An Introduction to Moses & GIZA++ Toolsets

Anoop Kunchukuttan
anoopk@cse.iitb.ac.in

CS626
30 Jul 2013

What is Moses?

- Most widely used **phrase-based** SMT framework
  - *'Moses'* actually refers to the SMT decoder
  - However, includes training, tuning, pre-processing tools, etc.
  - Open-source, modular and extensible - developed primarily at the University of Edinburgh
- Written in C++ along with supporting scripts in various languages
  - https://github.com/moses-smt/mosesdecoder
- Also supports *factored, hierarchical phrase based, syntax based* MT systems
  - Other decoders of interest: cdec, Joshua, ISI ReWrite
- Visit: http://www.statmt.org/moses/

Recap: SMT basics

Generative Model

- Noisy channel model of translation from sentence f to sentence e.
- Task is to recover e from noisy f.

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e})$$

P(f|e): Translation model, addresses adequacy

P(e): Language model, addresses fluency

GIZA++ : translation model params
SRILM: language model
ISI ReWrite: decoder

Discriminative Model

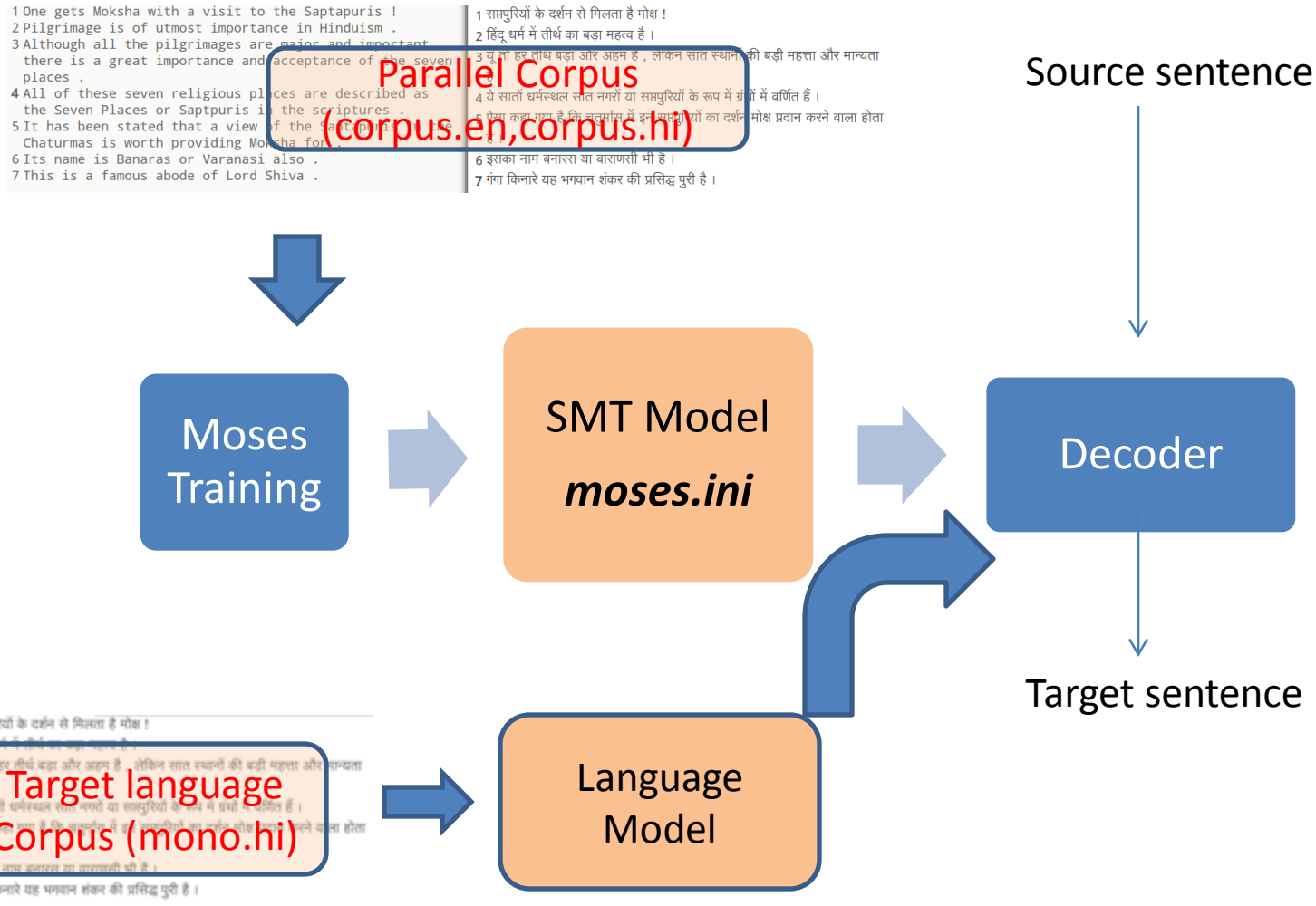
- Maximum Entropy based model, incorporating arbitrary features

$$\hat{\mathbf{e}} = \underset{e}{\operatorname{argmax}} \exp \sum_i \lambda_i h_i(f, e)$$

- h_i - features functions (phrase/lexical direct/inverse translation probability, LM probability, distortion score)
- λ_i are weights of the features

GIZA++,train_moses.perl : phrase, lexical, distortion probabilities
SRILM: language model score
moses: decoder

What does Moses do?



Installing Moses

- Compile and install the following:
  - Moses
  - GIZA++
  - Language Modelling toolkit (SRILM/IRSTLM)
- Installation Guides
  - From StatMT: http://www.statmt.org/moses_steps.html
  - Works best for Ubuntu: http://organize-information.blogspot.in/2012/01/yet-another-moses-installation-guide.html
  - A bit older guide: http://www.cfilt.iitb.ac.in/Moses-Tutorial.pdf
- Be ready for a few surprises !

Workflow for building a phrase based SMT system

- **Corpus Split:** Train, Tune and Test split
- **Pre-processing**: Normalization, tokenization, etc.
- **Training**: Learn Phrase tables from *Training* set
- **Tuning**: Learn weights of discriminative model on *Tuning* set
- **Testing**: Decode *Test* set using tuned data
- **Post-processing**: regenerating case, re-ranking
- **Evaluation**: Automated Metrics or human evaluation

Pre-processing -1 (Normalize the text) Case normalization

- Recasing method:
  - Convert training data to lowercase
  - Learn recasing model for target language

```
scripts/recaser/train-recaser.perl --dir MODEL --corpus CASED [--ngram-count NGRAM] [--train-script TRAIN]
```

  - Restore case in test output using recasing model

```
scripts/recaser/recase.perl --in IN --model MODEL/moses.ini --moses MOSES >OUT
```

- Truecasing method
  - Learnt via True casing model

```
scripts/recaser/train-truecaser.perl --model MODEL --corpus CASED
```

  - Convert words at start of sentence to lowercase (if they generally occur in lowercase in corpus)

```
scripts/recaser/truecase.perl --model MODEL < IN > OUT
```

  - Restore case in test output using truecasing model

```
scripts/recaser/detruecase.perl < in > out
```

Pre-processing -1 (Normalize the text) Character Normalization

Important for Indic scripts

- Multiple Unicode representations
  - e.g. ज़ can be represented as +u095B or +u091c (ज) +1093c (nukta)
- Control characters
  - Zero-Width Joiner/Zero-Width Non-Joiner
- Characters generally confused
  - Pipe character (|) with *poorna-virama* (।)
  - Colon(:) with *visarga* (ः)

Preprocessing-2 (Other steps)

- Sentence splitting
  - Stanford Sentence Splitter
  - Punkt Tokenizer (NLTK library)
- Tokenization
  - Scripts/tokenizer/tokenizer.perl
  - Stanford Tokenizer
  - Many tokenizers in the NLTK library

Train Language Model

- Supported LM tools:
  - KenLM comes with Moses
  - SRILM and IRSTLM are other supported language models
- Can train with one and test with another LM
  - All generate output in ARPA format
- **Training SRILM based language model**

```
ngram-count -order <n> -kndiscount -interpolate -text <corpus> -lm <lmfile>
```

Training Phrase based model

- The training script (train-model.perl) is a meta-script which does the following:
  - Run GIZA
  - Align words
  - Extract Phrases
  - Score Phrases
  - Learn Reordering model

- Run the following command

```
scripts/training/train-model.perl \
  -external-bin-dir <external_bin_dir>
  -root-dir <workspace_dir> \
  -corpus <train_path_without_ext> \
  -e <tgt_lang> -f <src_lang> \
  -alignment <phrase_extraction_strategy e.g. grow-diag-final-and> \
  -reordering <reordering_strategy e.g. msd-bidirectional-fe>
  -lm <lm_type, 0 for srilm>:<lm_order>:<lm_file>:0
```

More Training Options

- Configure maximum phrase length
  - -max-phrase-length
- Train the SMT system in parallel
  - -parallel
- Options for parallel training
  - -cores, -mgiza, -sort-buffer-size, -sort-parallel, etc.

The phrase table
($workspace_dir/model/phrase-table.tgz)

```
956 ' 'Twas he that ||| निखरे माती ||| 0.2 1.39907e-05 1 0.0834042 2.718 ||| 0-0 1-0 2-0 1-1 ||| 5 1 1
957 ' 'Twas he ||| निखरे माती ||| 0.2 0.00209263 1 0.0834042 2.718 ||| 0-0 1-0 2-0 1-1 ||| 5 1 1
958 ' 'Very good. ||| --ठीक तो है। ||| 1 0.0123742 1 7.53276e-05 2.718 ||| 0-0 1-0 2-0 2-1 2-2 ||| 1 1 1
959 ' 'Very well, sir. ||| हाँ सर! ||| 0.5 9.46519e-06 1 0.0063612 2.718 ||| 0-0 1-0 2-0 3-0 3-1 ||| 2 1 1
960 ' 'Very well, then. ||| ठीक ही है। ||| 1 2.77816e-12 1 9.01339e-06 2.718 ||| 0-0 1-0 2-0 3-0 2-1 2-2 ||| 1 1 1
961 ' 'Very well. ||| अच्छा! ||| 0.25 0.00115741 1 0.0434682 2.718 ||| 0-0 1-0 2-0 ||| 8 2 2
962 ' 'Watching me, of all persons. ||| --मुझको? ||| 1 2.14335e-05 1 0.169273 2.718 ||| 0-0 1-0 2-0 3-0 4-0 5-0 ||| 1 1 1
963 ' 'We have heard that you have ||| " "हमने सुना है कि आपने ||| 1 0.000316347 1 7.88927e-08 2.718 ||| 0-0 1-1 2-1 3-2 4-3 4-4 5-5 6-5 ||| 1 1 1
964 ' 'We have heard that ||| " "हमने सुना है कि ||| 1 0.00391593 1 2.99769e-06 2.718 ||| 0-0 1-1 2-1 3-2 4-3 4-4 ||| 1 1 1
965 ' 'We have heard ||| " "हमने सुना ||| 1 0.0118525 1 4.3827e-05 2.718 ||| 0-0 1-1 2-1 3-2 ||| 1 1 1
966 ' 'We have ||| " "हमने ||| 1 0.0282705 1 0.00021881 2.718 ||| 0-0 1-1 2-1 ||| 1 1 1
967 ' 'Well, I do take rest, father. ||| मृत्यु क्या है? ||| 1 5.60474e-20 1 1.34553e-05 2.718 ||| 0-0 1-0 2-0 4-0 5-0 6-0 1-1 3-1 1-2 ||| 1 1 1
968 ' 'Well, it happens. ||| --जरुर होता है। ||| 1 0.00130446 1 0.000452107 2.718 ||| 0-0 1-0 2-0 3-0 3-1 3-2 ||| 1 1 1
969 ' 'Well, people who are good at ||| जो ||| 7.19321e-05 7.11023e-21 1 0.299537 2.718 ||| 3-0 ||| 13902 1 1
```

- inverse phrase translation probability
- inverse lexical weighting
- direct phrase translation probability
- direct lexical weighting
- phrase penalty (always exp(1) = 2.718)
- Within-phrase alignment information

The model file ($workspace_dir/model/moses.ini)

```
#########################
### MOSES CONFIG FILE ###
#########################

# input factors
[input-factors]
0

# mapping steps
[mapping]
0 T 0

# translation tables: table type (hierarchical(0), textual (0), binary (1)), source-factors, target-factors, number of scores, file
# OLD FORMAT is still handled for back-compatibility
# OLD FORMAT translation tables: source-factors, target-factors, number of scores, file
# OLD FORMAT a binary table type (1) is assumed
[ttable-file]
0 0 0 5 /home/anoop/tmp/sample_data/workspace/moses_data/model/phrase-table.gz

# no generation models, no generation-file section

# language models: type(srilm/irstlm), factors, order, file
[lmodel-file]
0 0 3 /home/anoop/tmp/sample_data/sample_monolingual.en.lm


# limit on how many phrase translations e for each phrase f are loaded
# 0 = all elements loaded
[ttable-limit]
20

# distortion (reordering) files
[distortion-file]
0-0 wbe-msd-bidirectional-fe-allff 6 /home/anoop/tmp/sample_data/workspace/moses_data/model/reordering-table.wbe-msd-bidirectional-fe.gz

# distortion (reordering) weight
[weight-d]
0.3
0.3
0.3
0.3
0.3
0.3
0.3

# language model weights
[weight-l]
0.5000


# translation model weights
[weight-t]
0.20
0.20
0.20
0.20
0.20

# no generation models, no weight-generation section

# word penalty
[weight-w]
-1
```

Tuning the Model

- Tune the parameter weights to maximize translation accuracy on '*tuning set*'
- Different tuning algorithms are available:
  - MERT, PRO, MIRA, Batch MIRA
- Generally, a small tuning set is used (~500-1000 sentences)
- MERT (Minimum Error Rate Tuning) is most commonly used tuning algorithm:
  - Model can be tuned to various metrics (BLEU, PER, NIST)
  - Can handle only a small number of features

MERT Tuning

- Command:

```
scripts/training/mert-moses.pl <tun_src_file>
   <tun_tgt_file>  <decoder_binary_path>  \
   <untuned_model_file> --working-dir <workspace> --rootdir
   <moses_script_dir>
```

- Important Options
  - Maximum number of iterations. Default: 25

    `--maximum-iterations=ITERS`
  - How big nbestlist to generate

    `--nbest=100`
  - Run decoder in parallel

    `--jobs=N`

Decoding test data

- Decoder command

```
bin/moses  -config <moses_config> -input-file <input_file>
```

- Other common decoder options
  - alignment-output-file <file>: output alignment information
  - n-best-list: generate n-best outputs
  - threads: number of threads
  - ttable-limit: number of translations for every phrase
  - xml-input: supply external translations (named entities, etc.)
  - minimum-bayes-risk: use MBR decoding to get best translation
  - Options to control stack size

Evaluation Metrics

- Argument for validation of automated metrics: correlation with human judgments
- Automatic Metrics:
  - BLEU (Bilingual Evaluation Understudy)
  - METEOR: More suitable for Indian languages since it allows synonym, stemmer integration
  - TER, NIST
- Commands
  - Bleu scoring tool:

    scripts/generic/multi-bleu.perl
  - Mteval scoring tool: official scoring tool at many workshops (BLEU and NIST)

    scripts/generic/mteval-v13a.pl

More Moses Goodies

- XML RPC server
- Binarize the phrase tables
- Load Phrase table on demand
- Experiment Management System (EMS)
- A simpler EMS
  - https://bitbucket.org/anoopk/moses_job_scripts
- … continue exploring

What is GIZA++?

- GIZA++ is a system for training word alignment systems
- Uses of GIZA++:
  - Building block for phrase based MT system
  - Learning probabilistic lexicon from corpus
- Implementation of the IBM models
- GIZA++ does not contain a decoder
  - Try using ISI Rewrite decoder

Packages Needed to Run GIZA ++

(slides from : Bridget McInnes)

- GIZA++ package
  - developed by Franz Och
  - www-i6.informatik.rwth-aachen.de/Colleagues/och
- mkcls package
  - developed by Franz Och
  - www.-i6.informatik.rwth-aachen.de/Colleagues/och

Step 1

Retrieve data:

- Create a parallel corpus: one sentence per line format

Step 2

Create files needed for GIZA++:

- Run plain2snt.out located within the GIZA++ package
  - ./plain2snt.out french english
- Files created by plain2snt
  - english.vcb
  - french.vcb
  - frenchenglish.snt

Files Created by plain2snt

- english.vcb consists of:
  - each word from the english corpus
  - corresponding frequency count for each word
  - an unique id for each word
- french.vcb
  - each word from the french corpus
  - corresponding frequency count for each word
  - an unique id for each word
- frenchenglish.snt consists of:
  - each sentence from the parallel english and french corpi translated into the unique number for each word

Example of .vcb and .snt files

french.vcb:

2 Debates 4
3 du 767
4 Senate
5 (hansard) 1

english.vcb:

2 Debates 4
3 of 1658
4 the 3065
5 Senate 107
6 (hansard) 1

frenchenglish.snt

1
2 3 4 5
2 3 4 5 6
1
…

Step 3

Create mkcls files needed for GIZA++:

- Run _mkcls which is not located within the GIZA++ package
  - mkcls –pengish –Venglish.vcb.classes
  - mkcls –pfrench –Vfrench.vcb.classes
- Files created by _mkcls
  - english.vcb.classes
  - english.vcb.classes.cats
  - french.vcb.classes
  - french.vcb.classes.cats

Files Created by the mkcls package

- .vcb.classes files contains:
  - an alphabetical list of all words (including punctuation)
  - each words corresponding frequency count
- .vcb.classes.cats files contains
  - a list of frequencies
  - a set of words for that corresponding frequency

.vcb.classes ex:

"A 99
"Canadian 82
"Clarity 87
"Do 78
"Forging 96
"General 81

.vcb.classes.cats ex:

…
82: … "Candian, "sharp, 1993, …
…
87: "Clarity, "grants, 1215 , …
…
99: "A, 1913, Christian, …

Step 4

Run GIZA++:

- Generate co-occurrence file

Sn2cooc.out french.vcb english.vcb frenchenglish.snt > fe.cooc

- **Run GIZA++ located within the GIZA++ package**
  - **./GIZA++ -S french.vcb –T english.vcb –C frenchenglish.snt –CoocurrenceFile fe.cooc**
- **Files created by GIZA++:**
  - **Decoder.config**
  - **ti.final**
  - **actual.ti.final**
  - **perp**
  - **trn.src.vcb**
  - **trn.trg.vcb**
  - **tst.src.vcb**
  - **tst.trg.vcb**
  - **a3.final**
  - **A3.final**
  - **t3.final**
  - **d3.final**
  - **D4.final**
  - **d4.final**
  - **n3.final**
  - **p0-3.final**
  - **gizacfg**

Files Created by the GIZA++ package

- **Decoder.config**
  - **file used with the ISI Rewrite Decoder**
  - **developed by Daniel Marcu and Ulrich Germann**
  - **http://www.isi.edu/licensed-sw/rewrite-decoder/**
- **trn.src.vcb**
  - **list of french words with their unique id and frequency counts**
  - **similar to french.vcb**
- **trn.trg.vcb**
  - **list of english words with their unique id and frequency counts**
  - **similar to english.vcb**
- **tst.src.vcb**
  - **blank**
- **tst.trg.vcb**
  - **blank**

(cont) Files Created by the GIZA++ package

- ti.final
  - file contains word alignments from the french and english corpus
  - word alignments are in the specific words unique id
  - the probability of that alignment is given after each set of numbers
  - ex:
    - 3 0 0.237882
    - 1171 1227 0.963072
- actual.ti.final
  - file contains word alignments from the french and english corpus
  - words alignments are the actual words not their unique id's
  - the probability of that is alignment is given after each set of words
  - ex:
    - of NULL 0.237882
    - Austin Austin 0.963072

(cont) Files Created by the GIZA++ package

- **A3.final**
  - **matches the english sentence to the french sentence and give the match an alignment score**
  - **ex:**
    - **#Sentence pair (1) source length 4 target length 5 alignment score : 0.000179693**

      Debates of the Senate (Hansard)

      Null ({3}) Debats ({1}) du ({2}) Senat ({4}) (hansard) ({5})
- **perp**
  - **list of perplexity for each iteration and model**

    #trnsz tstsz iter model trn-pp test-pp trn-vit-pp tst-vit-pp

    2304 0 0 Model1 10942.2 N/A 132172 N/A

    - **trns – training size**
    - **tstsz – test size**
    - **iter – iteration**
    - **trn-pp – training perplexity**
    - **tst-pp – test perplexity**
    - **trn-vit-pp – training viterbi perplexity**
    - **tst-vit-pp – test viterbi perplexity**

(cont) Files Created by the GIZA++ package

- a3.final
  - contains a table with the following format:
    - i j l m p (i / j, l, m)
      - j = position of target sentence
      - i = position of source sentence
      - l = length of the source sentence
      - m = length of the target sentence
      - p(i / j, l, m) = is the probability that a source word in position i is moved to position j in a pair of sentences of length l and m
  - ex:
    - 0 1 1 60 5.262135e-06
      - 0 – indicates position of target sentence
      - 1 – indicates position of source sentence
      - 1 – indicates length of source sentence
      - 60 indicates length of target sentence
      - 5.262135e-06 – is the probability that a source word in position 1 is moved position 0 of sentences of length 1 and 60
- d3.final – similar to a3.final with positions i and j switched

(cont) Files Created by the GIZA++ package

- **n3.final**
  - **contains the probability of the each source token having zero fertility, one fertility, … N fertility**
- **t3.final**
  - **table after all iterations of Model 4 training**
- **d4.final**
  - **translation table for Model 4**
- **D4.final**
  - **distortion table for IBM-4**
- **gizacfg**
  - **contains parameter settings that were used in this training.**
  - **training can be duplicated exactly**
- **p_03.final**
  - **probability of inserting null after a source word**
  - **file contains: 0.781958**

References

- Moses Manual (Your complete ref. to Moses)
- Hoang, Hieu, and Philipp Koehn. "Design of the moses decoder for statistical machine translation." *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*. Association for Computational Linguistics, 2008.
- NLTK
- Unicode Tutorial