Computational Linguistics HT 2013: Weeks 3-5, Practical 1

Lecturers: Phil Blunsom and Ed Grefenstette
Demonstrator: Jan Botha and Nal Kalchbrenner

Introduction

In this practical you will compete against your classmates to build the best performing part of speech tagger. The main aim will be to design and implement a Hidden Markov Model tagger of the type described in lecture 2. Due to your lecturer's over indulgence of BBC crime drama, you will train and test your tagger for Danish text. No knowledge of Danish is required to do well in this practical, though you may take the opportunity to learn about Danish morphology in order to develop innovative strategies for tagging unknown words (the Wikipedia page on Danish Grammar is a good start).

Files

The training and testing data for the competition can be downloaded through the web interface, as well as an example competition submission.

danish.train This file contains the training corpus with one sentence per line. The format is *word/tag*.

danish.test This file contains the testing data for your submission to the competition website. The format is one sentence per line with only the words supplied.

example.py A Python implementation of a simple zero order HMM that outputs the most frequent tag for each test word, or the globally most frequent tag (`NOUN`) for unknown words.

```
python example.py --train danish.train --test danish.test > danish.example
```

If used with the `--readable` flag this script outputs the tagged test corpus in the same *word/tag* format as the training corpus, rather than the one tag per line format for the competition submission.

danish.example This file is an example submission produced by the *example.py* code. The format is one tag per line for the words in *danish.test*.

Note that the training data is in UTF-8 encoding which, depending on the programming language you use, may require care when processing individual characters.

The tag set used for this data is a small generic set of tags intended to be universal across languages. For more details see the following paper:

A Universal Part-of-Speech Tagset (LREC 2012). Slav, Petrov Dipanjan Das, Ryan McDonald.
`http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf`

The test data contains words that do not occur in the tagged training data. As such you must devise a strategy for tagging unknown words.

Tasks

Task 1: Submit a basic first order HMM tagger In order to implement your tagger you will need to write a program which estimates the quantities $P(\text{tag}_i|\text{tag}_{i-1})$ and $P(\text{word}|\text{tag})$ by counting instances in the training data, and working out relative frequencies as described in the lectures. You may at this stage want to think what to do about unobserved tag transitions: there is an extensive literature on ‘smoothing’ for HMMs, but a simple method is to add 1 to each transition counts and then treat unobserved transitions as if they had actually been seen once (Google for “add 1 smoothing” or “Laplace smoothing”).

Next you will have to implement the Viterbi algorithm for tagging new sentences. You may want to extend it a little to cope with the case where you encounter an unknown word. Your technique for handling unknown words will be crucial to your taggers performance.

When you have built the training and tagging routines for your model, test its accuracy by a technique called 10-fold ‘cross-validation’. To do this you split the training corpus 90%/10% into training and test sections. Train your tagger on the 90% and test on the 10%. Now take a different 90%/10% split and do the same, and so on, rotating the 90%/10% split so that eventually each bit of the corpus has been used as testing data. You can then average the results for each 10% test set. You can use this testing technique to debug your code and to test your extensions to the tagger.

Once you are satisfied with your taggers performance you should use it to tag the test data supplied and submit the output to competition website. You may make up to five competition submissions per day.

You can use whatever programming language you like, but make sure that your code is fully and clearly documented so that someone who may not be completely fluent with that language (i.e. the demonstrator) can understand what you have done.

Task 2: Win the competitions by exploring more advanced tagging algorithms Once you have implemented a basic HMM you should then explore extensions to your system to improve its accuracy. Possible extension include implementing a second order HMM, more advanced smoothing, and exploring the feature based strategies for tagging unknown words discussed in the lectures.

Website

A competition website for this practical resides at (not yet active):

`inclass.kaggle.com/c/oxford-danish-part-of-speech-tagging`

You will need to setup an account in order to be able to submit the results from your system. This can be done at:

`inclass.kaggle.com/account/register`

The competition is restricted to Oxford University participants, so you will need to use an email address ending in `ox.ac.uk`.

Once you have registered you will be able to download the competition data (you will be asked to accept the competition rules the first time), and submit your results using the *Make a Submission* button. You can make up to five submissions per day.

Scoring

When you submit a set of test results to the competition webpage they will be scored against the hidden tags. Tag accuracy is used to rank the submissions. During the competition the leader-board will display your best submission’s error on a subset of the test data. At the end of the competition the results will be updated to display the error on the full test data.

Initially there is a benchmark submissions on the leader board corresponding to the zero order HMM implemented in the `example.py` code supplied.

Forum

The competition webpage includes a forum for discussing the practical. You can use this forum to ask questions relating to the practical, or to share insights with your classmates. You will also use this forum to submit your final report.

Assessment

To complete this practical you must setup an account on the competition website and submit a set of test results for your HMM tagger including at least one novel approach to handling unknown words. You must also submit a post to the competition forum describing your implementation, its strengths and weaknesses (approximately half a page of text in total).

To be signed off for this practical you must demonstrate your implementation, show your forum post, and identify your submission on the competition leader-board, to one of the practical demonstrators.