can be measured. In the case of term weights, the least weight of any term is zero, so the maximum angle between any two vectors is 90 degrees, or $\frac{\pi}{2}$.

What the cosine measure does, is that it returns the cosine of the angle between the vectors, giving the result 1 when the vectors are pointing in the same direction and 0 when the vectors are completely at right angles to each other.

From this way of calculating similarity, it can be seen that the only important property of the term frequencies is their distribution. For example, the vectors $(2, 5)$ and $(6, 15)$ (assuming that both vectors start at $(0, 0)$) both point in the same direction. Hence, the cosine measure will regard them as identical.

With the distribution similarity in mind, it is easy to see that when used to compare texts, it will be words with similar distributions which will tend to impact the similarity measure most. These words are of course the words in class one, since they are needed to construct understandable sentences. This, in turn, motivates the use of stop-lists for frequent words so that the distribution of the words from class two and three can dominate the similarity measure. Finally, using a distribution presupposes a sufficiently large sample, ie a large body of text. The collections of example texts used by PEFNA provides such a sample, but average size of the news articles is in comparison very short. This leads to skewed distributions, again dominated by the words from class one.

2.4.3 Sources of misclassifications

Using the raw occurence counts have the effect that user categories in PFAGENT with many or lengthy examples tend to attract messages simply because they contain a large number of common words from class one. Since this category of words is rather limited in its number of members and static in its distribution, it is more or less the same across several categories. Therefore, it is actually a smaller part of the problem than the words in the second class. As the number of example articles grow, so will the vocabulary representing class two, simply because more authors and subtopics are represented. The vocabulary in class three is underrepresented compared to class two because class three words are distinct and not exchangeable in the same way as words in class two.

The short average size of a Usenet News article compounds the problem because it may only contain a few or possibly no words from class three, but it must have some words from class two (and one) to be an understandable text. Thus, it is the words from class two which tends to dominate the similarity measure and these words appear more frequently in a category with many example articles.

2.4.4 Term frequency

The basic mechanism of the cosine measure appeared sound, so work was initiated to investigate alternative ways to weigh the terms (words) in the categories. The other option was to invent a completely different comparison algorithm, an unattractive concept under the resources available to the project.

The occurence count for each word in a category is the sum of the number of times the word occurs in all examples of that category. For an article, the occurence count of a word is naturally restricted to the number of times the word appears in the article. The default weight of a word is simply its occurence count.

Raw occurrence counts performs rather well, but does not take into account how a word is distributed *among the categories*, ie how well it can be used to distinguish one category from another. What was needed was a way to weigh each word in each category. Word which appeared in many categories should have lesser weight than those which appeared in one or a few categories.

The base case will be written

$$w_{t,c} = n \tag{2.2}$$

to indicate that the weight of term $w_{t,c}$ in category c is n, the number of times it occurs in category c.

In Salton and McGill [9], page 205, this is the *term frequency* (TF) weighing system.

2.4.5 Inverse document frequency

The first alternative replaced the plain term frequency with

$$w_t = \frac{n}{n_C}, \tag{2.3}$$

where n_C is the number of categories the word appears in. Since this is the *inverse document frequency* (IDF) it was expected to perform better than the base case, but this weighing system produced more misclassifications than the base case. The likely reason is that it needs stop lists to filter out the irrelevant terms. With all words available it doesn't really help the class three words to compete against the class two words. For instance, if the word *'ask'* has $n = 6$ and appears in four categories and the word *'hum'* has $n = 3$ in two categories their resulting weights will be equal ($\frac{6}{4} = \frac{3}{2}$), but *'ask'* is still more likely to occur and may therefore score more hits.

2.4.6 The probability of the word given the category

The second alternative attempt assigned the weight to a word as the probability that the word would appear in the category. This was simply

$$w_t = P(t|C)\,, \tag{2.4}$$

where C is the category which term t belongs to.

This turned out to be indistinguishable from the base case in terms of performance and since it does not alter the proportion between the words in a category compared to the base case, they are identical.

2.4.7 Term informativity

In the Machine Learning community there's a long and established technique for building decision trees from classified data [4, 8]. Independently developed by statisticians [2] the informativity measure used for classification trees can be adapted as a term weighing system.

The idea is that the discriminative power of each word is measured and used as the informativity of the word, ie how well the particular word can be used to determine membership in a category. The measure is computed like this:

$$I(t) = -\sum_{i=1}^{N} P(t|C_i) \log_2 P(t|C_i)\,. \tag{2.5}$$

The measure is open ended in that it will give a result of 0 if a word exists in only one category but yields higher values for words which appears in other categories. The absolute result depends on the number of categories. Another way of viewing the measurement is as a gauge on how a word is distributed accross the categories. A word which occurs evenly among the categories is penalized harder by the measure than one which dominates a single category but has occasional occurences in others.

This is not exactly the same as the *term discrimination theory* discussed by Salton and McGill, nor is it exactly the same as *term relevance* or *term utility* ([9], pg 205) although the informativity measure is similar in spirit to all three.

Term weighing systems like

$$w_t = \max(I(\cdot)) - I(t) \tag{2.6}$$

and

$$w_t = \frac{\max(I(\cdot)) - I(t)}{\max(I(\cdot))} \tag{2.7}$$

(which are basically the same although (2.7) is normalized) were both significantly better than the base case at classifying articles back to their original categories. The term $\max(I(\cdot))$ is the value of the least informative word regardless of the category.

2.5 Configuration of the test situation

The test runs were performed across three dimensions; the dataset, the fraction of test cases drawn from the dataset and the weighing method. In all cases, only the weights on the words in the categories were modified. The words in the article originated from a single document and had their frequencies assigned as their weights.

Datasets were available from three subjects. The subjects had all used PFCLIENT for several months to read Usenet News and to collect categories and examples. Subject A had 15 categories and 90 examples, subject B had 18 categories and 400 examples and subject C had 7 categories and 158 examples. The categories and examples had been created from each subject's personal preferences.

The fraction of test cases drawn from each dataset was $\{0.1, 0.2, 0.3, \ldots 0.9\}$, ie at least 10% and at most 90%. Rather than making a completely random selection which would favour big categories an attempt was made to draw the test fraction from each category. At least one test case was drawn from each category which held two or more examples.

Classification accuracy was measured as the average classification accuracy of the category. This gives a slightly lesser value than simply taking the number of correct classifications and dividing by the number of test cases, since that hides the difference between a big category with many classification attempts and a small category which only has one or two chances to prove its worth.

Finally, efficiency was be estimated from the classification accuracy by comparing the classification performance with a classifier without prior information, ie the probability of finding the right class by chance. For the three subjects this probability was $1/15 \approx 0.0667, 1/18 \approx 0.0556$ and $1/7 \approx 0.1428$ for subjects A, B and C respectively.

For each point in the exploration space (dataset, test fraction, weighing method), 50 independent runs were performed, each run making a new draw of test cases. This reduced the variance of the average classification accuracy enough to indicate a separation between the weighing methods with a certainty of 95%.

Although only a representative part of the data is reported here, over 24000 individual article measurements were run.

2.6 Running the tests

In the graphs showed in this section, the accuracy on the vertical axis is the average classification accuracy per category after 50 independent samplings of

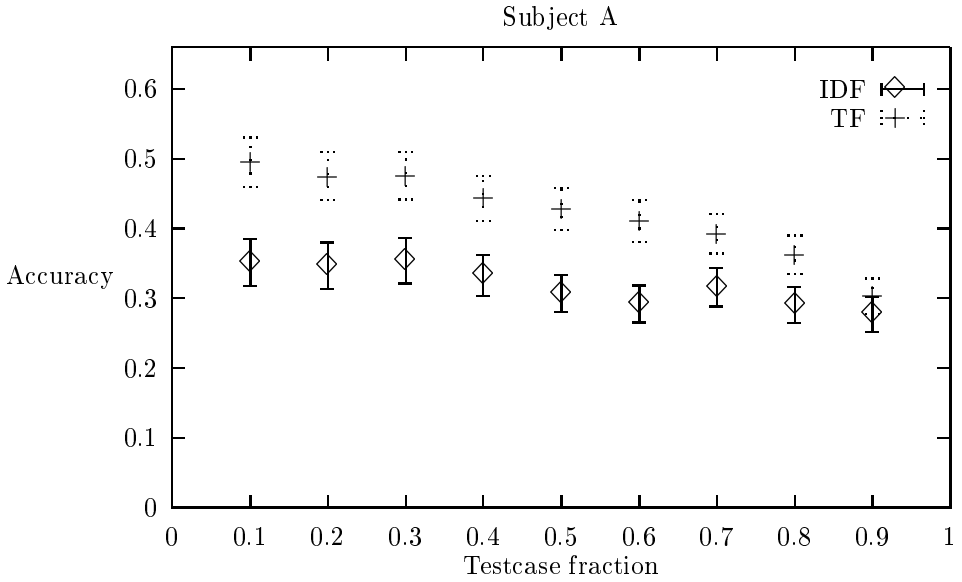

Figure 2.1: Comparing the term frequency against the inverse document frequency on the dataset from subject A.

test cases, ie

$$\frac{\frac{\sum_{i=1}^{C} H_{C_i}/T_{C_i}}{N_C}}{\sum_1^{50} 50} \tag{2.8}$$

where H_{C_i} is the number of test cases correctly assigned to category C_i, and T_{C_i} is the total number of test cases drawn from category C_i, and N_C is the number of categories. Errorbars for a 95% confidence interval of the true mean are provided.

2.6.1 Inverse document frequency

Figures 2.1–2.3 show how the inverse document frequency (IDF, 2.3) compares with term frequency (TF, 2.2, the base case). As can be seen, the term frequency is more accurate. The best efficiency of the term frequency (ie how many times more accurate was the classification compared to guessing the correct category) was $0.4956 * 15 \approx 7.43$ for subject A, $0.1885 * 18 \approx 2.83$ for subject B and $0.4076 * 7 \approx 6.11$ for subject C.

2.6.2 Informativity weighing

Figures 2.4–2.6 show how the term informativity in (2.5) compares with term frequency. In all three datasets the informativity term weighing method is significantly more accuracte than term weighing, although not in all test case fractions. For subjects A and C, informativity is more accurate for test case fractions

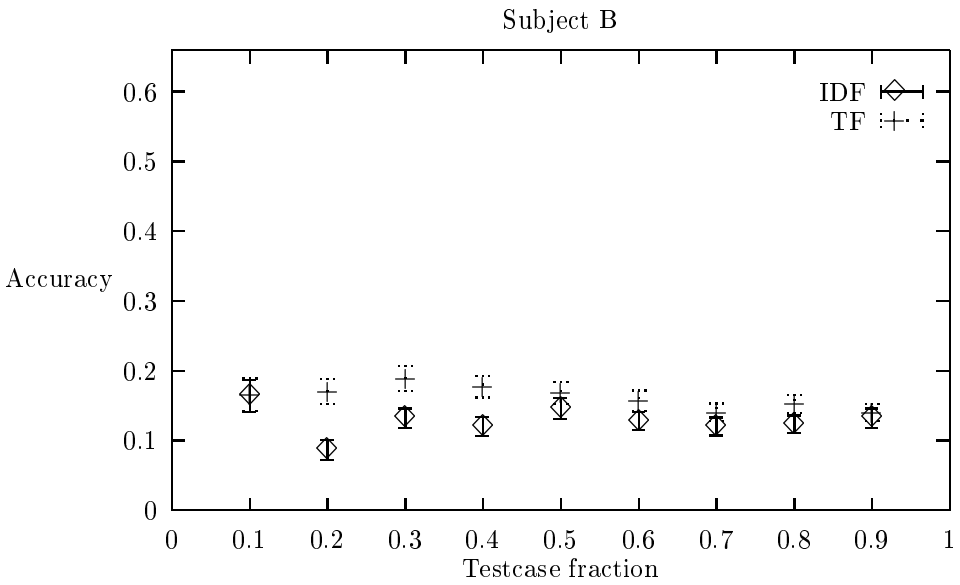

Figure 2.2: Comparing the term frequency against the inverse document frequency on the dataset from subject B.

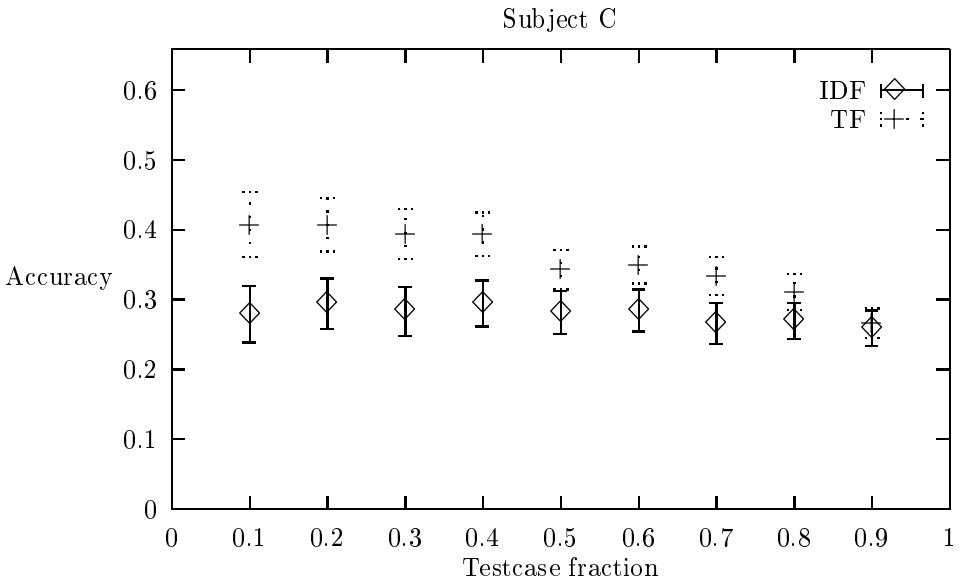

Figure 2.3: Comparing the term frequency against the inverse document frequency on the dataset from subject C.

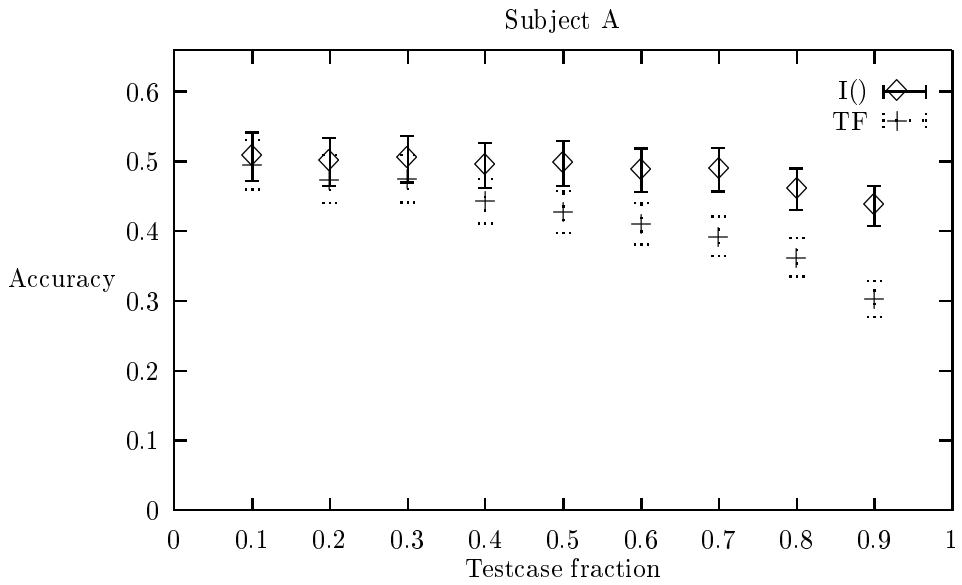

Figure 2.4: Comparing the term frequency against informativity weighing on the dataset from subject A.

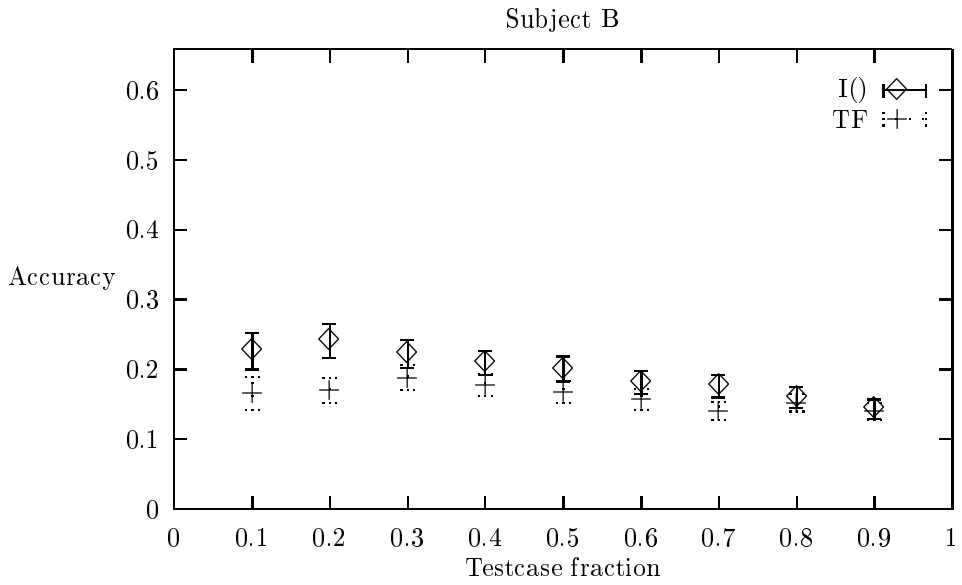

Figure 2.5: Comparing the term frequency against informativity weighing on the dataset from subject B.

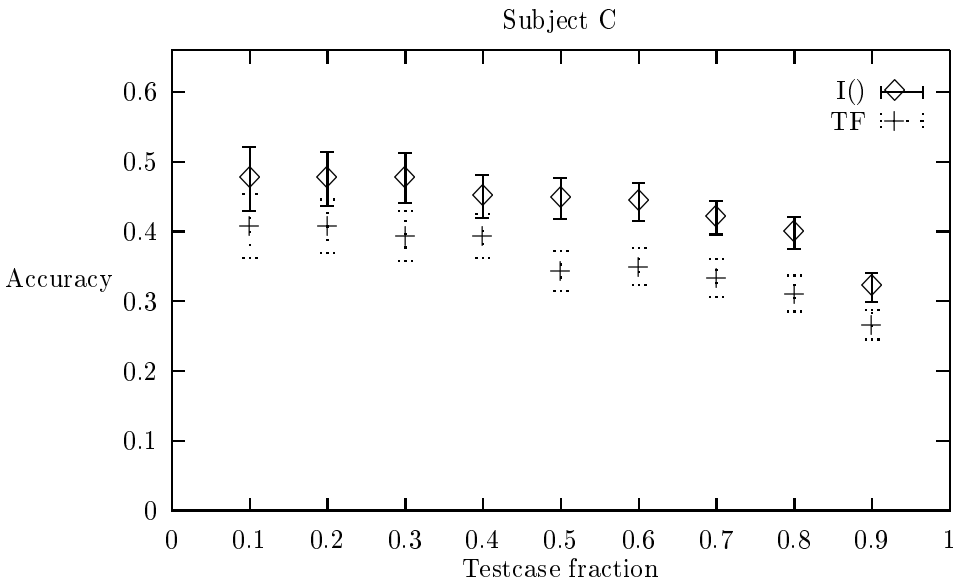

Figure 2.6: Comparing the term frequency against informativity weighing on the dataset from subject C.

$\{0.5, 0.6, 0.7, 0.8, 0.9\}$. For subject B, the only test case fractions where informativity is significantly more accurate are $\{0.1, 0.2\}$.

The efficiencies for informativity were $0.5067 * 15 \approx 7.60$ for subject A, $0.2407 * 18 \approx 4.33$ for subject B and $0.4762 * 7 \approx 3.33$ for subject C. Thus, even though the average accuracy for subject C was higher than for subject B the classification made a better job for subject B than for subject C.

2.6.3 Analysis

What happens when the term frequencies of the example category is replaced by the informativity of the term? At first, the similarity function computed must be rewritten:

$$sim_{a,c} = \frac{\sum_t w_{a,t} \cdot I(t_c)}{\sqrt{\sum_t w_{a,t}^2} \cdot \sqrt{\sum_t I(t_c)^2}} \,. \tag{2.9}$$

The most striking difference is that the identity of the category disappears and that only the informativity of the term remains. Since the informativity of the term is computed across all available categories, the modified measure actually removes almost all traces of term distribution from the categories. What is left, is the distribution in the article.

With the term frequencies removed, the vectors representing the categories are essentially discretized by *term membership*. If a category includes a specific term the vector will be influenced by the amount given by the informativity of the

term. All other categories who also include this term will have the same value on that particular dimension. As a result, they will appear more similar since their vectors will be identical when only defined by their common terms.

The collapse of the term distribution in the categories may at first appear to be a step backwards, but the empirical results indicate otherwise. The reason for this appears to be hidden in the dimensions where the categories does not share terms, ie those terms which are good indicators of category membership. Terms unique to a category will have higher informativity and thus influence the direction of the category vectors more than their less informative terms.

As an example, imagine a three-dimensional space with two axes on the ground and one axis up into the air. The two horizontal axes represent terms with low informativity. The vertical axis represents a term with high informativity. Categories which only share the low informativity terms will coincide in the horizontal plane. But, any category in which the term with high informativity is a member will point up into the air, away from the other.

When the distribution of terms in the article is used in the modified similarity measure above (2.9), the high frequency words will accumulate some importance since their high occurence count is multiplied with the low (but not zero) informativity value. This reduces the utility of rewarding informative terms and leads to the hypothesis that classification may be improved further by removing the distribution from the article as well. This would reduce the similarity measure to the point where the final decision is influenced by the intersection of terms, weighed by their informativity.

2.6.4 Removing term distribution from the article

$$sim_{a,c} = \frac{\sum_t I(t_a) \cdot I(t_c)}{\sqrt{\sum_t w_{a,t}^2} \cdot \sqrt{\sum_t I(t_c)^2}}\,. \tag{2.10}$$

Equation (2.10) shows the similarity measure used in a complementary run using otherwise the same parameters as in figure 2.4, 2.5 and 2.6. The difference from equation (2.9) is in the expression in the numerator. Both the article and the category are only represented by the (global) informativity of common terms. The term $\sum_t w_{a,t}^2$ in the denominator remained due to an oversight. This expression is a constant property of an article and does not unduly influence the comparison (although the expression no longer is a correct cross-product).

An even stricter formulation of (2.10) could be

$$sim_{a,c} = \frac{\sum_t I(t_a) \cdot I(t_c)}{\sqrt{\sum_t I(t_a)^2} \cdot \sqrt{\sum_t I(t_c)^2}} \tag{2.11}$$

where the informativity of unknown terms is zero.

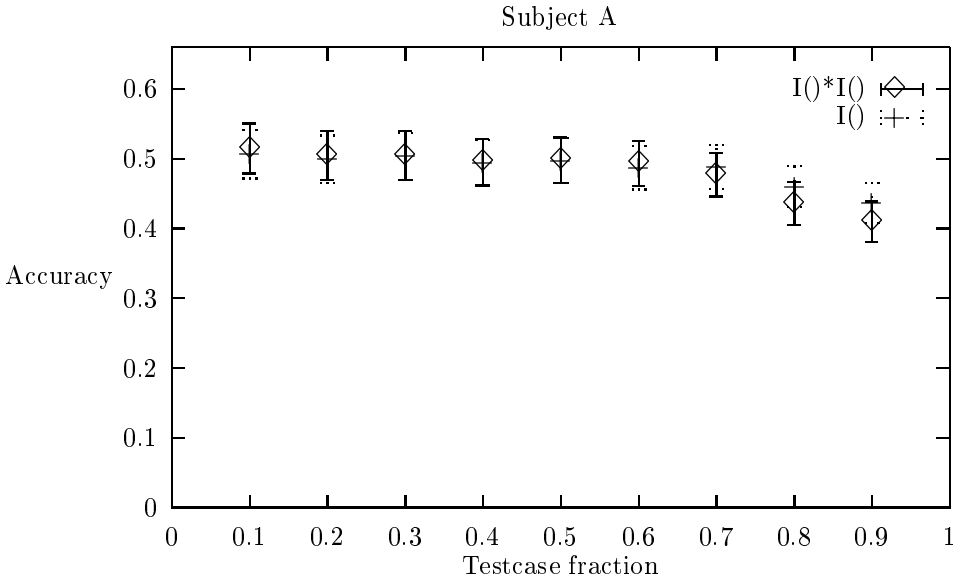

Figure 2.7: Informativity weighing of the category only compared with both category and article, for subject A.

In fact, (2.11) simplifies into

$$sim_{a,c} = \frac{1}{\sqrt{\sum_t I(t_c)^2}} \tag{2.12}$$

This is because the numerator and the left hand term of the denominator are equal when any unknown terms in the article fails to contribute to the sum of squares. Since both article and all categories share the usable informativity values, $I(t_a) = I(t_c)$ and the numerator is rewritable as $\sum_t I(t_a)^2$.

There is, however, no obvious reason to be drastic and remove the information that the article contains a lot of unrecognized words. In (2.10) that property of the article is retained and has an attenuating influence on the similarity figure. Although not demonstrated here, it appears that the presence of this property is indeed beneficial for the categorization accuracy.

Figures 2.7, 2.8 and 2.9 shows how measure (2.9) compares with (2.10). The points labelled $I()$ are the same as in figures 2.4, 2.5 and 2.6, and refer to equation (2.9). The points labelled $I() \cdot I()$ refer to equation (2.10).

Although no noticable improvement can be discerned for subject A, significant differences are present in figures 2.8 and 2.9 for subjects B and C. The reason why not all three subjects were subject to significant improvement is unclear, but is obviously hidden in the properties of the data sets. Subject B had both the largest number of categories and articles.

Table 2.6.4 contains the best and average predictive efficiencies, that is, improvements on categorization related to random choice.

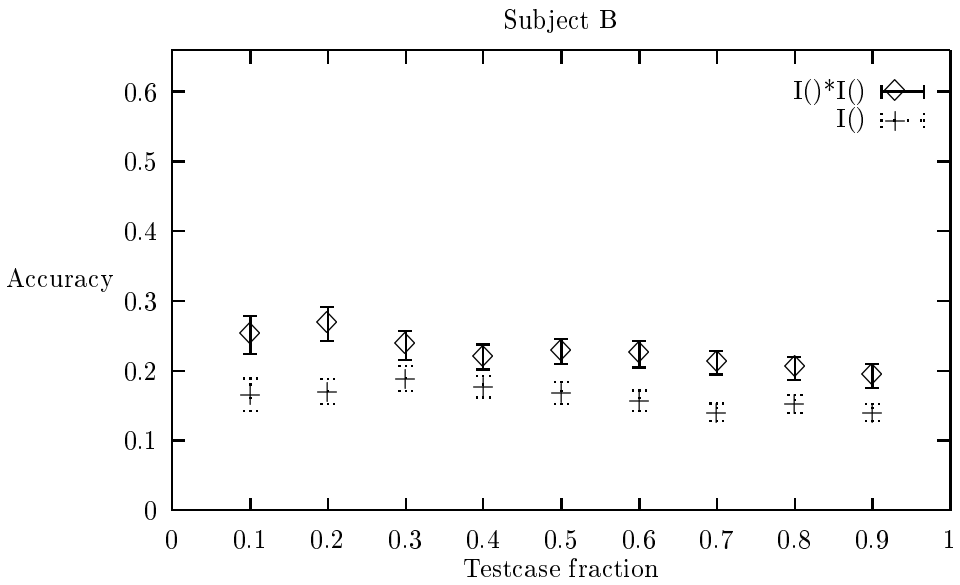

Figure 2.8: Informativity weighing of the category only compared with both category and article, for subject B.

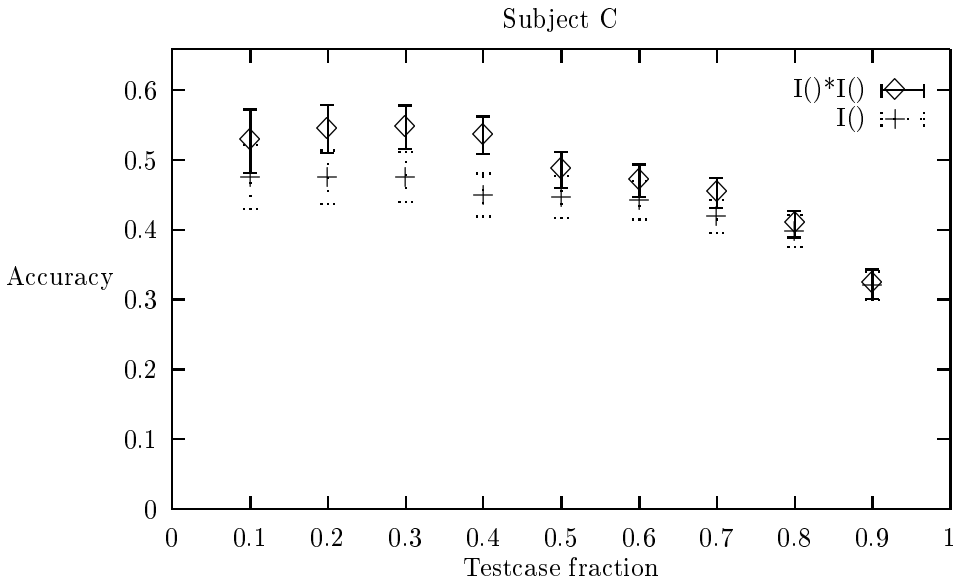

Figure 2.9: Informativity weighing of the category only compared with both category and article, for subject C.

	Subject A		Subject B		Subject C	
Nof categories	15		18		7	
Similarity measure	avg	best	avg	best	avg	best
IDF	4.777	5.307	2.304	2.953	1.953	2.065
TF	6.309	7.433	2.920	3.393	2.494	2.853
I()	7.288	7.600	3.517	4.333	3.037	3.333
I()*I()	7.227	7.726	4.068	4.801	3.340	3.828

Table 2.1: Average and best predictive efficiencies.

2.7 Conclusions

An interesting feature of figures 2.4 and 2.6 is that the informativity weighing method loses accuracy less rapidly than the term frequency method as the test case fraction increases. This can not be observed in figure 2.5 but gives a hint that informativity weighing may be better for smaller example sets.

Another feature present in all tests is that the variance (as indicated by the 95% confidence intervals) of the classification accuracy is decreasing as the number of test cases increases. This can be interpreted as support for the hypothesis that a small number of examples in each category is better than a diffuse and wide category, since it reduces noise created by class two words. However, this presupposes that the user is categorizing messages according to topic and not to writing style. Should the user label texts according to style, the noise would be generated by the class three words.

The term informativity methods makes a better job than term frequency of classifying articles in the lab test situation reported here. All that is required of a good similarity measure is that it increases the space between the categories, moving similar categories closer to the article and non-similar ones farther away. The recognition of discriminatory words which are good indicators of category membership is a key aspect of this process and the term informativity measurement appears to be a feasible way of doing it.

Bibliography

[1] 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.

[2] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmost California, 1994.

[3] 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=.

[4] Bojan Cestnik, Igor Kononenko, and Ivan Bratko. Assistant 86: A knowledge elicitation tool for sophisticated users. In *Proceedings of the Second European Working Session on Learning*, pages 31–45. Sigma Press, 1987.

[5] 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.

[6] 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.

[7] 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.

[8] Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

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

[10] 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.