GIZA++ and Moses

Outline

- GIZA++
  - Steps
  - Output files
- Moses
  - Training pipeline
  - Decoder
- Demo

GIZA++

- A statistical machine translation toolkit used to train IBM Models 1-5 (moses only uses output of IBM Model-1)
- Step-1: plain2snt.out - convert corpus to GIZA++ format
  - Generates vcb files and snt files
- Step-2: snt2cooc.out - generates the concurrence file
- Step-3: GIZA++ - generates alignment files

Output Files (1/3)

- Translation table (*.t*.*)
  - s_id t_id P(t_id|s_id)
- Fertility table (*.n3.*)
  - s_id p0 p1 p2 . . . pn
  - where p0 is the probability that the source token has zero fertility; p1, fertility one, ...,
- Probability of inserting a null after a source word (*.p0*)

Output Files (2/3)

- Alignment tables (*.a*.*)
  - *Format: i j l m P(i | j , l, m)*
  - Where,

    j = position in target sentence

    i = position in source sentence

    l = length of source sentence *m* = length of 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*

Output Files (3/3)

- Distortion table(*.d3.*)
  - The format is similar to the alignment tables but the position of i and j are switched:

    j i l m P(j | i, l, m)
- Alignment probability table for HMM alignment mode (*.A3.*)
- Perplexity File (*.perp)
- Revised vocabulary files (*.src.vcb, *.trg.vcb)
- Final parameter file: (*.gizacfg)

Moses

- An SMT system that allows you to automatically train translation models for any language pair
- Requires a parallel corpus
- An efficient search algorithm: finds the highest probability translation among the exponential number of choices
- Main components
  - Training Pipeline (mainly in perl)
  - Decoder (C++)

Training Pipeline

- Preprocessing
  - Corpora cleaning
  - Tokenization
  - Case conversation
- Word alignments (Using GIZA++)
- Language model building (Using SRILM)

Decoder

- Find the highest scoring sentence in the target language according to the translation model
- Four modules:
  - **Input:** a plain sentence/annotated with xml-like elements /complex structure like a lattice or confusion network
  - **Translation model:** phrase-phrase rules/hierarchical rules. Also supports **features** to add extra information to the translation process
  - **Decoding algorithm:** several different strategies for the search, such as stack-based, cube-pruning, chart parsing *etc.*
  - **Language model:** supports several different language model toolkits (SRILM, KenLM, IRSTLM, RandLM)

Step-1: Preparing Training Data

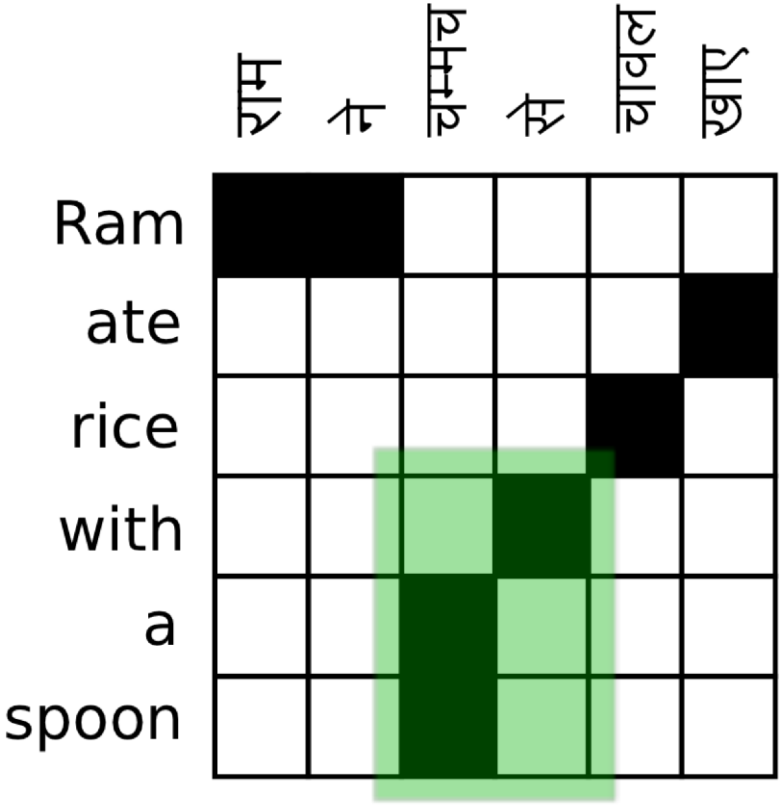
- Training data
  - sentence aligned data (one sentence per line) in two files
- Cleaning the corpus
  - removes empty lines, redundant space characters
  - drops lines (and their corresponding lines), that are empty, too short, too long

Step-2 : Run GIZA++ for Word Alignments

- Step-1: plain2snt.out - convert corpus to GIZA++ format
  - Generates vcb files and snt files
- Step-2: snt2cooc.out - generates the concurrence file
- Step-3: GIZA++ - generates alignment files (moses uses only IBM Model-1 output)

Step-3: Align Words in sentences

- Different heuristics
- Default: grow-diag-final



Step-4: Get Lexical Translation Table

- Estimates a maximum likelihood lexical translation table from the alignments (in both the directions)

  e_word f_word p(e_word | f_word)

Step-5: Extract Phrases

wiederaufnahme ||| resumption ||| 0-0

wiederaufnahme der ||| resumption of the ||| 0-0 1-1 1-2

wiederaufnahme der sitzungsperiode ||| resumption of the session ||| 0-0 1-1 1-2 2-3

der ||| of the ||| 0-0 0-1

der sitzungsperiode ||| of the session ||| 0-0 0-1 1-2

Step-6: Score Phrases

- Format

  phrase_f ||| phrase_e ||| a ||| b ||| c ||| d ||| e

  Where,

  a = Inverse phrase translation probability

  b = Inverse lexical weighting

  c = Direct phrase translation probability

  d = Direct lexical weighting

  e = phrase penalty

Step-7: Reordering Model

Step-8: Create configuration file

- Configuration file for the decoder (moses.ini)

Tuning

- Decoding uses different features in a linear model
  - probabilities from the language models
  - phrase/rule tables
  - reordering models,
- **Tuning:** the process of finding the optimal weights for this linear model
- Optimal weights are those which maximise translation performance on the tuning dataset

Other Functionalities

- Output a ranked list of the translation candidates
- Various types of information about how it came to its decision
- Binarisation of translation model for faster loading
- Evaluation of translations
- Alternative phrase scoring methods
- **Moses server:** provides an xml-rpc interface to the decoder
- **Web translation:** a set of scripts to be used to translate web pages
- **Analysis tools:** scripts to analyse and visualise Moses output, in comparison with a reference

References

- Moses homepage

  http://www.statmt.org/moses/
- Moses installation tutorials
  - http://organize-information.blogspot.in/2012/01/yet-another-moses-installation-guide.html
  - www.cfilt.iitb.ac.in/Moses-Tutorial.pdf
- How to run GIZA++?

  https://github.com/tetsuok/giza-pp/tree/master/GIZA%2B%2B-v2