Unsupervised Semantic Parsing (USP) User Guide

Hoifung Poon
hoifung@cs.washington.edu

September 30, 2009

1 Introduction

In the following sections, we will provide step-by-step instructions for using the USP system as described in [2] in unsupervised semantic parsing and question answering. Throughout this document, we will assume that you are at the directory where you unpack `usp-code.tar.gz`, and we will use the experiments in [2] as a running example, which apply USP to the GENIA dataset [1] and answer questions about biological entities and relations.

2 Files Included

`poon09.pdf:` the paper on unsupervised semantic parsing and the USP system.

`questions.txt:` the questions used in the question-answering evaluation.

`usp.eval:` the answers extracted by the USP system and labels.

`usp-code.tar.gz:` the USP code archive, which includes the following files:

`usp.jar:` Java executable for USP.

`src/:` source code for USP.

`genia/:` default data directory containing the input files for replicating the USP experiments in [2]. It contains raw text and syntactic analyses (morphologies and Stanford dependencies) of 1999 PubMed abstracts in GENIA [1].

`results/:` default directory for the output of USP.

`eval/:` default directory for the question files.

3 Input To USP

The input to USP consists of syntactic analyses of source text. In particular, USP assumes that the data directory (`genia` in our example) contains three subdirectories:

`text/` contains the raw text files.

`morph/` contains input tokens, stems and POS tags.

`dep/` contains the Stanford dependencies obtained with the format of `collapsedTree`.

4 Unsupervised Semantic Parsing

To run USP with the default directories, run:

```
java -mx20000m -cp usp.jar semantic.Parse genia results
```

This will generate the following files in `results`:

`genia.mln` contains counts tallied from the MAP parse, which implicitly represent the learned parameters. Each cluster starts with a line containing the unique ID along with core forms and counts. It is followed by a number of argument types, each of which consists of three lines: first the ID and counts for various argument numbers, second the argument forms and counts, third the argument clusters and counts.

`genia.parse` contains the MAP semantic parses. Every sentence is partitioned into subformulas, each of which is specified by four lines: first the unique ID (docId+sentenceId+partId) and the text fragment, second the cluster ID and core form, third the ID of the parent subformula and cluster, fourth the argument type ID along with the argument form and ID.

`genia.clustering` contains the core forms of non-unit clusters. Each cluster is represented by a line containing the core forms and counts.

The current implementation requires substantial amount of memory. With heap size of 20GB, It takes about 20 minutes on the GENIA dataset on a 64-bit linux machine with eight cores (Intel Xeon 2.3GHz). With a smaller heap size (e.g. -mx8000m), the running time is substantially longer (e.g., 80 minutes). Running USP on the Penn-Treebank WSJ data takes about 50 minutes on the same machine and requires about 25GB of memory. Running USP on a 32-bit machine will result

in out-of-memory error due to heap size limit. In future work, we will seek to reduce memory requirement per processor by making computation distributed.

The exponential priors can be specified through optional parameters priorNumParam and priorNumConj. For a full list of options, run the program without any parameter.

5 Question Answering

To replicate the question answering experiment with the learned model and MAP parses, run

```
java -cp usp.jar eval.USP eval results genia
```

This assumes that `eval/` contains the question files, and `results/` contains the learned model and parses. The output consists of the answers extracted by USP for the questions. Currently, our implementation only handles simple factoid questions as specified in the paper.

References

[1] Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:180–82, 2003.

[2] Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore, 2009. ACL.