Web Categorisation Using Distance-Based Decision Trees

V. Estruch\textsuperscript{2} C. Ferri\textsuperscript{2} J. Hernández-Orallo\textsuperscript{2} M.J. Ramírez-Quintana\textsuperscript{2}

DSIC, Universidad Politécnica de Valencia, Camino de Vera s/n, Apdo. 22012, 46071 Valencia, Spain.

Abstract

In Web classification, web pages are assigned to pre-defined categories mainly according to their content (content mining). However, the structure of the web site might provide extra information about their category (structure mining). Traditionally, both approaches have been applied separately, or are dealt with techniques that do not generate a model, such as Bayesian techniques. Unfortunately, in some classification contexts, a comprehensible model becomes crucial. Thus, it would be interesting to apply rule-based techniques (rule learning, decision tree learning) for the web categorisation task. In this paper we outline how our general-purpose learning algorithm, the so called distance based decision tree learning algorithm (DBDT), could be used in web categorisation scenarios. This algorithm differs from traditional ones in the sense that the splitting criterion is defined by means of metric conditions (“is nearer than”). This change allows decision trees to handle structured attributes (lists, graphs, sets, etc.) along with the well-known nominal and numerical attributes. Generally speaking, these structured attributes will be employed to represent the content and the structure of the web-site.

Keywords: web mining, classification, structured data, decision trees, distance-based methods.

1 Introduction

Etzioni \cite{4} defined Web mining as the use of data mining techniques for extract information from Web documents and services. Given the large amount of documents available in the Web, one of the most common task performed on the Web is the classification of documents into one or more categories. For

\textsuperscript{1} This work has been partially supported by ICT for EU-India Cross Cultural Dissemination Project ALA/95/23/2003/077-054 and Generalitat Valenciana under grant GV04B/477 and CICYT under grant TIN 2004-7943-C04-02.

\textsuperscript{2} Email: \{vestruch, cferri, jorallo, mramirez\}@dsic.upv.es
instance, this is essential in applications that have to catalog news articles, sort and filter electronic mail, recommend films or music or search information about a topic (search engines). Although some authors distinguish classification from categorisation, for the sake of simplicity, in this paper we use both of them as synonyms since a categorisation problem can be solved by several classifiers. The simplest approach to the categorisation of Web documents is to take only the textual part of them into account (Text categorisation). The basic idea is to classify a document as of class \( c \) if certain words relevant to the \( c \) definition are present in the document.

However, Web documents are more than just plain text and the information contained in other parts like the hyper-links can also be relevant to the categorisation process. For instance, if we are classifying sports news, a more accurate classification can be obtained if our classifier considers that a piece of sports news contains words like team, play or stadium, or contains links to other sports news. Therefore, recent research solves this problem by merging ideas from Web content mining and Web structure mining. For instance, [7] appends the text of the links to the text of the target page. [1] considers the text of a Web page along with the text and the category of its neighbouring pages. Some other approaches are able to handle both the text components in the pages and the links among them, such as [2], [5], or [6].

In this paper, we study how the DBDT approach fits to the web classification problem. This method allows us to integrate both the Web content and the Web structure mining in a unique framework by using structured data types for representing each component or context feature (title, keywords, text, links, ...) found in the pages. This evidence is then used by the DBDT in that the splitting criterion is defined by means of metric conditions (“is nearer than”) and handle structured attributes. We illustrate that the method is suitable for this kind of application by applying it to a simple example of Web classification and we briefly discuss about how the metric conditions can be expressed in an equivalent but more comprehensible form.

The paper is organised as follows. In Section 2 the DBDT algorithm is outlined. An illustrative example of our approach is shown in Section 3. Finally, Section 4 presents some conclusions.

2 Distance Based Decision Trees

In [3] we defined a learning method named Distance Based Decision Trees. This proposal is based on the use of prototypes and distances to define the

---

3 The classification is the process of inducing a model in that only one class is assigned to each document, whereas categorisation concerns with the situation in that a document can belong to more than one class.
partitions for the decision tree. Our decision tree inference strategy is a modification of the centre splitting method [8] consisting in to compute a set of attribute prototypes unlike the other one which takes all the attributes into account. Basically, for each attribute and for each class, a prototype (that value which minimises the sum of all the distances from it to the others) is calculated, considering only the values belonging to that attribute and that class. Once this process is finished, an attribute is chosen in order to split the data set. The split proceeds by associating every instance to its closest attribute prototype. The splitting attribute is selected according to some of the well-known heuristic functions (gain ratio, GINI index, etc). For this purpose, a metric space is associated to every attribute. Note that the fact of handling all the attributes as whole entity, just as centre splitting does, turns the comprehensible model extraction into a harder task, even if the involved attributes are nominal or numerical.

The result of this adaptation of centre splitting is not very different from classical decision trees (see the algorithm below), when attributes are either nominal and numeric, but in our case, we are able to deal with data containing structured attributes such as sets, lists, or trees.

**PROCEDURE** DBDT(S, m); // Single Attribute Centre Splitting. Learns a decision tree based on attribute distances

**INPUT**: A training set S as a set of examples of the form: (x₁,...,xₙ), n ≥ 1 where every attribute is nominal, numerical or structured. A metric space is associated to every attribute. m is the maximum # of children per node.

**BEGIN**

C ← {Class(e) : e ∈ S} // C is the set of existing classes

If |C| < 2 Then RETURN End If

For each attribute xⱼ:

If Values(xⱼ,S) < 2 Then CONTINUE End If //next iteration

ProtList ← ComputePrototypes(xⱼ,S,m,C). 

If Size(ProtList) ≤ 1 Then RETURN End If

Splitⱼ ← ∅ // Set of possible splits for attribute xⱼ

For i ← 1 to length(ProtList) // for all the prototypes

Śᵢ ← {e ∈ S : i = Attracts(e, ProtList, xⱼ)} // Śᵢ contains the examples attracted by prototype i

Splitⱼ = Splitⱼ ∪ Śᵢ // We add a new child to this split

i ← i + 1;

End For

BestSplit = Argmaxⱼ∈Splitⱼ(Op-timality(Splitⱼ)) // GainRatio, MDL, ...

For each set Šⱼ in BestSplit

DBDT(Šⱼ, m) // go on with each child

End For
The auxiliary functions \texttt{Attracts} and \texttt{Compute Prototypes} are inherent to the method. In a nutshell, the function \texttt{Attracts} just determines which prototype is assigned with a new example and, the function \texttt{Compute Prototypes} obtains a set of prototypes for each attribute.

3 An illustrative example

The previous step, before running the DBDT algorithm, consists of deciding what sort of data types are going to be used, as well as their associated metric functions. Let us consider the following example. A user is interested in seeking sports news from the Internet using a search engine. This search engine must “decide” automatically which available documents fit the search parameters. Thus, this task can be addressed as a two class classification problem. The information, extracted from an HTML document for this purpose, can be grouped in these three categories:

- **Structure**: it refers how the pages from a web site are connected one each others by means of hyper-links. Formally, it is represented as a graph. However, we will use a simpler approach but it is in its turn a very common proposal in the graph mining literature: we represent a graph as a set of ordered pairs where each pair encodes two linked pages. Concretely, each item in the ordered pair will store a set of key words. Also, for the sake of brevity, we use the well-known symmetric difference between sets as a metric function.

- **Content**: It deals with the information contained in a web page. Traditionally, this information is represented as a bag or a vector of words. In our example, we only consider one attribute, a set, reflecting the whole content \((\text{Content})\), and we use an adaptation of the symmetric difference between sets as a metric function.

- **Web use**: we mean by web use information the information derived from the HTTP connection to a web server. All these data can be encoded by means of nominal or numerical attributes. For these types we can use the discrete metric or the absolute value difference, respectively. In our example, this attribute is referred by \(\text{Connections}\) and it contains the number of daily connections.

The next step is to infer a classifier by training a model from a processed dataset that contains collected information from some web pages, such as that included in Table 1.

The set \(\{(\text{Olympics, games}],[\text{swim}]),([\text{swim}],[\text{win}]),([\text{win}],[\text{medal}])\}\) in the \texttt{Structure} attribute is interpreted in the following way. The first component of the list stands for words “Olympics” and “games” appear as keywords in a web page.
### Table 1
Information from a web server sample repository.

<table>
<thead>
<tr>
<th>Id.</th>
<th>Structure</th>
<th>Content</th>
<th>Conn.</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{((Olympics, games), [swim]), ([win], [win])}, ((Olympics, games), [boxing]), ([win], [medal])}</td>
<td>((Olympics, 30), (held, 10)) (summer, 40)</td>
<td>10</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>{((Olympics, games), [swim]), ([win], [win]), ([win], [medal])}</td>
<td>((Olympics, 15), (summer, 20)) (Athens, 40)</td>
<td>20</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>{([football], [Europe]), ([Europe], [final])}, ([final], [best, player])}</td>
<td>(football, 20), (champion, 10)</td>
<td>40</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>{([football], [match]), ([match], [team, players]), ([football], [referees]), ([match], [results])}</td>
<td>(football, 20), (Europe, 10), (champion, 12)</td>
<td>40</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>{([football], [match]), ([match], [team, players]), ([match], [scores])}</td>
<td>(football, 20), (Europe, 10)</td>
<td>40</td>
<td>Yes</td>
</tr>
</tbody>
</table>

If we apply the DBDT algorithm (using an accuracy-based heuristic), we find that the first attribute to be selected, as the first split, is *Connection*, being the values 40 (*Conn* value for the 4th instance) and 10 (*Conn* value for the 1st instance) the prototypes for the class “yes” and “no” respectively. Iterating the process, attributes *Structure* and *Content* are used to split the left and the right first level nodes, respectively. Finally, the new obtained nodes are pure and the process stops, getting the distance based decision tree (see figure below\(^4\)).

![Decision tree](image-url)

**Fig. 1.** a) Decision tree after the first split. b) Decision tree after finishing the process.

### Table 2
Information from a web server sample repository.

<table>
<thead>
<tr>
<th>Id.</th>
<th>Structure</th>
<th>Content</th>
<th>Conn.</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>{([football], [match]), ([match], [players]), ([match], [results])}</td>
<td>(football, 30), (held, 10) (Europe, 7)</td>
<td>36</td>
<td>No</td>
</tr>
</tbody>
</table>

Imagine now that a web site described as in Table 2 is stored in the list along with other web sites which are candidates to be shown to the customer. Before listing them directly we should classify the web site repository in order to

---

\(^4\) The numbers correspond to instance id, and the bold numbers stand for the prototype of each class for a particular partition.
filter not suitable information. First, we look inside the connection attribute. As the number of daily connections is closer to 40 than 10, the instance is hooked up to the left first-level node. Then, we repeat the same process for the structure attribute, in this case, the structure of this new web site is more similar to the structure of the fourth instance in the table than the third one. Then, this instance would be classified as sport news site, and, consequently, listed to the user.

Currently, we are thinking over how the metric conditions could be expressed into terms of patterns associated to the metric function (for instance, belong to could be a pattern for sets) [9], and obtain a transformed (and more comprehensible) model containing rules as this one: IF the word “football” appears in Content and the connections \{([football],[match]), ([match],[team,players])\} are found in Structure THEN this web-site is a sport media web-site.

4 Conclusions

In this paper, we have studied the feasibility of DBDT proposal to tackle web categorisation problems. DBDT has been developed in Java (www.dsic.upv.es/users/elp/soft/) and tested for both, structured and non structured, well-known classification problems, showing a really interesting performance. For this reason, we consider that this algorithm could be applied for more concrete scenarios, such as categorisation web.

References


