Tutorial 8: Connectionist and Statistical Language Processing

Decision Trees

Date: 09.01.2002
Student's name:

1 The Task: Compound Interpretation

We will again use the Weka machine learning package, as introduced in the last tutorial. The files for this tutorial reside in:

```
/courses/connectionism.winter.01/tutorial8/
```

This tutorial is based on a dataset of Lapata (2000) that deals with the interpretation of nominalizations. Nominalizations are compounds that can receive either an object or a subject interpretation, or both. Examples include:

(1) a. SUBJ: child behavior ⇒ child behaves
b. OBJ: car lover ⇒ loves car
c. SUBJ|OBJ: satellite observation ⇒ satellite observes or observes satellite

In (1a), the modifier *child* is the subject of the head of the compound *behavior*. In (1b), the modifier *car* is the object of the head *lover*. In (1c), both interpretations are possible.

The machine learning task is to assign the right interpretation (i.e., the value `SUBJ` or `OBJ` in the attribute `compound_class`) to a nominalization using a decision tree that incorporates two sets of attributes. The first set of attributes provides information about the compound and the context it occurs in:

(2) `pos_left`: part of speech of the word immediately preceding the compound.
(3) `pos_mod`: part of speech of the modifier of the compound.
(4) `pos_head`: part of speech of the head of the compound.
(5) `pos_right`: part of speech of the word immediately following the compound.

The second set of attributes contains disambiguation information provided by a naive Bayes classifier (we will learn more about this in the next lecture). There are several variants of this classifier, each of which uses a different smoothing method:

(6) `backoff`: smoothing based backing off to *n*-grams.
(7) `wordnet`: classed-based smoothing based on the WordNet taxonomy.
(8) `roget`: classed-based smoothing based on Roget's Thesaurus.
(9) `jensen`: similarity-based smoothing using the Jensen-Shannon divergence.
(10) `confusion`: similarity-based smoothing using confusion probability.

How these smoothing methods work doesn't need to concern us here. Important is that all of them propose either the classification `subj` or the classification `obj` for a given instance. Of course they don't necessarily agree in their classification.

2 Building a Decision Tree Using Context Information

An important concept in Weka is the *filtering* of attributes. This allows us to select a subset of the attributes in the dataset to work with. Let's assume we want to build a decision tree that classifies nominalizations using only the POS information.

Start the Weka Explorer, and load the file `compounds.arff` in the directory `tutorial8` (see above). Then select all the POS attributes: `pos_left`, `pos_mod`, `pos_head`, `pos_right`, and the target attribute `compound_class`. Now click on the button `Apply filters` to create a working relation with the five attributes that you have selected. All further operations will apply to the working relation (not to the base relation).

Now click on the `Classifer` button and select the classifier `Id3`, which implements the ID3 decision tree induction algorithm discussed in the lecture. In the field `Test option` select `Percentage split` with 75%. This tells Weka to use 75% of the data for training and the remaining 25% for testing. Clicking on `Start` will generate the decision tree and will output its performance on the data set in `compounds.arff`.

Have a look at the decision tree that ID3 generates. Why is it so huge? Why are there some leaves marked `null`?

Now look at the performance of the decision tree. What's its overall precision (percentage of correctly classified instances)? Why are there unclassified instances?

Assume a hypothetical classifier that always assigns the most frequent class (find out which one that is). Such a classifier is called a *frequency baseline* and is often used to establish a lower bound on the performance of a more complex model. Which precision does the frequency baseline achieve? How does it compare to the precision of the decision tree?

Now we will try to find out why the decision tree performs so badly. Under `Test option` select `Use training set`. This tells Weka to compute the performance of the model on the training set, instead of on the test set. Which precision do we get now? What does this tell us about why the decision tree performs so badly on the test set?

Now let's try another decision tree induction algorithm, C4.5, which is similar to ID3 but does pruning. To use C4.5, select `j48.J48` in the `Classifier` menu, then run this classifier on the data, again using the 75/25 split into training and test set. Inspect the resulting decision tree. Why does it perform better than the tree generated by ID3?

3 Building a Metaclassifier

An important use of decision trees is as *metaclassifiers*, i.e., as classifiers that work on the output of other classifiers. An example is the research in Lapata (2000), where the output of five classifiers is used as the input to a decision tree that interprets nominalizations. The task of the decision tree is to decide which classifier to trust under which circumstances.

Use the filtering mechanism of Weka to create a working relation with the attributes `backoff`, `wordnet`, `roget`, `jensen`, and `confusion`, each of which represents the output of a naive Bayes classifier. The target attribute is again `compound_class`. Now generate a decision tree using the ID3 algorithm and test it with a 75/25 split of the data into training and test set.

Draw the decision tree that ID3 comes up with. What's the precision now? How would you explain the improvement compared to the use of POS attributes? Do you get a further improvement if you use C4.5 instead of ID3 to induce the decision tree?

Now build a minimal decision tree that only incorporates one of the five classifiers (`backoff`, `wordnet`, `roget`, `jensen`, `confusion`). Give the precision for each of these classifiers. How much do we gain by using a metaclassifier that combines the individual classifiers?

Finally, build one big decision tree that incorporates all attributes (the four POS attributes plus the five classifiers). Do we get a further improvement in performance? Compare the trees

constructed by ID3 and C4.5. Which algorithm comes up with the better tree? What does the C4.5 tree tell us about the POS attributes?

References

Lapata, Maria. 2000. The automatic interpretation of nominalizations. In *Proceedings of the 17th National Conference on Artificial Intelligence*, 716–721. Cambridge, MA: MIT Press.