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
systems that recommend an item to a user based upon a description of the it
and a profile of the user's interests. Content-based recommendation syste
may be used in a variety of domains ranging from recommending web pag
news articles, restaurants, television programs, and items for sale. Although
details of various systems differ, content-based recommendation systems sh
in common a means for describing the items that may be recommended
means for creating a profile of the user that describes the types of items the u
likes, and a means of comparing items to the user profile to determine wha
recommend. The profile is often created and updated automatically in respo
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 applic
which a user interacts. Typically, a system presents a summary list of items
and the user selects among the items to receive more details on an item or
with the item in some way. For example, online news sites present web p
headlines (and occasionally story summaries) and allow the user to select a
to read a story. E-commerce sites often present a page with a list of indivi
ucts and then allow the user to see more details about a selected product and
the product. Although the web server transmits HTML and the user sees a
the web server typically has a database of items and dynamically const
pages with a list of items. Because there are often many more items avai
database than would easily fit on a web page, it is necessary to select a subse
to display to the user or to determine an order in which to display the items.

Content-based recommendation systems analyze item descriptions to ider
that are of particular interest to the user. Because the details of recommend
tems differ based on the representation of items, this chapter first discusses a
item representations. Next, recommendation algorithms suited for each repr
are discussed. The chapter concludes with a discussion of variants of the ap

P. Brusilovsky, A. Kobsa, and W. Nejdl (Eds.): The Adaptive Web, LNCS 4321, pp. 325 – 341,

10.1.1 Item Representation

Items that can be recommended to the user are often stored in a database ta
10.1 shows a simple database with records (i.e., "rows") that describe thr
rants. The column names (e.g., Cuisine or Service) are properties of restaura
properties are also called "attributes," "characteristics," "fields," or "var
different publications. Each record contains a value for each attribute. A un
tifier, ID in Table 10.1, allows items with the same name to be distingu
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	Mediu
10003	Jacques Bistro	French	Table	High

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

Of course, a web page typically has more information than is shown in T
such as a text description of the restaurant, a restaurant review, or even a me
may easily be stored as additional fields in the database and a web page ca
ated with templates to display the text fields (as well as the structured data).
free text data creates a number of complications when learning a user profile.
ple, a profile might indicate that there is an 80% probability that a particular u
like a French restaurant. This might be added to the profile because a user ga
tive review of four out of five French restaurants. However, unrestricted text
typically unique and there would be no opportunity to provide feedback on f
rants described as "A charming café with attentive staff overlooking the river."

An extreme example of unstructured data may occur in news articles. T
shows an example of a part of a news article. The entire article can be tr
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 a
pleted, lawmakers prepared to work through the weekend fine-tuning a
Gray Davis says will put the state in the power business for "a long time
The proposal involves partially taking over California's two largest utilitie
ing long-term contracts of up to 10 years to buy electricity from wholesalers

polysemous words (the same word may have several meanings) and synon
ferent words may have the same meaning). For example, in the article in T
"Gray" is a name rather than a color, and "power" and "electricity" refer to
underlying concept.

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

Many personalization systems that deal with unrestricted text use a tec
create a structured representation that originated with text search systems [3
formalism, rather than using words, the root forms of words are typical
through a process called stemming [30]. The goal of stemming is to create a
reflects the common meaning behind words such as "compute," "com
"computer" "computes" and "computers." The value of a variable associa
term is a real number that represents the importance or relevance. This valu
the *tf*idf* weight (term-frequency times inverse document frequency).
weight, $w(t,d)$, of a term t in a document d is a function of the frequency
document (tf_t,d), the number of documents that contain the term (df_t) and th
of documents in the collection (N).[1]

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

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

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

debt-0.128 lawmak-0.128 state-0.122 wholesal-0.119 partial-0.106 cons
alert-0.103 scroung-0.096 advoc-0.09 testi-0.088 bail-out-0.088 crisi-0.0
0.084 price-0.083 long-0.082 bond-0.081 plan-0.081 term-0.08 grid-0.07
0.077 blackout-0.076 bid-0.076 market-0.074 fine-0.073 deregul-0.07 sp
deplet-0.068 liar-0.066.

Of course, this representation does not capture the context in which a word loses the relationships between words in the description. For example, a desc a steak house might contain the sentence, “there is nothing on the menu th tarian would like” while the description of a vegetarian restaurant migh “vegan” rather than vegetarian. In a manually created structured database, t attribute having a value of “vegetarian” would indicate that the restaurant i vegetarian one. In contrast, when converting an unstructured text descriptio tured data, the presence of the word vegetarian does not always indicate tha rant is vegetarian and the absence of the word vegetarian does not always in the restaurant is not a vegetarian restaurant. As a consequence, techniques fo user profiles that deal with structured data need to differ somewhat from t niques that deal with unstructured data or unstructured data automatically a cisely converted to structured data.

One variant on using words as terms is to use sets of contiguous words For example, in the article in Table 10.2, terms such as “energy reserves” ar business” might be more descriptive of the content than these words treate vidual terms. Of course, terms such as “all but” would also be included, but expect that these have very low weights, in the same way that “all” and “but” ally have low weights and are not among the most important terms in Table 10

10.2 User Profiles

A profile of the user’s interests is used by most recommendation systems. T may consist of a number of different types of information. Here, we conc two types of information:

1. A model of the user’s preferences, i.e., a description of the types of interest the user. There are many possible alternative representations c scription, but one common representation is a function that for any iter the likelihood that the user is interested in that item. For efficiency pur function may be used to retrieve the n items most likely to be of interest t
2. A history of the user’s interactions with the recommendation system. include storing the items that a user has viewed together with other in about the user’s interaction, (e.g., whether the user has purchased the iter ing that the user has given the item). Other types of history include savi typed by the user (e.g., that a user searched for an Italian restaurant in zip code).

already purchased or read.[2] Another important use of the history in con recommendation systems is to serve as training data for a machine learning that creates a user model. The next section will discuss several different app learning a user model. Here, we briefly describe approaches of manually the information used by recommendation systems: user customization and recommendation systems.

In user customization, a recommendation system provides an interfac lows users to construct a representation of their own interests. Often ch are used to allow a user to select from the known values of attributes, e.g sine of restaurants, the names of favorite sports teams, the favorite sec news site, or the genre of favorite movies. In other cases, a form allows type words that occur in the free text descriptions of items, e.g., the name sician or author that interests the user. Once the user has entered this info simple database matching process is used to find items that meet the speci ria and display them to the user.

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

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

In rule-based recommendation systems, the recommendation system ha recommend other products based on the user history. For example, a sy contain a rule that recommends the sequel to a book or movie to people purchased the early item in the series. Another rule might recommend a n an artist to users that purchased earlier CDs by that artist. Rule-based sys capture several common reasons for making recommendations, but they do the same detailed personalized recommendations that are available with oth mendation systems.

---

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

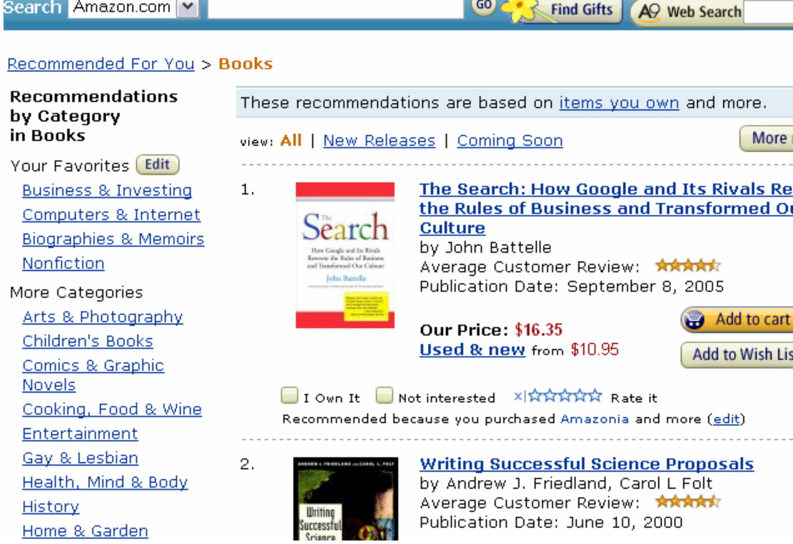

Fig. 10.1. Book recommendations by Amazon.com.

Fig. 10.2. User customization in Amazon.com

Creating a model of the user's preference from the user history is a form of
tion learning. The training data of a classification learner is divided into c
e.g., the binary categories “items the user likes” and “items the user doe
This is accomplished either through explicit feedback in which the user rates
some interface for collecting feedback or implicitly by observing the user
tions with items. For example, if a user purchases an item, that is a sign tha
likes the item, while if the user purchases and returns the item that is a sig
user doesn’t like the item. In general, there is a tradeoff since implicit me
collect a large amount of data with some uncertainty as to whether the use
likes the item. In contrast, when the user explicitly rates items, there is l
noise in the training data, but users tend to provide explicit feedback on on
percentage of the items they interact with.

Figure 10.3 shows an example of a recommendation system with exp
feedback. The recommender “MyBestBets” by ChoiceStream is a web ba
face to a television recommendation system. Users can click on the thum
thumbs down buttons to indicate whether they like the program that
mended. By necessity, this system requires explicit feedback because it is
grated with a television [1] and cannot infer the user’s interests by obse
user’s behavior.

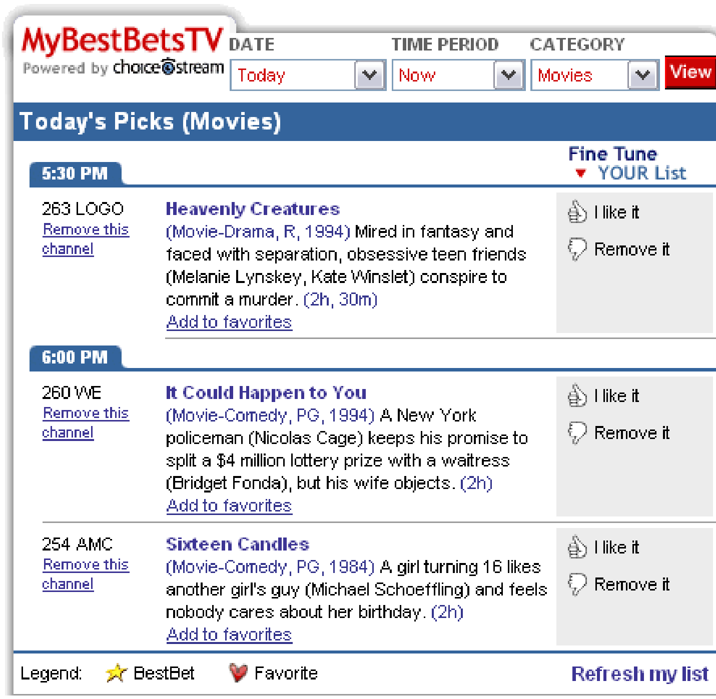

Fig. 10.3. A recommendation system using explicit feedback

model, the function predicts whether the user would be interested in the it of the classification learning algorithms create a function that will provide a of the probability that a user will like an unseen item. This probability may sort a list of recommendations. Alternatively, an algorithm may create a fu directly predicts a numeric value such as the degree of interest.

Some of the algorithms below are traditional machine learning algo signed to work on structured data. When they operate on free text, the free t converted to structured data by selecting a small subset of the terms as att contrast, other algorithms are designed to work in high dimensional spaces 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 ing training data, in this case text documents, into subgroups until those contain only instances of a single class. A partition is formed by a test on ture -- in the context of text classification typically the presence or abse individual word or phrase. Expected information gain is a commonly used c select the most informative features for the partition tests [38].

Decision trees have been studied extensively in use with structured da that shown in Table 10.1. Given feedback on the restaurants, a decision tree represent and learn a profile of someone who prefers to eat in expensive F taurants or inexpensive Mexican restaurants. Arguably, the decision tree b ideal for unstructured text classification tasks [29]. As a consequence of the tion-theoretic splitting criteria used by decision tree learners, the inducti decision trees is a preference for small trees with few tests. However, it can experimentally that text classification tasks frequently involve a large r relevant features [17]. Therefore, a decision tree's tendency to base classifi as few tests as possible can lead to poor performance on text classification. when there are a small number of structured attributes, the performance, and understandability of decision trees for content-based models are all a Kim et al. [18] describe an application of decision trees for personalizing ments on web pages.

RIPPER [9] is a rule induction algorithm closely related to decision operates in a similar fashion to the recursive data partitioning approach above. Despite the problematic inductive bias, however, RIPPER perform tively with other state-of-the-art text classification algorithms. In part, the ance can be attributed to a sophisticated post-pruning algorithm that opti fit of the induced rule set with respect to the training data as a whole. Fu RIPPER supports multi-valued attributes, which leads to a natural represe text classification tasks, i.e., the individual words of a text document can sented as multiple feature values for a single feature. While this is ess

such as sender, subject, and body text, can be represented as separate mu
features, which allows the algorithm to take advantage of the document's
in a natural fashion. Cohen [10] shows how RIPPER can classify e-mail
into user defined categories.

10.5 Nearest Neighbor Methods

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

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

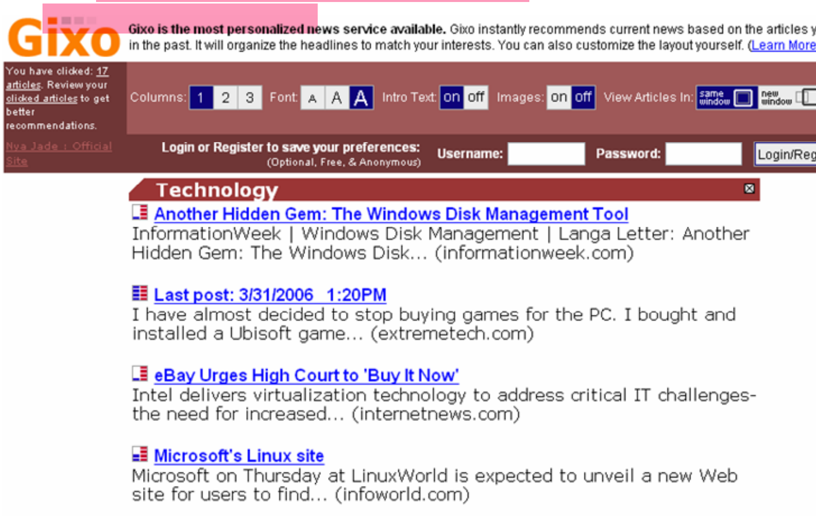

Fig. 10.4. Gixo presents personalized news based on similarity to articles that have
been read

The Daily Learner system uses the nearest neighbor algorithm to create a mo
user's short term interests [7]. Gixo, a personalized news system, also uses
larity as a basis for recommendation (Figure 10.4). The headlines are prece
icon that indicates how popular the item is (the first bar) and how similar tl
to stories that have been read by the user before (the second bar). The fact
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 depen
user's ability to construct queries by selecting a set of representative keyw
methods that help users to incrementally refine queries based on previous
sults have been the focus of much research. These methods are commonly r
as relevance feedback. The general principle is to allow users to rate docu
turned by the retrieval system with respect to their information need. Thi
feedback can subsequently be used to incrementally refine the initial query.
ner analogous to rating items, there are explicit and implicit means of colle
vance feedback data.

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

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

Here, Q_i is the user's query at iteration i, and α, β, and γ are parameters th
the influence of the original query and the two prototypes on the resulting
query. The underlying intuition of the above formula is to incrementally
query vector towards clusters of relevant documents and away from irrelev
ments. While this goal forms an intuitive justification for Rocchio's a
there is no theoretically motivated basis for the above formula, i.e., ne
formance nor convergence can be guaranteed. However, empirical ex
have demonstrated that the approach leads to significant improvements in
performance [33].

In more recent work, researchers have used a variation of Rocchio's alg
a machine learning context, i.e., for learning a user profile from unstruc
([15], [3], [29]). The goal in these applications is to automatically indu
classifier that can distinguish between classes of documents. In this con

to unlabeled documents can be used to assign class membership. Simi relevance feedback version of Rocchio's algorithm, the Rocchio-based tion approach does not have any theoretic underpinnings and there are no ance or convergence guarantees.

10.7 Linear Classifiers

Algorithms that learn linear decision boundaries, i.e., hyperplanes sepa stances in a multi-dimensional space, are referred to as linear classifiers. T large number of algorithms that fall into this category, and many of them successfully applied to text classification tasks [20]. All linear classifiers scribed in a common representational framework. In general, the outco learning process is an n-dimensional weight vector w, whose dot product dimensional instance, e.g., a text document represented in the vector spa results in a numeric score prediction. Retaining the numeric prediction linear regression approach. However, a threshold can be used to convert predictions to discrete class labels. While this general framework holds fo classifiers, the algorithms differ in the training methods used to derive t vector w. For example, the equation below is known as the Widrow-Hoff rule or gradient descent rule and derives the weight vector w by incremen movements in the direction of the negative gradient of the example's squ [37]. This is the direction in which the error falls most rapidly.

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

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

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

An important advantage of the above learning schemes for linear algorit they can be performed on-line, i.e., the current weight vector can be modi

hyperplanes that separate the training data accurately, the hyperplane's gene
performance might not be optimal. A related approach aimed at improving g
tion performance is known as support vector machines [36]. The central ide
ing support vector machines is to maximize the classification margin, i.e., th
between the decision boundary and the closest training instances, the so-c
port vectors. A series of empirical experiments on a variety of benchmark
indicated that linear support vector machines perform particularly well on t
fication tasks [17]. The main reason for this is that the margin maximiza
inherently built-in overfitting protection mechanism. A reduced tendency
training data is particularly useful for text classification algorithms, becau
domain high dimensional concepts must often be learned from limited trai
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
been much work on probabilistic text classification approaches. This section
one such example, the naïve Bayesian classifier. Early work on a probabilis
fier and its text classification performance was reported by Maron [24]. T
algorithm is commonly referred to as a naïve Bayesian Classifier [13]. R
have recognized Naïve Bayes as an exceptionally well-performing text cla
algorithm and have frequently adopted the algorithm in recent work ([27], [2

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

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

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

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

from training data.

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

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

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

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

In contrast to the binary document representation of the multivariate Bernou
the multinomial formulation captures word frequency information. This
sumes that documents are generated by a sequence of independent trials dra
multinomial probability distribution. Again, the naïve Bayes independence
tion allows $p(d_i|c_j;\ \theta)$ to be determined based on individual word probabilitie

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

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

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

Even though the naïve Bayes assumption of class-conditional attribute ence is clearly violated in the context of text classification, naïve Bayes very well. Domingos and Pazzani [12] offer a possible explanation for this p showing that class-conditional feature independence is not a necessary con the optimality of naïve Bayes. The naïve Bayes classifier has been used 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 recommendatio and noted that they made use of technology related to information retriev tf*idf and Rocchio's method. Indeed, some of the early work on content-b ommendation used the term "query" to refer to user models. In this vie model is a saved query (or a set of saved queries) that can retrieve additio information of interest to the user. Some representative early systems inclu tem at Bellcore [14] that found new technical reports related to previousl ports and LyricTime [22] that recommended songs in a multimedia player b profile learned from the user's feedback on prior songs played.

The creation and rapid growth of the World Wide Web in the mid 19 access to vast amounts of information possible and created problems of loc

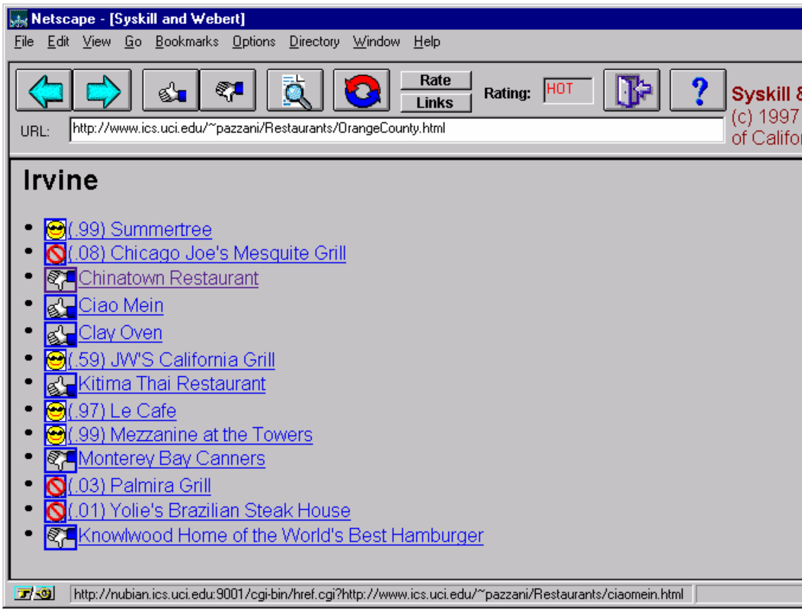

Fig. 10.5. The Syskill & Webert system learns a model of the user's preference for

vant words using techniques such as information theory or tf*idf. Some repr
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 int content-based recommendation, no content-based recommendation system good recommendations if the content does not contain enough information guish items the user likes from items the user doesn't like. In recommend items, e.g., jokes or poems, there often isn't enough information in the word to model the user's interests. While it would be possible to tell a lawyer jo chicken joke based upon word frequencies, it would be difficult to distinguis lawyer joke from other lawyer jokes. As a consequence, other recommenda nologies, such as collaborative recommenders [35], should be used in such situ

In some situations, e.g., recommending movies, restaurants, or telev grams, there is some structured information (e.g., the genre of the movie actors and directors) that can be used by a content-based system. Howeve formation might be supplemented by the opinions of other users. One way the opinions of other users in the frameworks discussed in Section 10.2 is to tional data associated to the representation of the examples. For example, B [4] add features to examples that indicate the identifiers of other users wh item. Ripper was applied to the resulting data that could learn profiles with laborative and content-based features (e.g., a user might like a science fictio USER-109 likes it). Although not strictly a content-based system, the sam ogy as content-based recommenders is used to learn a user model. Indeed, B Pazzani [6] have shown that any machine learning algorithm may be used as for collaborative filtering by transforming user ratings to attributes. Chapter book [8] discusses a variety of other approaches to combining content and tive information in recommendation systems.

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

10.11 Summary

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

laborative Filtering Architecture. In: Proceedings of the Tenth ACM SIGKDD In
Conference on Knowledge Discovery and Data Mining. Seattle, WA. (2004) 394

2. Allan, J., Carbonell, J., Doddington, G., Yamron, J., Yang, Y.: Topic Detection ing Pilot Study Final Report. In: Proceedings of the DARPA Broadcast News Tr and Understanding Workshop. Lansdowne, VA (1998) 194-218
3. Balabanovic, M., Shoham Y.: FAB: Content-based, Collaborative Recommenda munications of the Association for Computing Machinery 40(3) (1997) 66-72
4. Basu, C., Hirsh, H., Cohen W.: Recommendation as Classification: Using Socia tent-Based Information in Recommendation. In: Proceedings of the 15th Natio ence on Artificial Intelligence, Madison, WI (1998) 714-720
5. Belkin, N., Croft, B.: Information Filtering and Information Retrieval: Two S Same Coin? Communications of the ACM 35(12) (1992) 29-38
6. Billsus, D., Pazzani, M.: Learning Collaborative Information Filters. In: Procee International Conference on Machine Learning. Morgan Kaufmann Publishers WI (1998) 46-54
7. Billsus, D., Pazzani, M., Chen, J.: A Learning Agent for Wireless News Acce ceedings of the International Conference on Intelligent User Interfaces (2002) 33
8. Burke, R.: Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A. (eds.): The Adaptive Web: Methods and Strategies of Web Personalization. Le in Computer Science, Vol. 4321. Springer-Verlag, Berlin Heidelberg New York volume
9. Cohen, W.: Fast Effective Rule Induction. In: Proceedings of the Twelfth In Conference on Machine Learning, Tahoe City, CA. (1995) 115-123
10. Cohen, W.: Learning Rules that Classify E-mail. In: Papers from the AAAI Spri sium on Machine Learning in Information Access (1996) 18-25
11. Cohen, W., Hirsh, H. Joins that Generalize: Text Classification Using WHIRL. I ings of the Fourth International Conference on Knowledge Discovery & Data M York, NY (1998) 169-173
12. Domingos, P., Pazzani, M. Beyond Independence: Conditions for the Optima Simple Bayesian Classifier. Machine Learning 29 (1997) 103-130.
13. Duda, R., Hart, P.: Pattern Classification and Scene Analysis. New York, NY: Sons (1973)
14. Foltz, P., Dumais, S.: Personalized Information Delivery: An Analysis of I 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: S on Document Analysis and Information Retrieval, Las Vegas, NV (1995) 301-31
16. Joachims, T., Freitag, D., Mitchell, T.: WebWatcher: A Tour Guide for the W Web. In: Proceedings of the 15th International Joint Conference on Artificial I Nagoya, Japan (1997) 770 -775
17. Joachims, T.: Text Categorization With Support Vector Machines: Learning Relevant Features. In: European Conference on Machine Learning, Chemnitz (1998) 137-142
18. Kim, J., Lee, B., Shaw, M., Chang, H., Nelson, W.: Application of Decision-Tre Techniques to Personalized Advertisements on Internet Storefronts. International Electronic Commerce 5(3) (2001) 45-62
19. Kivinen, J., Warmuth, M.: Exponentiated Gradient versus Gradient Descent Predictors. Information and Computation 132(1) (1997) 1-63

21. Lewis, D.: Naive (Bayes) at Forty: The Independence Assumption in Informatio
In: European Conference on Machine Learning, Chemnitz, Germany (1998) 4-15
22. Loeb, S.: Architecting Personal Delivery of Multimedia Information. Commun
the ACM 35(12) (1992) 39-48
23. Mandel, M., Poliner, G., Ellis, D.: Support Vector Machine Active Learning
Retrieval. ACM Multimedia Systems Journal 12(1) (2006) 3-13
24. Maron, M.: Automatic Indexing: An Experimental Inquiry. Journal of the Ass
Computing Machinery 8(3) (1961) 404-417
25. McCallum, A., Rosenfeld, R., Mitchell T., Ng, A.: Improving Text Classi
Shrinkage in a Hierarchy of Classes. In: Proceedings of the International Co
Machine Learning. Morgan Kaufmann Publishers. Madison, WI (1998) 359-367
26. McCallum, A., Nigam, K.: A Comparison of Event Models for Naive Bayes Te
cation. In: AAAI/ICML-98 Workshop on Learning for Text Categorization, Te
port 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 fr
and Unlabeled Documents. In: Proceedings of the 15th International Conferenc
cial Intelligence, Madison, WI (1998) 792-799
29. Pazzani M., Billsus, D.: Learning and Revising User Profiles: The Identification
ing 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.). T
System: Experiments in Automatic Document Processing. NJ: Prentice Hall (197
34. Salton, G. Automatic Text Processing. Addison-Wesley (1989)
35. Schafer, B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative Filtering Re
Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The Adaptive Web: M
Strategies of Web Personalization. Lecture Notes in Computer Science,
Springer-Verlag, Berlin Heidelberg New York (2007) this volume
36. Vapnik, V.: The Nature of Statistical Learning Theory. Springer: New York (199
37. Widrow, A., Hoff, M.: Adaptive Switching Circuits. WESCON Convention
(1960) 96-104
38. Yang, Y., Pedersen J.: A Comparative Study on Feature Selection in Text Cat
In: Proceedings of the Fourteenth International Conference on Machine Lear
ville, TN (1997) 412-420
39. Yang, Y.: An Evaluation of Statistical Approaches to Text Categorization. I
Retrieval 1(1) (1999) 67-88