Models for Inexact Reasoning

Academic Year: 2009/2010

Instructor: Miguel García Remesal

Practical Assignment #1: Building an expert system for document classification using uncertain knowledge

Introduction

In this assignment we will review the theoretical methods studied this semester to reason with uncertain knowledge. We will focus in applying such methods to a real world problem: the automated classification of textual documents.

Using a document collection, the aim of this practical is creating a knowledge base that together with a suitable reasoning mechanism, they can be used to automatically decide whether a document belongs to a given category.

In this practical, we will use a corpus composed of web pages—i.e. HTML documents—collected from several computer science departments of different universities of the U.S. in January 1997. This is an adapted version of the larger corpus that can be found at http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/. The adapted version of the corpus is composed of 80 web pages: 40 belonging to members of the faculty (professors) and the remaining 40 pages belonging to members of the staff (post-doctoral researchers, system administrators, secretaries, operators, etc.).

The aim of this practical exercise is building an expert system to automatically decide whether an unseen web page belongs to a member of the faculty or conversely, to a member of the staff. To complete this assignment, students must carry out the following tasks:

1. Using a training set of 40 web pages labeled with their corresponding class—i.e. either *faculty* or *staff*—students must create a rule base that given different pieces of evidence—e.g. the occurrence of a term, the co-occurrence of two or more words in a sentence, the instantiation of given textual pattern, etc.—supports (or discards) to some degree the hypothesis of the document belonging to any of the classes.
2. We will use the MyCIN approach to build the inference engine. Note that the rule base created at the previous step must be specifically designed to

properly work with the MyCIN inference method (check your lecture notes and slides for further details).

3. Use the test set of 40 previously unseen labeled web pages to evaluate the performance of the developed classifier and generate a report describing and discussing the results.

In the next sections we further elaborate on the details regarding the different tasks.

Building the knowledge base

To create the knowledge base, students must use the training examples that can be found in the file *dataset.rar*. This file can be downloaded from the course's webpage. Once uncompressed, the RAR file creates two directories: *faculty* and *staff*. The former contains both the training examples and the test set for web pages belonging to members of the faculty, while the latter folder contains the training examples and the test set for members of the staff. Training examples can be found under the relative path */faculty/training* (20 documents for faculty members) and */staff/training* (20 documents for staff members).

Using the provided examples, students must generate a set of rules to support/discard the hypotheses *class (d) = faculty* and *class (d) = staff*. The former stands for the hypothesis "document *d* belongs to class *faculty*" and the latter represents the hypothesis "document *d* belongs to class *staff*".

Rules can be generated either by: (1) manually inspecting the documents and detecting the occurrence of relevant words, co-occurrence of words or detection of syntactic patterns or (2) using a machine learning method for rule induction (ID3, C4.5, ID3, etc.). In any case, the construction of the knowledge base has to be properly described and documented.

To illustrate the required knowledge base, let us suppose that we have created a set of rules to decide whether a document is about the baseball *World Series*. As shown in the sample rule base shown in Figure 1, the first rule asserts that the document is about the topic *World Series* with certainty factor 1 if it is about the topics *team* or *event*. Similarly, rule 2 suggests that the document is about the topic *team* with CF 1 if it is about the topics *St._Louis_Cardinals* or *Milwakee_Brewers*. On the other hand, we will consider that a document is about the topic *St._Louis_Cardinals* with CF 0.7 (rule 3) if the string *"Cardinals"* appears in the text. Note that quoted (terminal) symbols stand for strings that must occur in the text, while non-quoted (non terminal) symbols represent topics. The remaining rules are interpreted in a similar manner.

Note that this naïve knowledge base only considers the occurrence of words in the document to determine whether a document is about a topic. However, in this

assignment, students are encouraged to use additional features such as term frequencies, word co-occurrence in the same sentence, and others at the student's discretion.

Students must take into account that they need to include rules to identify documents referring both to members of the faculty and members of the staff. Besides, the generated knowledge base must be included and properly documented (in first order logic) in the assignment's report. Certainty factors must be assigned by students in a subjective manner, i.e. based on the study of the learning examples, rather than using any statistics. Note that it is required to include into the assignment's report the rationale for the certainty factor assigned to each rule.

```
team | event => World_Series

St._Louis_Cardinals | Milwaukee_Brewers => team

"Cardinals" => St._Louis_Cardinals (0.7)
Cardinals_full_name => St._Louis_Cardinals (0.9)

saint & "Louis" & "Cardinals"
    => Cardinals_full_name

"St." => saint (0.9)
"Saint" => saint

"Brewers" => Milwaukee_Brewers (0.5)
"Milwaukee Brewers" => Milwaukee_Brewers (0.9)

"World Series" => event
baseball_championship => event (0.9)

baseball & championship => baseball_championship

"ball" => baseball (0.5)
"baseball" => baseball

"championship" => championship (0.7)
```

Figure 1 Rule base for the World Series example

Implementing the inference engine

As stated previously, students will use the MyCIN approach to build the knowledge base. Therefore it is necessary to implement a MyCIN inference engine that given (1) a target hypothesis, (2) the knowledge base and (3) the evidence (a document), automatically calculates the certainty value associated to the target hypothesis.

To illustrate this scenario, let us suppose that we are given the hypothesis *class(d) = World Series*, the rule base shown in Figure 1 and a document *d*. Given the hypothesis, the implemented inference engine would automatically generate the inference tree shown in Figure 2. As explained in the lectures, the inference tree is built using backward chaining starting from the main goal: i.e. the target hypothesis.

Once the inference tree has been created, it is necessary to assign CF values to the leaf nodes based on the evidence (i.e. the text of the document to be classified). For this concrete document, we would assign values to the leaf nodes as shown in Figure 3. As can be seen, the leaf nodes “Cardinals”, “St.”, “Saint”, “Louis”, “Brewers”, “Milwaukee Brewers” and “World Series” are assigned a CF of -1, since their corresponding terms do not appear in the document. Conversely, the leaf nodes whose corresponding terms do appear in the document (i.e. “ball”, “baseball” and “championship”) have been assigned a CF of 1.

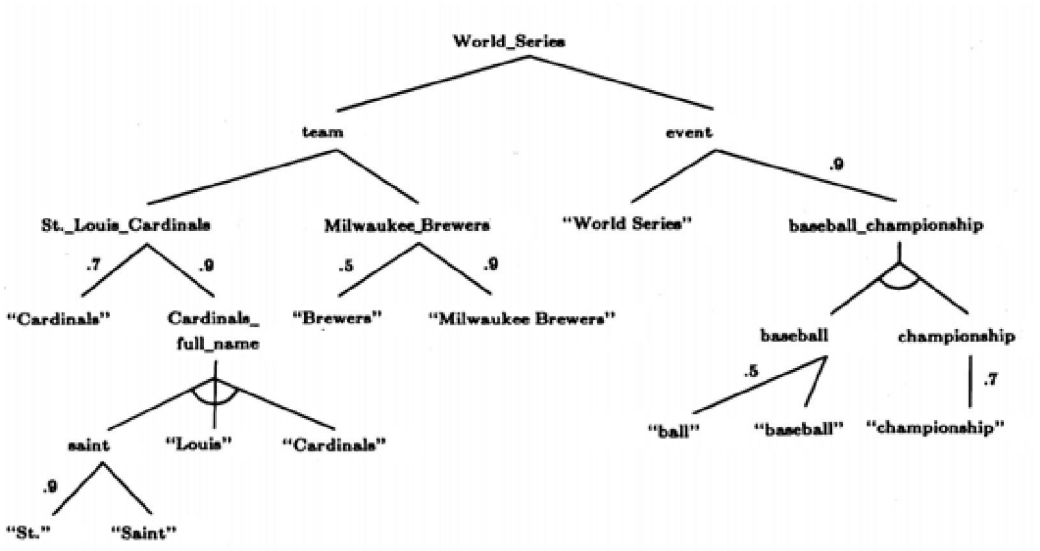

Figure 2 Inference tree for the sample scenario

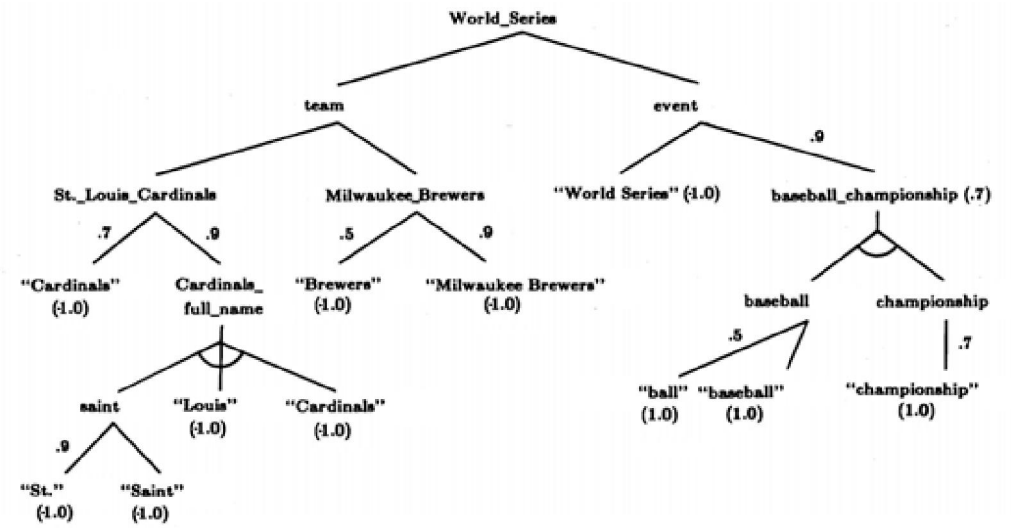

Figure 3 Assignment of CF values for the leaf nodes based on the facts (i.e. the document to be classified)

Once the leaf nodes have been assigned a CF value, it is possible to calculate the CF associated to the target hypothesis using the MyCIN method (see your lecture notes and slides for further details).

Based on the above guidelines, students must implement a program that given a sample document and a target hypothesis, automatically calculates the value associated to the target hypothesis. The inference engine can be implemented using any programming language. Please, note that the source code has to be properly commented to make it understandable by anyone. Code violating such requirements will not be accepted.

System Validation

Once the knowledge base has been created and the inference engine implemented, students must validate their systems using a test set. The latter can be found under the relative path */faculty/test* (20 documents for faculty members) and */staff/test* (20 documents for staff members).

Each document belonging to the test set must then be tested against two different hypotheses using the implemented program: (1) the web page belongs to a faculty member and (2) the web page belongs to a staff member. The document will be thus assigned the class with greater CF.

Once all the documents in the test set have been classified, students must create a summary of results including a confusion matrix. The latter is a visualization tool used in supervised learning to analyze the results of a classifier. Each column of the matrix represents the number of actual instances for each class, while each row represents the predicted number of instances of each class. Figure 4 shows a sample confusion matrix.

	Faculty (actual)	**Staff (actual)**
Faculty (predicted)	TF	FF
Staff (predicted)	FS	TS

Figure 4 Sample confussion matrix

As shown in Figure 4, the system has correctly classified TF + TS documents, while FS + FF documents were incorrectly classified. Other valuable statistics to be documented in the report are the following:

- Precision: proportion of predicted instances of a class which are indeed actual instances of the class. This value is calculated as follows for each class:

$$P_{faculty} = \frac{TF}{TF + FF}$$
$$P_{staff} = \frac{TS}{TS + FS}$$

- Recall: proportion of actual instances of a class which were correctly predicted to be instances of the class. This value is calculated as follows for each class:

$$R_{faculty} = \frac{TF}{TF + FS}$$
$$R_{staff} = \frac{TS}{TS + FF}$$

Besides collecting these measurements, students must also provide a written discussion of the results in a separate section of the assignment report.

Evaluation

The written report and the programs alone will be graded with a maximum of 6.9 points (in a 0-10 scale). To obtain a higher grade, it is necessary to do an oral presentation of the results of the assignment on a date to be determined (students will be informed in advance).

Submission Instructions

1. All students must submit all the items in the following checklist:
   - A report in PDF format no longer than 25 pages, including all the details of the development of the knowledge base, the implementation the inference engine and the system evaluation.
   - The source code and executables (if any) of all programs/libraries together with all required non-standard libraries (i.e. ready to be executed).
2. All the items from the checklist must come packed in a single RAR file named as follows:

   0910·MIR·[LASTNAME_FIRSTNAME].rar

Middle/compound names and multiple last names must be joined using a dash.

Some examples:

Martin Wheeler: 0910-MIR-[Wheeler_Martin].rar

Marcos Ángel Martínez Díaz: 0910-MIR-[Martinez-Diaz_Marcos-Angel].rar

The RAR file must be submitted by email to Prof. Miguel García (mgremesal@fi.upm.es) by January 29th 2010. Assignments not respecting the deadline will not be accepted.

Note on Plagiarism

Plagiarism is a serious offense, and we DO check for it. Any instance of plagiarism will result on failing the assignment, and by extension, the course (both in February and September).