Generating Rule-Based Trees from Decision Trees for Concept-based Information Retrieval

Ronnie Fanguy  
Nicholls State University  
is-raf@nicholls.edu

Vijay Raghavan  
Center for Advanced Computer Studies  
University of Louisiana at Lafayette

Abstract

Web-based information retrieval systems may result in poor levels of precision and recall when users are required to articulate their own queries. Concept-based information retrieval attempts to solve this problem by allowing users to select from concept definitions specified by experts. However, it is unrealistic to expect experts to define every concept which will be of interest to users. Therefore, we propose a system for generating concept definitions from decision trees.

1. Introduction

As the World Wide Web grows in importance as a medium for the exchange of information and ideas among various groups of individuals, the need for efficient and effective web-based information retrieval systems is heightened. One major challenge in this arena is retrieving relevant documents with high precision and high recall—with the ultimate goal being retrieval of all relevant documents (perfect recall) and only relevant documents (perfect precision). However, it is often difficult for users to articulate exactly the right query to achieve high levels of precision and recall. Therefore, retrieval requests often result in the user obtaining many irrelevant documents while missing documents that are relevant.

One attempt to solve this problem is found in the adaptation of concept-based information retrieval to web-based information retrieval systems. Most web-based search engines are based on the Boolean Retrieval Model, which requires a user to specify a query as some boolean combination of search terms [1, 2]. To ease the burden of query formulation and to deal with the disparity between a user’s vocabulary and the vocabulary used in a document collection, researchers have proved the effectiveness of replacing the boolean search engine’s interface with that of a concept-based information retrieval system [3, 4].

In concept-based information retrieval, a user issues a document retrieval request by selecting a concept, as opposed to specifying a boolean expression. The concept-based retrieval system uses the definition of the selected concept to generate one or more appropriate boolean search requests which are automatically submitted to the underlying boolean search engine. The documents retrieved from the underlying system are then merged and presented to the user. Such a system greatly relieves the user from the burden of specifying appropriate queries—as they simply select the concept of interest.

A key difficulty with concept-based retrieval is obtaining an adequate collection of concept definitions. What a user is allowed to retrieve is limited by the concepts that are defined. The number of concepts that are required to make the system usable may be quite large.

Another key difficulty exists when there is disagreement on the definition of a concept among users. If the system only allows a concept to be defined in one way, then the system will not be able to model these differences. Unless we move from a situation where concept definitions are presumed relevant for all users toward personalized concept definitions, precision and recall of our retrieval system will suffer. The interested reader is referred to [9] for an excellent discussion on the need for personalized information retrieval.

In this paper, we address these problems by presenting a system which generates personalized concept definitions from decision trees. We further explain concept-based information retrieval and rule-based concept trees in section 1.1. Then, in section 1.2, we discuss the structure and functioning of a decision tree classifier. In section 2, we present the overall framework of our system for generating rule-based concept trees and describe its components. Section 3 then presents our algorithm for converting a decision tree into a set of rule-based concept trees. Finally, sections 4 and 5 respectively evaluate the system and conclude the paper discussing future work.

1.1. Concept-Based Information Retrieval

The original concept-based information retrieval system was introduced in [5]—with their system called Rule Based Retrieval of Information by Computer
(RUBRIC). With RUBRIC, a concept is represented by a tree structure. The tree structure used in this system, called a rule-based tree, represents a concept as a boolean (AND/OR) combination of subconcepts and index terms. In this structure, subconcepts are included as a tree’s intermediate nodes and are associated with either the boolean operator AND or the boolean operator OR. The terminal (leaf) nodes of a tree are each associated with an index term. Consider the rule-based concept definition of human health science shown in Figure 1. (This example is borrowed from [3]). Notice that human health science is defined as a conjunction of the subconcepts human and health science. Furthermore, these two subconcepts are further defined. The definitions of subconcepts continue until leaf nodes are reached. At this point, the definitions are based on the index terms from the document collection.

In RUBRIC, rule-based trees are used to retrieve documents by applying bottom-up processing for each document in the document collection. Initially, each of a tree’s leaf nodes are assigned a value to indicate whether or not they are satisfied based on the contents of a document being considered for retrieval. A leaf node is satisfied if the document contains the leaf node’s associated term. Otherwise, the leaf node is not satisfied. From the leaf nodes, values are propagated up the tree towards the root node to determine whether or not the concept is satisfied. Based on the value propagated, the system decides whether or not the document is retrieved (whether or not the document matches the root concept).

When propagating a value to an OR intermediate node, if any of its children are satisfied, then the OR node is satisfied. However, for an AND intermediate node, all of the children must be satisfied in order for the node to be satisfied. When the root node is satisfied, the concept associated with the tree is considered to be satisfied. Thus, documents for which the root node is satisfied are retrieved for the user.

It can be very expensive to retrieve documents in this way. Alsaffar et.al. [6] recognize this shortcoming and propose preprocessing the rule base to speed up retrieval requests. In their solution, an AND/OR tree is converted into a group of Minimal Term Set (MTS) expressions. A MTS expression is simply a minimal set of terms that may be used to represent the root concept. That is, if all of its terms are found within a document, then the document satisfies the root concept and should be retrieved. Any one of the MTS expressions in the group may be satisfied in order for the root concept to be satisfied. Only when none of the MTS expressions are satisfied do we say that the document does not match the root concept.

Figure 1 shows the group of MTS expressions for the human health science concept. Notice that {man, genes} is one MTS expression. Therefore, if the terms “man” and “genes” both appear in a document, then we know that the root concept human health science is satisfied. Reexamine the rule-based tree defining the human health science concept, and notice that the term “man” satisfies the subconcept human and the term “genes” satisfies the subconcept health science. Together these two subconcepts satisfy the root concept human health science. Satisfaction of any one of the MTS expressions listed will similarly result in satisfaction of the root concept human health science.

![Rule-Based Concept Tree](image_url)

**Figure 1. Definition of concept health_science (human).**
The usability of a concept-based information retrieval system is limited by its ability to collect concept definitions. Such systems will only be successful if the concepts that users are interested in retrieving are represented in the system’s rule base. Therefore, a large number of rule-based trees is typically required of the system. Unfortunately, each of these rule-based concept definitions must be composed by experts who are familiar with the document collection and the index terms used therein and also with the mechanics of constructing rule-based trees. Expecting experts to define rule-based trees for every concept in which users may be interested is obviously unreasonable as the cost in terms of time and money is extremely prohibitive.

Kim, et. al. [7] recognize this problem and propose a solution for automatically generating concept definitions in the form of rule-based trees. However, their approach requires the existence of a machine-readable thesaurus. This may be appropriate for some domains; however, it will not be applicable in many others. Unless these limitations are overcome in a more general sense, concept-based information retrieval systems will by and large be limited to the academic and theoretical realm.

In this paper, we propose a solution to overcome these limitations by automatically constructing concept definitions from decision trees. Thus, when using our approach, the requirement may be translated into the ability to build a decision tree. Algorithms for the efficient induction of decision trees have been studied and are well understood. Such algorithms require as input a set of pre-classified training data. If we are able to gather a representative set of documents and elicit concept labels for each document in the set, then we can apply these algorithms. The result of which may be processed by the algorithms presented in this paper to generate a concept definition for the concept(s) of interest. Subsequently, the concept definition may be used by the system to retrieve documents which are relevant to the concept.

1.2. Decision Tree Classifier

A decision tree is a classification mechanism whereby an object is subjected to a series of tests. Each possible sequence of test outcomes yields an appropriate class label to associate with the object being classified. A decision tree is structured as a tree whose branch (non-terminal) nodes are tests and whose leaf (terminal) nodes represent class labels. Each branch node in a decision tree has a number of child nodes—one for each possible test outcome. For example, if a branch node is associated with the test Height > 5, then the possible test outcomes are {true, false}. This branch node will, therefore, have two children—one connected via an arc associated with the true outcome and one associated with the false outcome.

When classifying an object with a decision tree, we begin at the root node of the decision tree and apply its associated test. Depending on the result of the test, the appropriate arc corresponding to this result is traversed to the appropriate child node in the tree. If the subsequent node is a branch node, then its associated test is applied and the appropriate arc is again traversed. This process continues until a leaf node is reached. At this time, the concept label associated with the leaf node is returned as the appropriate concept for the object being classified.

We are interested in examining the features that cause an object to be assigned to a particular concept or category. The decision tree certainly contains information appropriate for distinguishing between sets of concepts. If we are able to extract this information from the decision tree, then we will be a step closer to creating concept definitions.

2. System Framework

Our system for generating rule-based trees for conceptual retrieval is shown in Figure 2. In this system, the set of rule-based trees is derived from decision trees.
Decision tree generators require a training set of data as input. Therefore, before we can generate decision trees, we must prepare a training set. In the current context, a training set is a set of labeled attribute vectors describing a subset of documents from the document collection. The attribute vectors describing each document include the index terms associated with the document collection and perhaps other document data that are available. The labels associated with each attribute vector are obtained from the user. The user will select an appropriate label to indicate which concept applies from a set of concept labels. If there is only a single concept of interest—the label will indicate whether or not each document is relevant with respect to that concept.

Once the training set is prepared, it will be used as input to the system component responsible for generating a decision tree. See5, the Windows version of the popular decision tree generator c4.5 [8], is the program we use to perform this step. By identifying the distinguishing features of documents in the training set, See5 constructs a decision tree which may be used to assign an appropriate concept label to a document.

### 3. Transforming Decision Trees into Concept Trees

We propose an algorithm which converts a decision tree into a number of concept definitions—one for each concept label used within the decision tree. The structure we use to store these concept definitions is RUBRIC's rule-base tree.

The rule-base tree structure may be used to represent the concepts stored within a decision tree. The path from the root node to a leaf node in a decision tree represents a series of tests that must all be passed in order for the concept associated with the leaf to be satisfied. That is, the conjunction of each of these tests (along with their appropriate results) gives us one way to satisfy the concept—the very essence of the AND operator. However, there may be many such paths through the tree for a single concept. Any one of these paths will yield satisfaction of the concept—the very essence of the OR operator. Hence, with an appropriate conversion mechanism, the distinguishing concept characteristics stored within a decision tree may be restructured using the AND and OR nodes of a rule-based tree.

However, the restructured version of the decision tree must support the same types of tests that are used within the decision tree. RUBRIC’s rule-based concept trees have leaf nodes that are associated with terms. The condition associated with the leaf nodes may be interpreted as “the associated term is present in the document.” However, this type of test is not flexible enough to support decision tree tests. Therefore, some modification must be made to either the rule-based concept tree structure or the information stored for each document.

By expanding the concept tree structure to allow more flexible types of conditions, we can represent the tests that are used within a decision tree. Instead of storing just a term, each leaf node may be extended to support conditions of the form `attribute op value`, where

- **attribute** is some property of set of objects being considered,
- **op** is an equality (==, !=) or relational (<, <=, >, >=) operator,
- **value** is some literal.

This extension will enable support for the types of tests commonly used within decision trees. Alternatively, the result of the test conditions could be

---

**Example Decision Tree**

- **T1**
  - True
  - False
- **T2**
  - True
  - False
- **T3**
  - True
  - False
- **T4**
  - True
  - False

**Simplified Decision Tree for Concept +**

- **T1**
  - True
  - False
- **T2**
  - True
  - False
- **T3**
  - True
  - False
- **T4**
  - True
  - False

**Simplified Decision Tree for Concept -**

- **T1**
  - True
  - False
- **T2**
  - True
  - False
- **T3**
  - True
  - False
- **T4**
  - True
  - False

---

**Figure 3. Simplifying decision trees.**
stored as part of the document indexing structure. Thereby, the concept tree’s leaf node may store the test condition as if it were a term. With this change, determining whether “the associated term is present in the document” will appropriately correspond to evaluating the decision tree test result. In this way, decision tree tests may be modeled while requiring no change to the existing rule-based tree structure.

We split the task of transforming decision trees into concept trees into two main parts—simplifying decision trees and inverting the simplified decision trees. Given a particular concept that is represented within a decision tree, we can perform these two steps to construct a rule-based tree that represents the concept.

3.1. Simplifying Decision Trees

The first step necessary to construct an AND/OR concept tree from a decision tree is to simplify the decision tree. In simplifying the decision tree, we consider only the paths that lead to a given concept label. The decision tree is simplified by removing any components (subtrees) that are not associated with the concept being defined—so that only those paths through the tree that lead to the concept being defined remain. An example is shown in Figure 3. Here, we see how a decision tree, which classifies objects as either + or −, may be simplified to decision trees containing only relevant information for each of these two concepts. In this example, each non-leaf test node results in either true or false. Notice that in the simplified tree for concept + only the components of the tree that lead to + leaf nodes remain in the tree. Similarly, only the components of the tree that lead to − leaf nodes remain in the simplified tree for the concept −. By performing this step, we are able to focus solely on the portion of the tree that is relevant to the concept being defined.

3.2. Inverting Simplified Decision Trees

After the decision tree is simplified so as to focus on only those paths that lead to the concept being defined, we proceed to invert the simplified decision tree. This process is demonstrated by the recursive function shown in Figure 4. This function takes two parameters as input:
- D—the current node within the simplified decision tree and
- ψ—the AND/OR concept tree being constructed.

Initially, D should refer to the root of the decision tree, and ψ should be an empty AND/OR concept tree. The function proceeds to process a single node of the decision tree at a time. It begins at the root node and relies on recursive calls to process the entire tree. As the function traverses the simplified decision tree, it constructs an appropriate AND/OR concept tree—which it returns as output.

As this algorithm processes each node in the simplified decision tree, it first considers the number of children at the current decision tree node (D). If there is more than one child, then there is more than one path through this node which will lead to the concept being defined. In this case, any one of the paths may be taken. Therefore, an OR node is needed in the concept tree.

Next, we proceed to iterate through each of the child nodes of the current node in the decision tree. In processing the children, we consider whether or not the child is a leaf node. If it is a leaf node, then we simply add a node to the concept tree which represents the test result that leads from the current node in the decision tree to the child leaf node. However, if the child is not a leaf node, then there are other nodes which must be processed along the current path through the decision tree. In this case, we must add an AND node because we must satisfy this test along with the other subsequent test nodes in the current path. Along with this AND node, we add a child that is responsible for handling the test that corresponds to the arc which leads from the current node to the non-leaf child within the decision tree. Other children of the AND node are added as the non-leaf child nodes of D nodes are recursively processed.

Notice also that the algorithm gives considerable attention to maintaining the current parent within the concept tree being constructed. This attention is required to ensure that when the processing of each child node is complete, the appropriate AND/OR concept tree is returned as output.

ψ —AND/OR concept tree being constructed
η —Current parent node in Concept Tree
D —Current node in the decision tree
C —Child node in decision tree

function ψ = ConstructConcept(D, ψ)
η = ψ.CurrentNode
If (D.NumChildren > 1)
η = ψ.AddORNode()
For each C ∈ D.Children
If η != ψ.CurrentNode
ψ.MakeCurrent(η)
If (!IsLeafNode(C)) AND (IsAndNode(ψ.CurrentNode))
η = ψ.AddANDNode()
ψ.MakeCurrent(η)
ψ.AddLEAFNode(D.Test, C.ParentTestResult)
If (!IsLeafNode(C))
ψ = ConstructConcept(C, ψ)
return

Figure 4. Algorithm to invert a simplified decision tree.
complete (after arbitrarily many recursive calls) we return to the correct positioning within the concept tree being constructed to continue appending other child nodes—if necessary. Figure 5 shows the result of applying this algorithm to the simplified decision tree for the concepts + and -.

4. System Evaluation

There are three major advantages of this approach over existing approaches to constructing concept definitions in the form of rule-based trees. The first and most obvious advantage is that a concept may be constructed in the absence of experts and thesauri. Any user who is willing to supply class labels is able to construct concept definitions.

Second, using the characteristics identified by the decision tree generator as key to distinguishing one concept from others has particularly valuable benefits to an information retrieval system. Rather than simply getting a definition which serves to characterize a concept, we obtain a rule-based tree which is focused on the characteristics that distinguish a concept from others within the document collection. Therefore, the resulting concept definition is composed of the features that allow a user to precisely retrieve the documents that are relevant to the concept.

When analyzing this advantage, the reader may be prone to question whether there is any advantage of the rule-based tree representation over the decision tree representation. Recall that a decision tree is a classifier which distinguishes between a number of different classes. A rule-based tree is different in that it defines a single concept. However, the rule-based tree representation is not just a classifier. It is a hierarchy which defines the root concept in terms of subconcepts. This is an important distinction because we are not only defining the concept associated with the root node of the tree, but we are also identifying some structure within that concept by the hierarchy of subconcepts. We do not make any claims regarding the quality of this structure. Presently, we are able to generate some structure with our
algorithm whose complexity is a function of the number of nodes in the decision tree—O(n). We are currently examining other more computationally expensive alternatives that generate trees which have greater structural quality.

Third, the concept definitions may be personalized to a much higher degree than existing approaches allow. Rather than offering only an expert’s definition of a concept or relying on the word relationships stored in a thesaurus, our approach focuses on defining concepts that are tailored to a particular user’s definition of the concept—based on their personal labeling of the training data. When the definition is not agreed upon, this characteristic is a major benefit.

However, these benefits do not come without a cost. The benefits are gained at the cost of greater user involvement in the construction of concept definitions. Instead of simply being given a concept definition, the user bears the burden of assigning concept labels to documents. However, this cost may be reduced in several ways. A user’s personal concept definition may evolve during several iterations. Initially, a rough concept definition may be formed by evaluating a small subset of documents. Documents that are subsequently retrieved may also be evaluated to make the definition more precise. Over several iterations, the concept definition will more precisely meet the user’s needs. Another mechanism which may be employed to ease the user’s burden is to allow groups of users who share similar concept definitions to share labeled examples. In this way, the burden of labeling examples is shared among the group.

5. Conclusion/Future Work

In this paper, we have presented an approach to generating concept definitions from decision trees. Generating concept definitions without the help of an expert greatly improves the usability of a concept-based information retrieval system. Such systems consequently increase the likelihood of boolean search engine effectiveness, as the system does not rely on user-specified queries. Also, the user is likely to be pleased to be relieved of the burden of query articulation. Our system replaces this burden with the requirement of assigning concept labels to documents. We believe that the user will find this burden far lighter than the frustration which may develop from trial and error query development.

Future work on this research project involves carrying out experiments which quantitatively evaluate the quality of our system relative to the existing approaches, its implications for personalization, and its usefulness in aiding human understandability of concepts. Furthermore, rule-based trees may also be extended to include weighted arcs. We plan to extend our conversion algorithm to calculate these weights. Additionally, we are examining alternative means of automatically generating rule-based concept trees.

Additionally, we are exploring the potential of this approach to expand the usefulness of browsing directories—such as those found in Yahoo, Google, Altavista, etc. These directories enable users to find relevant web pages by allowing them to start at some high level category and traverse down to more and more specific categories. These directories can be thought of as trees which are defined by OR nodes, where each child is a specific subcategory of the general parent node—similar to an is-a relationship (generalization). By applying rule-based trees, we can model not only OR nodes but also AND nodes. This allows us to model aggregation (the part-of relationship) as well as generalization. Determining the usefulness of this extension will require additional research.

6. References