Content-based recommendation

3.0

Content-based recommendation

3.1

Architecture of content-based systems, Content representation and content similarity, Item profiles, Discovering features of documents, Obtaining item features from tags, Representing item profiles, Methods for learning user profiles, Similarity based retrieval, The Role of User Generated Content in the Recommendation Process.

3.2

Bayes classifier for recommendation, Regression based recommendation system. Advantages and drawbacks of content-based filtering
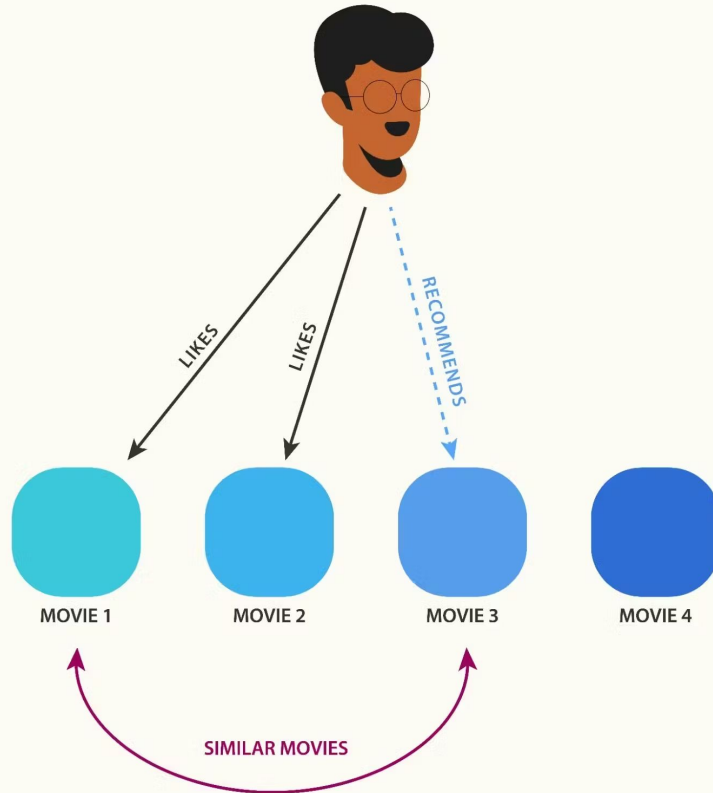
The basic process performed by a content-based recommender consists in matching up the attributes of a user profile in which preferences and interests are stored, with the attributes of a content object (item), in order to recommend to the user new interesting items.

2 Basics of Content-based Recommender Systems

Systems implementing a content-based recommendation approach analyze a set of documents and/or descriptions of items previously rated by a user, and build a model or profile of user interests based on the features of the objects rated by that user [63]. The profile is a structured representation of user interests, adopted to recommend new interesting items. The recommendation process basically consists in matching up the attributes of the user profile against the attributes of a content object. The result is a relevance judgment that represents the user's level of interest in that object. If a profile accurately reflects user preferences, it is of tremendous advantage for the effectiveness of an information access process. For instance, it could be used to filter search results by deciding whether a user is interested in a specific Web page or not and, in the negative case, preventing it from being displayed.
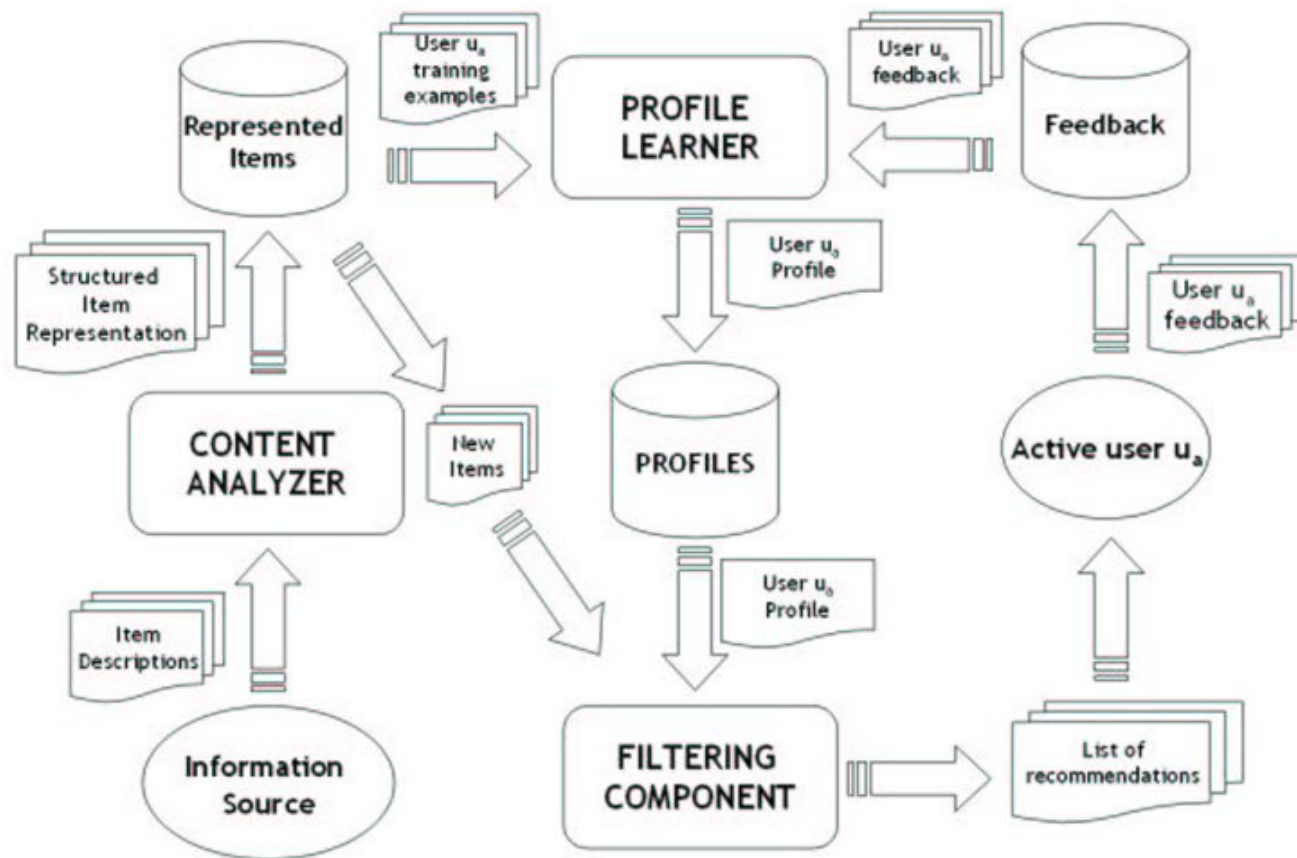
Fig. 1 High level architecture of a Content-based Recommender

- CONTENT ANALYZER – When information has no structure (e.g. text), some kind of pre-processing step is needed to extract structured relevant information. The main responsibility of the component is to represent the content of items (e.g. documents, Web pages, news, product descriptions, etc.) coming from information sources in a form suitable for the next processing steps. Data items are analyzed by feature extraction techniques in order to shift item representation from the original information space to the target one (e.g. Web pages represented as keyword vectors). This representation is the input to the PROFILE LEARNER and FILTERING COMPONENT;

Item descriptions coming from InformationSource are processed by the CONTENT ANALYZER,that extracts features(keywords,n-grams,concepts,...) from unstructured text to produce a structured item representation, stored in the repository Represented Items.

Borrows techniques from Information Retrieval systems

like/dislike   ratings   Text comments

Explicit feedback   Implicit feedback

- PROFILE LEARNER – This module collects data representative of the user preferences and tries to generalize this data, in order to construct the user profile. Usually, the generalization strategy is realized through machine learning techniques [61], which are able to infer a model of user interests starting from items liked or disliked in the past. For instance, the PROFILE LEARNER of a Web page recommender can implement a relevance feedback method [75] in which the learning technique combines vectors of positive and negative examples into a prototype vector representing the user profile. Training examples are Web pages on which a positive or negative feedback has been provided by the user;

- FILTERING COMPONENT – This module exploits the user profile to suggest relevant items by matching the profile representation against that of items to be recommended. The result is a binary or continuous relevance judgment (computed using some similarity metrics [42]), the latter case resulting in a ranked list of potentially interesting items. In the above mentioned example, the matching is realized by computing the cosine similarity between the prototype vector and the item vectors.

In order to build the profile of the active user u_a, the training set TR_a for u_a must be defined. TR_a is a set of pairs $\langle I_k, r_k \rangle$, where r_k is the rating provided by u_a on the item representation I_k. Given a set of item representation labeled with ratings, the PROFILE LEARNER applies supervised learning algorithms to generate a predictive model – the *user profile* – which is usually stored in a *profile repository* for later use by the FILTERING COMPONENT. Given a new item representation, the FILTERING COMPONENT predicts whether it is likely to be of interest for the active user, by comparing features in the item representation to those in the representation of user preferences (stored in the user profile). Usually, the FILTERING COMPONENT implements some strategies to rank potentially interesting items according to the relevance with respect to the user profile. Top-ranked items are included in a *list of recommendations L_a*, that is presented to u_a. User tastes usually change in time, therefore up-to-date information must be maintained and provided to the PROFILE LEARNER in order to automatically update the user profile. Further feedback is gathered on generated recommendations by letting users state their satisfaction or dissatisfaction with items in L_a. After gathering that feedback, the learning process

is performed again on the new training set, and the resulting profile is adapted to the updated user interests. The iteration of the feedback-learning cycle over time allows the system to take into account the dynamic nature of user preferences.

Applications of Recommendation Systems

- Product Recommendations
- Movie Recommendations
- News Articles

Models for recommendation systems

The Utility Matrix

The data itself is represented as a utility matrix, giving for each user-item pair,a value that represents what is known about the degree of preference of that user for that item.Values come from an ordered set, e.g., integers 1–5 representing the number of stars that the user gave as a rating for that item. We assume that the matrix is sparse,meaning that most entries are"unknown."An unknown rating implies that we have no explicit information about the user's preference for the item.

Foundation of Data Representation: The utility matrix is the core structure where we represent user-item interactions in a recommendation system.
Rows and Columns:
Rows represent individual users.
Columns represent items (movies, products, articles, etc.).
Entries: Each cell in the matrix holds a value indicating the known preference of a user for a particular item. This value could be:
Explicit Ratings: Numerical ratings given by a user on a specific scale (e.g., 1-5 stars for a movie).
Implicit Feedback: Inferred preferences from user behavior (e.g., purchase history, time spent on an article, number of views).
Ordered Value Set

Values in the matrix typically belong to an ordered set, allowing for ranking and comparison of preferences.
5-Star Ratings: A standard example, where 5 stars signifies high preference and 1 star signifies low preference.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Figure 9.1: A utility matrix representing ratings of movies on a 1–5 scale

Sparsity

The Core Challenge: Most of the utility matrix is "unknown." This means that for most user-item pairs, we don't have any explicit or implicit information on the user's preference. This is common because users rarely rate or interact with every single item in a system.
Recommender System Goal: The main goal of a recommendation system is to fill in the "unknown" values in the matrix to predict the ratings a user might give to items they haven't interacted with.
Key Takeaways

The utility matrix is a critical data structure in building recommendation systems.
Entries in the matrix represent user-item preferences, but they are often sparse.
Collaborative filtering and content-based filtering techniques aim to address sparsity and recommend items to users based on patterns in the utility matrix.

Long Tail

Sparsity and the Long Tail:

Sparsity: The key challenge in collaborative filtering and utility matrices is sparsity, meaning most entries represent unknown user preferences.
Long Tail Distribution: In statistics, the long tail refers to the large number of infrequent events or items that contribute a significant portion of the overall activity or sales in a given system.
Connecting Sparsity and Long Tail:

While not a direct application of the long tail concept, the unknown entries in the utility matrix, representing items users haven't interacted with, could be considered analogous to the long tail in some ways:

They represent a large number of infrequent interactions (unknown preferences).
Accurately predicting preferences within these "unknown" entries is crucial for effective recommendations.
Focus of Recommender Systems:

It's important to remember that the primary focus of recommender systems is not to recommend everything in the long tail (all unknown entries). Instead, the aim is to utilize the information in the known entries (user-item interactions) to predict and recommend the most relevant items for each user, even if they haven't explicitly interacted with those items before.

Conclusion:

While the long tail concept doesn't directly translate to the context of recommender systems and utility matrices, the notion of sparsity and the large number of unknown entries does share some similarities with the idea of infrequent events in the long tail. However, the focus of recommender systems remains on identifying and recommending relevant items to users, not necessarily exploring all possible unknown interactions.
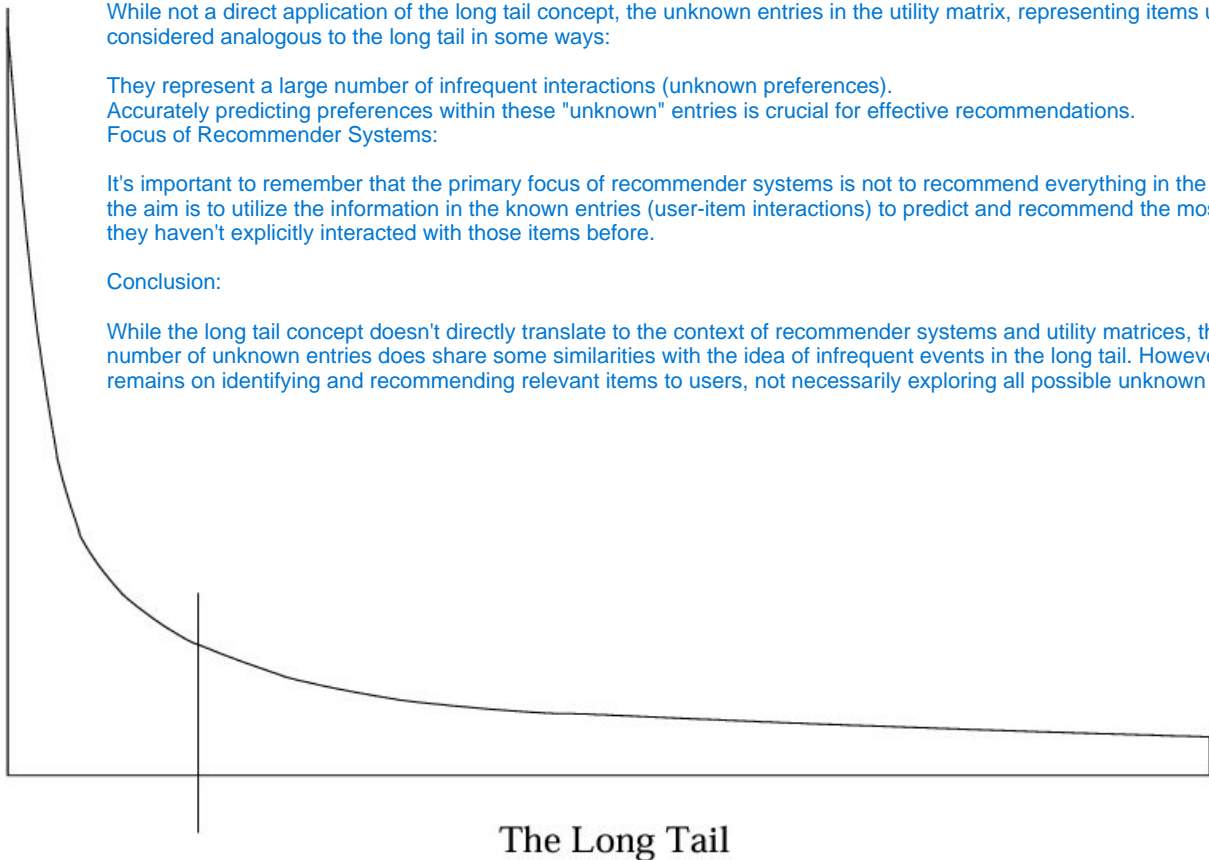
The Long Tail

Populating the Utility Matrix

Implicit "dislike" difficult: Unlike explicit dislikes, implicit data often only captures positive interactions (likes). It can be challenging to infer dislikes solely from user behavior.
Combination: Many recommender systems utilize a combination of explicit and implicit feedback for a more comprehensive understanding of user preferences and improved recommendation accuracy.
Additionally:

The statement "this sort of rating system really has only one value: 1 means the user likes the item" is partially correct.
While user behavior often indicates "like," the strength of that preference might not be captured. Some implicit feedback approaches try to address this by assigning weights to interactions based on factors like frequency or duration (e.g., time spent watching a video).

1. Ask users to submit online rating

2. We can make inferences from users behavior. Most Obviously, if a user buys a product at Amazon,watches a movie on YouTube,or reads a news article, then the user can be said to "like" this item.Note that this sort of rating system really has only one value: 1 means that the user likes the item.

1. Explicit Feedback:

This involves directly asking users to submit their ratings for items. This can be done through:
Star ratings: Assigning a rating on a scale (e.g., 1-5 stars) indicating the user's preference for an item.
Thumbs up/down: Simpler approach indicating whether the user likes or dislikes an item.
Numeric ratings: Users provide a numerical score on a specific scale (e.g., 1-10).
Advantages:

Provides clear and direct information about user preferences.
Can be used to personalize recommendations more accurately.
Disadvantages:

Low engagement: Users might be unwilling or hesitant to provide ratings frequently.
Cold start problem: New users or items with no ratings lack data for recommendations.

2. Implicit Feedback:

Involves inferring user preferences from their behavioral data, as you correctly mentioned. Examples include:
Purchase history: Assuming a user "likes" an item they purchased.
Viewing history: Videos watched on YouTube, articles read online, etc., can indicate interest.
Search history: Products or services searched for might indicate potential preferences.
Click-through rates (CTR): Clicking on links or ads can signify interest in the associated item.

Item profile

In a content-based system, we must construct for each item profile, which is a record or collection of records representing important characteristics of that item.

Eg the features of a movie

1. The set of actors of the movie. Some viewers prefer movies with their favorite actors.

2. The director. Some viewers have a preference for the work of certain directors.

3. The year in which the movie was made. Some viewers prefer old movies; others watch only the latest releases.

4. The *genre* or general type of movie. Some viewers like only comedies, others dramas or romances.

Discovering Features of Documents

There are many kinds of documents for which a recommendation system can be useful.

For example,

- there are many news articles published each day,and we can not read all of them.A recommendation system can suggest articles on topics a user is interested in
- Web pages are also a collection of documents

Steps

1. Remove stop words
2. For the remaining words,compute the TF.IDF score for each word in the document. Theoneswiththehighest scoresarethewords that characterize the document.
3. We may then take as the features of document then words with the highest TF.IDF scores. It is possible to pick "**n**" to be the same for all documents,or to let n be a fixed percentage of the words in the document.We could so choose to make all words whose TF.IDF scores are above given threshold to be a part of the featureset
4. To Measure The Similarity Of Two documents, there are several natural distance measures we can use : Jaccard distance or cosine distance

TF - IDF

TERM FREQUENCY

$$= \frac{\text{No. of rep of words in sentence}}{\text{No. of words in sentence}}$$

FINALLY

TF*IDF $IDF = \log\left(\frac{\text{No. of sentences}}{\text{No. of sentences containing words}}\right)$

Sent1 → good boy

Sent2 → good girl

Sent3 → boy girl good

Words	Frequency
good	3
boy	2
girl	2

Vectors
⟹
TF IDF

TF

	Sent1	Sent2	Sent3
good	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$
boy	$\frac{1}{2}$	0	$\frac{1}{3}$
girl	0	$\frac{1}{2}$	$\frac{1}{3}$

*

IDF

Words	IDF
good	$\log(3/3) = 0$
boy	$\log(3/2)$
girl	$\log(3/2)$

	f_1	f_2	f_3	O/P
	good	boy	girl	
Sent1	0	$\frac{1}{2} * \log(3/2)$	0	
Sent2	0			

Obtaining Item Features From Tags

Let us consider a database of images as an example of a way that features have been obtained for items. The problem with images is that their data, typically an array of pixels, does not tell us anything useful about their features. We can calculate simple properties of pixels, such as the average amount of red in the picture, but few users are looking for red pictures or especially like red pictures.

There have been a number of attempts to obtain information about features of items by inviting users to *tag* the items by entering words or phrases that describe the item. Thus, one picture with a lot of red might be tagged "Tiananmen Square," while another is tagged "sunset at Malibu." The distinction is not something that could be discovered by existing image-analysis programs.

Almost any kind of data can have its features described by tags.

Representing Item Profiles

- Sometime features are added as 1 or 0 (like or dislike) sometime numerical values are used (ratings)
- There is no harm if some components of the vectors are Boolean and others are real-valued or integer-valued.
- We can still compute the cosine distance between vectors, although if we do so, we should give some thought to the appropriate scaling of the non Boolean components, so that they neither dominate the calculation nor are they irrelevant.

Example 9.2: Suppose the only features of movies are the set of actors and the average rating. Consider two movies with five actors each. Two of the actors are in both movies. Also, one movie has an average rating of 3 and the other an average of 4. The vectors look something like

$$
\begin{array}{cccccccccc}
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 3\alpha \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 4\alpha
\end{array}
$$

However, there are in principle an infinite number of additional components, each with 0's for both vectors, representing all the possible actors that neither movie has. Since cosine distance of vectors is not affected by components in which both vectors have 0, we need not worry about the effect of actors that are in neither movie.

The last component shown represents the average rating. We have shown it as having an unknown scaling factor α. In terms of α, we can compute the cosine of the angle between the vectors. The dot product is $2 + 12\alpha^2$, and the lengths of the vectors are $\sqrt{5 + 9\alpha^2}$ and $\sqrt{5 + 16\alpha^2}$. Thus, the cosine of the angle between the vectors is

$$\frac{2 + 12\alpha^2}{\sqrt{25 + 125\alpha^2 + 144\alpha^4}}$$

$\alpha = 1$,

0.816.

Learning User Profiles and Filtering

We assume that we have a set **D_L** of training documents, which are labeled by a specific user. Furthermore, the training data contain the ratings assigned by the active user to these documents. These documents are used to construct a training model.

The labels on the documents correspond to the numeric, binary, or unary ratings.

Assume that the ith document in D_L has a rating denoted by c_i. We also have a set D_U of testing documents, which are unlabeled. Note that both D_L and D_U are specific to a particular (active) user

D_U might correspond to candidate Web documents for recommendation to the active user. The precise definition of D_U depends on the domain at hand, but the individual documents in D_U are extracted in a similar way to those in D_L. The training model on D_L is used to make recommendations from D_U to the active user.

Problem is of classification or regression.

Nearest Neighbor Classification

The nearest neighbor classifier is one of the simplest classification techniques, and it can be implemented in a relatively straightforward way. The first step is to define a similarity function, which is used in the nearest neighbor classifier. The most commonly used similarity function is the cosine function. Let $\overline{X} = (x_1 \ldots x_d)$ and $\overline{Y} = (y_1 \ldots y_d)$ be a pair of documents, in which the normalized frequencies of the ith word are given by x_i and y_i, respectively, in the two documents. Note that these frequencies are normalized or weighted with the use of unsupervised tf-idf weighting or the supervised methods discussed in the previous section. Then, the cosine measure is defined using these normalized frequencies as follows:

$$\text{Cosine}(\overline{X}, \overline{Y}) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2} \sqrt{\sum_{i=1}^{d} y_i^2}} \qquad (4.9)$$

For each document in D_U, its k-nearest neighbors in D_L are determined using the cosine similarity function. The average value of the rating for the k neighbors of each item in D_U is determined. This average value is the predicted rating for the corresponding item in D_U. An additional heuristic enhancement is that one can weight each rating with the similarity value.

The main challenge with the use of this approach is its high computational complexity. Note that the nearest neighbor of each document in $\mathcal{D}_U$ needs to be determined, and the time required for each nearest neighbor determination is linear to the size of $\mathcal{D}_L$. Therefore, the computational complexity is equal to $|\mathcal{D}_L| \times |\mathcal{D}_U|$. One way of making the approach faster is to use clustering to reduce the number of training documents in $\mathcal{D}_L$. For each distinct value of the rating, the corresponding subset of documents in $\mathcal{D}_L$ are clustered into $p \ll |\mathcal{D}_L|$ groups. Therefore, if there are s distinct values of the ratings, then the total number of groups is $p \cdot s$. Typically, a fast centroid-based (i.e., k-means) clustering is used to create each group of p clusters. Note that the number of groups $p \cdot s$ is significantly

small

Bayes Classifier

In this case, we have a set $\mathcal{D}_L$ containing the training documents, and a set $\mathcal{D}_U$ containing the test documents. For ease in discussion, we will assume that the labels are binary in which users specify either a like or a dislike rating as $+1$ or -1, respectively for each of the training documents in $\mathcal{D}_L$. It is, however, relatively easy to generalize this classifier to the case where the ratings take on more than two values.

In the Bernoulli model, the frequencies of the words are ignored, and only the presence or absence of the word in the document is considered. Therefore, each document is treated as a binary vector of d words containing only values of 0 and 1. Consider a target document $\overline{X} \in \mathcal{D}_U$, which might correspond to the description of an item. Assume that the d binary features in $\overline{X}$ are denoted by $(x_1 \ldots x_d)$. Informally, we would like to determine $P(\text{Active user likes } \overline{X} | x_1 \ldots x_d)$. Here, each x_i is a 0-1 value, corresponding to whether or not the ith word is present in the document $\overline{X}$. Then, if the class (binary rating) of $\overline{X}$ is denoted by $c(\overline{X})$, this is equivalent to determining the value of $P(c(\overline{X}) = 1 | x_1 \ldots x_d)$. By determining both $P(c(\overline{X}) = 1 | x_1 \ldots x_d)$ and $P(c(\overline{X}) = -1 | x_1 \ldots x_d)$ and selecting the larger of the two, one can determine whether or not the active user likes $\overline{X}$. These expressions can be evaluated by using the Bayes rule and then applying a *naive assumption* as follows:

$$P(c(\overline{X}) = 1 | x_1 \ldots x_d) = \frac{P(c(\overline{X}) = 1) \cdot P(x_1 \ldots x_d | c(\overline{X}) = 1)}{P(x_1 \ldots x_d)}$$

$$\propto P(c(\overline{X}) = 1) \cdot P(x_1 \ldots x_d | c(\overline{X}) = 1)$$

$$= P(c(\overline{X}) = 1) \cdot \prod_{i=1}^{d} P(x_i | c(\overline{X}) = 1) \quad [\text{Naive Assumption}]$$

The naive assumption states that the occurrences of words in documents are conditionally independent events (on a specific class), and therefore one can replace P(x1 ...xd|c(X) = 1) with d i=1 P(xi|c(X) = 1).

In cases where such a ranking of the items is needed, the constant of proportionality is no longer irrelevant. This is particularly common in recommendation applications where it is not sufficient to determine the relative probabilities of items belonging to different rating values, but to actually rank them with respect to one another. In such cases, the constant of proportionality needs to be determined. Assume that the constant of proportionality in the relationship above is denoted by K

$$\llcorner \qquad \qquad \imath{-}\imath \qquad \qquad \qquad \imath{-}\imath \qquad \qquad \lrcorner$$

Therefore, we can derive the following value for K:

$$K = \frac{1}{P(c(\overline{X}) = 1) \cdot \prod_{i=1}^{d} P(x_i|c(\overline{X}) = 1) + P(c(\overline{X}) = -1) \cdot \prod_{i=1}^{d} P(x_i|c(\overline{X}) = -1)}$$

Table 4.1: Illustration of the Bayes method for a content-based system

Keyword ⇒ Song-Id ⇓	Drums	Guitar	Beat	Classical	Symphony	Orchestra	Like or Dislike
1	1	1	1	0	0	0	Dislike
2	1	1	0	0	0	1	Dislike
3	0	1	1	0	0	0	Dislike
4	0	0	0	1	1	1	Like
5	0	1	0	1	0	1	Like
6	0	0	0	1	1	0	Like
Test-1	0	0	0	1	0	0	?
Test-2	1	0	1	0	0	0	?

Rule-based Classifiers

Rule-based classifiers in content-based systems are similar to rule-based classifiers in collaborative filtering. In the item-item rules of collaborative filtering, both the antecedents and consequents of rules correspond to ratings of items. The main difference is that the antecedents of the rules in collaborative filtering correspond to the ratings of various items,

Item contains keyword set A ⇒ Rating= Like

Item contains keyword set B ⇒ Rating=Dislike

The first step is to leverage the active user profile (i.e., training documents) to mine all the rules at a desired level of support and confidence. As in all content-based methods, the rules are specific to the active user at hand. For example, in the case of Table 4.1, the active user seems to be interested in classical music. In this case, an example of a relevant rule, which has 33% support and 100% confidence, is as follows:

{Classical, Symphony} ⇒ Like

overall approach for rule-based classification can be described as follows:

1. (Training phase:) Determine all the relevant rules from the user profile at the desired level of minimum support and confidence from the training data set DL.

2. (Testing phase) For each item description in DU , determine the fired rules and an average rating. Rank the items in DU on the basis of this average rating

Rule 1: {Classical} ⇒ Like (50%, 100%)

Rule 2: {Symphony} ⇒ Like (33%, 100%)

Rule 3: {Classical, Symphony} ⇒ Like (33%, 100%)

Rule 4: {Drums, Guitar} ⇒ Dislike (33%, 100%)

Rule 5: {Drums} ⇒ Dislike (33%, 100%)

Rule 6: {Beat} ⇒ Dislike (33%, 100%)

Rule 7: {Guitar} ⇒ Dislike (50%, 75%)

It is evident that rule 2 is fired by Test-1, whereas rules 5 and 6 are fired by Test-2. Therefore, Test-1 should be preferred over Test-2 as a recommendation to the active user. Note that the rules fired by Test-1 also provide an understanding of why it should be considered the best recommendation for the active user.

Regression-Based Models

Regression-based models have the merit that they can be used for various types of ratings such as binary ratings, interval-based ratings, or numerical ratings. Large classes of regression models such as linear models, logistic regression models, and ordered probit models can be used to model various types of ratings. Here, we will describe the simplest model, which is referred to as linear regression

Let D_L be an $n \times d$ matrix representing the n documents in the labeled training set $\mathcal{D}_L$ on a lexicon of size d. Similarly, let $\overline{y}$ be an n-dimensional column vector containing the ratings of the active user for the n documents in the training set. The basic idea in linear regression is to assume that the ratings can be modeled as a linear function of the word frequencies. Let $\overline{W}$ be a d-dimensional row vector representing the coefficients of each word in the linear function relating word frequencies to the rating. Then, the linear regression model assumes that the word frequencies in the training matrix D_L are related to rating vectors as follows:

$$\overline{y} \approx D_L \overline{W}^T \tag{4.12}$$

Therefore, the vector $(D_L \overline{W}^T - \overline{y})$ is an n-dimensional vector of prediction errors. In order to maximize the quality of the prediction, one must minimize the squared norm of this vector. Furthermore, a regularization term $\lambda \|\overline{W}\|^2$ may be added to the objective function in order to reduce overfitting. This form of regularization is also referred to as *Tikhonov regularization*. Here, $\lambda > 0$ is the regularization parameter. Therefore, the objective function O can be expressed as follows:

$$\text{Minimize } O = \|D_L \overline{W}^T - \overline{y}\|^2 + \lambda \|\overline{W}\|^2 \tag{4.13}$$

The problem can be solved by setting the gradient of this objective function with respect to $\overline{W}$ to 0. This results in the following condition:

$$D_L^T (D_L \overline{W}^T - \overline{y}) + \lambda \overline{W}^T = 0$$

$$(D_L^T D_L + \lambda I) \overline{W}^T = D_L^T \overline{y}$$

Table 4.2: The family of regression models and applicability to various types of ratings

Regression Model	Nature of Rating (Target Variable)
Linear Regression	Real
Polynomial Regression	Real
Kernel Regression	Real
Binary Logistic Regression	Unary, Binary
Multiway Logistic regression	Categorical, Ordinal
Probit	Unary, Binary
Multiway Probit	Categorical, Ordinal
Ordered Probit	Ordinal, Interval-based

Advantages

It is noteworthy that the ratings of the other users usually play no role in a content-based recommendation algorithm.

This is both an advantage and a disadvantage, depending on the scenario at hand. On the one hand, in cold-start scenarios, where little information about the ratings of other users is available, such an approach can still be used as long as sufficient information about the user's own interests are available. This, at least, partially alleviates the cold-start problem when the number of other users in the recommender system is small. Furthermore, when an item is new, it is not possible to obtain the ratings of other users for that item. Content-based methods enable recommendations in such settings because they can extract the attributes from the new item, and use them to make predictions

Disadvantages

On the other hand, the cold-start problem for new users cannot be addressed with content-based recommender systems.

Furthermore, by not using the ratings of other users, one reduces the diversity and novelty of the recommended items. In many cases, the recommended items may be obvious items for the user, or they may be other items that the user has consumed before. This is because the content attributes will always recommend items with similar attributes to what the user has seen in the past

Representing user profile

Informing Recommendation Algorithms: UGC provides valuable data points that recommendation algorithms can analyze to understand user preferences and behaviors. By incorporating user reviews, ratings, comments, and interactions, recommendation systems can better tailor their suggestions to individual users' tastes and needs.

Enhancing Personalization: UGC enriches the personalization capabilities of recommendation systems by offering insights into users' diverse interests and preferences. By leveraging UGC, recommendation algorithms can deliver more relevant and targeted recommendations, increasing user satisfaction and engagement.

Building Trust and Credibility: User-generated reviews and recommendations carry significant weight in building trust and credibility. Consumers often rely on the experiences and opinions of their peers when making purchasing decisions. By showcasing UGC, recommendation systems can instill confidence in their suggestions and encourage users to explore recommended content or products.

Diversifying Recommendations: UGC helps recommendation systems avoid homogeneity by diversifying the range of content and products suggested to users. By considering a broader spectrum of user-generated feedback, recommendation algorithms can offer more varied and inclusive recommendations, catering to the diverse preferences of their user base.

Enabling Social Influence: UGC facilitates social influence within recommendation systems. Users are influenced by the preferences and behaviors of their peers, making user-generated reviews, ratings, and recommendations powerful drivers of decision-making. Recommendation systems leverage this social influence to guide users towards content or products that align with popular sentiment or peer endorsement.

Feedback Loop for Improvement: User-generated feedback serves as a feedback loop for recommendation systems, enabling continuous improvement and refinement. By analyzing UGC data, recommendation algorithms can identify patterns, trends, and areas for enhancement. This iterative process ensures that recommendation systems evolve to better meet user needs and preferences over time.

Fostering Community Engagement: UGC fosters a sense of community and belonging within recommendation systems. Users can engage with each other through comments, discussions, and interactions around recommended content or products. This sense of community encourages user participation, generates valuable UGC, and enhances the overall user experience.

Introduction

The ratings of other users may not be required to make meaningful recommendations

Content-based recommender systems try to match users to items that are similar to what they have liked in the past. This similarity is not necessarily based on rating correlations across users but on the basis of the attributes of the objects liked by the user.

Unlike collaborative systems, which explicitly leverage the ratings of other users in addition to that of the target user, content-based systems largely focus on the target user's own ratings and the attributes of the items liked by the user

At the most basic level, content-based systems are dependent on two sources of data:

1. The first source of data is a description of various items in terms of content-centric attributes. An example of such a representation could be the text description of an item by the manufacturer.

2. The second source of data is a user profile, which is generated from user feedback about various items. The user feedback might be explicit or implicit. Explicit feedback may correspond to ratings, whereas implicit feedback may correspond to user actions. The ratings are collected in a way similar to collaborative systems.

Uses of content based systems

Content-based systems are largely used in scenarios in which a significant amount of attribute information is available at hand. In many cases, these attributes are keywords, which are extracted from the product descriptions. In fact, the vast majority of contentbased systems extract text attributes from the underlying objects. Content-based systems are, therefore, particularly well suited to giving recommendations in text-rich and unstructured domains. A classical example of the use of such systems is in the recommendation of Web pages.

Content-based systems are closely related to knowledge-based recommender systems.

Table 1.2: The conceptual goals of various recommender systems

Approach	Conceptual Goal	Input
Collaborative	Give me recommendations based on a collaborative approach that leverages the ratings and actions of my peers/myself.	User ratings + community ratings
Content-based	Give me recommendations based on the content (attributes) I have favored in my past ratings and actions.	User ratings + item attributes
Knowledge-based	Give me recommendations based on my explicit specification of the kind of content (attributes) I want.	User specification + item attributes + domain knowledge

Basic Components of Content-Based Systems

preferred to convert the item descriptions into keywords

natural applications of content-based systems are also text-centric.

text classification and regression modeling methods remain the most widely used tools for creating content-based recommender systems.

- (offline) preprocessing portion,
- (offline) learning portion,
- (online) prediction portion.

The various components of content-based systems are as follows:

1. Preprocessing and feature extraction:

2. Content-based learning of user profiles:

3. Filtering and recommendation:

1. Preprocessing and Feature Extraction

1.1 Feature Extraction

- the descriptions of various items are extracted
- the most common approach is to extract keywords from the underlying data
  - descriptions of the books and keywords describing the content, title, and author. In some cases, these descriptions can be converted into a bag of keywords
- Feature weighting is closely related to feature selection, in that the former is a soft version of the latter. In the latter case, attributes are either included or not included depending on their relevance, whereas in the former case, features are given differential weight depending on their importance.

Example of Product Recommendation : IMDb rating

1. Domain-specific knowledge can be used to decide the relative importance of keywords. For example, the title of the movie and the primary actor may be given more weight than the words in the description. In many cases, this process is done in a heuristic way with trial and error.

2. In many cases, it may be possible to learn the relative importance of various features in an automated way. This process is referred to as feature weighting, which is closely related to feature selection. Both feature weighting and feature selection are described in a later section.

Example of Web Page Recommendation

Web documents require specialized preprocessing techniques because of some common properties of their structure and the richness of the links inside them. Two major aspects of Web document preprocessing include the removal of specific parts of the documents (e.g., tags) that are not useful and the leveraging of the actual structure of the document.

- For example, the title of a document is considered more important than the body and is weighted more heavily
- Anchor text contains a description of the Web page pointed to by a link. Because of its descriptive nature, it is considered important, but it is sometimes not relevant to the topic of the page itself. Therefore, it is often removed from the text of the document.
- where possible, anchor text could even be added to the text of the document to which it points.
- the structure of the layout is learned from the documents at the site by extracting tag trees from the site. Other main blocks are then extracted through the use of the tree-matching algorithm [
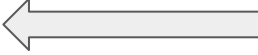
Example of Music Recommendation

The user feedback is used to build a more refined model for music recommendation

Pandora Internet radio

2 Feature Representation and Cleaning

There are several steps in the cleaning process:

1. Stop-word removal

2. Stemming

3. Phrase extraction

commonly occurring words are often statistically less discriminative

:Such words are often discounted by down-weighting.

Soft

After executing these steps, the keywords are converted into a vector-space representation. Each word is also referred to as a term. In the vector-space representation, documents are represented as bags of words, together with their frequencies.

How are words discounted? This is achieved by using the notion of *inverse document frequency*. The inverse document frequency id_i of the ith term is a decreasing function of the number of documents n_i in which it occurs.

$$id_i = \log(n/n_i) \qquad (4.1)$$

Here, the number of documents in the collection is denoted by n.

Furthermore, care needs to be taken that the excessive occurrence of a single word in the collection is not given too much importance. For example, when item descriptions are collected from unreliable sources or open platforms, such as the Web, they are liable to contain a significant amount of spam. To achieve this goal, a damping function $f(\cdot)$, such as the square root or the logarithm, is optionally applied to the frequencies before similarity computation.

$$f(x_i) = \sqrt{x_i}$$
$$f(x_i) = \log(x_i)$$

Frequency damping is optional and is often omitted. Omitting the damping process is equivalent to setting $f(x_i)$ to x_i. The *normalized* frequency $h(x_i)$ for the ith word is defined by combining the inverse document frequency with the damping function:

$$h(x_i) = f(x_i)id_i \qquad (4.2)$$

This model is popularly referred to as the tf-idf model, where tf represents the term frequency and idf represents the inverse document frequency.

Collecting User Likes and Dislikes

Aside from the content about the items, it is also necessary to collect data about the user likes and dislikes for the recommendation process. The data collection is done during the offline phase, whereas recommendations are determined during the online phase when a specific user is interacting with the system.

The data about user likes and dislikes can take on any of the following forms

Rating

Implicit feedback

Text opinions

Cases

Supervised Feature Selection and Weighting

The goal of feature selection and weighting is to ensure that only the most informative words are retained in the vector-space representation.

The basic idea is that the noisy words often result in overfitting and should therefore be removed a priori. This is particularly important, considering the fact that the number of documents available to learn a particular user profile is often not very large. When the number of documents available for learning is small, the tendency of the model to overfit will be greater. Therefore, it is crucial to reduce the size of the feature space

One is feature selection, which corresponds to the removal of words. The second is feature weighting, which involves giving greater importance to words. Note that stop-word removal and the use of inverse-document frequency are examples of feature selection and weighting, respectively

However, these are unsupervised ways of feature selection and weighting, where user feedback is given no importance. In this section, we will study supervised methods for feature selection, which take the user ratings into account for evaluating feature informativeness.

- Gini Index
- Entropy
- χ2-Statistic
- Normalized Deviation
- Feature Weighting

Gini index

Let t be the total number of possible values of the rating. Among documents containing a particular word w, let $p_1(w) \ldots p_t(w)$ be the fraction of the items rated at each of these t possible values. Then, the Gini index of the word w is defined as follows:

$$\text{Gini}(w) = 1 - \sum_{i=1}^{t} p_i(w)^2 \tag{4.3}$$

The value of $\text{Gini}(w)$ always lies in the range $(0, 1-1/t)$, with smaller values being indicative of greater discriminative power. For example, when the presence of the word w always results in the document being rated at the jth possible rating value (i.e., $p_j(w) = 1$), then such a word is very discriminative for rating predictions. Correspondingly, the value of the Gini index in such a case is $1 - 1^2 = 0$. When each value of $p_j(w)$ takes on the same value of $1/t$, the Gini index takes on its maximum value of $1 - \sum_{i=1}^{t}(1/t^2) = 1 - 1/t$.

4.3.4.2 Entropy

Entropy is very similar in principle to the Gini index except that information-theoretic principles are used to design the measure. As in the previous case, let t be the total number of possible values of the rating and $p_1(w) \ldots p_t(w)$ be the fraction of the documents containing a particular word w, which are rated at each of these t possible values. Then, the entropy of the word w is defined as follows:

$$\text{Entropy}(w) = -\sum_{i=1}^{t} p_i(w) \log(p_i(w)) \qquad (4.4)$$

Same as Gini Index