Machine Translation
Decode

Haitang Hu
hthu@cs.jhu.edu

March 5, 2015

This greedy decoder is based on the work of Philippe Langlais et al.[1]

1 Intuition

The intuition behind the greedy decoder is a simple hill climb process: we start from a **seed** of translation, as our initial start point. Then, we define a set of **neighbors**, as possible options, and we use our **score** function to choose the best translation. We keep doing iteration until we find a local optimal(namely the hill), and we are done.
Note that this method does not guarantee the best translation, since it is not search all the possible space, but practically this works pretty well.

2 Components

2.1 Seed

"In our case, we select the segmentation which involves the minimum number of source phrases belonging to the translation model that cover maximally the source sentence.[1]"
This could gain us a better search space for translation options, since the minimum coverage could give us more freedom to combine **split**, **swap** tranformations, so that we have more possbility to find a better local optimal(jump out a local optimal). But this is currently unimplemented, instead, we are currently using a *stack decode* algorithm as our initialization, which simply consider the best probability on the sum of **translation model** score and **language model** score.

2.2 Score

Our score function is composed by:

- Translation Model Score
  Just simply the phrase translation log-probability.
- Language Model Score
  A trigram language model log-probability.
- Distortion Penalization
  The score of the translation position difference j and j'.

Combine above, our score function[2] can be written as:

$$Score(f, e) = \log p_{lm}(f) + \sum_{i}^{f} \log p_{tm}(f_i|e_i) + \sum_{j}^{f} d(a_j, a_{j-1})$$

Where

$$d(a_j, a_{j-1}) = \alpha^{|a_j - a_{j-1} - 1|}$$

The value of α will be discussed in later section.

2.3 Neighbors

We defines 6 kinds of operations, to form a neighbor sets:

a. **Move**
If the adjacent source are translated to be far from each other, we try to move this two phrase closer.

b. **Swap**
Swap the translation for any two pair of phrase. Note that this actually includes **Move**, while the standard version of **Swap** is only applied on adjacent source pairs.

c. **Split**
Split the current source phrase into 2 phrases.

d. **Merge**
Merge two adjacent source phrase into one.

e. **Replace** & **Bi-replace**
Replace the current phrase with other translation, or replace adjacent source pairs simultaneously.

These operations allow us to explore more of the search space.

3 Evaluation

3.1 Phrase Options

Since we are allowing 6 operations above, we want to explore more search space. It is natural that we put more phrase translation options, so that we could form different language model and settings. After several experiments, it shows that from 1 phrase options to 5, the score is raised most, after 5 the score is imporved slowly while taking a long time to train.

Phrase Options	Score	Time
1	-1352.89	$8.7s$
5	-1308.87	$35.8s$
10	-1307.65	$63.4s$

Table 1: Phrase Options

3.2 Distortion Base

The distortion base decides how much will we penalize for our distortion. Since we want more search space, so we should not penalize too heavy. $\alpha = 0.05$ is a decent value to get a better score.

3.3 Stack Size

Since we are only using *Stack Decode* for initialization, the stack size affects a little on the final score, only around 0.5 are improved while increasing stack size from $1 \rightarrow 10$. So for efficiency we just set the stack size to be equal to 1.

References

[1] Philippe Langlais, Alexandre Patry and Fabrizio Gotti. *A Greedy Decoder for Phrase-Based Statistical Machine Translation*, 2007, `http://www.iro.umontreal.ca/~felipe/bib2webV0.81/cv/papers/paper-tmi-2007.pdf`

[2] Philipp Koehn, Franz Josef Och, Daniel Marcu. *Statistical Phrase-Based Translation*, 2003, `http://www.isi.edu/~marcu/papers/phrases-hlt2003.pdf`

[3] Philipp Koehn. *Decoding*, 2015, `http://mt-class.org/jhu/slides/lecture-decoding.pdf`