A Survey on Personalised Information Filtering Systems for the World Wide Web.

Kjersti Aas

December 1997

Contents

Chapter 1

Introduction

Technological advances have made wide-area information sharing commonplace. A suite of tools has emerged for network information finding and discovery, e.g., World Wide Web, Archie, and Gopher. These tools provide a means to search for existing information, but the exploding volume of digital information makes it difficult for the user to keep up with the fast pace of information generation. Instead of making the user go after information, it is desirable to have information selectively flowed to the user. In particular, there is a need for tools to capture *profiles* of users' information needs, and to find documents relevant to these needs, as these needs change over time.

An intriguing thought for the future involves the different uses to which we can put such a user profile. Over a period of time, it will become an increasingly accurate predictor of the user's interests. One can imagine using the profile for many tasks: filtering incoming e-mail, picking out interesting Usenet news articles, creating personalised newspapers, automatic selection of goods to browse in an on-line shop, and so on.

A very simple kind of tool to capture profiles of users' information needs is already available on the internet: mailing lists. Hundreds of mailing lists exist, covering a wide variety of topics. The user subscribes to lists of interest to him and receives messages on the topic via email. He may also send messages to the lists to reach other subscribers. A problem with mailing lists, is that a user whose information does not exactly match certain lists will either receive too many irrelevant, or too few relevant messages.

Most people using Netscape or other Web browsers maintain a bookmark list that contains pointers to favourite Web pages. Often the pointers are grouped together into categories. If it is possible to discover salient features of the already established categories, one can create a software agent that can use existing Web search tools on behalf of an individual user to automatically seek new Web documents that are potentially interesting.

Recent work that arises at the intersection of information retrieval and software agents offers some novel solutions to this problem. Information retrieval is a well-established field of information science that addresses issues of retrieval from a large collection of documents in response to user queries. By comparison, agent research is a relatively new field of study, which has grown out of artificial intelligence. Agent research is concerned with issues of designing intelligent and autonomous software for a variety of tasks.

Chapter 2 describes some of the systems that have been developed the last few years with the aim of assisting a user in finding information on the World Wide Web. Most of these systems can be classified into one out of three categories: search engines, assisted browsing systems and personalised information filtering systems. In this report, we focus on personalised information filtering, and in Chapters 3–6 the past work done in this area is surveyed.

Chapter 2

Systems

This chapter describes some of the current systems that assist a user in finding information on the World Wide Web. Most of these systems can be classified into one out of three categories: search engines, information filtering systems, and assisted browsing systems. Sections 2.1–2.3 describe the differences between the categories and give examples of systems from each category. An overview of the systems is given in Table 2.1.

Name	*Type*	*Site*	*Application area*
Lycos	Search engine	Commercial	The Web
Alta Vista	Search engine	Commercial	The Web
WebCrawler	Search engine	Commercial	The Web
OpenText	Search engine	Commercial	The Web
Syskill & Webert	Information filtering	University of California	The Web
Lira	Information filtering	Stanford University	The Web
IDD News Browser	Information filtering	George Mason University	Usenet news
SIFT	Information filtering	Stanford University	Usenet news
PSUN	Information filtering	University College Cork	Usenet news
Newt	Information filtering	MIT	Usenet news
NewsWeeder	Information filtering	Carnegie Mellon	Usenet news
InformationFinder	Information filtering	Andersen Consulting	Lotus Notes
WebHunter	Information filtering	MIT	e-mail
Tapestry	Information filtering	Xeroc PARC	e-mail
WebWatcher	Assisted browsing	Carnegie Mellon	The Web
Letizia	Assisted browsing	MIT	The Web

Table 2.1: *Current systems that assist a user in finding information on the web.*

2.1 Search engines

Currently Web searching is based on traditional information retrieval techniques and is typically based on boolean operations on keywords. Major Web services such as Lycos, Alta Vista, WebCrawler, and OpenText employ software robots (also called spiders) which traverse the Web and create an index based on the full text of the documents they find. A user submits a keyword query to one of these services and receives the location of any document found in the index which matches the query. Different services vary in their methods of constructing the index. For instance, Alta Vista ranks the results based on a scoring algorithm where documents with a higher score appear at the head of the ranking list. A document has a higher score if the following hold:

- the query words or phrases are found in the first few words of the document (i.e. in the title of a Web page)
- the query words or phrases are found close to one another in the document.
- the document contains more than one instance of the query word or phrase. However, repeating a word over and over in a Web page has no effect on the ranking.

The enormous growth of the Web makes the search engines less favourable to the user because of the large number of pages they return for a single search. Thus it is time consuming for the user to traverse the lists of pages just to find the relevant information. To remedy this problem many researchers are currently investigating more efficient approaches for assisting users in finding information on the Web. In Sections 2.2 and 2.3 we describe two categories of such approaches.

2.2 Personalised information filtering systems

The main difference between search engines and information filtering systems, is that while search engines find relevant items in a database, information filtering systems remove irrelevant items from a constant stream of documents. Personalised information filtering systems generally embody one or more of a user's interests via a user profile, use this profile to filter incoming articles to that user, and update the profile based on relevance feedback information provided by the user after assessing the received articles. The filtering systems are often classified into one of two categories, depending on the manner in which the documents are filtered. Cognitive systems choose documents based on the characteristics of their contents, while social systems select documents based on recommendations and annotations of other users. Below we give some examples of systems from both categories.

2.2.1 Cognitive filtering systems

The cognitive systems, also referred to as content-based systems, choose documents based on the characteristics of their contents. Examples of such systems are Syskill & Webert [22], Lira [2], IDD News Browser [5], SIFT [31], PSUN [28], Newt [26], NewsWeeder [15], and InformationFinder [14].

Syskill & Webert

Starting from a manually constructed index page for a particular topic, the user can rate hyperlinks off this page. The system uses the ratings to learn a user specific topic profile that can be used to suggest unexplored hyperlinks on the page. Syskill & Webert also use Lycos to retrieve pages by turning the topic profile into a query.

Lira

In this system, which is also known as Fab, a general model of one user's interest is learned by asking the user to rate pages. Lira uses the model to browse the Web offline and returns a set of pages that match the user's interest.

IDD News Browser

When the user accesses this system for the first time, he is asked to set up an initial profile. This profile is periodically run against various collections built from live internet newspaper and Usenet feeds to generate matches in the form of a personalised newspaper. When reading his personalised newspaper, the user provides positive or negative feedback to the system, which are then used by a learner to induce new profiles. These system-generated profiles may then be used to make recommendations to the user about new articles and collections.

SIFT

The SIFT system is a tool for filtering Usenet news articles. A user submits profiles, one for each topic he is interested in, giving keywords in articles he would like to read. Moreover he may specify the frequency of updates and the amount of information to receive. When a new article comes in, it is matched against all the profiles. If the similarity between an article and a particular profile is above a certain threshold, the document is mailed to the user.

PSUN

PSUN is another system for filtering Usenet news articles. The user profile is constructed automatically from documents previously deemed to be of interest to the user; no explicit specification of key words and terms are required. It is similar to the other systems in that it act as a broker on behalf of a user, embodying his interests and goals, accepting feedback from the user, and adjusting the user goals based on this relevance feedback. However, the designers of PSUN claim that their system is better than the others when it comes to adapting a profile to a user's changing requirements.

Newt

Newt is very similar to PSUN. The user profile is modelled as a population of profile individuals, each of which searches for articles in a small domain. The user can provide feedback to the system for the articles recommended and this feedback is used to modify the profile(s).

NewsWeeder

When a user wants to read Usenet news articles using NewsWeeder, he go to his own virtual newsgroup. The newsgroup contains a list of one-line article summaries, which are sorted by how well they match a model of the user's interests. The user selects a group of articles from these summaries, reads and then rates them. NewsWeeder collects the user's ratings and each night, it uses this information to learn a new model of the user's interests.

InformationFinder

InformationFinder, or InfoFinder for short, is an experimental system that learns what types of internet pages a user is interested in and periodically sends him suggested URLs that it thinks will match his interests. It learns the user's interests from pages that the user has selected as being relevant or not relevant to a particular category. InformationFinder differs from many of the other approaches in this section in that is use phrases instead of single words to represent the content of the pages.

2.2.2 Social filtering systems

Social filtering, also referred to as collaborative filtering, is the filtering of information based on the advice of others. This method of information filtering is often used by people in their daily lives. People seek recommendations from others, and they use this advice to filter information. For example, a person who wants to see a movie might try to get recommendations from friends. This information is processed, and a decision is made on which movie to see. The recommendations of some friends are trusted more than recommendations from others. This trust is weighted accordingly in the decision making process. Examples of social filtering systems are WebHunter [16] and Tapestry [11].

WebHunter

WebHunter (also known as WebHound) is one of the social filtering systems that has been developed. In this system, the user submits a list of pages together with the ratings of these pages. The agent finds other users with similar ratings and suggests unread pages that are liked by others with similar interests.

Tapestry

Tapestry is an experimental system for filtering e-mail. The system implements a novel mechanism for collaborative filtering in which users annotate documents before the documents are filtered. Because annotations are not available at the time a new document arrives, the system supports continuous queries that examine the entire database of documents and take into account newly introduced annotations during the filtering process.

2.3 Assisted browsing systems

Rather than providing the user with selected pages, some systems instead assist the users in their browsing. These systems track and observe users' behaviour and reactions as they traverse the Web. Based on the collected data, the systems explore and examine links ahead of the users and recommend the most promising links. Examples of such systems are WebWatcher [1] and Letizia [19].

WebWatcher

The WebWatcher system requires the user to state a particular goal, and then interactively offers advice on which link to follow next given the page the user is looking at. The system learns by keeping track of whether its advice is followed, and also from asking the users to signal success or failure when their search has been completed.

Letizia

Letizia is similar to WebWatcher in the sense that the system accompanies the user while browsing. As the user operates a conventional Web browser such as Netscape, the agent tracks the user behaviour and attempts to anticipate items of interest by doing exploration of links from the user's current position. Letizia differs from WebWatcher in that it does not require the user to state a goal at the outset, instead trying to infer goals implicitly from the user's browsing behaviour.

Chapter 3

Personalised information filtering systems

Information filtering deals with the delivery of information that is relevant to the user in a consistent and timely manner. An information filtering system assists users by filtering the data stream and delivering the relevant information to the user. Information preferences vary greatly across users; therefore, filtering systems must be highly personalised to serve the individual interests of the user. The system must also accommodate changes in the user needs and adapt to those changes. A personalised filtering system should satisfy three requirements:

- **Specialisation.** A personalised filtering system should be highly responsive to the needs of the user. Since a filtering system involves repeated interactions with the user, the system should get a better feel for the user's needs over time. Assuming that a lot of the user actions are consistent, the system should get increasingly better at matching his needs over time. The system should be able to gradually converge to that part of user needs which is predictable and consistent.

- **Adaptation.** Since the interaction could extend over a long period of time, the user's interests cannot be assumed to stay constant. The change in interest could be anything from a slight shift in relative priorities to completely losing interest in some domain and gaining interest in another. The system must be able to detect or must allow the user to indicate the change of interests and should respond by apdapting to these changes.

- **Exploration.** Considering the possibility that the information seeker might not be highly motivated, the system should be capable of recommending new potentially interesting information based on what it already knows about him. The system should be able to explore newer information domains to find something of potential interest to the user.

In a personalised information filtering system the interaction between the system and the user runs in cycles. When the user accesses the system the first time he defines his fields of interest. The system formalises this information and builds a user profile based on it. The system then daily searches a predefined set of Web sources and indexes all new documents that are found. Each time the user accesses the system the following steps take place:

- All unread documents are presented to the system, and the system generates confidence scores based on the user profile and the documents.
- According to the confidence scores and a given threshold, the system decides which documents to route to the user.
- The user receives a list of titles of interesting documents together with the confidence values, and based on the titles he decides whether to read the documents or not.
- For each document he reads, he provides one of three types of actions: Positive feedback, negative feedback, or not feedback at all.
- The feedback is recorded and later used to improve the performance of the system by modifying the system's classification behaviour in the light of changing user requirements.

The personalised information filtering systems are often classified into one of two categories depending on the manner in which the documents are score. *Cognitive systems* choose documents based on the characteristics of their contents, while *social systems* select documents based on recommendations and annotations of other users. The cognitive and social approaches are just as valid for selecting documents. If information is being gathered for keeping up to date with a certain community, social filtering is the way to choose documents. However, if information is being gathered based on the topic, independent of other users, the cognitive systems are more appropriate.

In this report we concentrate on the cognitive systems, for which there are four essential requirements:

- A representation of a Web page.
- A representation of the user's interests.
- A function to determine the pertinence of a Web page given a user's interests.
- A function returning an updated user profile given the user's feedback on a page.

The process of developing a document representation is denoted document indexing, and this issue is treated in Chapter 4. The representation of the user's interests is very dependent on the choice of method for scoring the documents. Hence, both tasks are discussed in Chapter 5. Finally, the construction of a user profile given the feedback from the user is described in Chapter 6.

Chapter 4

Indexing Web Documents

Indexing is the process of developing a document representation by assigning content descriptors or terms[1] to the document. These terms are used in assessing the relevance of a document to the user's interests. Terms are of two types: objective and non-objective. Examples of objective terms include author name, document URL, and date of publication. The non-objective terms are intended to relate to the information content manifested in the document. While there in general is no disagreement about how to assign objective terms, there are several possible ways of assigning non-objective terms to documents, and in this chapter, we describe some of the approaches that have been used.

The first step in the indexing process usually consists of a transformation of the text into words. This issue is treated in Section 4.1. Indexing can be based on single or multiple words. In Section 4.2 we consider indexing based on single words, while the focus in Section 4.3 is on multi-word or phrase indexing. A major characteristic, or difficulty, of text filtering problems is the high dimensionality of the feature space. Hence, there may be a need for dimensionality reduction. This issue is treated in Section 4.4.

4.1 Text transformation

The first step in the feature extraction process usually consists of a text transformation of following kind:

- Remove HTML mark-up tags
- Recognise individual words
- Use a stop-list to eliminate unwanted words

[1] We use *terms* rather than *words* because the *terms* that are considered may be parts of words, single words or combinations of words.

- Perform suffix removal to generate word stems

Most current systems assume that one can represent the content of a page purely by considering the words contained in the text. Hence, they ignore HTML mark-up tags, images and other multimedia information.

By **stemming** we mean the process of removing prefixes and suffixes from words in a document or a query. This is done to group words that have the same conceptual meaning, such as *walk, walker, walked,* and *walking.* The Porter stemmer is a well-known algorithm for this task.

Stopwords are frequent words that carry no information. These are words like pronouns, prepositions, conjunctions etc. Stopwords are usually removed from the term set of the document.

4.2 Single-term indexing

The perhaps most commonly used document representation is the so called vector space model [25]. In the vector space model, documents are represented by vectors of terms. If m terms are used for content identification, then a document D_i is represented as an m-dimensional vector $D_i = (w_{i1}, ...w_{im})$, where w_{ik} is the weight of term T_k in document D_i. There are several ways of choosing the weights, and in this section we describe three approaches for it: Boolean, statistical, and probabilistic.

4.2.1 Boolean model

Under the Boolean model, each term is viewed as a Boolean variable and has the value TRUE if the term is present in the document. There are no weighting of terms, meaning that all terms are considered to be equally important in representing the document content.

4.2.2 Statistical model

In the statistical model, the weight of a term signifies the term's importance, such as its frequency in the document.

A well-known approach for computing term weights is the TFDIF scheme [25]. The TFDIF scheme assigns weight w_{ik} to term T_k in document D_i in proportion to f_{ik}, the number of occurrences of term T_k in document D_i, and in inverse proportion to n_k, the number of documents in which T_k occurs at least once. The assumptions behind TFDIF are based on two empirical observations regarding text. First, the more times a term appears in a document, the more relevant it is to the topic of the document. Second, the more times the term occurs throughout all documents in the collection to be filtered, the more poorly it discriminates between documents. The weight is given as

$$w_{ik} = f_{ik} * \log(N/n_k) \tag{4.1}$$

where N is the total number of documents in the collection and the factor $log(N/n_k)$ is an inverse collection frequency factor which is small for terms that are widely used in the total collection. It is not very easy to define the meaning of the *total collection* in the context of an information filtering system. Hence, one has no prior knowledge of the inverse collection frequency factor. Estimates of this factor based on previous filtered documents, have however shown to produce useful results.

4.2.3 Probabilistic model

In the probabilistic model, the difference in the distributional behaviour of words over all documents in a collection is used as a guide to whether a word should be assigned as an indexing term. Two types of words are defined, *speciality words* that are content bearing and *function words* that aren't. Bookstein et al. [6] found that function words were randomly distributed according to a Poisson process, whereas speciality words did not follow this distribution. Hence, the content-bearing words can be mechanically detected by measuring the extent to which their distributions deviate from a Poisson process.

4.3 Multi-term or phrase indexing

The indexing schemes in the previous section are based on the assignment of single-term elements to documents. Assignment of single terms to documents is not ideal for two reasons. First, single terms used out of context often carry ambiguous meaning (i.e the word mouse has a different meaning in the context of conputer science and the context of zoology), and conversely, the same concept can be described by many different words (i.e World Wide Web, Web, WWW). Second, many single terms are either too specific or too broad to be useful in indexing. Term phrases, on the other hand, carry more meaning and thus have more discriminating power than the individual terms. For HTML-documents, there is a well-defined underlying structure that could be used. There is also additional information in plain text, for example structure of sentences, position of words or neighbouring words.

Semantic information can be helpful in almost all aspects of natural language understanding, but it is difficult to obtain. In most cases, semantic information is encoded manually, but several techniques have been developed for generating semantic knowledge automatically. Some systems learn the meanings of unknown words using expectations derived from other word definitions in the surrounding context. Other approaches use example- or case-based methods to match unknown word contexts against previously seen word contexts. In the following some of these techniques are described.

Latent semantic indexing (LSI) [3, 4, 10] is an extension of the standard vector space model which was designed to incorporate semantic information. This approach is based on the assumption that there is some underlying or latent structure in the pattern of word usage across documents, and that statistical techniques can be used to estimate this structure. LSI uses singular-value decomposition (SVD), a technique closely related to eigenvector decomposition and factor analysis. SVD takes a large word-by-document matrix and decomposes it into a set of k (typically 100 to 300), orthogonal factors from which the original matrix can be approximated by linear combination. Instead of representing documents and queries directly as vectors of independent words, LSI represents them as continuous values on each of the k dimensions derived from the SVD analysis. One advantage of this approach is that queries can retrieve documents even if they have no words in common. This may occur, for example, if the keywords used in each of the documents co-occur frequently in other documents.

Sorensen et al. [28] use weighted n-grams (n words found to occur after each other a significantly number of times) as opposed to single words. This means that for a document to be interesting, not only the single words have to be interesting, but they also have to appear in the same context as they did in documents previously found interesting by the user.

In [24], Riloff and Shepherd present a corpus-based method that can be used to build semantic dictionaries for specific categories. The input to the system is a small set of seed words for a category and a representative text corpus. The output is a ranked list of words that is associated with the category. A user then reviews the top-ranked words and decides which ones that should be entered in the semantic lexicon. The work in [24] is based on the observation that category members are often surrounded by other category members in text, meaning that given a few category members, one may collect surrounding contexts and use statistics to identify other words that belong to the same category.

4.4 Dimensionality reduction

A central problem in statistical text classification is the high dimensionality of the feature space, where there exists one dimension for each unique term found in the collection of documents, typically hundreds of thousands. Standard classification techniques cannot deal with such a large feature set, since processing is extremely costly in computational terms, and the results become unreliable due to the lack of sufficient training data. Hence, there is a need for a reduction of the original feature set, which is commonly known as dimensionality reduction in the pattern recognition literature. Most of the dimensionality reduction approaches can be classified into one of two categories: feature selection and re-parameterisation.

4.4.1 Feature selection

Feature selection aims at reducing the number of features by selecting the best ones according to some criterion. An approach used by many authors [1, 2, 12, 17, 22] is to prune the words with low weights, where the weights are computed using the TFDIF scheme described in Section 4.2.2. Another method [27] is to determine the most important features by applying a χ^2 measure of dependence to a contingency table containing the number of relevant and irrelevant documents in which the word occurs, and the number of relevant and irrelevant documents in which the word doesn't occur. Some of the machine learning techniques for feature selection may also be used (e.g. sequential forward selection, sequential floating feature selection, and genetic algorithm search [30]). Krulwich et al. [14] use heuristics to extract significant phrases from documents rather than standard mathematical techniques. Their approach is built upon the observation that document authors often use a variety of visual techniques to convey significant pieces of information to readers, such as capitalisation, italicising, underlining, section headings, list of items, and heavily repeated phrases. More information about this issue can be found in [33], which is a comparative study of feature selection methods in statistical learning of text categorisation.

4.4.2 Re-parameterisation

Re-parameterisation is the process of constructing new features as combinations or transformations of the original features. Latent Semantic Indexing [3, 4, 10] (which was described Section 4.3) in is one such approach. In addition to incorporating underlying semantics, it serves as a means of data compression, capturing the important information contained in a large number of words with a much smaller number of factors. This is particularly useful for eliminating the redundancy in word features that is due to word dependence, since the factors are constructed to be orthogonal.

Chapter 5

Document Scoring

In order to determine the similarity of a document to a user profile a typical text filtering system applies a method to the document vector, producing a score. There are a number of such methods, and the choice of method is closely related to the document and the user profile representations. In this chapter we discuss four types of methods, *keyword-based approaches, the vector-space model, probabilistic models*, and *instance-based approaches.*

5.1 Keyword-based approaches

In the keyword-based approaches user interests are specified as arbitrary boolean expressions formed by linking keywords using the standard boolean logical operators AND, OR, and NOT. We describe two such approaches, the boolean model and the slightly different fuzzy set model.

5.1.1 Boolean model

If the boolean expression corresponding to the user interests evaluates to TRUE, the score returned from this model is 1, and otherwise it is 0. The boolean model is simple to implement and many commercial systems are based on it. One disadvantage with this method is that it always returns either 0 or 1. Hence, it is not possible to rank the output. Moreover, the model does not allow weighting of user profile terms to indicate their relative importance.

5.1.2 Fuzzy set model

In this approach we define a membership function M which measures the degree of importance of a term T_k in a document D_i by

$$M(D_i, T_k) = d, \text{ for } 0 <= d <= 1$$

where d for instance may be computed using the TFDIF-scheme described in Section 4.2.2. User interests are expressed as in the case of the boolean model and are also processed in a similar manner. Fuzzy set models have almost the same disadvantages as boolean models. Weighting of user profile words is not possible, and they suffer from lack of discrimination among the output nearly to the same extent as the boolean-based systems.

5.2 Vector space model

In this approach, both the document and the user profile are represented as vectors. The document vector may be produced by one of the methods described in Chapter 4, while methods for learning a user profile are described in Chapter 6. In order to determine the similarity of the document, $D = (w_1, ...w_m)$, to the profile $P = (p_1, ...p_m)$, the dot product, or equivalently the cosine of the angle between the vectors, is used as a metric.

$$f(D) = \sum_{k=1}^{m}(w_k * p_k) \tag{5.1}$$

The effect of this metric can be seen in the following example. Suppose that the document vector and the profile vector each contain a single word, but the words are different. The similarity of the document and the profile will then be zero, since the cosine between the two perpendicular vectors is zero. A more unbiased technique, however, will usually regard the two vectors as being very similar, since all but two of the elements in the vectors agree.

5.3 Probabilistic models

The information-filtering problem is by nature a classification problem; all documents can be classified as belonging to either a positive class, c, (relevant to the user) or a negative class, $\bar{c}$, (irrelevant to the user). Assuming that two training sets, one with relevant documents and one with irrelevant documents, are available, a classification model may be built. Several well-known classification approaches may be chosen for this task, and in this section we will describe in more detail some of these methods.

5.3.1 Linear discriminant analysis

Linear discriminant analysis [9] for the two-class (relevant/irrelevant) problem can be derived as follows. If the mean vectors for the two training sets are μ_c and $\mu_{\bar{c}}$, an unknown document D is classified to be relevant if

$$(\mu_c - \mu_{\bar{c}})' S^{-1} D \geq \frac{1}{2} (\mu_c - \mu_{\bar{c}})' S^{-1} (\mu_c + \mu_{\bar{c}})$$

and classified to be irrelevant otherwise. Here,

$$S = \frac{(N_c - 1) S_c + (N_{\bar{c}} - 1) S_{\bar{c}}}{(N_c + N_{\bar{c}} - 2)} \tag{5.2}$$

where N_c and $N_{\bar{c}}$ are the number of documents in the two training sets and S_c and $S_{\bar{c}}$ are the covariance matrices. Rather than using a pooled estimate of the covariance matrix S, one can estimate a separate covariance matrix for each class. This is an approach known as quadratic discriminant analysis. However, this approach is only effective if the number of feature vectors in each class is significantly larger than the dimension of the feature vectors, which is almost never the case for the information filtering problem.

5.3.2 The Bayesian classifier

According to Bayes' rule [9] a document $D = (w_1, ... w_m)$ should be classified as relevant if

$$P(C = c|D) > P(C = \bar{c}|D) \tag{5.3}$$

where

$$P(C = c|D) = \frac{P(C = c)\, P(D|C = c)}{P(C = \bar{c})\, P(D|C = \bar{c}) + P(C = c)\, P(D|C = c)} \tag{5.4}$$

The denominator in the above equation can be left out. Hence, assuming that the words in the document are conditionally independent, Bayes' rule becomes

$$P(C = c) \prod_{k=1}^{m} P(w_k|C = c) > P(C = \bar{c}) \prod_{k=1}^{m} P(w_k|C = \bar{c}) \tag{5.5}$$

Estimates for the prior probabilities $P(C = c)$ and $P(C = \bar{c})$ and the conditional probabilities $P(w_k|C = c)$ and $P(w_k|C = \bar{c})$ can be determined from the training samples.

5.3.3 Logistic regression

Logistic regression [27] is a statistical technique for modelling a binary response variable (relevant/irrelevant) by a linear combination of one or more predictor variables (document features), using a logic link function. With this model, the probability of relevance of a document is given by

$$p = \frac{1}{1 + \exp(-\beta^T D)} \tag{5.6}$$

If this probability is close to 0, the document is not relevant to the user. On the other hand, if it is close to 1, the document is highly relevant to the user.

The optimal value of the coefficient vector β can be derived from the relevant and irrelevant document vectors in the training set using maximum likelihood and the Newton-Raphson method of numerical optimisation.

5.3.4 Neural networks

A neural network is a network of units, of which some may be hidden. Schütze et al. [27] describe two neural networks for document filtering, one linear and one non-linear. The linear network consists only of two input units and one output unit. The input units correspond to the features of the document, while the output unit correspond to the probability of relevance. The probability of relevance for a document is computed by propagating the activation of the input units through the network to the output unit, and then propagating the error back through the network, using a gradient descent algorithm. As activation function for the units of the network Schütze et al. use the sigmoid

$$f(x) = \frac{e^x}{1 + e^x}.$$

This means that this specific linear neural network performs maximum likelihood estimation of the same model as the logistic regression approach described in the previous section. The main difference lies in the training algorithm, while Newton-Raphson is used for the logistic regression, back-propagation is used for neural networks.

5.3.5 Decision trees

In this approach, the document vector D is matched against a decision tree to determine whether the document is relevant to the user or not. The decision tree is constructed from training samples, and a well-known approach for this task is Quinlan's ID3 algorithm [23]. This approach starts with a set of documents serving as training samples and looks for the most informative component of the document feature vectors, whose values are used to partition the set in as homogenous subsets as possible. For the sake of simplicity, we assume that the training vectors D_i are binary, i.e. $w_k = 1$ if the corresponding term T_k is present in the document and $w_k = 0$ otherwise. Thus, ID3 creates a binary classification tree. The most informative component can be calculated with help of the entropy-formula:

$$-\, p_c \log(p_c) - p_{\bar{c}} \log(p_{\bar{c}}) \tag{5.7}$$

where p_c and $p_{\bar{c}}$ are the frequencies of a relevant and irrelevant document, respectively. For each component of the document vectors, the entropy resulting when this component is chosen as the next decision node, is computed. The component with the highest difference between the entropy of the old set and the weighted sum of the entropies of the new partitions is selected. This procedure is repeated until no sets can be partitioned any further.

5.3.6 Belief networks

Belief networks are directed a-cyclic graphs used to represent probability distributions. They have been referred to by many other names, including Bayesian networks and probabilistic expert systems. The knowledge base is represented by a qualitative description of the dependencies between the variables in the system, and the quantitative description of the numerical values of those dependencies. An example of a Bayesian network is given in [13], where Koller et al. propose an approach that utilises the hierarchical topic structure of documents to decompose the classification tasks into a set of simpler problems.

5.4 Instance-based approaches

The instance-based approaches differ from the approaches in the previous section in that they store all training samples together with their class label (relevant/irrelevant) and postpone any form of generalisation until classification time. In this section we will describe two such methods; the nearest neighbour classifier and the exemplar-based learning.

5.4.1 The nearest neighbour classifier

To classify an unknown document D, the k-nearest neighbour (kNN) algorithm [9] ranks the document's neighbours among the training documents, and use the class labels of the k most similar neighbours to predict the class of the input document. The classes of these neighbours are weighted using the similarity of each neighbour to D, where similarity may be measured by for example the Euclidean distance or the cosine between the two document vectors.

5.4.2 Exemplar-based learning

Foltz et al. [10] apply an exemplar-based learning technique to a selection of internet news articles. In a training set he had 100 articles, which the user designated as interesting, and then ranked new articles by the cosine between their representation and the nearest article in the training set. In other words, articles were given a high score if they were very similar to at least one training sample.

Chapter 6

User Profile Construction

If an information filtering system is to be of assistance, it requires knowledge about the user. The most common and simplest way is to ask the user to type in the keywords that he is interested in [31]. An alternative is to acquire knowledge about the user through learning. Many systems employ machine learning techniques to induce a user profile. This does not only eliminate the need for users to define their own keywords, but will also allow the system to adapt to changes. However, for the system to be really intelligent and adaptive, it should be able to learn from the user's behaviour. Many systems let the user's behaviour be reflected through his choice of documents and through his feedback for these documents. In Section 6.1 we discuss different ways of obtaining feedback from the user, while the actual learning process is described in Section 6.2

6.1 User feedback

Many of the classification methods described in Chapter 5 require a set of positive examples (the documents that are relevant to the user) and a set of negative examples (the documents that are irrelevant to the user). Hence, for many systems, when the user access the system for the first time, he is often asked to provide feedback on a number of documents. The system formalises this information and build an initial user profile based on it. An alternative procedure which does not require interaction from the user, is to let the user start with a very comprehensive profile. In the beginning he then will get many irrelevant documents, but after a while the system will learn from his feedback and update the profile relative to his response.

Feedback from the user may be obtained in two ways; the user can either explicitly evaluate documents or the system can observe some implicit evidence about the user's interest in each document. When it comes to explicit feedback, the user usually provides a score for the documents retrieved by the system. Explicit feedback from the user requires his conscious involvement, which is often annoying. Hence, there may be a need for methods for implicit registration of user interests. In a news filtering system, for instance, information of the user's interest in a message may be obtained from the answers of the following questions:

- Did the user read message ?
- Did the user save the message ?
- Did the user reply the message ?

For a more general filtering system, one could for instance consider the time that the user spends on each article [21]. The problem with this approach, is that it assumes that users devote their entire concentration to the task of reading the article and don't take breaks, which is not a very realistic assumption.

6.2 Adaptive Learning

In Chapter 5 we described four types of methods for determining the similarity between a document and a user profile; keyword-based approaches, instance-based approaches, probabilistic models, and the vector space model. The keyword-based approaches rely on the user to specify the keywords of the user profile and hence do not require any learning process. This is also the case for the instance-based approaches, which postpone any form of generalisation until classification time. The way of learning the parameters of the probabilistic models was indicated when describing the models (see Chapter 5). Hence, only the vector space model remains, and in this section we will describe different approaches for learning a user profile under this model. Throughout this section, the class labels of the training document (relevant/irrelevant) are assumed to be known, either because the user explicitly has evaluated the documents or because the system has observed some implicit evidence about the user's evaluation in each document.

6.2.1 The Rocchio Algorithm

The Rocchio algorithm [25] is one of the most widely applied learning algorithms for text classification. A document is represented as a vector of weighted terms, where the weights are computed using the TFDIF weighting system described in Section 4.2. Learning is achieved by combining document vectors into a profile $P = (p_1, ...p_m)$. An initial profile can be computed as the weighted difference between the average of the relevant training samples, μ_c, and the average of the irrelevant training samples, $\mu_{\bar{c}}$,

$$P = \alpha\mu_1 - \beta\mu_0 \tag{6.1}$$

where the parameters α and β control the relative impact of positive and negative examples, respectively. The lengths of the document vectors are usually normalised before the computations. Typically, the profile resulting from the Rocchio algorithm is restricted to have non-negative weights, meaning that if the computed p_k's are negative, they are set to 0. The initial profile may be used to fetch new documents to the user, and the feedback from the user (relevant or irrelevant) can be utilised to update the initial profile:

$$P^{new} = \gamma\, P^{old} + \alpha\mu_c - \beta\mu_{\bar{c}} \tag{6.2}$$

In [7] γ, α, and β are set to 8, 16, and 4, respectively.

6.2.2 The Widrow-Hoff Algorithm

While the TFIDF classifier computes the profile from the entire set of training data at once, the Widrow-Hoff Algorithm [18] runs through the training documents one at a time updating the profile vector at each step. The new profile is computed from the old profile using training document D_i with class label C_i (1 if the document is relevant and 0 otherwise):

$$P^{new} = P^{old} - 2\eta(P^{old}\, D_i - C_i)D_i. \tag{6.3}$$

Initially the profile is typically set to the all zeros vector $P = (0, 0, .., 0)$, but other initial settings are possible. The parameter η (which is always greater than 0) controls how quickly the profile is allowed to change. Moreover it determines the degree of influence of each new training document on the profile.

6.2.3 Kivinen & Warmuth's EG Algorithm

This algorithm is similar to the Widrow-Hoff algorithm in that it maintains a profile P and runs through training documents one at a time. With the EG algorithm [18], however, the components of the profile are restricted to be non-negative and sum to one. In this approach the user profile is initially set to $P = (1/m, ...1/m)$ and the jth component of the new profile is computed from the old profile using training document D_i with class label C_i:

$$p_j^{new} = \frac{p_j^{old} \exp(-2\eta(P^{old}\, D_i - C_i)w_{ij})}{\sum_{k=1}^{t} p_j^{old} \exp(-2\eta(P^{old}\, D_i - C_i)w_{ik})} \tag{6.4}$$

As before, the learning rate η controls the impact of each new training example.

Chapter 7

Conclusion

Designers of personalised information filtering systems can benefit from research in text retrieval, user modeling, and a number of other fields. Personalised information filtering is, however, a unique information seeking process that is distinguished by a focus on satisfying relatively stable user interests. This report has reviewed progress in the field with particular emphasis on content-based information filtering systems.

Information filtering systems must develop representations of both documents and user interests, they must be equipped with some way of comparing documents with interests, and they must possess some way of using the results of those comparisons to assist the user with document selection. Research in this area has produced a number of content-based representations, of which the majority uses the frequency with which words appear in documents. When combined with implicit or explicit feedback from the user about documents they have examined, those representations provide a basis for construction of profiles which represent the user's interests. Both text retrieval and machine learning offer techniques for comparing document representations with profiles, and this is still an area of active research.

During the last few years we have witnessed an intensive research in the form of both comercially and freely available information filtering systems offering a range of solutions that can help users achieve their information seeking goals. The complexity of models range from straightforward rule-based approaches to more complex machine learning methods. Many problem areas have been identified and targeted, such as filtering e-mail, locating relevant information sites, and filtering Usenet news. While there still remain additional issues that need to be adressed, the results so far indicate that effective systems for personalised information filtering can be developed. As the quantity of online information continues to increase, such systems will play an even bigger part in bringing together producers and consumers of information.

Bibliography

[1] R. Armstrong, D. Freitag, T. Joachims and T. Mitchell, *WebWatcher: A Learning Apprentice for the World Wide Web*, In AAAI Spring Symposium on Information Gathering, Stanford, CA, March 1995.

[2] M. Balabanović and Y. Shoham, *Learning Information Retrieval Agents: Experiments with Automated Web Browsing*, AAAI-95 Spring Symposium on Information Gathering from Heterogenous, Distributed Environments, 1995

[3] B. T. Bartell, G. W. Cotrell and R. K. Belew, *Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling*, Proc. ACM SIG Information Retrieval, Copenhagen 1992.

[4] M. W. Berry, S. T. Dumais and G. W. O'Brien, *Using linear algebra for intelligent information retrieval*, SIAM Review, Vol. 37, pp. 573–595, December 1995.

[5] E. Bloedhorn, I. Mani and T. R. MacMillan, *Representational Issues in Machine Learning of User Profiles*, AAAI Spring Symposium on Machine Learning in Information Access, 1996.

[6] A. Bookstein and D. R. Swanson, *Probabilistic models for automatic indexing*, Journal of the American Society for Information Sciene, Vol. 25, pp. 312–318, 1974.

[7] C. Buckley, G. Salton, J. Allan and A. Singhal, *Automatic Query Expansion Using SMART: TREC3*, Proc. 3rd Text Retrieval Conf., NIST, 1994.

[8] W. W. Cohen, *Learning to Classify English Text with ILP Methods*, Workshop on Inductive Logical Programming, Leuven, September 1995.

[9] R. O. Duda and P. E. Hart, *Pattern Classification and Scene analysis*, John Wiley & Sons, 1973.

[10] P. W. Foltz and S. T. Dumais, *Personalized information delivery: An analysis of information filtering methods*, Communications of the ACM, Vol. 35, pp. 51–60, 1992.

[11] D. Goldberg, D. Nichols, B. M. Oki and D. Terry, *Using collaborative filtering to weave an information tapestry*, Communications of the ACM, Vol. 35, pp. 61–70, 1992.

[12] T. Joachims, *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization*, Technical report, CMU-CS-96-118, Carnegie Mellon, March 1996.

[13] D. Koller and M. Sahami, *Hierarchically Classifying Documents Using Very Few Words*, Proc. 14th Int. Conf. on Machine Learning, 1997.

[14] B. Krulwich and C. Burkey, *Learning user information interests through the extraction of semantically significant phrases*, AAAI Spring Symposium on Machine Learning in Information Access, 1996.

[15] K. Lang, *NewsWeeder: Learning to filter netnews*, Proc. 12th Int. Mach. Learning Conf. (ICML'95), San Francisco, pp. 3311-339, 1995.

[16] Y. Lashkari, M. Metral and P. Maes, *Collaborative Interface Agents*, In Proc. of the 12th Nat. Conf. Artificial Intelligence, 1995.

[17] D. D. Lewis and W. A. Gale, *A Sequential Algorithm for Training Text Classifiers*, Proc. 7th Annual Int. ACM-SIGIR Conf. on Research and Development in Information Retrieval, Dubline, 1994.

[18] D. D. Lewis, R. E. Shapire, J. P. Callan and R. Papka, *Training algorithms for linear text classifiers*, Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 298–306, 1996.

[19] H. Lieberman, *Letizia: An agent That Assists Web Browsing*, Int. Joint Conf. on Artificial Intelligence, Montreal, August 1995.

[20] D. Mladenić, *Personal WebWatcher: design and implementation*, Technical Report IJS-DP-7472, Carnegie Mellon, 1996.

[21] D. S. W. Ngu and X. Wu, *Site Helper: A Localized Agent that Helps Incremental Exploration of the World Wide Web*, Proc. 6th Int. World Wide Web Conf., Santa Clara California USA, April 1997.

[22] M. Pazzani, J. Muramatsu and D. Billsus, *Syskill & Webert: Identifying interesting web sites*, AAAI Spring Symposium on Machine Learning in Information Access, Stanford, May 1996.

[23] J. R. Quinlan, *Learning efficient classification procedures and their application to chess end games*, In Machine Learning, An Artificial Intelligence Approach, pp. 463–482, Tioga Publishing Company, Palo Alto, CA, 1983.

[24] E. Riloff and J. Shepherd, *A Corpus-Based Approach for Building Semantic Lexicons* Proc. 2nd Conf. on Empirical Methods in Natural Language Processing, 1997.

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

[26] B. D. Sheth, *A Learning Approach to Personalized Information Filtering*, Master Thesis, MIT, 1994.

[27] H. Schütze, D. A. Hull and J. O. Pedersen, *A Comparison of Classifiers and Document Representations for the Routing Problem*, Proc. SIGIR-95, pp. 229–237, 1995.

[28] H. Sorensen and M. Mc Elligott, *PSUN: A Profiling System for Usenet News*, CIKM'95 Intelligent Information Agents Workshop, Baltimore, December 1995.

[29] E. Wiener, J. Pedersen and A. S. Weigend, *A neural network approach to topic spotting*, Proc. 4th Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, 1995.

[30] M. R. Wulfekuhler and W. F. Punch, *Finding Salient Features for Personal Web Page Categories*, Proc. 6th Int. World Wide Web Conf., April 1997.

[31] T. W. Yan and H. Garia-Molina, *SIFT - A Tool for Wide-Area Information Dissemination*, Proc. 1995 USENIX Technical Conf., pp. 177-186, 1995.

[32] Y. Yang, *Sampling Strategies and Learning Efficiency in Text Categorization*, AAAI Spring Symposium on Machine Learning in Information Access, Stanford, March 1996.

[33] Y. Yang and J. O. Pedersen, *A Comparative Study on Feture Selection in Text Categorization*, Proc. 14th Int. Conf. on Machine Learning ICML'97, 1997.