Statistical and Learning Methods in Natural Language Processing

Ido Dagan and Shuly Wintner

Project description

Objectives

The final projects deal with one particular problem: part-of-speech tagging for Hebrew. Each team will implement a different machine learning method for solving this problem: Naïve Bayes, Winnow/Perceptron, HMM and KNN. This will enable us to compare the performance of different ML algorithms on this task.

Data representation

In order to give each implementation a fair chance, we prefer to have a unique data representation model for all projects. In other words, we prefer that all projects use, as much as possible, the same feature set for training and representing test examples. We define the standard set of features as:

- The lemma (citation form, or lexical entry)
- Part of speech
- Morphological features: Number (sg/pl/other); Gender (f/m/other); Person (1/2/3/other); Tense (past/-present/future/imperative/infinitive); Status (absolute/construct/other)
- Attached pronominal suffix (a combination of number, gender and person)
- Prefixes: Definite article, the Prepositions b/k/l/m, the Conjunctions w/$. Each of these features can have the values y/n. The order of the prefixes is ignored.

Then, use the same features for a window of ± 2 words around the target word.

Morphological disambiguator

The system you implement should read from the standard input a file with the results of morphological analysis of a Hebrew text. It should produce to the standard output a file with a single analysis for each of the words in input, chosen from the analyses in the input.

Input The format of the input file is: for each word, an integer number n indicating the number of analyses, followed by n analyses, each on a new line. The format of the analyses is explained on the web site.

Output The format of the output file: one analysis per line. If more than one analysis has the same part-of-speech, produce one of them.

Operation The disambiguator should operate from the command line:

```
disambiguate < input_file > output_file
```

For training you will also receive a gold standard: a file with the correct analysis of each of the words in the input file. The format of this file is exactly the same as the output file.

Evaluation

You should also implement an evaluation procedure. This system should take the output of the morphological disambiguation system, compare it with a gold standard and compute accuracy (percentage of the words for which your system produces the correct part of speech). Note that the evaluation should not compare the entire analysis, only the part of speech.

Input Two files in the output format.

Output A single number in the range $[0 - 1]$ indicating accuracy.

Operation From the command line:

```
evaluate results_file gold_file > accuracy_file
```

Wrap these procedures in a script which performs 10-fold cross validation evaluation.

Report

Once the system is working, perform some error analysis. Try to locate instances where the system goes wrong and generalize them. Play around with the set of features (either heuristically or using information-gain measures) and try to improve the results.

Suggest ideas for improvement. If additional resources are needed, specify how they can be acquired. Implement as many improvements as you wish and show better performance.

Use any programming language, as long as it can be compiled and executed on an ordinary linux/unix machine. These include Perl, Java, C++ etc. Please be conservative with the use of state-of-the-art libraries which may not be supported on all platforms.

Submission

Submit two systems: the morphological disambiguator and the evaluation script. For each system, submit the (well-documented) sources along with (very simple) installation and operation instructions. We will test the system on held-out data and compute accuracy for them. Submit a report which describes the system (conceptually, not technically). Report on the feature set you have found most useful and how it was selected. Report on evaluation results and on any improvements you have made to the original algorithm you implemented.

Deadline: TBD.

Good luck!