Prototype Sequence Modeling Toolkit (PSMT) Quick-Start Guide

Aria Haghighi (aria42@cs.berkeley.edu)

June 18, 2008

1 Introduction

In this document, we'll provide step-by-step instructions for training and using a prototype sequence model. Throughout this document, we'll assume that you are at a terminal where the current directory is the root of the `pmst` project. Our running example will involve English POS induction where all data has been provided in the `data` sub-directory. If you want to make substantial changes to the code, feel free to examine the source code in the `src` sub-directory.

2 Make a prototype file

Every prototype sequence model must have a specified *prototype file*, where each line specifies a label and any numbers of prototypes (including none) associated with that label. Each line is tab-seperated and formated as follows:

```
LABEL PROTO_1 PROTO_2 ... PROTO_N
```

For an example of prototype file in the English POS domain, see `protoFiles/sample_proto.txt`. Typically, you will manually create this file for each domain or task.

3 Build a Similarity Model (Optional)

If you want to use prototype similarity features, you will need to train a word similarity model. We have provided a command-line tool to build word similarity models based on word contexts.[1]

[1] If you want to create a similarity model based on different criteria, you can simply implement the Java interface *edu.berkeley.nlp.prototype.PrototypeSimilarityModel*.

3.1 Context Similarty Model

A context similarity model represents a word w by a context vector $f(w)$ aggregated from the context surrounding w's usage in a corpus; a context feature consists of a word used near w along with a description of where the context word is relative to w. At its most specific a context feature can specify the direction and distance from w, i.e.`the#-1` means `the` occurs immediately to left of the target word. At it's least specific, a context feature only encodes counts of words which occur within a context window.

Then collection of word context vectors optionally undergos dimensionality reduction. It is possible to use SVD for dimensionality reduction as in [2], but this requires `matlab` to be available on the system. Another alternative is random projection [1] which is much faster than SVD with only a marginal cost in quality. The similarity between two words is given by the normalized dot-product between possibly reduced context vectors. To train a context model, run

```
scripts/build_context_model.rb conf/sample_context.conf
```

where `conf/sample_context.conf` is a sample configuration file specifying options for building a context model. To see available options run the above command with no argument.

4 Train Sequence Model

To train the sequence model run,

```
scripts/train_seq_model.rb conf/sample_train.conf
```

again `conf/sample_train.conf` specifies options for training the sequence model. To see all available options, run the above script without arguments. Once training completes, which should take about 10 minutes. Once it is complete, the sequence model will be written to the `models/seq.model`.

In terms of features that are utilized, a feature for each whole word is used as well as a range of other word features which are configurable via options. The sample configuration `sample_train.conf` turns on all word features relevant for POS tagging. Note that only features on *word-types* are allowed.

5 Test Sequence Model

To train the sequence model you build in the last section, run

```
scripts/test_seq_model.rb conf/sample_test.conf
```

As before, *conf/sample_test.conf* specifies options for testing the sequence model, including where to find the files to tag. To see all available options, run the above script without arguments. Once tagging completes, the tagged files will appear in the `output` directory (this is configurable via the configuration file). Each token in an output file consists of the word and tag conjoined (*word##tag*) by `##`; this delimeter can be configured via the `delimeter` option.

6 POS Induction Experiment

If you want to re-run the POS induction experiment, you must fill in the location of the WSJ portion of the treebank in the configuration file `conf/pos_induction.conf` as follows

```
treebankPath path-to-wsj
```

You will also need to download `legacy_context.model` from the project page[2] and move it into the `models/` sub-directory. Then you can run the experiment using the following command,

```
scripts/pos_induction_test.rb conf/pos_induction.conf
```

This experiment should yield an 80.5% accuracy.

References

[1] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Knowledge Discovery and Data Mining*, 2001.

[2] Aria Haghighi and Dan Klein. Prototype-driven learning for sequence modeling. In *HLT-NAACL*, 2006.

[2] http://code.google.com/p/prototype-sequence-toolkit