Content-Based Recommendation Systems

Michael J. Pazzani[1] and Daniel Billsus[2]

[1] Rutgers University, ASBIII, 3 Rutgers Plaza
New Brunswick, NJ 08901
pazzani@rutgers.edu
[2] FX Palo Alto Laboratory, Inc., 3400 Hillview Ave, Bldg. 4
Palo Alto, CA 94304
billsus@fxpal.com

Abstract. This chapter discusses content-based recommendation systems, i.e., systems that recommend an item to a user based upon a description of the item and a profile of the user's interests. Content-based recommendation systems may be used in a variety of domains ranging from recommending web pages, news articles, restaurants, television programs, and items for sale. Although the details of various systems differ, content-based recommendation systems share in common a means for describing the items that may be recommended, a means for creating a profile of the user that describes the types of items the user likes, and a means of comparing items to the user profile to determine what to recommend. The profile is often created and updated automatically in response to feedback on the desirability of items that have been presented to the user.

10.1 Introduction

A common scenario for modern recommendation systems is a Web application with which a user interacts. Typically, a system presents a summary list of items to a user, and the user selects among the items to receive more details on an item or to interact with the item in some way. For example, online news sites present web pages with headlines (and occasionally story summaries) and allow the user to select a headline to read a story. E-commerce sites often present a page with a list of individual products and then allow the user to see more details about a selected product and purchase the product. Although the web server transmits HTML and the user sees a web page, the web server typically has a database of items and dynamically constructs web pages with a list of items. Because there are often many more items available in a database than would easily fit on a web page, it is necessary to select a subset of items to display to the user or to determine an order in which to display the items.

Content-based recommendation systems analyze item descriptions to identify items that are of particular interest to the user. Because the details of recommendation systems differ based on the representation of items, this chapter first discusses alternative item representations. Next, recommendation algorithms suited for each representation are discussed. The chapter concludes with a discussion of variants of the approaches, the strengths and weaknesses of content-based recommendation systems, and directions for future research and development.

10.1.1 Item Representation

Items that can be recommended to the user are often stored in a database table. Table 10.1 shows a simple database with records (i.e., “rows”) that describe three restaurants. The column names (e.g., Cuisine or Service) are properties of restaurants. These properties are also called “attributes,” “characteristics,” “fields,” or “variables” in different publications. Each record contains a value for each attribute. A unique identifier, ID in Table 10.1, allows items with the same name to be distinguished and serves as a key to retrieve the other attributes of the record.

Table 10.1. A restaurant database

ID	Name	Cuisine	Service	Cost
10001	Mike’s Pizza	Italian	Counter	Low
10002	Chris’s Cafe	French	Table	Medium
10003	Jacques Bistro	French	Table	High

The database depicted in Table 10.1 could be used to drive a web site that lists and recommends restaurants. This is an example of structured data in which there is a small number of attributes, each item is described by the same set of attributes, and there is a known set of values that the attributes may have. In this case, many machine learning algorithms may be used to learn a user profile, or a menu interface can easily be created to allow a user to create a profile. The next section of this chapter discusses several approaches to creating a user profile from structured data.

Of course, a web page typically has more information than is shown in Table 10.1, such as a text description of the restaurant, a restaurant review, or even a menu. These may easily be stored as additional fields in the database and a web page can be created with templates to display the text fields (as well as the structured data). However, free text data creates a number of complications when learning a user profile. For example, a profile might indicate that there is an 80% probability that a particular user would like a French restaurant. This might be added to the profile because a user gave a positive review of four out of five French restaurants. However, unrestricted text fields are typically unique and there would be no opportunity to provide feedback on five restaurants described as “A charming café with attentive staff overlooking the river.”

An extreme example of unstructured data may occur in news articles. Table 10.2 shows an example of a part of a news article. The entire article can be treated as a large unrestricted text field.

Table 10.2. Part of a newspaper article

Lawmakers Fine-Tuning Energy Plan
SACRAMENTO, Calif. -- With California's energy reserves remaining all but depleted, lawmakers prepared to work through the weekend fine-tuning a plan Gov. Gray Davis says will put the state in the power business for "a long time to come." The proposal involves partially taking over California's two largest utilities and signing long-term contracts of up to 10 years to buy electricity from wholesalers.

Unrestricted texts such as news articles are examples of unstructured data. Unlike structured data, there are no attribute names with well-defined values. Furthermore, the full complexity of natural language may be present in the text field including polysemous words (the same word may have several meanings) and synonyms (different words may have the same meaning). For example, in the article in Table 10.2, "Gray" is a name rather than a color, and "power" and "electricity" refer to the same underlying concept.

Many domains are best represented by semi-structured data in which there are some attributes with a set of restricted values and some free-text fields. A common approach to dealing with free text fields is to convert the free text to a structured representation. For example, each word may be viewed as an attribute, with a Boolean value indicating whether the word is in the article or with an integer value indicating the number of times the word appears in the article.

Many personalization systems that deal with unrestricted text use a technique to create a structured representation that originated with text search systems [34]. In this formalism, rather than using words, the root forms of words are typically created through a process called stemming [30]. The goal of stemming is to create a term that reflects the common meaning behind words such as "compute," "computation," "computer" "computes" and "computers." The value of a variable associated with a term is a real number that represents the importance or relevance. This value is called the *tf*idf* weight (term-frequency times inverse document frequency). The *tf*idf* weight, *w(t,d)*, of a term *t* in a document *d* is a function of the frequency of *t* in the document (tf_t,d), the number of documents that contain the term (df_t) and the number of documents in the collection *(N)*.[1]

$$w(t,d) = \frac{tf_{t,d} \log\left(\frac{N}{df_t}\right)}{\sqrt{\sum_i (tf_{t_i,d})^2 \log\left(\frac{N}{df_{t_i}}\right)^2}} \quad \textbf{(10.1)}$$

Table 10.3 shows the *tf*idf* representation (also called the vector space representation) of the complete article excerpted in Table 10.2. The terms are ordered by the *tf*idf* weight. The intuition behind the weight is that the terms with the highest weight occur more often in that document than in the other documents, and therefore are more central to the topic of the document. Note that terms such as "util" (a stem of "utility"), "power," "megawatt," are among the highest weighted terms capturing the meaning.

[1] Note that in the description of *tf*idf* weights, the word "document" is traditionally used since the original motivation was to retrieve documents. While the chapter will stick with the original terminology, in a recommendation system, the documents correspond to a text description of an item to be recommended. Note that the equations here are representative of the class of formulae called *tf*idf*. In general, *tf*idf* systems have weights that increase monotonically with term frequency and decrease monotonically with document frequency.

Table 10.3. *tf*idf* representation of the article in Table 10.2

util-0.339 power-0.329 megawatt-0.309 electr-0.217 energi-0.206 california-0.181 debt-0.128 lawmak-0.128 state-0.122 wholesal-0.119 partial-0.106 consum-0.105 alert-0.103 scroung-0.096 advoc-0.09 testi-0.088 bail-out-0.088 crisi-0.085 amid-0.084 price-0.083 long-0.082 bond-0.081 plan-0.081 term-0.08 grid-0.078 reserv-0.077 blackout-0.076 bid-0.076 market-0.074 fine-0.073 deregul-0.07 spiral-0.068 deplet-0.068 liar-0.066.

Of course, this representation does not capture the context in which a word is used. It loses the relationships between words in the description. For example, a description of a steak house might contain the sentence, “there is nothing on the menu that a vegetarian would like” while the description of a vegetarian restaurant might mention “vegan” rather than vegetarian. In a manually created structured database, the cuisine attribute having a value of “vegetarian” would indicate that the restaurant is indeed a vegetarian one. In contrast, when converting an unstructured text description to structured data, the presence of the word vegetarian does not always indicate that a restaurant is vegetarian and the absence of the word vegetarian does not always indicate that the restaurant is not a vegetarian restaurant. As a consequence, techniques for creating user profiles that deal with structured data need to differ somewhat from those techniques that deal with unstructured data or unstructured data automatically and imprecisely converted to structured data.

One variant on using words as terms is to use sets of contiguous words as terms. For example, in the article in Table 10.2, terms such as “energy reserves” and “power business” might be more descriptive of the content than these words treated as individual terms. Of course, terms such as “all but” would also be included, but one would expect that these have very low weights, in the same way that “all” and “but” individually have low weights and are not among the most important terms in Table 10.3.

10.2 User Profiles

A profile of the user’s interests is used by most recommendation systems. This profile may consist of a number of different types of information. Here, we concentrate on two types of information:

1. A model of the user’s preferences, i.e., a description of the types of items that interest the user. There are many possible alternative representations of this description, but one common representation is a function that for any item predicts the likelihood that the user is interested in that item. For efficiency purposes, this function may be used to retrieve the *n* items most likely to be of interest to the user.
2. A history of the user’s interactions with the recommendation system. This may include storing the items that a user has viewed together with other information about the user’s interaction, (e.g., whether the user has purchased the item or a rating that the user has given the item). Other types of history include saving queries typed by the user (e.g., that a user searched for an Italian restaurant in the 90210 zip code).

There are several uses of the history of user interactions. First, the system can simply display recently visited items to facilitate the user returning to these items. Second, the system can filter out from a recommendation system an item that the user has already purchased or read.[2] Another important use of the history in content-based recommendation systems is to serve as training data for a machine learning algorithm that creates a user model. The next section will discuss several different approaches to learning a user model. Here, we briefly describe approaches of manually providing the information used by recommendation systems: user customization and rule-based recommendation systems.

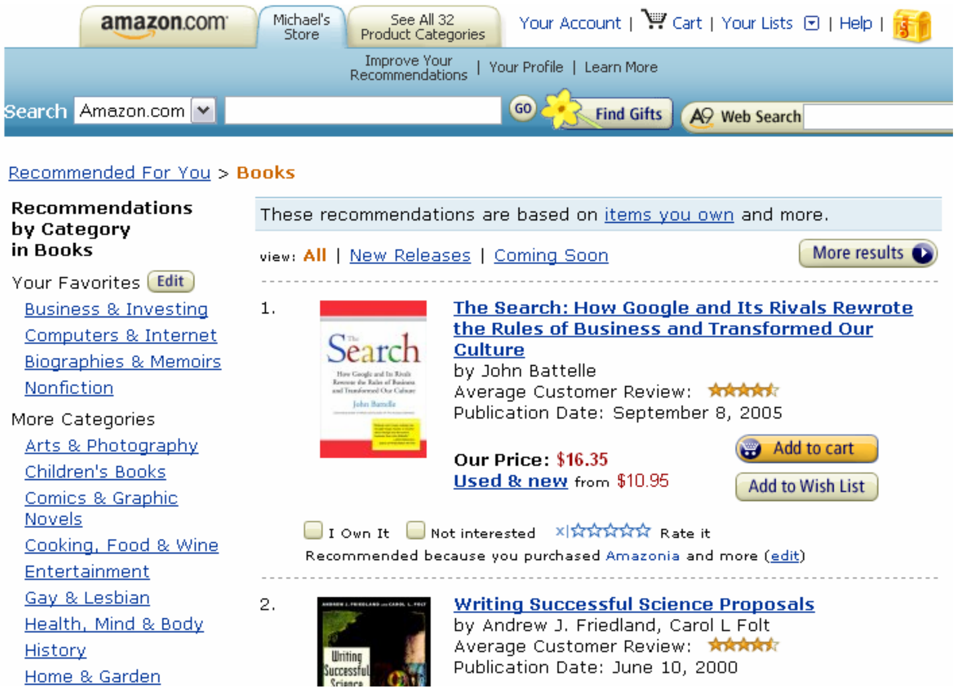
In user customization, a recommendation system provides an interface that allows users to construct a representation of their own interests. Often check boxes are used to allow a user to select from the known values of attributes, e.g., the cuisine of restaurants, the names of favorite sports teams, the favorite sections of a news site, or the genre of favorite movies. In other cases, a form allows a user to type words that occur in the free text descriptions of items, e.g., the name of a musician or author that interests the user. Once the user has entered this information, a simple database matching process is used to find items that meet the specified criteria and display them to the user.

There are several limitations of user customization systems. First, they require effort from the user and it is difficult to get many users to make this effort. This is particularly true when the user's interests change, e.g., a user may not follow football during the season but then become interested in the Superbowl. Second, customization systems do not provide a way to determine the order in which to present items and can find either too few or too many matching items to display.

Figure 10.1 shows book recommendations at Amazon.com. Although Amazon.com is usually thought of as a good example of collaborative recommendation (see Chapter 9 of this book [35]), parts of the user's profile can be viewed as a content-based profile. For example, Amazon contains a feature called "favorites" that represents the categories of items preferred by users. These favorites are either calculated by keeping track of the categories of items purchased by users or may be set manually by the user. Figure 10.2 shows an example of a user customization interface in which a user can select the categories.

In rule-based recommendation systems, the recommendation system has rules to recommend other products based on the user history. For example, a system may contain a rule that recommends the sequel to a book or movie to people who have purchased the early item in the series. Another rule might recommend a new CD by an artist to users that purchased earlier CDs by that artist. Rule-based systems may capture several common reasons for making recommendations, but they do not offer the same detailed personalized recommendations that are available with other recommendation systems.

[2] Of course, in some situations it is appropriate to recommend an item the user has purchased and in other situations it is not. For example, a system should continue to recommend an item that wears out or is expended, such as a razor blade or print cartridge, while there is little value in recommending a CD or DVD a user owns.


amazon.com
Michael's Store
See All 32 Product Categories
Your Account | Cart | Your Lists | Help |
Improve Your Recommendations | Your Profile | Learn More
Search Amazon.com GO Find Gifts Web Search

Recommended For You > Books

Recommendations by Category in Books

Your Favorites Edit
- Business & Investing
- Computers & Internet
- Biographies & Memoirs
- Nonfiction

More Categories
- Arts & Photography
- Children's Books
- Comics & Graphic Novels
- Cooking, Food & Wine
- Entertainment
- Gay & Lesbian
- Health, Mind & Body
- History
- Home & Garden

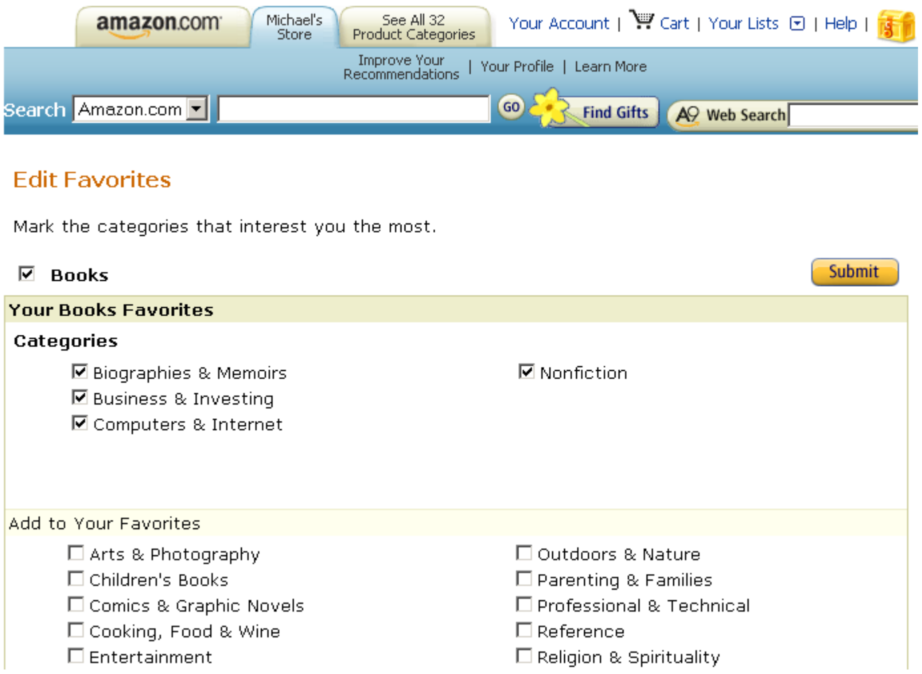
These recommendations are based on items you own and more.

view: All | New Releases | Coming Soon — More results

1. **The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture**
by John Battelle
Average Customer Review:
Publication Date: September 8, 2005
Our Price: $16.35
Used & new from $10.95
Add to cart — Add to Wish List
I Own It — Not interested — Rate it
Recommended because you purchased Amazonia and more (edit)

2. **Writing Successful Science Proposals**
by Andrew J. Friedland, Carol L Folt
Average Customer Review:
Publication Date: June 10, 2000

Fig. 10.1. Book recommendations by Amazon.com.


amazon.com
Michael's Store
See All 32 Product Categories
Your Account | Cart | Your Lists | Help |
Improve Your Recommendations | Your Profile | Learn More
Search Amazon.com GO Find Gifts Web Search

Edit Favorites

Mark the categories that interest you the most.

[✓] **Books** — Submit

Your Books Favorites

Categories

[✓] Biographies & Memoirs	[✓] Nonfiction
[✓] Business & Investing	
[✓] Computers & Internet	

Add to Your Favorites

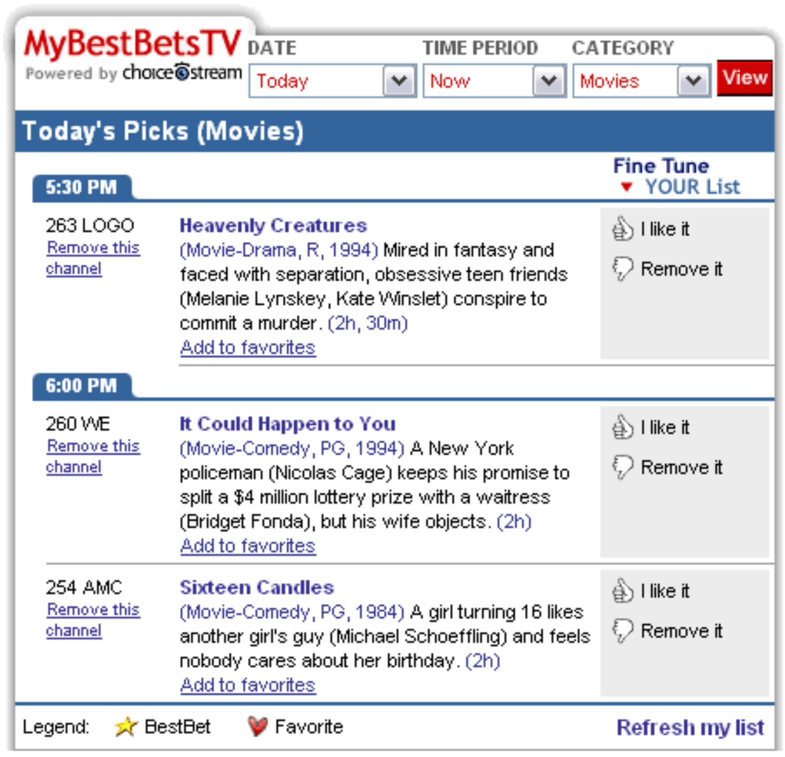
☐ Arts & Photography	☐ Outdoors & Nature
☐ Children's Books	☐ Parenting & Families
☐ Comics & Graphic Novels	☐ Professional & Technical
☐ Cooking, Food & Wine	☐ Reference
☐ Entertainment	☐ Religion & Spirituality

Fig. 10.2. User customization in Amazon.com

10.3 Learning a User Model

Creating a model of the user's preference from the user history is a form of classification learning. The training data of a classification learner is divided into categories, e.g., the binary categories "items the user likes" and "items the user doesn't like." This is accomplished either through explicit feedback in which the user rates items via some interface for collecting feedback or implicitly by observing the user's interactions with items. For example, if a user purchases an item, that is a sign that the user likes the item, while if the user purchases and returns the item that is a sign that the user doesn't like the item. In general, there is a tradeoff since implicit methods can collect a large amount of data with some uncertainty as to whether the user actually likes the item. In contrast, when the user explicitly rates items, there is little or no noise in the training data, but users tend to provide explicit feedback on only a small percentage of the items they interact with.

Figure 10.3 shows an example of a recommendation system with explicit user feedback. The recommender "MyBestBets" by ChoiceStream is a web based interface to a television recommendation system. Users can click on the thumbs up or thumbs down buttons to indicate whether they like the program that is recommended. By necessity, this system requires explicit feedback because it is not integrated with a television [1] and cannot infer the user's interests by observing the user's behavior.



Today's Picks (Movies)

5:30 PM

263 LOGO
Remove this channel

Heavenly Creatures
(Movie-Drama, R, 1994) Mired in fantasy and faced with separation, obsessive teen friends (Melanie Lynskey, Kate Winslet) conspire to commit a murder. (2h, 30m)
Add to favorites

6:00 PM

260 WE
Remove this channel

It Could Happen to You
(Movie-Comedy, PG, 1994) A New York policeman (Nicolas Cage) keeps his promise to split a $4 million lottery prize with a waitress (Bridget Fonda), but his wife objects. (2h)
Add to favorites

254 AMC
Remove this channel

Sixteen Candles
(Movie-Comedy, PG, 1984) A girl turning 16 likes another girl's guy (Michael Schoeffling) and feels nobody cares about her birthday. (2h)
Add to favorites

Fig. 10.3. A recommendation system using explicit feedback

The next section reviews a number of classification learning algorithms. Such algorithms are the key component of content-based recommendation systems, because they learn a function that models each user's interests. Given a new item and the user model, the function predicts whether the user would be interested in the item. Many of the classification learning algorithms create a function that will provide an estimate of the probability that a user will like an unseen item. This probability may be used to sort a list of recommendations. Alternatively, an algorithm may create a function that directly predicts a numeric value such as the degree of interest.

Some of the algorithms below are traditional machine learning algorithms designed to work on structured data. When they operate on free text, the free text is first converted to structured data by selecting a small subset of the terms as attributes. In contrast, other algorithms are designed to work in high dimensional spaces and do not require a preprocessing step of feature selection.

10.4 Decision Trees and Rule Induction

Decision tree learners such as ID3 [31] build a decision tree by recursively partitioning training data, in this case text documents, into subgroups until those subgroups contain only instances of a single class. A partition is formed by a test on some feature -- in the context of text classification typically the presence or absence of an individual word or phrase. Expected information gain is a commonly used criterion to select the most informative features for the partition tests [38].

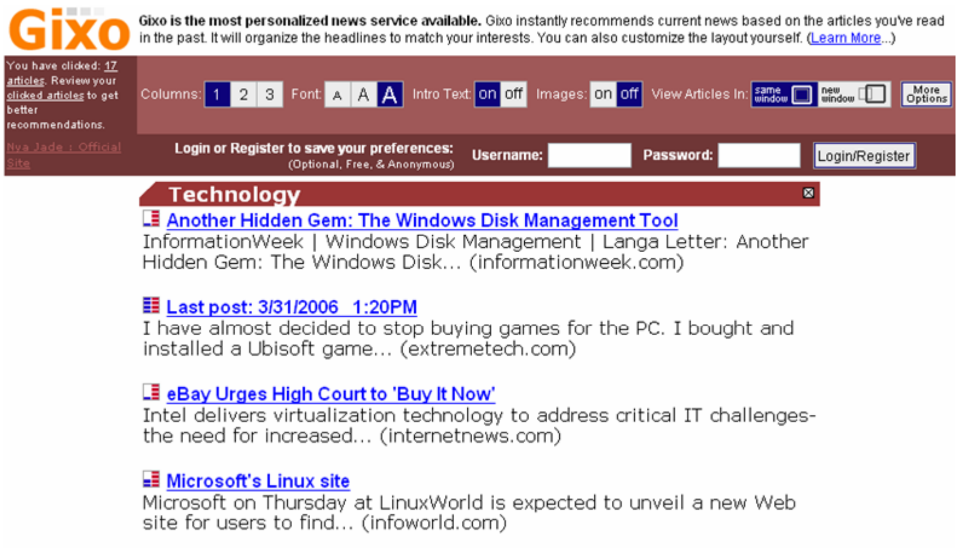
Decision trees have been studied extensively in use with structured data such as that shown in Table 10.1. Given feedback on the restaurants, a decision tree can easily represent and learn a profile of someone who prefers to eat in expensive French restaurants or inexpensive Mexican restaurants. Arguably, the decision tree bias is not ideal for unstructured text classification tasks [29]. As a consequence of the information-theoretic splitting criteria used by decision tree learners, the inductive bias of decision trees is a preference for small trees with few tests. However, it can be shown experimentally that text classification tasks frequently involve a large number of relevant features [17]. Therefore, a decision tree's tendency to base classifications on as few tests as possible can lead to poor performance on text classification. However, when there are a small number of structured attributes, the performance, simplicity and understandability of decision trees for content-based models are all advantages. Kim et al. [18] describe an application of decision trees for personalizing advertisements on web pages.

RIPPER [9] is a rule induction algorithm closely related to decision trees that operates in a similar fashion to the recursive data partitioning approach described above. Despite the problematic inductive bias, however, RIPPER performs competitively with other state-of-the-art text classification algorithms. In part, the performance can be attributed to a sophisticated post-pruning algorithm that optimizes the fit of the induced rule set with respect to the training data as a whole. Furthermore, RIPPER supports multi-valued attributes, which leads to a natural representation for text classification tasks, i.e., the individual words of a text document can be represented as multiple feature values for a single feature. While this is essentially a representational convenience if rules are to be learned from unstructured text documents, the approach can lead to more powerful classifiers for semi-structured text documents. For example, the text contained in separate fields of an email message, such as sender, subject, and body text, can be represented as separate multi-valued features, which allows the algorithm to take advantage of the document's structure in a natural fashion. Cohen [10] shows how RIPPER can classify e-mail messages into user defined categories.

10.5 Nearest Neighbor Methods

The nearest neighbor algorithm simply stores all of its training data, here textual descriptions of implicitly or explicitly labeled items, in memory. In order to classify a new, unlabeled item, the algorithm compares it to all stored items using a similarity function and determines the "nearest neighbor" or the *k* nearest neighbors. The class label or numeric score for a previously unseen item can then be derived from the class labels of the nearest neighbors.

The similarity function used by the nearest neighbor algorithm depends on the type of data. For structured data, a Euclidean distance metric is often used. When using the vector space model, the cosine similarity measure is often used [34]. In the Euclidean distance function, the same feature having a small value in two examples is treated the same as that feature having a large value in both examples. In contrast, the cosine similarity function will not have a large value if corresponding features of two examples have small values. As a consequence, it is appropriate for text when we want two documents to be similar when they are about the same topic, but not when they are both not about a topic.


Gixo is the most personalized news service available. Gixo instantly recommends current news based on the articles you've read in the past. It will organize the headlines to match your interests. You can also customize the layout yourself. (Learn More...)

You have clicked: 17 articles. Review your clicked articles to get better recommendations.

Nya Jade : Official Site

Columns: [✓ 1] 2 3 Font: A A [✓ A] Intro Text: [✓ on] off Images: on [✓ off] View Articles In: [✓ same window] new window More Options

Login or Register to save your preferences: (Optional, Free, & Anonymous) Username: Password: Login/Register

Technology

Another Hidden Gem: The Windows Disk Management Tool
InformationWeek | Windows Disk Management | Langa Letter: Another Hidden Gem: The Windows Disk... (informationweek.com)

Last post: 3/31/2006 1:20PM
I have almost decided to stop buying games for the PC. I bought and installed a Ubisoft game... (extremetech.com)

eBay Urges High Court to 'Buy It Now'
Intel delivers virtualization technology to address critical IT challenges- the need for increased... (internetnews.com)

Microsoft's Linux site
Microsoft on Thursday at LinuxWorld is expected to unveil a new Web site for users to find... (infoworld.com)

Fig. 10.4. Gixo presents personalized news based on similarity to articles that have previously been read

The vector space approach and the cosine similarity function have been applied to several text classification applications ([11], [39], [2]) and, despite the algorithm's unquestionable simplicity, it performs competitively with more complex algorithms. The Daily Learner system uses the nearest neighbor algorithm to create a model of the user's short term interests [7]. Gixo, a personalized news system, also uses text similarity as a basis for recommendation (Figure 10.4). The headlines are preceded by an icon that indicates how popular the item is (the first bar) and how similar the story is to stories that have been read by the user before (the second bar). The fact that these bars differ shows the value of personalizing to the individual.

10.6 Relevance Feedback and Rocchio's Algorithm

Since the success of document retrieval in the vector space model depends on the user's ability to construct queries by selecting a set of representative keywords [34], methods that help users to incrementally refine queries based on previous search results have been the focus of much research. These methods are commonly referred to as relevance feedback. The general principle is to allow users to rate documents returned by the retrieval system with respect to their information need. This form of feedback can subsequently be used to incrementally refine the initial query. In a manner analogous to rating items, there are explicit and implicit means of collecting relevance feedback data.

Rocchio's algorithm [33] is a widely used relevance feedback algorithm that operates in the vector space model. The algorithm is based on the modification of an initial query through differently weighted prototypes of relevant and non-relevant documents. The approach forms two document prototypes by taking the vector sum over all relevant and non-relevant documents. The following formula summarizes the algorithm formally:

$$Q_{i+1} = \alpha Q_i + \beta \sum_{rel} \frac{D_i}{|D_i|} - \gamma \sum_{nonrel} \frac{D_i}{|D_i|} \qquad \textbf{(10.2)}$$

Here, Q_i is the user's query at iteration i, and α, β, and γ are parameters that control the influence of the original query and the two prototypes on the resulting modified query. The underlying intuition of the above formula is to incrementally move the query vector towards clusters of relevant documents and away from irrelevant documents. While this goal forms an intuitive justification for Rocchio's algorithm, there is no theoretically motivated basis for the above formula, i.e., neither performance nor convergence can be guaranteed. However, empirical experiments have demonstrated that the approach leads to significant improvements in retrieval performance [33].

In more recent work, researchers have used a variation of Rocchio's algorithm in a machine learning context, i.e., for learning a user profile from unstructured text ([15], [3], [29]). The goal in these applications is to automatically induce a text classifier that can distinguish between classes of documents. In this context, it is assumed that no initial query exists, and the algorithm forms prototypes for classes analogously to Rocchio's approach as vector sums over documents belonging to the same class. The result of the algorithm is a set of weight vectors, whose proximity to unlabeled documents can be used to assign class membership. Similar to the relevance feedback version of Rocchio's algorithm, the Rocchio-based classification approach does not have any theoretic underpinnings and there are no performance or convergence guarantees.

10.7 Linear Classifiers

Algorithms that learn linear decision boundaries, i.e., hyperplanes separating instances in a multi-dimensional space, are referred to as linear classifiers. There are a large number of algorithms that fall into this category, and many of them have been successfully applied to text classification tasks [20]. All linear classifiers can be described in a common representational framework. In general, the outcome of the learning process is an *n*-dimensional weight vector *w*, whose dot product with an *n*-dimensional instance, e.g., a text document represented in the vector space model, results in a numeric score prediction. Retaining the numeric prediction leads to a linear regression approach. However, a threshold can be used to convert continuous predictions to discrete class labels. While this general framework holds for all linear classifiers, the algorithms differ in the training methods used to derive the weight vector *w*. For example, the equation below is known as the Widrow-Hoff rule, delta rule or gradient descent rule and derives the weight vector *w* by incremental vector movements in the direction of the negative gradient of the example's squared error [37]. This is the direction in which the error falls most rapidly.

$$w_{i+1,j} = w_{i,j} - 2\eta(w_i \cdot x_i - y_i)x_{i,j} \qquad \textbf{(10.3)}$$

The equation shows how the weight vector *w* can be derived incrementally. The inner product of instance x_i and weight vector w_i is the algorithm's numeric prediction for instance x_i. The prediction error is determined by subtracting the instance's known score, y_i, from the predicted score. The resulting error is then multiplied by the original instance vector x_i and the learning rate η to form a vector that, when subtracted from the weight vector *w,* moves *w* towards the correct prediction for instance x_i. The learning rate η controls the degree to which every additional instance affects the previous weight vector.

An alternative algorithm that has experimentally been shown to outperform the approach above on text classification tasks with many features is the exponentiated gradient (EG) algorithm. Kivinen and Warmuth [19] prove a bound for EG's error, which depends only logarithmically on the number of features. This result offers a theoretic argument for EG's performance on text classification problems, which are typically high-dimensional.

An important advantage of the above learning schemes for linear algorithms is that they can be performed on-line, i.e., the current weight vector can be modified incrementally as new instances become available. This is a crucial advantage for applications that operate under real-time constraints.

Finally, it is important to note that while the above approaches tend to converge on hyperplanes that separate the training data accurately, the hyperplane's generalization performance might not be optimal. A related approach aimed at improving generalization performance is known as support vector machines [36]. The central idea underlying support vector machines is to maximize the classification margin, i.e., the distance between the decision boundary and the closest training instances, the so-called support vectors. A series of empirical experiments on a variety of benchmark data sets indicated that linear support vector machines perform particularly well on text classification tasks [17]. The main reason for this is that the margin maximization is an inherently built-in overfitting protection mechanism. A reduced tendency to overfit training data is particularly useful for text classification algorithms, because in this domain high dimensional concepts must often be learned from limited training data, which is a scenario prone to overfitting.

10.8 Probabilistic Methods and Naïve Bayes

In contrast to the lack of theoretical justifications for the vector space model, there has been much work on probabilistic text classification approaches. This section describes one such example, the naïve Bayesian classifier. Early work on a probabilistic classifier and its text classification performance was reported by Maron [24]. Today, this algorithm is commonly referred to as a naïve Bayesian Classifier [13]. Researchers have recognized Naïve Bayes as an exceptionally well-performing text classification algorithm and have frequently adopted the algorithm in recent work ([27], [28], [25]).

The algorithm's popularity and performance for text classification applications have prompted researchers to empirically evaluate and compare different variations of naïve Bayes that have appeared in the literature (e.g. [26], [21]). In summary, McCallum and Nigam [26] note that there are two frequently used formulations of naïve Bayes, the multivariate Bernoulli and the multinomial model. Both models share the following principles. It is assumed that text documents are generated by an underlying generative model, specifically a parameterized mixture model:

$$P(d_i \mid \theta) = \sum_{j=1}^{|C|} P(c_j \mid \theta) P(d_i \mid c_j; \theta) \qquad \textbf{(10.4)}$$

Here, each class c corresponds to a mixture component that is parameterized by a disjoint subset of θ, and the sum of total probability over all mixture components determines the likelihood of a document. Once the parameters θ have been learned from training data, the posterior probability of class membership given the evidence of a test document can be determined according to Bayes' rule:

$$P(c_j \mid d_i; \hat{\theta}) = \frac{P(c_j \mid \hat{\theta}) P(d_i \mid c_j; \hat{\theta})}{P(d_i \mid \hat{\theta})} \qquad \textbf{(10.5)}$$

While the above principles hold for naïve Bayes classification in general, the multivariate Bernoulli and multinomial models differ in the way $p(d_i|c_j;\ \theta)$ is estimated from training data.

The multivariate Bernoulli formulation was derived with structured data in mind. For text classification tasks, it assumes that each document is represented as a binary vector over the space of all words from a vocabulary V. Each element B_{it} in this vector indicates whether a word appears at least once in the document. Under the naïve Bayes assumption that the probability of each word occurring in a document is independent of other words given the class label, $p(d_i|c_j;\ \theta)$ can be expressed as a simple product:

$$P(d_i \mid c_j;\theta) = \prod_{t=1}^{|V|} (B_{it}P(w_t \mid c_j;\theta) + (1 - B_{it})(1 - P(w_t \mid c_j;\theta))) \qquad \textbf{(10.6)}$$

Bayes-optimal optimal estimates for $p(w_t|c_j;\ \theta)$ can be determined by word occurrence counting over the data:

$$P(w_t \mid c_j;\theta) = \frac{1 + \sum_{i=1}^{|D|} B_{it}P(c_j \mid d_i)}{2 + \sum_{i=1}^{|D|} P(c_j \mid d_i)} \qquad \textbf{(10.7)}$$

In contrast to the binary document representation of the multivariate Bernoulli model, the multinomial formulation captures word frequency information. This model assumes that documents are generated by a sequence of independent trials drawn from a multinomial probability distribution. Again, the naïve Bayes independence assumption allows $p(d_i|c_j;\ \theta)$ to be determined based on individual word probabilities:

$$P(d_i \mid c_j;\theta) = P(|d_i|)\prod_{t=1}^{|d_i|} P(w_t \mid c_j;\theta)^{N_{it}} \qquad \textbf{(10.8)}$$

Here, N_{it} is the number of occurrences of word w_t in document d_i. Taking word frequencies into account, maximum likelihood estimates for $p(w_t|c_j;\ \theta)$ can be derived from training data**:**

$$P(w_t \mid c_j;\theta) = \frac{1 + \sum_{i=1}^{|D|} N_{it}\, P(c_j \mid d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N_{is}\, P(c_j \mid d_i)} \qquad \textbf{(10.9)}$$

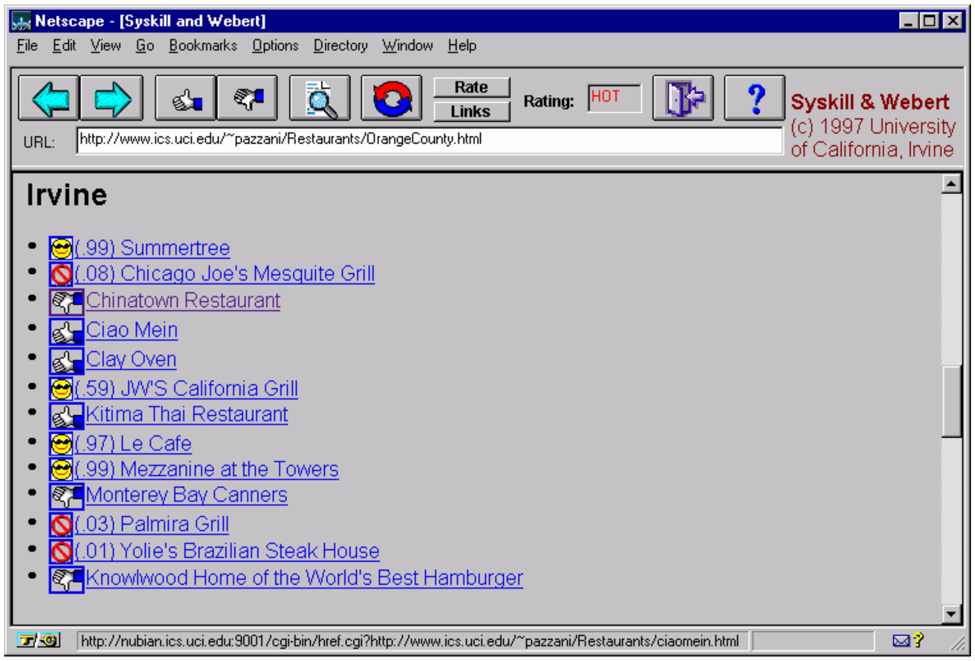
Empirically, the multinomial naïve Bayes formulation was shown to outperform the multivariate Bernoulli model. This effect is particularly noticeable for large vocabularies (McCallum and Nigam, 1998).

Even though the naïve Bayes assumption of class-conditional attribute independence is clearly violated in the context of text classification, naïve Bayes per-forms very well. Domingos and Pazzani [12] offer a possible explanation for this paradox by showing that class-conditional feature independence is not a necessary condition for the optimality of naïve Bayes. The naïve Bayes classifier has been used in several content-based recommendation systems including Syskill & Webert [29].

10.9 Trends in Content-Based Filtering

Belkin & Croft [5] surveyed some of the first content-based recommendation systems and noted that they made use of technology related to information retrieval such as tf*idf and Rocchio's method. Indeed, some of the early work on content-based recommendation used the term "query" to refer to user models. In this view, a user model is a saved query (or a set of saved queries) that can retrieve additional or new information of interest to the user. Some representative early systems include a system at Bellcore [14] that found new technical reports related to previously read reports and LyricTime [22] that recommended songs in a multimedia player based on a profile learned from the user's feedback on prior songs played.

The creation and rapid growth of the World Wide Web in the mid 1990s made access to vast amounts of information possible and created problems of locating and


Netscape - [Syskill and Webert]
File Edit View Go Bookmarks Options Directory Window Help
Rate
Links
Rating:
HOT
Syskill & Webert
(c) 1997 University of California, Irvine
URL:

http://www.ics.uci.edu/~pazzani/Restaurants/OrangeCounty.html

Irvine

- (.99) Summertree
- (.08) Chicago Joe's Mesquite Grill
- Chinatown Restaurant
- Ciao Mein
- Clay Oven
- (.59) JW'S California Grill
- Kitima Thai Restaurant
- (.97) Le Cafe
- (.99) Mezzanine at the Towers
- Monterey Bay Canners
- (.03) Palmira Grill
- (.01) Yolie's Brazilian Steak House
- Knowlwood Home of the World's Best Hamburger

http://nubian.ics.uci.edu:9001/cgi-bin/href.cgi?http://www.ics.uci.edu/~pazzani/Restaurants/ciaomein.html

Fig. 10.5. The Syskill & Webert system learns a model of the user's preference for web pages

identifying personally relevant information. Some in the Machine Learning community applied traditional machine learning methods to user modeling of document interests. These methods reduced the text training data to a few hundred highly relevant words using techniques such as information theory or tf*idf. Some representative systems included WebWatcher [16] and Syskill & Webert [29].

10.10 Limitations and Extensions

Although there are different approaches to learning a model of the user's interest with content-based recommendation, no content-based recommendation system can give good recommendations if the content does not contain enough information to distinguish items the user likes from items the user doesn't like. In recommending some items, e.g., jokes or poems, there often isn't enough information in the word frequency to model the user's interests. While it would be possible to tell a lawyer joke from a chicken joke based upon word frequencies, it would be difficult to distinguish a funny lawyer joke from other lawyer jokes. As a consequence, other recommendation technologies, such as collaborative recommenders [35], should be used in such situations.

In some situations, e.g., recommending movies, restaurants, or television programs, there is some structured information (e.g., the genre of the movie as well as actors and directors) that can be used by a content-based system. However, this information might be supplemented by the opinions of other users. One way to include the opinions of other users in the frameworks discussed in Section 10.2 is to add additional data associated to the representation of the examples. For example, Basu et al. [4] add features to examples that indicate the identifiers of other users who like an item. Ripper was applied to the resulting data that could learn profiles with both collaborative and content-based features (e.g., a user might like a science fiction movie if USER-109 likes it). Although not strictly a content-based system, the same technology as content-based recommenders is used to learn a user model. Indeed, Billsus and Pazzani [6] have shown that any machine learning algorithm may be used as the basis for collaborative filtering by transforming user ratings to attributes. Chapter 12 of this book [8] discusses a variety of other approaches to combining content and collaborative information in recommendation systems.

A final usage of content in recommendations is worth noting. Simple content-based rules may be used to filter the results of other methods such as collaborative filtering. For example, even if it is the case that people who buy dolls also buy adult videos, it might be important not to recommend adult items in a particular application. Similarly, although not strictly content-based, some systems might not recommend items that are out of stock.

10.11 Summary

Content-based recommendation systems recommend an item to a user based upon a description of the item and a profile of the user's interests. While a user profile may be entered by the user, it is commonly learned from feedback the user provides on items. A variety of learning algorithms have been adapted to learning user profiles, and the choice of learning algorithm depends upon the representation of content.

References

1. Ali, K., van Stam, W.: TiVo: Making Show Recommendations Using a Distributed Collaborative Filtering Architecture. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Seattle, WA. (2004) 394-401
2. Allan, J., Carbonell, J., Doddington, G., Yamron, J., Yang, Y.: Topic Detection and Tracking Pilot Study Final Report. In: Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop. Lansdowne, VA (1998) 194-218
3. Balabanovic, M., Shoham Y.: FAB: Content-based, Collaborative Recommendation. Communications of the Association for Computing Machinery 40(3) (1997) 66-72
4. Basu, C., Hirsh, H., Cohen W.: Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In: Proceedings of the 15th National Conference on Artificial Intelligence, Madison, WI (1998) 714-720
5. Belkin, N., Croft, B.: Information Filtering and Information Retrieval: Two Sides of the Same Coin? Communications of the ACM 35(12) (1992) 29-38
6. Billsus, D., Pazzani, M.: Learning Collaborative Information Filters. In: Proceedings of the International Conference on Machine Learning. Morgan Kaufmann Publishers. Madison, WI (1998) 46-54
7. Billsus, D., Pazzani, M., Chen, J.: A Learning Agent for Wireless News Access. In: Proceedings of the International Conference on Intelligent User Interfaces (2002) 33-36
8. Burke, R.: Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization. Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York (2007) this volume
9. Cohen, W.: Fast Effective Rule Induction. In: Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, CA. (1995) 115-123
10. Cohen, W.: Learning Rules that Classify E-mail. In: Papers from the AAAI Spring Symposium on Machine Learning in Information Access (1996) 18-25
11. Cohen, W., Hirsh, H. Joins that Generalize: Text Classification Using WHIRL. In: Proceedings of the Fourth International Conference on Knowledge Discovery & Data Mining, New York, NY (1998) 169-173
12. Domingos, P., Pazzani, M. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. Machine Learning 29 (1997) 103-130.
13. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. New York, NY: Wiley and Sons (1973)
14. Foltz, P., Dumais, S.: Personalized Information Delivery: An Analysis of Information Filtering Methods. Communications of the ACM 35(12) (1992) 51-60
15. Ittner, D., Lewis, D., Ahn, D.: Text Categorization of Low Quality Images. In: Symposium on Document Analysis and Information Retrieval, Las Vegas, NV (1995) 301-315
16. Joachims, T., Freitag, D., Mitchell, T.: WebWatcher: A Tour Guide for the World Wide Web. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence. Nagoya, Japan (1997) 770 -775
17. Joachims, T.: Text Categorization With Support Vector Machines: Learning with Many Relevant Features. In: European Conference on Machine Learning, Chemnitz, Germany (1998) 137-142
18. Kim, J., Lee, B., Shaw, M., Chang, H., Nelson, W.: Application of Decision-Tree Induction Techniques to Personalized Advertisements on Internet Storefronts. International Journal of Electronic Commerce 5(3) (2001) 45-62
19. Kivinen, J., Warmuth, M.: Exponentiated Gradient versus Gradient Descent for Linear Predictors. Information and Computation 132(1) (1997) 1-63

20. Lewis, D., Schapire, R., Callan, J., Papka, R.: Training Algorithms for Linear Text Classifiers. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Konstanz, Germany (1996) 298-306
21. Lewis, D.: Naïve (Bayes) at Forty: The Independence Assumption in Information Retrieval. In: European Conference on Machine Learning, Chemnitz, Germany (1998) 4-15
22. Loeb, S.: Architecting Personal Delivery of Multimedia Information. Communications of the ACM 35(12) (1992) 39-48
23. Mandel, M., Poliner, G., Ellis, D.: Support Vector Machine Active Learning for Music Retrieval. ACM Multimedia Systems Journal 12(1) (2006) 3-13
24. Maron, M.: Automatic Indexing: An Experimental Inquiry. Journal of the Association for Computing Machinery 8(3) (1961) 404-417
25. McCallum, A., Rosenfeld, R., Mitchell T., Ng, A.: Improving Text Classification by Shrinkage in a Hierarchy of Classes. In: Proceedings of the International Conference on Machine Learning. Morgan Kaufmann Publishers. Madison, WI (1998) 359-367
26. McCallum, A., Nigam, K.: A Comparison of Event Models for Naive Bayes Text Classification. In: AAAI/ICML-98 Workshop on Learning for Text Categorization, Technical Report WS-98-05, AAAI Press (1998) 41-48
27. Mitchell, T.: Machine Learning. McGraw-Hill (1997)
28. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Learning to Classify Text from Labeled and Unlabeled Documents. In: Proceedings of the 15th International Conference on Artificial Intelligence, Madison, WI (1998) 792-799
29. Pazzani M., Billsus, D.: Learning and Revising User Profiles: The Identification of Interesting Web Sites. Machine Learning 27(3) (1997) 313-331
30. Porter, M.: An Algorithm for Suffix Stripping. Program 14(3) (1980) 130-137
31. Quinlan, J.: Induction of Decision Trees. Machine Learning 1(1986) 81-106
32. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kauffman (1993)
33. Rocchio, J.: Relevance Feedback in Information Retrieval. In: G. Salton (ed.). The SMART System: Experiments in Automatic Document Processing. NJ: Prentice Hall (1971) 313-323
34. Salton, G. Automatic Text Processing. Addison-Wesley (1989)
35. Schafer, B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization. Lecture Notes in Computer Science, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York (2007) this volume
36. Vapnik, V.: The Nature of Statistical Learning Theory. Springer: New York (1995)
37. Widrow, A., Hoff, M.: Adaptive Switching Circuits. WESCON Convention Record 4 (1960) 96-104
38. Yang, Y., Pedersen J.: A Comparative Study on Feature Selection in Text Categorization. In: Proceedings of the Fourteenth International Conference on Machine Learning, Nashville, TN (1997) 412-420
39. Yang, Y.: An Evaluation of Statistical Approaches to Text Categorization. Information Retrieval 1(1) (1999) 67-88