Abstract

Incremental syntactic language models score sentences in left-to-right fashion and are therefore a good mechanism for incorporating syntax into phrase-based translation. We integrate an incremental syntactic LM, ModelBlocks, into the Moses phrase-based translation system. We document the novel contributions to software which accompany the paper *Incremental Syntactic Language Models for Phrase-based Translation* (Schwartz et al., 2011). The exact models used in these experiments are also released.

1 Contribution Summary

The exact versions of the ModelBlocks HHMM parser and the Moses phrase-based machine translation systems used in *Incremental Syntactic Language Models for Phrase-based Translation* are located in the software directory. The specific modifications and additions we made to ModelBlocks and Moses have been copied into the software/diffs directory.

The exact versions of the ModelBlocks parsing model and Moses phrase table and n-gram language model files used are located in the data-models directory.

This document summarizes our code and data contributions, and illustrates how to compile and run our code, for the purposes of replicating and building on our work.

2 External Dependencies

The following external code and external data sets, distributed separately, are required to replicate our results:

- SRI Language Modeling Toolkit (Stolcke, 2002)
- Penn Wall Street Journal Treebank (Marcus et al., 1993)
- NIST Open MT Urdu-English data, as partitioned and preprocessed in (Baker et al., 2009)

3 Code: Moses

The code for the Moses decoder is available at http://www.sf.net/projects/mosesdecoder. The results in this submission were based on svn trunk revision 3739, with additions and modifications as listed below.

3.1 Code Additions

We implemented a syntactic language model within Moses:
3.2 Code Modifications

We modified existing Moses source code to compile and integrate the new syntactic language model feature:

moses-decoder/moses/src
   SyntacticLanguageModel.h
   SyntacticLanguageModel.cpp
   SyntacticLanguageModelFiles.h
   SyntacticLanguageModelState.h

3 Code: ModelBlocks

The code for the parser is available at http://www.sf.net/projects/modelblocks. The results in this submission were based on git master tree revision 839b44d2d7e7c2d0845401b4c77f3070a665bde7, with additions and modifications as listed below.

4.1 Code Additions

We implemented the following scripts to calculate language models and to interpolate language models for perplexity calculations:

wsjparse/scripts/
   calc-ngram-counts.sh
   calc-ngram-lm-ppl.sh
   calc-ngram-lm.sh
   elim-rare-words-from-sents.py
   interpolate-lm.rb
   interpolate-ngram-lms.rb

4.2 Code Modifications

rvtl/include/nl-hmm.h Modified HMM implementation to calculate beam probability sum, and to appropriately handle unknown words.

wsjparse/Makefile Modified to appropriately handle unknown words.

wsjparse/include/HHMMParser.h Enhanced parser to calculate and output perplexity values.

wsjparse/include/TextObsModel.h Modified parser lexical model to appropriately handle unknown words.

5 Data: Models

The exact data models used during parsing and translation are provided below:

wsjTRAIN-pu-unk-nr.gf-hhmm.model Parser model, trained on WSJ Treebank

order-5.srilm 5-gram language model, trained on WSJ Treebank

phrase-table.gz Moses Urdu-English translation phrase table

reordering-table.wbe-msd-bidirectional-fe.gz Moses reordering table

moses.ini Urdu-English Moses configuration, tuned using the above models

6 How to Reproduce Results

Figure 1 lists the steps required to reproduce our translation results (for Moses configured using the syntactic language model) on the NIST Open MT 2008 devtest set.

Figure 2 lists the steps required to use the HHMM parser to parse and calculate perplexity values.

The code additions in 4.1 can be used to calculate perplexity data using SRILM (calc-ngram-lm-ppl.sh), to interpolate HHMM and n-gram language models (interpolate-ngram-lms.rb), and to interpolate two different n-gram language models (interpolate-lm.rb).
To compile parser:

```
$ cd software/wsjparse
$ echo "USER_TREEBANK_LOCATION = /path/to/WSJ_Treebank" > user-treebank-location.txt
$ make bin/parser-gf-hhmm
```

To rebuild the parser model:

```
$ mkdir genmodel
$ make genmodel/wsjTRAIN-pu-unk-nr.gf-hhmm.model
```

To parse and calculate perplexity:

```
$ cat /path/to/devtest.en | bin/parser-gf-hhmm --beam=2000 --perplexity \  
  data-models/wsjTRAIN-pu-unk-nr.gf-hhmm.model > devtest.ur.hhmm.ppl
```

Figure 2: To compile the parser with our modifications, run the above commands.

References


