Content-Based Recommendation Systems

Charilaos Zisopoulos
Universität des Saarlandes
5 PUBLICATIONS 37 CITATIONS
SEE PROFILE

Savvas Karagiannidis
Aristotle University of Thessaloniki
6 PUBLICATIONS 36 CITATIONS
SEE PROFILE

Stefanos Antaris
KTH Royal Institute of Technology
28 PUBLICATIONS 122 CITATIONS
SEE PROFILE

Content-Based Recommendation Systems

Author team (in alphabetical order):
Antaris Stefanos , AEM : 1444 , santaris@csd.auth.gr

Demirtsoglou Georgios , AEM : 1251 , gdemirts@csd.auth.gr
Zisopoulos Xarilaos , AEM : 1460 , chazisop@csd.auth.gr
Karagiannidis Savvas , AEM : 1473

The author team consists of undergraduate students of the Computer Science Department, Aristotle University of Thessaloniki.

Abstract

In this paper we study Content-Based Recommendation Systems. This definition refers to systems used in the Web in order to recommend an item to a user based upon a description of the item and a profile of the user's interests.

To start with, we will give a definition of a Recommendation system in generally.Then, we will discuss why recommendation systems are necessary for Web users nowadays and pinpoint the problem that are trying to solve. Furthermore, we will focus on techniques used in content-based recommendation systems in order to create a model of the user's interests and analyze an item collection, using the representation of the items. Additionally, we will emphasize on the advantages and drawbacks of recommendation systems, both in the context of making recommendations and in contrast with other types of recommendation systems.
Finally, we will discuss the LIBRA content-based recommendation system and we will emphasize on the CBMRS, PRES and COBRA systems, which are implemented using the Java Platform.

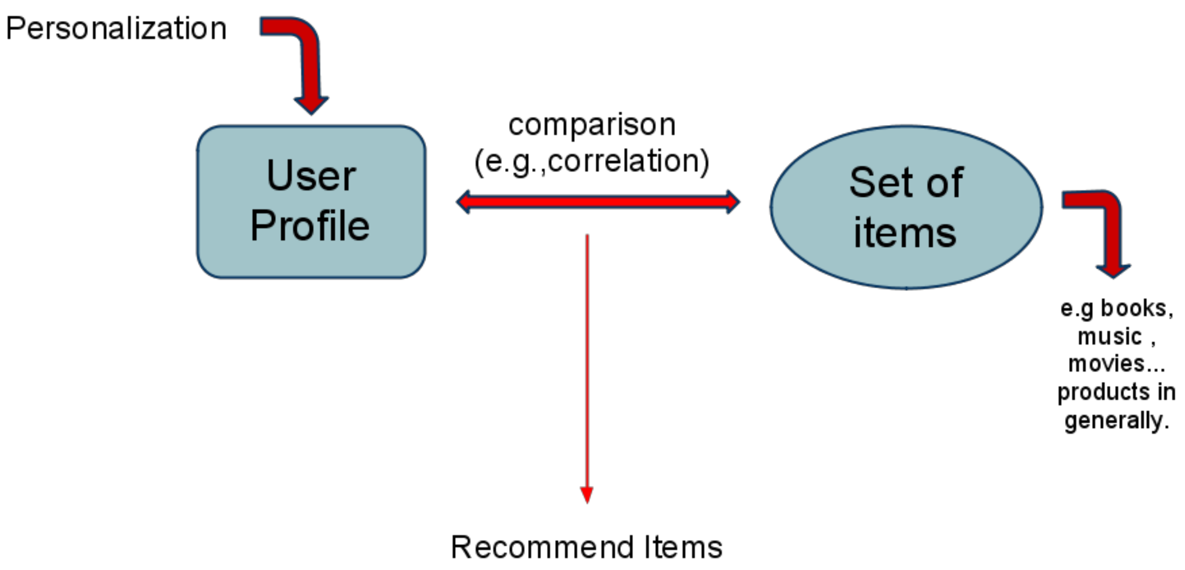
1.Introduction

As the World Wide Web is expanding in an exponential rate, the size and the complexity of the information are increasing along with it. The Web now contains a massive amount of information, most of it does not interest the user, either as unwanted information (advertisements, spam etc.) or as content irrelevant to his interests. However, every user has unique and peculiar interests, which might correlate with a small percentage of the Web's content. Therefore, it has become even more difficult and time consuming for the users to find information that they are interested it. To help users find the information that is in accordance with their interests the Web can be personalized, using recommender systems.

Nowadays,everyone has purchased a product electronically by using e-commerce technology that is conducted between businesses and consumers (is it also referred as B2C). Consider the example of buying an item from an e-shop.A user creates an account in the website and then he can purchase products on-line and rate the products according to his preferences and interests. We can use this information, combined with the item's representation, to create a profile of the user and then suggest products which are relative to his interests. The underlying technology that provides this functionality is a recommendation system.

Recommendation systems are a special type of information filtering systems. They are software applications that aim to support users in their decision making while interacting with a vast amount of information. They recommend items of interest to the users based on preferences they have expressed, either explicitly or implicitly. The continuously expanding volume and increasing complexity of information on the Web has therefore made such systems essential tools for users in a variety of applications , usually including information seeking or e-commerce activities. Recommendation systems help users overcome the information overload problem by exposing them to the most interesting items, and by offering novelty, insight, and relevance. Recommendation technology is hence an important part of the information seeking problem that has emerged along with the World Wide Web. Major e-commerce sites such as Amazon is using recommendation technology in a number of different ways. Many adopters are on their way and enterprises are competing with each other in order to find the right and efficient way to use this technology, thus providing a more efficient service to their customers.

Recommender Systems general model


Personalization
User Profile
comparison (e.g.,correlation)
Set of items
e.g books, music , movies... products in generally.
Recommend Items

We may say that recommendation systems resemble search engines, however the user is not searching explicitly for an item but the engine recommends items to the user based on his previous interactions with the system, which are used to model his preferences.

2. Techniques for User Modeling and Item Analysis

In a formal definition, recommendation systems address the following problem: Given a set of items A, a subset B of A, with $|B| << |A|$ and the values $f(b)$ $\forall b \in B$ of a function $f(x)$, $A \to \{0,1\}$, where the closed type of $f(x)$ is unknown, find an estimate of $f(x)$, namely $g(x)$, $A \to \{0,1\}$, and a subset C of A, with $|C| << |A|$ and $B \cap C = \varnothing$, so that the probability $P(g(c) = f(c) = 1)$, $\forall c \in C$, is maximized. The common target set of $f(x)$ and $g(x)$ can be also numerical, using a type of threshold to distinguish between different classes. For the rest of this paper, we will consider the target set to be boolean, unless noted differently. Or is it?

This definition can be interpreted as such: A is the set of items being offered to the user through a web service, B is the subset of items already rated negatively (0) or positively (1) and C is the subset of items to be recommended. The function f(x) represents the like or dislike of any item by the user, which obviously is not known and the function g(x) is our estimate for the user's interests, derived from a variety of methods, examined later in this section. The size of sets B and C need to be very much smaller (denoted by the use of <<) than A for practical reasons , such as the rating of a reasonable number of items by the user used as training data or recommending a number of items that the user can examine in a short time and can be presented in the limited visual space of a web page. Our goal is to create the best estimate possible g(x) of the user's interest function f(x), for every item in set A, in order to recommend items that have a high probability to match the user's interests.

In the case of content-based recommendation systems, we use the item's description and the user's rating of a small set of items in order to create an estimate of the user's interest. Then, we use this estimate to decide which items fit best the user's interests and therefore should be recommended. The problem is actually a problem of classification or regression , with both categories been studied extensively and can be solved with a variety of algorithms derived from the field of machine learning.

The problem can be broken down into 3 separate sub-problems, which address different areas of the problem. First of all , before we can apply mathematical models and algorithms to create an estimate of the user's interest, we must create a representation of the real world items in a way that they can be processed by an algorithm. In other words, we must decide the description that will be used to classify the items. Then, we must decide what model or algorithm we will use to create an estimate of the user's interests and implement it into a software system. Finally, we use this estimate in order to decide if we should recommend an item or not. The items that seem to match the user's interests the most are selected and displayed to the user.

In the rest of this section , we will analyze every subproblem and present a variety of known solutions to them, along with the limitations , advantages and disadvantages of those solutions. Furthermore , we display how this subproblems are linked to solve the original problem and how decisions in one of those subproblems might affect the decisions for the other ones. Moreover, when available, we provide experimental data on the accuracy of the results produced by the method.

2.1 Item representation

The items' representation is the cornerstone of a content based system. Combined with the user's ratings on the training data examples, they are the data that allows us to create an estimate of the user's interests.

Furthermore, once such a model has been created, the item's representation is used to decide if an item correlates highly with the user's interests. However, items are usually objects of the real world like books, movies,songs and restaurants, that an algorithm cannot process directly, so they must be converted into a data representation that the recommendation system can handle. This process is crucial to the system's effectiveness since poor representations will produce equally poor results that do not reflect reality. Therefore, the accuracy of the representation of the real world objects is a definitive characteristic of the recommending process.

Another issue is whether the representations are created by a human or the process is automated. Although an automated process is a lot more efficient in terms of performance , we must be able to certify the accuracy of the representations generated. It is not unusual for the nature of items to prohibit the use of an automated process. For example, we cannot expect (at least with current technology) a computer program to generate the description of a restaurant. Usually we rely on advances from the fields of information retrieval, data mining and pattern recognition to automatically create representations. However, it is a well established fact that techniques from those fields can be applied effectively only to a limited variety of data and even when possible , sometimes the results are disappointing compared to those produced by the descriptive skill of any human. On the other hand, allowing a human to intervene in the process may insert subjectivity into the representation, thus decreasing the accuracy of the items' representation.

Bearing those facts in mind, there usually two approaches in representing an item[2]. Although they have complementary strengths , they also have conflicting attributes , therefore not allowing for a combination that would bear the advantages of both. Those approaches are :

a) *Structured data* : Structured data refers to sets of data that have a known structure defined in a data model, usually a database schema.Items are represented using a small number of characteristics, not unlike as in a relational database. Every characteristic has a well-defined value set, while every item is described by the values of each characteristic. A unique identifier, also known as a primary key is used to identify each item, in order to distinguish them.

b) *Unstructured data*: By unstructured data we refer to data that does not have a known data model. Almost every data that does not follow a database schema can fall into this category, such as unrestricted text and multimedia data, even if they have a underlying structure, like a language's grammar. Although unstructured data is a wider term, it is common in recommendation systems to represent multimedia data using textual descriptions. Although in the usual case this requires human intervention, this representation allows us not to analyze multimedia data which usually has a much greater size than its textual description and requires complex and time-consuming analyzing techniques. Furthermore, as noted before, the modern techniques of pattern recognition from multimedia data are still in their infancy and do not always produce satisfying results. Therefore, in the rest of this chapter we will focus on studying how recommender systems treat unrestricted text rather than other unstructured data.

Frequently in the bibliography we also encounter the category of semi-structured data. This refers to data that is organized in distinct characteristics, like in the structured data model. However, the well-defined value set rule is broken and some of the characteristics are allowed to contain unstructured data with no restrictions on their value. Furthermore, the schema is not as strict as that of relational databases. This is not a representation methodology created for combining the advantages of the previous approaches, but rather used to satisfy the need for representing items that have characteristics with both well-defined and unrestricted values. Movies are such an item. The characteristics of year of release and duration have well-defined values, whereas the actors' names, the summary and the title are examples of characteristics with unrestricted values. In information extraction and data mining, semi-structured data is usually partitioned in structured and unstructured data and then treated using different techniques for each kind of data, that we will illustrate for every representation.

While both structured and unstructured data can be used to create an item's representation, structured data can be used more efficiently in order to correlate two items. This is a direct result of the small number of characteristics used to represent an item and the well-defined values of every characteristic. This strict structure allows us to treat every item as a n-dimensional vector, where n is the number of characteristics used to describe an item. Then, we can apply well known techniques from the fields of Information Theory and Information Retrieval , such as cosine similarity and Pearson correlation, in order to measure the similarity of items.

On the other hand, unstructured data consists of attribute values that are typically unique. For example, for a single book, they are an infinite number of summaries, which will deliver the same meaning to a human reader. This is a result of the inherent attributes of unstructured data, such as synonyms and semantics in any natural language or a plot and scene sequence in a movie. Those are concepts that even the most sophisticated algorithms of Artificial Intelligence fail to grasp, for the time being (let us not forget the Church-Turing thesis, which has not been disproved and seems unlikely to). Therefore, one cannot hope to use known techniques in order to find which items correlate, without transforming the unstructured data to a more convenient form. Therefore, the usual approach with unstructured data is to transform it into structured data using a number of techniques. Those techniques come from the field of Pattern Recognition when we handle multimedia data and from the field of Information Extraction in the more common case of unrestricted text documents[4].

One of the most common techniques in transforming unrestricted text into structured data is the tf-idf technique, common in Information Extraction and Data Mining. Given a set of documents, we can determine which terms are more relevant to a document's subject. The idea behind this technique is a simple one, yet very efficient. It states that the most important terms of a document are the ones that occur most often in a document, while they do not occur as often in other documents. This principle allows us to avoid words common in any language or to a set of documents to occur as highly relevant terms. Furthermore, excluding stop words from the process and group words with the same root, a process called stemming, can be used to boost the effectiveness on the method[3]. The result of this method is a number of weight values that denote how relevant a term is to a document. Obviously, terms that do not exist in a specific document have zero weights. Therefore, a document can be represented as a m-vector, where m is the number of terms present in all the documents in the set. A more efficient solution that has provided similar results is using a constant number of the most important terms for every document, thus reducing the dimensionality of the problem and as a result requiring less computing time to process the data.

The ineffectiveness of algorithms to process unstructured data without transforming it, does not imply that the structured data representation is superior. The selection of characteristics used to describe an item in a database is based on the judgment of the person that creates it. Any item in the real world has a vast number of characteristics , which are not always known in their entirety, whereas its representation has a limited number of them. Therefore, any selection of characteristics will result in loss of information, which could prove to be vital in the accuracy of the recommender's results. Furthermore, the strict structure typical of structured data can prove a severe disadvantage if proper care is not taken in its design. We cannot always predict characteristics that are needed in describing an item. For example, nowadays an important aspect of a home computer system is the number of cores of its processor units. A database containing computer system items would need to have a field added to describe this information and this field would require a value for every computer item existing in the database. Another common problem in structured data is that the range of values of a characteristic might be modified. This is common with enumerated types of data, when a new value emerges that was not anticipated. Again taking computer systems as an example, as new technologies emerge, new values might occur in fields such as "network type", which we cannot predict. We come to the conclusion that while structured data is more convenient for algorithms, it is not flexible enough to the ever-changing nature of the real world. However, given the vast research and advance of database systems and other related fields, it's a powerful tool that we should not ignore.

On the contrary, unstructured data has a significant advantage. Free of any predefined structure, it allows us a great degree of flexibility in its use. Therefore we can use data easily interpreted by the human brain, such as multimedia data and textual descriptions, without transforming them to a more formal representation. Although this may allow a human being (or an algorithm made to imitate human behavior) to describe an item given a greater level of detail, it is more difficult for an algorithm to use it for correlating items, as we've discussed previously in this section. However, the process of transforming unstructured into structured data allows us to create a unique structure for any set of given items, based on the representation of the items themselves rather than known facts. While this may not be an optimal solution for items that can be described efficiently using a small number of characteristics, it allows us to extract the terms that define an item the most, especially when not much is known about the abstract nature of the item. Usually, this is achieved using special techniques to decide which parts of the unstructured data representation are more important in its definition, in relevance of all the other items' descriptions. For example, we have demonstrated how the tf-idf technique allows us to find the most definitive terms of any document in a known set of documents.

After deciding on a representation of the real world items, we can use them as input for our problem. There is a great selection of algorithms and techniques that can process this data. However, most algorithms that derive from the Data Mining and Machine Learning fields were designed with structured data in mind, therefore they are more efficient with it rather than with unstructured data, even after being transformed, due to its high dimensionality. That should not be disheartening in the use of unstructured data, since many algorithms exist that can analyze it efficiently. On the following sections, we will demonstrate some well known algorithms and discuss whether they are suitable for one representation or the other, among other issues.

2.2 User Modeling, Item Analysis and Recommendations

The model of the user's preferences is one of them most important aspects of any recommendation systems. It allows us to make a prediction about the user's interests and when combined with other information found on the user's profile, to recommend the items that the user would be interested in the most. Of course, an algorithm is needed in order to construct a non-trivial model. This algorithm is the core of a recommendation system and uses examples from items that the user has already examined, also referred as the training data set. Having created a user model, we can use it in order to find which items correlate the most with the user's interest, in order to recommend them. Usually, the same algorithm involves both creating a user model and estimating a user's preference for an unrated item. This is a result of the assumptions every algorithm makes and the method it uses to classify the results. In order to provide accurate results, every algorithm consists of a methodology to create a user model and a similar methodology used to estimate the ratings of yet to be rated items.

Although a vast variety of user modeling algorithms exist , almost all of them have a common feature. They need a training data set, which in the case of recommender-systems , is a number of items that the user has rated either positively or negatively. The minimum number of items needed to create an accurate model of the user's preferences can be considered one of the factors contributing to an algorithm's efficiency. For this fact to become clear, we first need to examine the different ways of rating the items in the training data.
There are generally two approaches in learning the user's like or dislike of a certain item [2]: Explicit and implicit feedback. Explicit feedback involves the straightforward approach of requesting a user to rate an item, using a number of different rating systems , from the "thumbs-up or thumbs-down" approach of a binary scale to a numerical scale of ratings, e.g. from 1 to 10. On the other hand, implicit methodology involves monitoring the user's behavior towards items, such as the time of viewing the item's web page, whether the item was purchased or not and so on.

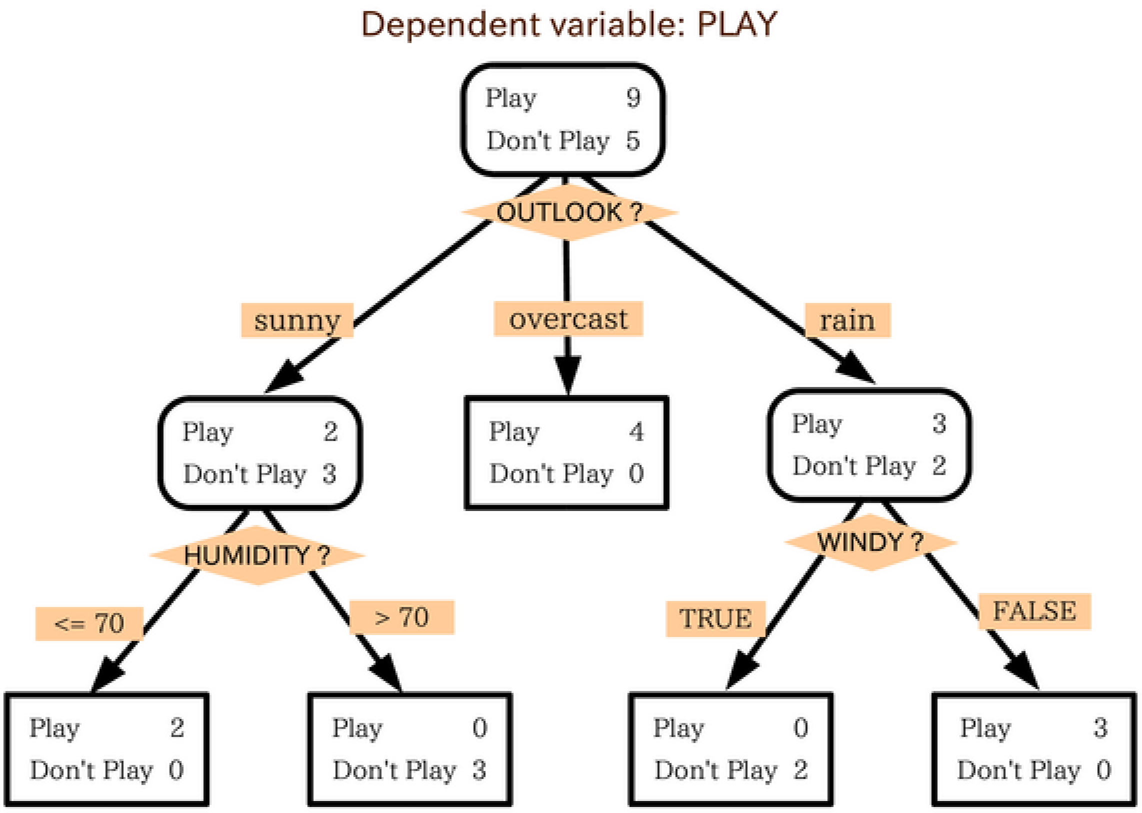
In both methodologies , the smallest the number of items needed for accurate results , the better. In the case of explicit feedback, we must keep the number of items rated by the user to a minimum, as to ensure that the ratings are sincere and that we do not degrade the user's experience of using our service, whereas in implicit feedback, for every user time and computational resources must be allocated in order to monitor his actions. Furthermore , those methods offer conflicting benefits. While the explicit methodology offers reliable results, we must expect the user to provide a small number of rated items, whereas in implicit feedback we can have a much larger number of rated items, although we cannot be certain about the accuracy of the results, since we depend on empirical rules[2]. Another problem that may arise is that the nature of the problem might prohibit the use of implicit methods, for the simple reason that we do not have a sufficient number or strong enough indications in order to rate an item. For instance, consider the rating of movie items. We cannot record the user's behavior accurately, therefore we do not have enough clues to rate an item implicitly. In this case , explicit rating by the user is the only way of action.

There are problems shared by all techniques in user modeling and item analysis. Those problems arise from certain properties that characterize the training data set. The most common one is the size of the training data set. We've already mentioned that we would like a small training set and we've analyzed the reasons behind this decision. However, when we have a very small training data set not by choice, but by necessity, that might be a problem. The number of items might not meet the necessary requirements needed for an accurate method. Therefore, the results of the model created could be inaccurate. This fact makes the selection of the training data set size a problem of optimization, where little adjustments might introduce significant differences in the accuracy of the recommender system. Furthermore, another property of the items of the training data set might be critical in creating our model. Although this may vary from technique to technique, generally speaking, we prefer items that have regularities in their general content. Since most techniques use similar values in the same characteristic to classify items, it is more efficient to create a model when the properties of every class are clear and distinct from other classes. Therefore the accuracy of the model of the user's interests, relies on whether those properties are inherent in the training data set or not. An extreme instance of this problem is when we have only positives examples, therefore we must use specialized techniques in order to create a model that can classify items correctly. There is a lot of research carried out in this problem, with the latest trend being the use of anti-patterns, where we look for items that are dissimilar, rather than the more common approach of searching for patterns occurrences in the item set.

Once we have acquired a sufficiently large training data set, we can apply a user modeling algorithm. There are many algorithms, with a number of advantages and disadvantages , while no-one of the existing algorithms has a clear advantage, at least backed by solid proof. Some algorithms provide better results under certain circumstances, while they may not be efficient in every recommendation problem. A common feature that distinguishes algorithms is whether they work better with structured or unstructured data. We will now present three of the most common techniques applied , in their theoretical basis and provide experimental results when possible. In order to investigate the inherent advantages and disadvantages in every method, we will study them in theory, rather than deal with their various implementations. However, we will note significant implementations wherever possible.

2.2.1.Decision Trees

Decision trees are a methodology that derives from the field of Decision Theory. Given a training set, they create a model that relies on the values of items in certain characteristics in order to accurately classify a previously unclassified item, using the least number of characteristics possible. Every node of the tree represents a set of items, with restricted values on their characteristics. The root node does not follow that rule, including the whole set of items, whereas the leaf nodes consist of items that belong only to one class or have restricted values in every characteristic. To create those sets, data metrics from the field of Information Theory are used, commonly Expected Information Gain and Information Entropy, in order to select the characteristic that partitions the set of items the most , depending on the values the items have in this particular characteristic. This partitioning of one set into several subsets is represented by branches in the decision tree, leading to different nodes, each for every value of the selected characteristic. While this technique is suitable for discrete value characteristics, continuous value ones need to be subjected to a process of discretization. An example of a decision tree follows, using the characteristics of atmospheric conditions during a tennis game, namely outlook, humidity and wind, in order to determine whether or not to play on a certain day.



A decision tree for playing tennis

The decision tree can be created using a number of algorithms, ID3 and its variations being the most common ones. However, the common principle of the methodology is to select the attribute that creates the most heterogeneous sets possible, thus allowing for the quickest classification possible, while creating sets as uniform as possible. Nodes on the same tree level need not use the same partitioning characteristic, as displayed in the above example, since every characteristic selection depends on the set that the node represents.Therefore, different branches of the tree may follow a different series of characteristics, allowing for maximum efficiency on the classification process. However, the leaf nodes do not always consist of items that belong to only one class. Given the nature of the training data set, it is possible that we may not be in the position to be definite of the class a set of items belong to. However, we can provide estimates of whether an item of that set belongs to one class or not, in the form of probabilities. Those probabilities are calculated as the ratio of training data set items in that leaf node that belong to a specific class, to the total number of items in the leaf node. Then, we can either classify an item with a certain level of confidence or choose not to classify it, depending on the value of the specific probability and how crucial for the application the result can be.

In the case of recommender systems, the training data set is the number of items rated. We usually have only two classes, consisting of the items favored or not by the user. The training data set is used in order to create the decision tree, following the process explained above. Then, we simply estimate if an item might be relevant to the user's interests by inserting it in the root node of the decision tree and then traversing the tree until we reach a leaf node, given the item's value on the partitioning characteristics. Then , we may have either a definite rating, having 100% confidence, or a probabilistic one, which we can use in our recommendations. In order to interpret the results, we need to select items that have classified as positive, while the level of confidence in this classification can be used as a comparison measure. Then, we can recommend a number of those items to the user.

One of the greatest advantages of decision trees is their performance in terms of time needed to estimate an item. Once we've build the user model, we need only a few comparisons to classify an unrated item, namely a logarithmic number of comparisons (O(logn)) , since we actually execute a tree traversing. Furthermore, decisions trees are a well-documented tool, used in Decision Support Systems, being implemented by a wide number of algorithms, each invented with a certain goal in mind, thus having certain strengths. Thus, we are able to search through bibliography or implementations of algorithms, in order to find one that suits us the most, depending on the nature of the items' representation and the particular instance of the recommendation problem.

However, this method is quite static and does not function well in the dynamic environment of the Web. First of all, the vast majority of data in the web is unstructured. From our analysis it is clear that decision trees is a machine-learning method used with structured data, with well-defined value ranges in their characteristics. Thus, any unstructured data needs preprocessing before being eligible to be processed by a decision tree implementation, as we've shown in the previous section.

Another drawback of decision trees is that they depend highly in the training data set to build the user model. We would like to be able to rebuild the user model with information that we gather after the initial model has been build, such as new ratings or purchases the user has made. Although techniques exist that allow us to perform this adjustment, in decision trees we would have to repeat the whole process of creating the tree, since the result depends on the set used. This may be a major problem, as the number of rated items increases by time.

Furthermore, decision trees' dependence on the training set introduces a problem common to machine-learning algorithms. Overtraining is the problem we face when the model we have gives accurate results for the training set it was created from, but fails to be acceptably efficient in more general subsets of the problem. That can be attributed to the fact that the model was build on assumptions that are true for this particular training set although they do not apply in the much large set of unrated items. Therefore, they produce poor results when applied to sets foreign to the training data set, which is always the items that we're interested in, since we want to estimate their value in a function, being the user's interest in recommender systems.

Even if we assume that we have eliminated the danger of overtraining, either by carefully selecting our data set or by using specialized techniques that address this problem, decision trees have a significant problem when used for recommendation systems. The only comparison measure for the item's correlation with a known class is the confidence of the classification, which is not a very strong one. This may become clear if we take into consideration that it is common to have many items that are classified with a 100% confidence. Thus, the question arises of selecting a method in order to select a constant number of items to be recommended. We may assume that selecting them randomly will provide efficient results, since they have a similar rating in the model, however this assumption quickly collapses when we consider the heuristic nature and limitations of this particular technique, which is common with machine-learning algorithms. Therefore, we do not have a measure strong enough in order to guarantee that the selection of items recommended are the ones the user is most likely to favor, which is the problem recommender systems are trying to solve.

In the light of those facts, we should be disheartened in the use of decision trees. They are a powerful technique, but they should be used with caution and by always taking into consideration the nature of the items' representation. However, wherever applicable, decision trees guarantee accurate results while being efficient on resources usage.

2.2.2.Bayesian Classifiers

Bayesian classifiers use techniques from the field of Probability Theory, namely Bayes possibilities and the Bayes theorem, in order to classify items. A Bayes probability gives the probability of an event A, given that an event B has occured. This allows us to have a better view of a probabilistic system, given certain events that may occur. Furthermore, the Bayes theorem allows to calculate such Bayes probabilities, represented by P(A|B), given that we know the probability of event B, the probability of event A and the probability of event B given that event A occurred.The formula for Bayes theorem is the following:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Although this formula may seem impractical at first, since we need three probabilities in order to calculate one, the Bayes theorem is not only one of the foundations of Probability Theory, but also used widely in many fields of Artificial Intelligence, one of them being machine learning algorithms. This importance lies in the usage of the Bayes theorem to perform stochastic root cause analysis, in which we discover the cause of a problem, by studying its symptoms, like in modern medicine diagnostics.

Bayes classifiers differ greatly from the decision tree technique we've analyzed, particularly because they're one of the most popular methods in analyzing unrestricted text data. Furthermore, the classifiers used for this purpose have been altered as to encounter the particular problem of representing adequately unrestricted text documents. In particular, one special class of classifiers is being used for this purpose, naive Bayes classifiers. Their naivety lies in certain hypotheses they make as a basis to characterize the items they're classifying. In naive Bayes document classifiers, the hypothesis is made that every word in a document is dependent of the class the document belongs to, for example in terms of genre, but are independent of one another. Although this hypothesis may seem weak, since it is common in documents that certain words are used together or in the same context, naive Bayes classifiers have demonstrated dependable results in text classification [4].

Furthermore, Bayes classifiers do not require documents to be represented by a vector, as it is common in the so-called vector space model that is used widely for unstructured data. They rather treat texts us a collection of words, thus being able to use the product rule of Probability Theory and the probabilities of distinct words, in order to calculate the probability of a document, given its classification. Then, by using an altered version of the Bayes theorem that does not require to calculate the independent probability of every document, we can calculate the probability of every class, given a document, which is a process of classification. Therefore, in order to classify an item we need the following terms :

a) The probability of every word, given that it is used in the context of a class. This probability is not used in the classification directly, but rather is used to calculate the probability of a document. For every class, we calculate the probability of a word, given a class, or $P(w|c)$, where w depicts a word and c a class. For this purpose, the training data is used, since the user provides classification for every document. A number of methods are used to calculate this probability, where some take into account the existence of a word or not while others count the number of occurrences of a certain word in a document, as described in [2]. Therefore, different techniques create different classifiers from the same data training set.

b) The probability of a document, given a class. As we've noted before, this can be achieved by using the probability of the words that comprise the document. As in the case of calculating the probability for every word, we can again either calculate this probability using a variety of different methods, the most common ones taking either the existence or the occurrences of a word in a document. Usually, the same method is used in both probabilities, in order to maintain uniformity in the process and guarantee the accuracy of the results.

c) The probability for every class. As with the probability for every word, this can be simply extracted by the training data. Although the usual approach is to calculate probabilities as the percentage of every class in the training data set, it is not uncommon to use certain techniques to normalize the results, in order to create more accurate results, as those applied in LIBRA [4].

Once we've calculated these three figures, we can calculate the probability of a class, given a document. Then , we can classify any item by considering it to be in the class that had the greatest probability to contain this document. This approach gives naive Bayesian classifiers certain inherent advantages, that seem to overcome certain disadvantages found in decision trees, possibly due to the different approach those two methodologies follow.

Although we've mentioned before that Bayes classifiers are considered as one of the most efficient tools in text analysis, they have no inherent limitation that prohibits the use of structured data. Although the vast expansion of relational databases and Data Mining have given us extraordinary tools in structured data processing, we may choose to use a Bayes classifier, depending on the nature of the problem. Though in the general case Bayes classifiers do not provide as accurate results as Data Mining techniques, their ability to be used with any form of data makes them a very flexible tool, which can be used with a variety of data, in contrary to many machine-learning algorithms, such as decision trees.

Furthermore, while we noted that decision trees do not possess a strong enough measure to ensure the accuracy of our results, that is not the case with Bayes classifiers. Instead of a more or less strict classification, the output of this method is, for every class, the probability that the item in question belongs to that class. Therefore, we have a metric that we can use in order to select a constant number of results, noted as top-k, to display to the user. Furthermore, we use a threshold of confidence in those probabilities, in order to classify an item. Therefore, we can validate the accuracy of our results, since in some applications it is preferred that ambiguous items are not classified at all, rather than be classified incorrectly. As a consequence, we may use a Bayesian classifier, even if we don't have solid proof on its accuracy on a specific problem, by introducing a confidence threshold.

Another difference between decision trees and Bayes classifiers is that Bayes classifiers do not only depend on the training data set, but also on their hypotheses, or their "naivety", and the technique used to calculate probabilities, in order to produce results. Therefore, from a single training data set, a vast number of naive Bayes classifiers may be modeled. Usually, we construct our classifiers in order to reflect certain properties that characterize the problem. For example, in text classification, classifiers that use the number of occurrences of a term rather than its existence, a metric which provides a better insight to the topic of a document, produce better results. This allows us to avoid the great problem of many machine-learning algorithms, namely overtraining, by adjusting the classifier to the specific problem.

As we've noted before, no technique is not without its weaknesses. In Bayes classifiers, this weakness lies in the hypotheses they make. A probabilistic classifier that would not make any hypotheses, would maybe be more complex to create and use, but would produce more accurate results. However, we do not yet have knowledge of a method to create such classifiers, at least not in non-exponential times, depending on the input size.However, even though their naivety, Bayes classifiers have shown to be more accurate than more complex methods used in text classification. For example, in the case of LIBRA [4] , experimental evaluation of a Bayes classifiers has shown promising results. Bayes classifiers are considered to be almost 70% accurate on their ratings, a percentage that is acceptable in most applications. For example, consider that in a system that recommends three items to a user, two of them would be likely to be favored by the user, therefore the goal of the recommender system is met.

2.2.3. Relevance feedback

So far we've seen two methods, decision trees and Bayes classifiers. Although both methods have proven to provide adequate results in a specific context, they lack an important factor needed in recommender systems: adaptability in new input data. Once a Bayes classifier or a decision tree model has been created, there is no way to introduce new data in it, we can only use it as it is. The only way of taking into consideration new data provided by the user, either explicitly or implicitly, is to build the model again from scratch. However, this approach is time-consuming, since it is dependent of the number of items that are used to build the model. Therefore, as a user uses the recommender system more and more, the system will not be able to adapt as easily. Therefore, we reach the dilemma of either providing less accurate results or finding a means for providing more computational power, thus raising costs, without any guarantee that we will meet the users' needs.

In response to this problem, the technique of relevance feedback has been introduced. This technique allows us to adjust our model to new user data, therefore providing more accurate results without the need to rebuild the model from scratch. The most popular algorithm using relevance feedback is Rocchio's algorithm [3]. This algorithm operates in the vector-space model. Therefore, unstructured data must be converted to structured one, as we've demonstrated with the tf-if technique. Then, every item is represented as a vector in a n-dimension space, where n is the number of characteristics. In the case of unrestricted text, the most important terms play the role of characteristics, each of them assigned a weight, that represents the term's importance in the classification procedure. The user profile is represented as a vector too, one that correlates highly with positively rated results, while it does not correlates as much with negatively rated results. As a measure for this correlation, a number of techniques can be used, depending on the case, where cosine measurement and Pearson correlation are the most common ones.

The following formula can be used to describe the function of Rocchio's algorithm:

P' = a*P + b*ST(rel) - c*ST(non-rel)

P indicates the current user profile, ST(rel) the sum of all relative rated items, while ST(non-rel) the sum of all unrelated rated items. Finally, P' is the updated user profile. Parameters a,b and c control how each one of these vectors control the outcome. Therefore, if we have a new rated item that we want to add in the user's profile, we only have to execute the following operation:

- P' = a*P + b*T , if the item T has been estimated as relevant.

- P' = a*P - c*T , if the item T has been estimated as non relevant.

Therefore, we can update the user's profile, until the vector that represents it correlates highly only with relevant, yet unrated items.Parameters b and c control how fast this conversion occurs, for relevant and irrelevant items respectively, while parameter a controls how long previous preferences affect the user's profile. In some systems, like PRES [3], that deal with contemporary preferences, parameter a has a small enough value, as to indicate the rapid change of the user's interests.

Feedback relevance has a great advantage as a method. Being able to update the model of the user's interests allows us to create more accurate results, therefore providing better recommendations. This can be very useful, especially in today's Web 2.0, where users spend a lot of time on-line, browsing a lot more web pages than before. Recommendations that adopt to the user's interests almost instantly means that we can even offer recommendations of items even as the user browses an item, rather than build its profile off-line, due to the great computational cost.

However, relevance feedback has two significant disadvantages. First of all, as we noted, it works only with structured data. Therefore, unstructured data must be first converted. Although this is rather asn easy task for text documents, considering the advances in Information Retrieval, when taking into consideration multimedia unstructured data, there does not exist a method that provides accurate results. However, the greatest drawback of relevance feedback is that there is no strong proof of its accuracy. It has not been proven that for every case, the algorithm will perform adequately and that the user profile vector will converge to the related documents, although that seems to be the case. However, without knowledge of the method's precision, adoption is rather unlikely by recommender systems, which are highly prized by web services companies, as a valuable service to their customer base. Therefore, although relevance feedback shows significant results, we must wait until strong scientific proof exists, in order to determine whether this technique can compete with the vast number of algorithms that exist. However, until now experimental data has been promising.

3.Advantages and drawbacks

In the previous chapter we offered a technical presentation of recommender systems, we examined popular classification algorithms and we presented their advantages and disadvantages. In this section, we will examine the advantages and drawbacks of content-based recommendation systems in general, without taking into account the algorithm being used. As we've shown, content-based recommender systems rely heavily in the items they recommend, hence their name. We will examine how this approach in making recommendations generates inherit advantages and disadvantages in any content-based system. In this context, we will examine content-based systems in comparison to other recommendation methodologies that exist. Furthermore, we will present some advantages and drawbacks of recommender systems in general.

3.1 Advantages

One of the greatest advantage of content-based recommender systems is realized when we compare the methodology used by them and by collaboration filtering systems. The latter rely primarily on user ratings, in order to make recommendations, by creating user groups that are described by common interests, also reffered to as a user stereotype. Therefore, in order to make recommendations for any item, we need them to be rated by a number of users, in order to recommend items to other users, based on their similarity to users that have already rated the item. That is not the case with content-based systems. Since they rely on the content of each item, they do not use the information of ratings by other users. Therefore, the only prerequisite for making a recommendations is the rating of the training data set. This fact gives recommender systems a number of advantages.

First of all, since recommendations depend only on the user's ratings, we can make recommendations that match the unique taste of every user. That is not the case with collaboration systems, where users are categorized using stereotypes. As a result , content-based systems offered a greater level of personalization in the recommendations. Furthermore, since recommendations are not influenced by the number of users, we can make recommendations even for users with peculiar interests, since we take into account only the content of the items. This characteristic of content-based systems is a great advantage, especially in the case of vast sets of items. When we have a number of items available that greatly exceed the number of users, we cannot rate all of the items as training data. That would prevent a collaborative system form making recommendations, however a content-based system would just use the content of every item in order to make them, therefore being more efficient. This same principle also applies in the case of newly added items. Although in collaborated systems new items would have to be rated by users, that is not necessary in content based systems. An analysis of their contents would provide the information necessary to make the recommendation.

Furthermore, since content-based systems make recommendations for a single users that are not dependent of the interests of other users, in contrary to collaborative filtering systems, the number of resources and time needed to make a recommendation does not increases as more users are added to the system. Therefore, the algorithm of making recommendations is scalable in terms of the number of users in the system. It must be noted that this refers to the process of recommendation and not the system. It is inevitable, as more and more users use the system simultaneously, that resources must be allocated to provide recommendations for every user. However, the recommendation process itself for every single user, does not depend on the number of users in general registered in our service, both off-line and on-line. This fact becomes more clear when we consider that content-based systems examines the items
themselves and not the users of the system. As an immediate result, the system is not scalable to the number of items in the item set, a disadvantage we will analyze later in this section. Taking these facts into consideration, content-based system should be preferred when the size of the item set is stable or increases in a slow rate, whereas many new users are registered every day. However, most web services are characterized of a great rate of growth in both the number of items and the number of users.

In addition, not depending on the basis of users for making recommendations gives another advantage to content-based recommender systems : inherent security from malicious item creation. This allows us to prevent viral marketing methodologies that can be used in order to promote or denote a certain item. In collaborative filtering systems, we can create a vast number of fake user profiles, each with different ratings, with a common point : giving a positive or a negative rating in a specific item, depending on either the malicious users wants to promote it or denote it, respectively. Therefore, it is possible through spam user registrations, a technique commonly employed in the Web and rather difficult to defend against, even with the use of reverse Turing tests [9], to affect the accuracy of the system's performance. In a content-based system, recommendations depend on the content of the item, therefore, such an attack would be almost impossible to carry out. Furthermore, since the user's model of interests is not affected by other users preferences, malicious advertisement would be more difficult, if not impossible, to carry out, even in the case of an attack.

Another advantage of content-based recommender systems is that they have the ability to present the logic behind their recommendations. We can inform the user which items that he has previously rated had a greater weight in the recommendations process. Therefore, we can help the user understand its preferences in a greater degree and strengthen the accuracy of our results, by giving the user a simplified view of the system. On the other hand, in a collaborative filtering system, that would be impossible due to a number of reasons. First of all, it would be insignificant to the user what other user profiles affected the recommendation. Since the Web is worldwide and its users prefer anonymity, it's highly unlikely that this information is of any value to the user. Furthermore , users are always sensitive of their private data, therefore it would be unethical and in most cases illegal to reveal private information concerning their preferences. This advantage of content-based systems has been adopted by the hybrid model and in particular by demographic recommender systems, that allow us to create users groups relevant to the content, therefore being able to explain their recommendations, without exposing private data.

3.2 Drawbacks

One of the greatest problems of content-based recommender systems, is the vast size of the item set. Since we need to find items in a set that correlate the most with the user's interests, we're obliged to examine all the items. In any other case, we cannot eliminate the possibility that items we haven't examined are not more relevant than the ones we did. Moreover, in the case of content-based recommender systems we must examine the content of every item in order to make a recommendation, whereas in collaborative filtering systems, we only need to examine their ratings by the users. Therefore, the number of items rises very quickly, as is the case in most e-commerce services, the performance of a content-based system decreases. As a result, when we intend to use a recommender system for a web service with a vast number of either old or new items, the solution of a content-based system would not be likely to meet our expectations in terms of performance.

However, the size of the item set as a whole is not the only problem of content-based systems. In addition, every item has its own content that the algorithm must use. Although this usually is not a problem of time or computer resources, since most item representations tend to be small compared with the number of items, how this content is derived from the original item is one of the most important problem we encounter when building a content-based system. In the previous chapter, we've analyzed how items are represented and the different techniques. We also noted that unstructured data is not easy to handle, especially when it comes to multimedia data. However, multimedia data are prevalent in today's Web 2.0, while a vast number of new multimedia items is being added to the Web every day. While we still do not possess techniques that produce satisfying results, the user's need for recommending multimedia items increases every day. Although several attempts have been made to solve this problem and progress has been made [1], the problem still remains. Therefore, the content analysis needed by content-based systems in order to make recommendations, is an inherent problem that might discourage their use when we deal with multimedia items.

Even when all other problems are ignored, there is a problem common in all recommender systems. Due to the difficulty of estimating the user's problem, the estimations given by any system are far from perfect. Although the fact that with hybrid systems we've combined the advantages of both content-based and collaborative filtering systems, still the results are satisfying but not certain. That is a problem that cannot be countered, at least with the techniques we possess today. However, it is unlikely that any recommendation system would solve this problem, because it is not in its entirety a common computational problem. Its great complexity lies in that it is not easy for the user to provide some kind of input that would fully describe its interests. Usually, although most humans have a common basis of interest, their individual choices may vary, due to factors that may be unclear, as their current mood or events that occurred in their lives. For example, a researcher might need to read a number of books for a new project. The recommender system cannot predict this change of interest, therefore making the problem a difficult one. Furthermore, even when it is easy to process the content of an item, like in the case of structured data, we cannot always interpret the nature of the item itself, in its entirety. As we've mentioned before, this is a result of the great number of characteristics an item has in the real world, whereas its representation has a limited number of them. Therefore, it is inevitable that the representation is only an approximation of a real world item. However, since algorithms cannot process real world objects, this procedure is necessary. Therefore, it becomes even more difficult to analyze the true meaning that an item might have to a human being, therefore making the process of recommending an item even more difficult.

Although many of the content-based advantages derive from the fact that the recommendations for every single users are independent to the user's preferences, that might be the cause for one of the its disadvantages. Content-based systems emerged over a decade ago, when the Web was still young and not widely adopted. User communities were more primitive, where the user's profile consisted of a few fields providing information like his name and his age, whereas communications was restricted to text. However, nowadays, with the use of social networks, the Web is more and more used in the context of a community, where user profiles are extensive and constantly updated with information about them, while every user is connected with hundreds of other users and communication between them is achieved using a variety of ways. Although content-based systems allow us to make recommendations to user with unique interests, they fail to group users that have the same interests.Therefore, the lack of a content-based recommendation system to create a group of users that share common interests, might prove to be a great drawback. However, collaborative filtering systems cannot create user groups neither, because they do not know the common interests of the users, but only their existence. Therefore, the hybrid model has again proven superior. By combining the item's content with the user stereotypes, it allow us to create cluster of users that share interests, as a result making it possible to create communities of users that share common interests. The result of this feature is not only to improve the user's experience of using our service, but also providing the accuracy of our recommender, since through social interaction the user's provide information that might prove very useful to the recommender system. This realization has created a new trend in recommender systems, where an item is recommended to a group rather than to individuals, thus increasing the possibility that the results are accurate, at least for most of the users of the community.

4.Examples of Content Based Recommendation Systems

Nowadays, recommendation systems have been widely adopted in the Web. Although some users may have concerns about the use of their private data in order to receive a greater quality of service, most users are amazed by the services that recommender systems offer them and trust the company's confidence agreement. Companies that operate in the web services market, have acknowledged this fact and have adopted accordingly their policies. Especially in the e-commerce market, recommender systems are viewed as one of the most valuable assets a company possesses. This can be attributed to the unique opportunity that recommender systems offer, that allows a company to provide personalized services to every customer, with minimum overhead and cost, in an automated way.

Content-based recommendation systems were the first approach to recommender systems, being developed since the mid 90's and they were quickly adopted by major web companies on their web sites. Therefore, although many of the recommender systems employed in the Web are content-based, they are not dominating the field. In the early years of recommender systems, the main competition was collaborative filtering systems, however the hybrid version prevailed, combining the advantages of both methodologies, thus offering better results. However, many companies still prefer the usage of content-based systems, that decision based on either the specific nature of the problem, for example uncertainty in the user's ratings, or business policies, like the use of a pre-existing content-based system and know-how the company has invested on.

During the period content-based recommender systems were in their prime, the Java platform was not as popular as it is today. However, there are many content-based recommender systems built using Java technology. The use of the Java platform has transformed the way we build Web applications, including recommender systems. It allows us to develop large-scale Web applications in a single robust framework, without the need for individual tools, while providing tools for important operations, like connecting databases to our applications or providing secure services.With the use of Java technologies, like servlets, the JDBC library and the Java Security library, we are able to provide a unified service, which is easier to develop and maintain.

In the spirit of this paper, we are going to present four examples of Content-Based Filtering Recommendation Systems. Three of them are developed using the Java language, therefore we will analyze their architecture, in order to demonstrate the use of the Java platform in recommender systems. In the remainder of this section, we will discuss the following systems :

1. LIBRA : A Content-Based Book Recommending System, using learning for text categorization.
2. CBMRS : A Content-Based Music Recommendation System.
3. PRES : A Content-Based Home Improvement Recommender System.
4. Cobra : A Content-Based Filtering and Aggregation of Blogs and RSS Feeds

4.1. LIBRA

LIBRA (Learning Intelligent Book Recommending Agent) is a content-based recommendation system that uses machine learning methodology in order to extract semi-structured text data from the web, for the purpose of making book recommendations [4]. The data extracted is relevant to a specific book and it is used to represent it in the system. LIBRA analyses the books' meta-data, in order to find books that correlate highly with the user's interests, thus making them a suitable candidate for a recommendation. LIBRA is operational in an experimental environment. The following link , http://www.cs.utexas.edu/users/libra/help.html provides more details in order to use LIBRA on-line.

Rather than manually labeling each book, a methodology used widely in earlier content-based book recommendation systems but rather inefficient, LIBRA uses machine learning techniques, implemented in an automated procedure that creates a database consisting of book information. In order to populate the database, data for every book is extracted from the web pages of the Amazon.com web site. In order to identify the site related to every book, the Amazon subject search service is used. Therefore, instead of using the entire text contained in the book, we represent every item using textual meta-data, thus greatly reducing the input data size for the recommendation system. When this procedure is completed, the user provides a 1 to 10 rating for a selected set of books, which comprises the training data set the algorithm uses in order to construct a model of the user's interests. The user model is created using a Bayesian learning algorithm that represents the user's interest as a ranked list of the features that are common in highly rated titles that exist in the system's catalog. In the rest of this section we provide the details of the information extraction from the semi-structured data and the user's profile learning procedure.

As we've mentioned before, the system at first creates the database which will be used by performing an Amazon subject search to obtain a list of book-description URLs of relevant titles. After these pages are downloaded, information is extracted from the semi-structured data they contain by locating specific information we're interested in. This allows us to represent every book as structured data, rather than in its native unstructured text data. In order to represent a book, LIBRA finds a set of substrings (wrappers) from the document for each set of pre-specified slots. These slots contain the data which is going to be used by the content-based algorithm to provide the recommendations.The particular algorithm of LIBRA utilizes the slots of title, authors, synopses, published reviews, customer comments, related authors, related titles and subject terms. LIBRA requires that books with at least one synopsis, review or customer comment , in order to have enough information to produce an accurate result. The strings that are contained in each slot are separated into atomic words, also known as tokens, which are then inserted into an unordered bag of words. As a result, there are as many bags as there are slots. This process allows us to represent every book in a vector space, thus allowing for the application of most machine learning algorithms.

After the creation of the system's database, the user is required to rate a set of training books. This set is selected either manually by the user by searching for particular authors or by the system, automatically providing random titles from the database. The learning algorithm of the current system is a simple Bayesian text classifier , extended by the creators of LIBRA to handle multiple bags of words. Through the user's ratings the set of books is separated into two classes, positively rated books , which are books that received a rating from 6 to 10, and negatively rated books, which received a rating from 1 to 5. Those numerical ratings are used to assign weights to the training examples when we are building the model of the user's preferences. Furthermore we extend the Bayes classifier with Laplace estimates to avoid zero probabilistic results. Details of the theory supporting Bayes classifiers were provided in chapter 2 of this paper. After we have learned the user's profile we find a constant number of books or as noted "the top-k books" that correlate the most with his profile. Then, we can recommend this items to the user, using a proper illustration and description.

What differentiates LIBRA from other book recommender systems is the use of machine learning methods to provide input for the content-based system. In terms of performance. the computational complexity analysis of the system revealed that the complexity is linear to the input size, whereas experimental data was used to calculate an average throughput of 200 books per second in the process of creating the user's profile. Furthermore, an extensive statistical analysis illustrated in [4], provides strong evidence that the system achieves a high level of accuracy in its results, while presenting a satisfactory performance in terms of resources and computational time.

4.2.CBMRS

CBMRS (Content-Based Music Recommendation System) is a system that implements the innovative idea of using not only the available data of the user's preferences but also dynamic data from the environment, in order to make music-related recommendations. The approach involves taking into consideration parameters such as weather, the user's pulses and mood, temperature and the user's current location and create a model based on how these parameters affect the user's preferences in music selection. By combining the current content-based filtering algorithms used for music recommendation and these new parameters that are known to affect the user's choice of music this system attempts to provide more accurate recommendations than the conventional content-based algorithms.

In this subsection we are going to discuss how the system collects the dynamic data from the enviroment and then uses it in combination with the static data collected from the user's preferences. Furthermore, we are going to discuss the general process of deciding which music titles to recommend to the user.

First of all, we will analyze how the system collects the data necessary to make a recommendation. For the purpose of measuring the user's pulses the system collects data from a pulse sensor attached to the user's watch. The sensor's signal is transmitted in real-time conditions using the Zigbee communication standard [5]. Then the data undergoes a process of discretization, into one of the following states: 0 to 40 is classified as dangerous, 41 to 65 as low, 66 to 120 as normal, 121 to 180 as high and 181 and up as dangerous.

In order to draw information about atmospheric conditions, the system collects weather data from the Web regarding the general area pinpointed from the user's IP. Then, this data is classified in seven distinct states for this variable, which are: Clear, sunny, cloudy, shower, rain, snow and storm. Furthermore, the atmosphere's temperature is measured, in order to provide additional information about atmospheric conditions. In order to collect this data, the system employs a temperature sensor attached on the user's watch. Then,this information is classified into one of the four distinct states for this variable: The temperature is considered cold when the readings are from -4 Fahrenheit degrees to 30.2 Fahrenheit degrees ,cool when it is in the range of 32 Fahrenheit degrees to 68 Fahrenheit degrees, warm from 69.8 Fahrenheit degrees to 86 Fahrenheit degrees and hot when the temperature is from 87.8 Fahrenheit degrees and up.

To pinpoint the user's location the system uses an RFID (Radio Frequency Identification) Tag also attached to the user's watch. This variable has six distinct states: Balcony, Bathroom, Bedroom, Guestroom, Kitchen and Living Room.

The age of the user can be traced using the RFID tag previously mentioned. Five distinct states are used for this variable: 0 to 7:Infant, 8 to 11:Child, 12 to 17:Young Adult, 18 to 61:Adult and 62 and up:Old Adult. There also exists a variable for the sex of the user which has only two distinct values and is traced using the RFID Tag.

The system collects statistic data about the user's choices, referred to as static data in contrary to the dynamic data provided by the previously mentioned sensors, the reason being that it applies to any state of the environment. The recommendation algorithm must be executed while the dynamic data collected from the environment has not changed, in order to provide accurate recommendations. This soft real-time restriction allows the system to make an estimate of the type of music the user would prefer to listen while certain environment variables apply, matching the mood of the user with his music preferences. After the system has stored the dynamic data concerning the state of the user's environment, this information is combined with the user's profile, which uses statistical data to model the user's preferences in music genre and style. Then, it recommends songs that not only match his interest but also the conditions of his current environment. A constant number of songs are recommended, referred to as "the top-k results".

The system is based on a Java-Based OSGi framework. The system was developed based on an OSGi gateway using Knopflerfish 1.3.3 [6], an open architecture source project which implements a service framework. OSGi is an industrial standard which is used to connect several different internet devices such as household information devices and security systems. It is a JES-based Gateway software based on an open architecture Java embedded server which can provide high quality multimedia regardless of the application software platform. It is an open network architecture that can support various network physical mediums and protocols (USB, Bluetooth, PNA, HAVi, RF, VESA etc.). As we have mentioned previously, the system consists of many different devices interconnected to each other providing data to the recommendation system, therefore an open architecture standard like OSGi is needed to make the interconnection faster and easier.

The CBMRS system, from the viewpoint of recommender systems, implements a content-based recommendation algorithm. We will describe not only the algorithm but also the process which the systems follows and how the different modules of the system interoperate in this process. There are three distinct modules in the CBMRS: the Context Manager, the Service Manager and the Music Recommendation Manager.

The Context Manager transfers data generated by events that change the state of the system. The change of state is caused by a new measurement on the sensors collecting the context data, which is then processed by a context analyzer. The analyzer's output is then transfered to an OWL (Web Ontology Language) inference engine [7]. In this process, OWL's purpose is to discover and represent the semantics and relationships that characterizes the system's data. After the data has been processed by the OWL inference engine, the data is sent to the Service Manager where it is transformed into valuable information using an OWL inferencer, including an OWL ontology object DB. The Inference engine in the Context Manager uses a Jena 2.0 ontology inferencer.

The second part of the architecture is the Service Manager. It consists of a Bundle Service, a term used in OSGi to refer to a collection of services, that provides the following services: a recommendation service as a bundle in a SOAP(Simple Object Access Protocol) [8] service, an OSGi framework installed device used to transfer information received from the OWL inference engine to the recommendation system and an Application and Bundle Manager Service that supports the management of the mobility of bundles. Communication between the Context Manager, the Music Recommendation Manager and the Music Service is performed using the SOAP service to process in real-time the data needed to offer music recommendations, using context information from as many different systems as possible, without interruption by a different device that has an OSGi middleware even when a user is on moving to a different location.

The third part of this architecture, the Music Recommendation Manager plays the role of deciding the optimal music list in a recommendation module. This is achieved by combining the data received from the recommendation list, which corresponds to a certain context information received from the Service Manager, with the user profile and information stored in the MCIDB (Music Content Information DataBase). This data is then used provided as input to a filtering process in a recommendation module. The Query Manager selects a proper profile according to the user context information and transfers the related data to the Recommendation Module. It also performs an updating process for the music selected by the users in the User Profile, further improving the statistical profile created for the current user. Then, the Recommendation Module performs a token analysis of the profile received from the Query Manager and configures a recommendation list by searching music from the MCIDB. Having created our recommendations, we can present them to the user in the form of a song playlist.

Summarizing, CBMRS uses a real-time open architecture implemented using Java, in which several different devices are used in order to obtain information critical to the system's accuracy. This information is then combined with information about the user's interest and a music related database in order to make a recommendation. Although the system is comprised of a rather complex architecture and uses many different sources of data, it allows us to make music recommendations to the user not only dependent on his preferences, but also taking into consideration his current mood, as well as other data. Therefore, although CBMRS is a soft real-time system, with strict performance requirements, it provides a service to users which was unavailable up to now.

4.3.PRES

PRES(Personalized Recommender System) is a content-based recommendation system that creates dynamic hyperlinks for a web site that contains a collection of advises about "do-it-yourself" home improvement. The most important aspect of this type of information is the fact that a certain topic is only interesting to a user for a short period of time. Once an improvement has been carried out or enough information has been provided, the user will lose interest in that topic.As a result, PRES should train the user model in order to classify unseen items onto a positive class (relevant to the user) or a negative class (irrelevant to the user).

PRES makes recommendations by comparing a user profile with the content of each document in a given collection. The content of a document can be represented with a set of terms. Terms are extracted from documents by running through a number of parsing steps. First all HTML tags and stop words, meaning words that occur very often and cannot be used as discriminators, are removed. The remaining words are reduced to their stem by removing prefixes and suffixes. The user profile is represented with the same terms as documents and it is built by analyzing the content of documents that the user found interesting.

In terms of a recommendation algorithm, PRES employs the relevance feedback method in order to determine which document the user finds interesting. Documents and profiles are represented as vectors in the vector space model. In the vector space model a document D is represented as an m-dimensional vector, where each dimension corresponds to a distinct term and m is the number of all terms in the collection of documents. Term weights can be determined by using the tf-idf technique. Because PRES operates in an environment in which the topic interest of users changes constantly only one vector is used.This vector is initially empty and is adjusted as the user navigates through the web site. When a user has spent a certain amount of time reading a document D, this document is considered relevant, therefore the user profile P is updated with the following equation:

$$P'=aP+bD$$

Weight b determines the relative importance of a document to the user. Because PRES only determines whether or not a document is relevant, weight b is always set to 1. The user's constant change of interest is reflected by adjusting the variable a, a weight between 0 and 1 that reduces the term weights in the profile, therefore reducing their importance in the user profile. This weight is determined via experimentation.In PRES the similarity between the profile vector and a document is determined by using the cosine measurement.

Considering the system's implementation, PRES has been largely written in the object-oriented programming language Java. Although Java is often used to make applets that run on client machines,PRES does not use them but it runs entirely on the server-side by using Java servlets and Java server pages(JSP). A servlet is a Java class that expands the functionality of a server. PRES uses the servlet engine of JRun that functions as a plug-in to an existing server. This means that the servlets run independently from the web server. This architecture has the advantage that servlet engines are independent of the server architecture, therefore they can run on any server machine, without changing the servlet code.

The architecture of the PRES implementation is shown below, in figure 1. The database contains two different types of data: Information about users and information about web pages. The information about users is obtained online either from users themselves or by observing their behavior. The information about web pages is gathered offline, in the greater part by the parser component which is run every time new pages are added to the web site. It analyses the collection of web pages and extracts terms and calculates their tf-idf weights which are then stored in the database. The profiler and membership component both react to user requests. The membership component enables the user to become a member by creating an account. Information that the users provide such as a unique user name and a password is stored in the database. A user needs to be a member in order to receive personalized recommendations. The profiler component keeps track of the pages that members visit. It also carries out the relevance feedback method for every member. All this information is stored in the user profile which is initially empty for new members. The recommender component tries to find relevant web pages by comparing the user profile with the pages that were indexed. Recommendations are presented to the user by inserting hyperlinks into the next JSP file that the user requests.

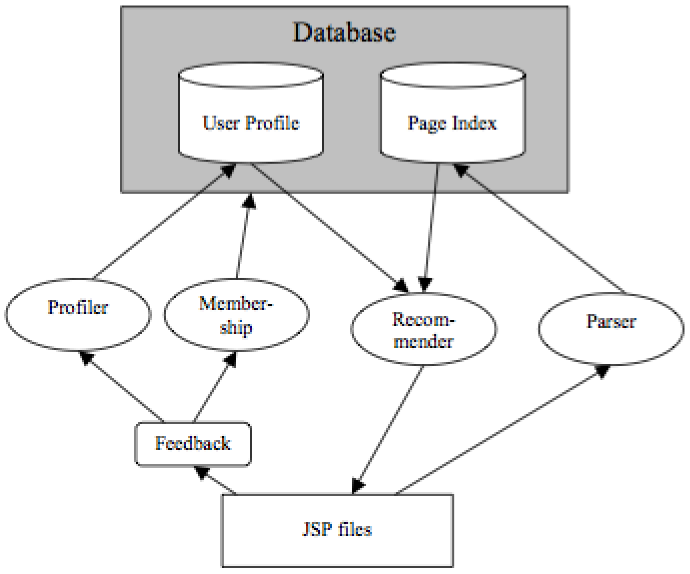


Figure 1 Architecture of the PRES implementation

In [3], the results of an experimental performance evaluation are reported. In this test, three fictitious users with different interests were created to evaluate PRES. the first one was interested in two topics about home improvement, the second one three topics and the third four topics. For each topic all the relevant documents in the collection were selected and classified as relevant if it contained information that was associated with the topic. The first user selected 80% of all the relevant documents about the two topics, the second user 50% and the third user 30%. The precision ratios differ significantly for different topics of interest. This is probably caused by the fact that documents about several topics contain many different terms which makes it difficult for the learning method to find a relation between these documents. Precision ratios also differ at different points in time for the same topics of interest. As more documents about a certain topic are selected it becomes easier for a learning method to make recommendations as it is provided with more similar terms. On the other hand, as more documents are selected the number of remaining documents that are relevant decreases ant it therefore also becomes more difficult for the learning method to find other relevant documents. This last factor will have effect in the case of the third user session as only 30%of the total amount of documents about a topic are selected. In this case, precision usually increases overtime. the exact effects however remain difficult to measure as other factors, such as a page change, also have an influence on these ratios.

4.4. Cobra

[this can be used anywhere else.It is an intro to blog and rss content-based filtering]

Blogs and RSS feeds are becoming increasingly popular. The blogging site LiveJournal has over 11 million user accounts, and over 1.6 million postings are made to blogs everyday. The "Blogosphere" is a new hotbed of Internet-based media that represents a shift from mostly static content to dynamic, continuously-updated discussions. The problem is that finding and tracking blogs with interesting content is an extremely cumbersome process. It is unclear that conventional Web search technology is well-suited to tracking and indexing such rapidly-changing content. Many users make use of RSS feeds, which is conjunction with an appropriate reader, allow users to receive rapid updates to sites of interest. However, existing RSS protocols require each client to periodically poll to receive new updates. In addition, a conventional RSS feed only covers an individual site, such as a blog. The current approach used by many users is to rely on RSS aggregators, such as SharpReader and FeedDemon, that collect stories from multiple sites along thematic lines.

It is obvious that the vision is to provide users with the ability to perform content-based filtering and aggregation across millions of Web feeds, obtaining a personalized feed containing only those articles that match the user's interests. Rather than requiring users to keep tabs on a multitude of interesting sites, a user would receive near-real-time updates on their personalized RSS feed when matching articles are posted.

[/end of intro]

Cobra(Content-based Filtering and Aggregation of Blogs and RSS Feeds) is a system that crawls,filters and aggregates vast numbers of RSS feeds, delivering to each user a personalized feed based on their interests. Cobra consists of a three-tiered network of crawlers that scan web feeds, filters that match crawled articles to user subscriptions and reflectors that provide recently-matching articles on each subscription as an RSS feed, which can be browsed using a standard RSS reader. Each of the three tiers of the Cobra network is distributed over multiple hosts in the Internet, allowing network and computational load to be balanced and permitting locality optimizations when placing services.

Cobra System Design

4.4.1 Crawler service

Crawlers are responsible for periodically crawling web feeds, such as blogs, new sites and other RSS feeds which are collectively call source feeds, and detecting new articles. A naive crawler would periodically download the contents of each source feed and push all articles contained therein to the filters. However, this approach can consume a considerable amount of bandwidth, both for downloading the source data and sending updates to the filters. In order to reduce bandwidth usage crawlers attempt to use the HTTP Last-Modified and ETag headers to check whether a feed has been updated since the last polling interval. Moreover, crawler makes use of HTTP delta encoding for those feeds that support it. If the HTTP headers indicate that the feed's content has changed, or if the server does not provide modification information,the crawler filters out articles that have been previously pushed to filters, reducing bandwidth requirements further and preventing users from seeing duplicate results. This can be done using two techniques. First, a while-document hash using Java's hashCode function is computed. If it matches the previous hash for this feed, the entire document is dropped. Second, each deed that has changed is broken up into its individual articles, which are henceforth processed individually. A hash is computed on each of the individual articles and those matching a previously-hashed article are filtered out.

4.4.2 Filter service

The filter service receives updated articles from crawlers and matches those articles against a set of subscriptions. Each subscription is a tuple consisting of a subscription ID, reflector ID and list of keywords. The subscription ID uniquely identifies the subscription and the reflector ID is the address of the corresponding reflector for that user. Subscription IDs are allocated by the reflectors when users inject subscriptions into the system. Each subscription has a list of keywords that must all be matched(in a case-insensitive fashion) in the article. One user may inject multiple concurrent subscriptions into the system.

Given a high volume of traffic from crawlers and a large number of users, it is essential that the filter be able to match articles against subscription efficiently. Cobra uses the matching algorithm proposed by Fabret et al.[]
This algorithm operates in two phases. In the first phase, the filter service uses an index to determine the set of all words (across all subscriptions) that are matched by any article. This has advantage that words that are mentioned in multiple subscriptions are only evaluated once. In the second phase, the filter determines the set of subscriptions in which all words have a match. This is accomplished by ordering subscriptions according to overlap and by ordering words within subscriptions according to selectivity, to test the most selective words first. If a word was not found in the first phase, all subscriptions that include that word can be discarded without further consideration. As a result, only a fraction of the subscriptions is considered if there is mach overlap between them.

4.4.3 Reflector service

The final component of the Cobra design is the reflector service, which receives matching articles from filters and reflects them as a personalized RSS feed for the corresponding user. The filter sends the complete article body to the reflector without a user list, and the reflector re-runs the matching algorithm against the list of active subscriptions it stores for the users it is serving. Since the matching algorithm is so efficient (taking 10ms for 1 million subscriptions), this appears to be the right tradeoff between bandwidth consumption and CPU overhead. For each subscription, Cobra caches the last k matching articles, providing a personalized feed which the user can access using a standard RSS reader. The value of k must be chosen to bound memory usage while providing enough content that a user is satisfied with the "hits" using infrequent polling. Typical RSS readers poll every 15-60 minutes.In Cobra's design, k is set to 10, a value that is typical for many popular RSS feeds.A user subscribes to Cobra by visiting a web site that allows the user to establish an account and submit subscription requests in the form of keywords. The web server coordinates with the reflectors and filters to instantiate a subscription by performing two actions:1 associating the user with a specific reflector node, and 2 injecting the subscription details into the reflector node and the filter nodes that feed data into that reflector. The response to the user's subscription request is a URL for a private RSS feed hosted by the chosen reflector node.

4.4.4 Implementation

Cobra's prototype is implemented in Java, and makes use of substrate for stream-based overlay networks (SBONs) for setting up and managing data flows between services. Cobra's implementation consists of 29178 lines of Java code in total. The crawler service is 2445 lines,the filter service 1258 lines and the reflector is 622 lines.

4.4.5 Performance

Cobra can scale well to handle a large number of source feeds and user subscriptions. Scalability is limited by service resource requirements (CPU and memory usage) as well as network bandwidth requirements. It can handle 10M users and 1M source feeds. Cobra offers low latencies for discovering matching articles and pushing those updates t users. The limiting factor for update latency is the rate at which source feeds can be crawled as well as the user's own polling interval. It is used a combination of real and synthesized web feeds to measure Cobra's performance. The real feeds consist of a list of 102,446 RSS feeds from syndic8.com, an RSS directory site. To scale up to larger numbers, an artificial feed generator was implemented. Each generated fed consists of 10 articles with words chosen randomly from a distribution of English words based on popularity rank from Brown corpus. Generated feed content changes dynamically with update intervals similar to those of real feeds. the feed generator is integrated into the crawler service and is enabled by a runtime flag.

As for crawlers performance, using last-modified checks for reading data from deeds reduces the inbound bandwidth by 43%. The combination of techniques for avoiding pushing updates to the filters results in a 99.8% reduction in the bandwidth generated by the crawlers, a total if 2.2KB/sec for 102.446 feeds. The use of locality-aware clustering should reduce the time crawl a set of source feeds, as well as reduce over all network load. From the initial set of 102.446 feeds, it was filtered out those that appeared to be down as well as feeds from two aggregator sites, topix.net and izynews.de, that together constituted 50.953 feeds. Those two sites host a large number of dynamically-generated feeds that exhibit a wide variation in crawl times, making it difficult to differentiate network effects.

As for filter performance, the matching algorithm is very fast, requiring less than 20 ms to match an article of 2000 words against 1 million user subscriptions. Of course, as the number of incoming articles increases, the overall matching time may become a performance bottleneck, although this process is readily distributed across multiple filters.

References

[1] **Content-Based Filtering for Music Recommendation Based on Ubiquitous Computing**
Jong-Hun Kim,Un-Gu Kang and Jung-Hyun Lee

Book chapter in Intelligent Information Processing III

[2] **Content-Based Recommendation Systems**, pp. 325-341
Michael J. Pazzani, Daniel Billsus
Lecture notes in Computer Science , "The Adaptive Web" by Peter Brusilovsky, Alfred Kobsa, Wolfgang Nejdi (Ed.)

http://www.springerlink.com/content/qq35wt68l6774261/

[3] **Using Content-Based Filtering for Recommendation**
Robin van Meteren and Maarten van Someren
http://www.ics.forth.gr/~potamias/mlnia/paper_6.pdf

[4] **Content-Based Book Recommending Using Learning for Text**

Categorization , Raymond J. Mooney and Loriene Roy, Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation, Berkeley, CA, August 1999
http://www.cs.utexas.edu/~ml/papers/libra-sigir-wkshp-99.pdf

[5] **The emergence of ZigBee in building automation and industrial control ,** Egan, D. , Computing & Control Engineering Journal , Published on April-May 2005, Volume 16, Issue 2

http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1454280

[6] **Knopflerfish OSGi Official Documentation**
http://www.knopflerfish.org/documentation.html

[7] **OWL Web Ontology Language Overview**
W3C Recommendation, 10 February 2004
Deborah L. McGuiness, Frank van Harmelen
http://ia.ucpel.tche.br/~lpalazzo/Aulas/TEWS/arq/OWL-Overview.pdf

[8] **Simple Object Access Protocol**
Gunnar Mein et al.
http://www.google.gr/patents?hl=el&lr=&vid=USPAT7146618&id=7Bt9AAAAE

BAJ&oi=fnd&printsec=abstract

[9] **Telling humans and computers apart automatically**
Luis von Ahn, Manuel Blum, John Langford,
Communications of the ACM, Volume 47, Issue 2, Pages 56-60
http://portal.acm.org/citation.cfm?id=966389.966390