SYSTEMS AND TOOLS FOR MACHINE TRANSLATION

GIZA++: Training of statistical translation models

Francisco Casacuberta  
fcn@iti.upv.es

Enrique Vidal  
evidal@iti.upv.es

June 4, 2007

Index

1 Introduction

2 Giza++
Statistical alignment models

\[
\hat{y} = \arg\max_y P_r(y \mid x) = \arg\max_y P_r(x \mid y) \cdot P_r(y)
\]

A "distorted (noisy) channel model"

Need: a target-language model + alignment and lexicon models
Alignments

- **Alignments**: (Brown et al. 90) $J = |x|, \ I = |y|$

$$a : \{1, \ldots, J\} \rightarrow \{0, \ldots, I\}$$

$a_j = 0 \Rightarrow j$ in $x$ is not aligned with any position in $y$.

- Set of possible alignments: $\mathcal{A}(x, y) = \{a : \{1, \ldots, J\} \rightarrow \{0, \ldots, I\}\}$

- The probability of translation $y$ to $x$ through an alignment $a$ is $\Pr(x, a \mid y)$

$$\Pr(x \mid y) = \Pr(J \mid y) \cdot \sum_{a \in \mathcal{A}(y, x)} \Pr(a \mid J, y) \cdot \Pr(x \mid a, J, y)$$

- **Length probability**: $\Pr(J \mid y) \approx n(J|I)$

Model 1

$$\Pr(x, a \mid J, y) = \prod_{j=1}^{J} \Pr(a_j \mid a_1^{j-1}, J, y) \cdot \Pr(x_j \mid x_1^{j-1}, a, J, y)$$

- $\Pr(a_j \mid a_1^{j-1}, J, y) \approx \frac{1}{(I+1)^J}$

- $\Pr(x_j \mid x_1^{j-1}, a, J, y) \approx l(x_j \mid y_{a_j})$  *statistical lexicon*

$$P_{M1}(x \mid y) = \frac{n(J|I)}{(I+1)^J} \prod_{j=1}^{J} \sum_{i=0}^{I} l(x_j \mid y_i)$$
Model 2

\[ Pr(x, a \mid J, y) = \prod_{j=1}^{J} Pr(a_j \mid a_{j-1}^i, J, y) \cdot Pr(x_j \mid x_{j-1}^i, a, J, y) \]

- \( Pr(a_j \mid a_{j-1}^i, J, y) \approx a(a_j \mid j, I) \)  
  *statistical alignments*
- \( Pr(x_j \mid x_{j-1}^i, a, J, y) \approx l(x_j \mid y_{a_j}) \)  
  *statistical lexicon*

\[ P_{M2}(x \mid y) = n(J \mid I) \cdot \prod_{j=1}^{J} \sum_{i=0}^{I} a(i \mid j, I) \cdot l(x_j \mid y_i) \]

Homogeneous HMM alignment

\[ Pr(x, a \mid J, y) = \prod_{j=1}^{J} Pr(a_j \mid a_{j-1}^i, J, y) \cdot Pr(x_j \mid x_{j-1}^i, a, J, y) \]

- \( Pr(a_j \mid a_{j-1}^i, J, y) \approx h(a_j \mid a_{j-1}, J, I) \)  
  *statistical alignments*
- \( Pr(x_j \mid x_{j-1}^i, a, J, y) \approx l(x_j \mid y_{a_j}) \)  
  *statistical lexicon*

\[ P_{HMM}(x \mid y) = n(J \mid I) \cdot \sum_{a} \prod_{j=1}^{J} h(a_j \mid a_{j-1}, J, I) \cdot l(x_j \mid y_{a_j}) \]
Optimal alignment with Model 2

Search for the “best” alignment from $A(x, y)$

$$\hat{P}_r(x \mid y) \approx \Pr(J \mid y) \cdot \max_{a \in A(y, x)} \Pr(x \mid J, y)$$

Using Model 2,

$$\hat{P}_{M2}(x \mid y) = n(J \mid I) \cdot \prod_{j=1}^{J} \max_{0 \leq i \leq I} [a(i \mid j, J, I) \cdot l(x_j \mid y_i)]$$

Viterbi algorithm $(x, y, l, a)$

For $j := 1$ until $J$

$$A[j] := \arg\max_{0 \leq i \leq I} a(i \mid j, J, I) \cdot l(x_j \mid y_i)$$

End-for

Return: $A$

The computational cost of this algorithm is $O(J \times I)$.

Model 3

$$\Pr(x \mid y) = \sum_a \Pr(x, a \mid y) = \sum_a \sum_{(\phi, \tau, \pi) \in F(x, a)} \Pr(\phi, \tau, \pi \mid y)$$

The probability for a tablet $\tau$ and a permutation $\pi$ is:

$$\Pr(\phi, \tau, \pi \mid y) = \Pr(\phi \mid y) \cdot \Pr(\tau \mid \phi, y) \cdot \Pr(\pi \mid \tau, \phi, y)$$

- $f(\phi_i \mid y_i)$: fertility probability
- $l(x_i \mid y_i)$: lexicon probability
- $d(j \mid i, J, I)$: distortion probability

$$P_{M3}(x \mid y) = \sum_{a_1=0}^{I} \cdots \sum_{a_J=0}^{I} \left( J - \phi_0 \right) p_0^{J-2\phi_0} p_1^J \phi_0^I \prod_{i=1}^{I} \phi_i! \cdot f(\phi_i \mid y_i) \prod_{j=1}^{J} l(x_j \mid y_{a_j}) \cdot d(j \mid a_j, J, I)$$
**Model 4**

The center of a target word $y_i$, $c(i) = \frac{\sum_k \pi_{i,k}}{\phi_i}$

- $f(\phi_i \mid y_i)$ \hspace{1cm} fertility probability
- $l(x \mid y_i)$ \hspace{1cm} lexicon probability
- $d_{=1}(j - c(i - 1) \mid \mathcal{C}_y(y_{i-1}), \mathcal{C}_x(x_j))$ \hspace{1cm} distortion probability for the first position in a tablet
- $d_{>1}(j - \pi_{i,k-1} \mid \mathcal{C}_x(x_j))$ \hspace{1cm} distortion probability for the rest of positions in a tablet

**Model 5**

For a target word $y_i$:

- For a target word $y_i$, number of vacant positions up to and including position $j$ just before $\tau_{i,k}$ is placed, $v(j, \tau_{i,k}^{i-1}, \tau_{i,k}^{i-1}) \equiv v_j$.

- $f(\phi_i \mid y_i)$ \hspace{1cm} fertility probability
- $l(x \mid y_i)$ \hspace{1cm} lexicon probability
- $d_{=1}(v_j \mid \mathcal{C}_x(x_j), v_{c(i-1)}, v_j - \phi_i + 1) \cdot (1 - \delta(v_j, v_{j-1}))$ \hspace{1cm} distortion probability for the first position in a tablet
- $d_{>1}(v_j - v_{\pi_{i,k-1}} \mid \mathcal{C}_x(x_j), v_j - v_{\pi_{i,k-1}} - \phi_i + k) \cdot (1 - \delta(v_j, v_{j-1}))$ \hspace{1cm} distortion probability for the rest of positions in a tablet
The training process

- Every model has a specific set of free parameters.
- For example for IBM Model 4: \( \theta = \left\{ \{l(x|y)\}, \{p_{\geq 1}(\Delta_j)\}, \{p_{>1}(\Delta_j)\}, \{p(\phi|x)\}, p_1 \right\} \)
- To train the model parameters \( \theta \): A maximum likelihood criterium, using a parallel training corpus consisting of \( S \) sentence pairs \( \{(x^{(n)}, y^{(n)}) : n = 1, \ldots, N\} \):
  \[
  \hat{\theta} = \arg\max_{\theta} \prod_{n=1}^{N} \sum_a p_{\theta}(x^{(n)}, a|y^{(n)}) .
  \]
- The training is carried out using the Expectation-Maximization (EM) algorithm.

Maximum likelihood by EM estimation.

- The counts in the reestimation are multiplied by \( Pr_{IM}(x, a | y) \) and are added for all possible alignment.
- No efficient method is computing these estimated counts.
- The estimated counts are approximate by:
  - Computing the (approximate) most probable alignment (Model 2)
  - Apply modifications: moves and swaps
  - Sum the estimated counts for all alignments whose probability is larger than the probability of the probable alignment times a given constant.
**Index**

1 Introduction » 2

○ 2 Giza++ » 14

---

**Tool-kits**

- The EGYPT Statistical Machine Translation Toolkit contains GiZA a training program that learns statistical translation models from bilingual corpora. GiZA is written C++ with the STL library (tested using gnu C++).
  
  http://www.clsp.jhu.edu/ws99/projects/mt/toolkit/

  (Developed in WS’99 Summer Workshop organized by the Center for Language and Speech Processing of the Johns Hopkins University)

- GiZA++ is an extension of the program GiZA
  
  Old version: http://www.fjoch.com/GIZA++.html
  
  Patched version: http://ling.umd.edu/~redpony/software/

- GiZA++ is used today to obtain word alignments in a bilingual corpus. These alignments are the basis to build phrase-based models, the state of the art in SMT.
**GIZA++ Package Programs**

- **GIZA++**: GIZA++ itself
- **plain2snt.out**: simple tool to transform plain text into GIZA format
- **plain2snt.out**: simple tool to transform GIZA format into plain text
- **trainGIZA++.sh**: Shell script to perform standard training given a corpus in GIZA format
- **mkcls**: Computes word classes in a monolingual corpus
- **snt2cooc**: Generates a cooccurrence file

**Input File Formats: vocabulary files**

Each entry is stored on one line as follows:

```
uniq_id1 string1 no_occurrences1
uniq_id2 string2 no_occurrences2
uniq_id3 string3 no_occurrences3
...
```

Here is an example:

<table>
<thead>
<tr>
<th>Source vocabulary file</th>
<th>Target vocabulary file</th>
</tr>
</thead>
<tbody>
<tr>
<td>176 desierto 8</td>
<td>731 eleccions 33</td>
</tr>
<tr>
<td>177 fueron 61</td>
<td>732 article 16</td>
</tr>
<tr>
<td>178 comprobar 6</td>
<td>733 nostra 23</td>
</tr>
<tr>
<td>179 instalaciones 15</td>
<td>734 alternativa 12</td>
</tr>
<tr>
<td>180 superado 4</td>
<td>735 contundent 3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

uniq.ids are sequential positive integer numbers.
0 is reserved for the special token NULL.
Input File Formats: bitext files

Each sentence pair is stored in three lines:

- The first line is the number of times of the sentence pair.
- The second line is the source sentence coded using the vocabulary file and
  the third is the target sentence in the same format.

Here’s a sample of 3 sentences:

```
... 1
119 109 120 20 121 122 7 123 124 29 72 125 126 57 22 127 128 129 10 11 12
63 29 3 129 9 130 131 8 132 133 55 78 134 135 60 124 136 137 66 9 13 12 14
1
130 131 132
138 139 140
1
114 133 134 12
123 8 141 142 14
...```

Input File Formats: dictionary File

- This is optional. The dictionary file is of the format:
  
  target_word_id source_word_id

- The list should be sorted by the target_word_id.

- If a dictionary is provided in the configuration file, GIZA++ will change the
  cooccurrence counting in the first iteration of model 1 to honor the so-called
  “Dictionary Constraint”: 

Config file for GIZA++

// general parameters:
// ---------------------
ml 101 (maximum sentence length)

// No. of iterations:
// -------------------
hmmiterations 5 (mh)
modelliterations 5 (number of iterations for Model 1)
model2iterations 0 (number of iterations for Model 2)
model3iterations 5 (number of iterations for Model 3)
model4iterations 5 (number of iterations for Model 4)
model5iterations 0 (number of iterations for Model 5)
model6iterations 0 (number of iterations for Model 6)

// parameter for various heuristics in GIZA++ for efficient training:
// --------------------------------------------------------------------
countincreasecutoff 1e-06 (Counts increment cutoff threshold)
countincreasecutoffal 1e-05 (Counts increment cutoff threshold for alignments in training of fertility models)
mincountincrease 1e-07 (minimal count increase)
peggedcutoff 0.03 (relative cutoff probability for alignment-centers in pegging)
probcutoff 1e-07 (Probability cutoff threshold for lexicon probabilities)
probsmooth 1e-07 (probability smoothing (floor) value)

// parameters for describing the type and amount of output:
// --------------------------------------------------------
compactalignmentformat 0 (0: detailed alignment format, 1: compact alignment format)
hmmdumpfrequency 0 (dump frequency of HMM)
log 0 (0: no logfile; 1: logfile)
model1dumpfrequency 0 (dump frequency of Model 1)
model2dumpfrequency 0 (dump frequency of Model 2)
model345dumpfrequency 0 (dump frequency of Model 3/4/5)
nbestalignments 0 (for printing the n best alignments)
nodumps 0 (1: do not write any files)
onlyaldumps 0 (1: do not write any files)
outputpath (output path)
transferdumpfrequency 0 (output: dump of transfer from Model 2 to 3)
verbose 0 (0: not verbose; 1: verbose)
verbo sesentence -10

IARFID-UPV June 4, 2007
// (negative: no output))

// parameters describing input files:
// ----------------------------------
// c  (training corpus file name)
// d  (dictionary file name)
// s  (source vocabulary file name)
// t  (target vocabulary file name)
// tc (test corpus file name)

// smoothing parameters:
// --------------------
emalsmooth 0.2
// (f-b-trn: smoothing factor for HMM alignment model (can be ignored by -emSmoothHMM))
model23smoothingfactor 0
// (smoothing parameter for IBM-2/3 (interpolation with constant))
model4smoothingfactor 0.2
// (smoothing parameter for alignment probabilities in Model 4)
model5smoothingfactor 0.1
// (smoothing parameter for distortion probabilities in Model 5)
nsmooth 64
// (smoothing for fertility parameters (good value: 64):

IARFID-UPV June 4, 2007 SMT -1: 22

// weight for wordlength-dependent fertility parameters
nsmoothgeneral 0
//{(smoothing for fertility parameters (default: 0): weight for word-independent fertility parameters)

// parameters modifying the models:
// ------------------------------
compactadtable 1
// (1: only 3-dimensional alignment table for IBM-2 and IBM-3)
deficientdistortionforemptyword 0
// (0: IBM-3/IBM-4 as described in (Brown et al. 1993);
// 1: distortion model of empty word is deficient;
// 2: distortion model of empty word is deficient (differently);
// setting this parameter also helps to avoid that during IBM-3 training too many words are aligned with the empty word)
depm4 76
// (d_{ 1}: &1:l, &2:m, &4:F, &8:E, d_{>1}&16:l, &32:m, &64:F, &128:E)
depm5 68
// (d_{ 1}: &1:l, &2:m, &4:F, &8:E, d_{>1}&16:l, &32:m, &64:F, &128:E)
emalignmentdependencies 2
// (lextrain: dependencies in the HMM alignment model.
// 1: sentence length;
// 2: previous class;
// 4: previous position;    

IARFID-UPV June 4, 2007 SMT-1: 23
Output file formats: probability tables

1. Translation table (*t.*

   prob_table.t1.n = t table after n iterations of Model1 training
   prob_table.t2.n = t table after n iterations of Model2 training
   prob_table.t2to3 = t table after transferring Model2 to Model3
   prob_table.t3.n = t table after n iterations of Model3 training
   prob_table.t4.n = t table after n iterations of Model4 training

   Each line is of the following format:

   \[ s_{id} \ t_{id} \ P(t_{id}|s_{id}) \]

2. Fertility table (*.n3.*

   Each line in this file is of the following format:

   \[ source_token_id \ p0 \ p1 \ p2 \ldots \ pn \]

   where \( p0 \) is the probability that the source token has zero fertility; \( p1 \), fertility one, ...., and \( n \) is the maximum possible fertility as defined in the program.
Output file formats: probability tables

3. Probability of inserting a null after a source word (*.p0*)
   Contains only one line with the probability of not inserting a NULL token.

4. Alignment tables (*.a.*)
   The format of each line is as follows:
   
   \[ i \ j \ l \ m \ P(i \mid j, l, m) \]

   where:
   
   \( i \) = position in source sentence
   \( j \) = position in target sentence
   \( l \) = length of source sentence
   \( m \) = length of target sentence

   and \( P(i \mid j, l, m) \) is the probability that a source word in position \( i \) is moved to position \( j \) in a pair of sentences of length \( l \) and \( m \).

5. Distortion table (*.d3.*)
   The format is similar to the alignment tables but the position of \( i \) and \( j \) are switched:
   
   \[ j \ i \ l \ m \ P(j \mid i, l, m) \]

6. Distortion table for IBM-4 (*.d4.*)
7. Distortion table for IBM-5 (*.d5.*)
8. Alignment probability table for HMM alignment mode (*.A3.*)
9. Perplexity File (*.perp*)
10. Revised vocabulary files (*.src.vcb, *.trg.vcb)
11. Final parameter file: (*.gizacfg*)