Practical Excercise 3: Part Of Speech disambiguation with SNoW

In this practical, you will investigate the performace of SNoW, a network of linear separators, in the task of Part Of Speech disambiguation. To do this, you are given a utility tool (FEX, see below), a learning program (SNoW, see below) and some data (BNC corpus, see below). You will partly try to reproduce the results at the following paper. Make sure you read it, you can find it at

http://12r.cs.uiuc.edu/~danr/Papers/pos.ps.gz

1 FEX

Before continuing, please read the FEX userguide, available at

http://12r.cs.uiuc.edu/~danr/Papers/fex.ps.gz

FEX allows you to specify the type of features you are interested in and extracts them from a given input text, creating an example file which will be the input for the next stage, SNoW (see below). FEX is still under development. Therefore, the userguide is not as exhaustive and informative in some cases as one might expect. Here is a list of some issues partially included in the userguide that you might find of interest:

- When the basic relation \texttt{base} is included in a feature pattern, FEX needs a file called \texttt{base.tag}. This file should be in the directory from where FEX is being launched. If FEX does not find it, it will stop and report the error.

To run FEX, execute \texttt{/g/150TP/files/PP3/fex}

2 SNoW

Before continuing, please read the SNoW userguide, available at

http://12r.cs.uiuc.edu/~danr/Papers/userguide.ps.gz

SNoW (Sparse Network of Winnows) is a system that learns a linear function over the feature space (specified in the previous step, FEX). To run SNoW, execute \texttt{/g/150TP/files/PP3/snow}
3 BNC Corpus

We converted part of the BNC Corpus (British National Corpus) to the format required by FEX. We split the 2,252,076 examples into 10 parts. The corpus files are named corpus<i>, where <i> is an integer between 0 and 9. These files are located at /g/150TP/files/PP3/corpus/.

We have also created 10 files called word_tag<i>, located in the same directory. These files contain every occurrence of a pair (tag, word) appearing in corpus<i>, one occurrence per line. A typical word_tag<i> file might look like:

```
... turned VVD out RP to TO cheer VVI ...
```

word_tag<i> files include the same information as the respective corpus<i> files, but in a different format. The reason we created these (redundant) files is to ease the task of constructing the base_tag file needed by FEX, since their format is more accessible than that of the corpus<i> files.

There are 143 different POS tags in the corpus.

4 Variations

You are asked to study how the performance obtained by SNoW on the BNC corpus is affected when trying the following variations.

- **Architecture**: Possible variations are (specify this to SNoW via an architecture file):
  - Different algorithms (*Winnow, Bayes* or *Perceptron*) for the update rule for the nodes in the SNoW network
  - Promotion/demotion parameters for *Winnow* and *Perceptron*
  - Number of passes through the data during training

- **Features**: Possible variations are (specify this to FEX via a feature script file):
  - From the set of features used in the FEX tutorial example (see FEX userguide), find the one that affects performance the most
  - Adding more conjunctive features
  - Varying the size of the windows in the feature patterns

- **Mode “cheat” vs. “nocheat”**: using statistics from the training set only vs. using statistics from the training and test sets. That is, when you run FEX with a feature script where base is specified, see how performance changes when file base_tag contains only tags from training
set or contains tags from both training and test sets. Notice that the latter is “cheating”, since, in theory, labels in the test set are only used for scoring purposes and should not be used for learning.

5 k-fold cross validation

You should use the k-fold cross-validation technique. This means that for every run you want to test, you have to choose k different splits of the corpus as follows (1 ≤ k ≤ 10):

- For every i ∈ (1,...,k) do
  - Let train = ∪j≠i corpus<j>
  - Let test = corpus<i>
  - Let P_i = performance obtained when running FEX and SNoW on train, test sets

Let final performance of run be average P = 1/k ∑_{i=1}^{k} P_i

With the symbol ∪ we mean concatenation of files.

6 What you should do

6.1 Main program

Write a program

    run <k> <-cheat|-nocheat> <script-file> <architecture-file>

which runs FEX and SNoW k times and outputs the average performance achieved, where:

- <k> is an integer between 1 and 10 denoting fold for cross-validation
- <-cheat|-nocheat> denotes if base_tag file should contain statistics from the training set only (-nocheat), or from both training and test sets (-cheat).
- <script-file> file containing feature patterns
- <architecture-file> file containing SNoW architecture

The program run should output one line like the following

    93.67% performance for k=3 -nocheat no_base.script winnow.arch

Notice that you have to build the base_tag file for the runs of FEX in which the feature relation base is specified.

Then, try the following runs of your program run:
1. Try all algorithms (Bayes, Winnow, Perceptron) with fixed parameters, all modes (-cheat, -nocheat), using features as in

http://12r.cs.uiuc.edu/~danr/Papers/pos.ps.gz

2. Using the -nocheat mode and the Winnow algorithm, vary Winnow’s parameters and/or features.

Examine the effects the variations have on performance and discuss the results.

6.2 Statistics on the corpus

Construct a table which says, for every $i = 1, 2, \ldots$, how many words have $i$ tags, and which fraction of the corpus they represent.

Compute the baseline for the BNC corpus. The baseline is the performance obtained by the process that assigns to every word its most common tag. This includes finding out which is the most common tag for every word in the corpus and then computing which fraction of the words in the corpus are labelled with precisely their most common tag.

7 What to hand in

Please submit your solution electronically to marias@eecs.tufts.edu before Wednesday, 4th April 2001, 5pm. You should include in your message:

- Statistics on BNC corpus.
- Performance results for every run tried, including information on which options were used.
- Discussion of results obtained. Please try to be concise, no need to write a whole dissertation.
- Clear, comprehensive documentation on how your programs work.
- Code (c++ programs, shell scripts, perl scripts) used. I highly recommend using Perl.

All the files handed in should be text files.