Automatically Derived Multi-level Word Classes for MT

Jacob Devlin

May 14, 2008

1 Introduction

Grouping words into semantic classes is a well studied task in NLP and MT research. The goal of this project was to automatically derive a multi-level hierarchy of bilingual word pair classes, for use in generalized rule extraction. This means that we have a set of k bilingual word pairs in our source and target languages $\langle s_1, t_1 \rangle, \langle s_2, t_2 \rangle, ...\langle s_k, t_k \rangle$, where each $s_j \rightarrow t_j$ has been extracted as a translation rule from our aligned training data. Note that each word *pair* must be unique, but there is no limit as to how many pairs any given source or target word can appear in. We want to group these word pairs into an n level class hierarchy, where there are m_j classes at the j^{th} level, and $m_j < m_{j+1}$.

If $k = 100000$, $n = 3$, $m_1 = 1000$, $m_2 = 100$, $m_3 = 10$, we would first group the 100000 word pairs into 1000 classes, then group these 1000 classes into 100 classes, then group these 100 classes into 10 classes, all based on semantic similarity. These classes would then be used as the basis for Chiang style generalized rule creation (the standard term is "hierarchical" rule creation, but we will use "generalized" to differentiate it from the hierarchical class structure). The implementation details are described in section 4, but we will first give an overview of our MT system, and then provide a constructed example, which should help the reader understand the intuition behind this project.

2 Overview of the MT system

We use state of the art hierarchcial MT system based off of David Chiang's Hiero [2]. The MT system can be divded into three major steps: Alignment of parallel training data, rule creation/generalization, and decoding. An overview of each step is provided below.

2.1 Alignment of parallel training data

Given a source/target sentence pair (s_1^J, t_1^I), we denote an alignment between these two sentence as a_1^J, where the j^{th} source word is aligned to (i.e. translates to) the ${a_j}^{th}$ target word. The special alignment $a_j = 0$ means that the source word at index j does not align to any target words. We use GIZA++ with IBM models 1-4 [1] and the HMM model [6] to align the parallel training corpus. Under this alignment model, we have a one-to-many mapping between the source words and target words of each training sentence pair. In order to get a more desirable many-to-many mapping, we first perform a 'backward" alignment from the target language to the source language, which is done by running GIZA++ again with the the source language and target language switched. These alignments are converted into two dimensional matrices and combined using a function that is "between" simple set union and set intersection [4].

The resulting alignment A is a many-to-many mapping between the source and target sentences, where any number of words on both the source and target side may be unaligned. The aligned parallel corpus, represented as a set of K triples (s_k, t_k, A_k), is directly used as the input to the rule extraction process.

2.2 Extraction of translation rules

We use a hierarchical rule extraction process as described in Chiang et al. (2005)[2]. As a first step, all phrase translations are extracted from each training triple $(\mathbf{s_k}, \mathbf{t_k}, \mathbf{A_k})$. We extract each possible phrase translation $\langle \hat{s}, \hat{t} \rangle$ such that no words in $\hat{s}$ are aligned to any other target words (other than the ones in $\hat{t}$), and no words in $\hat{t}$ are aligned to any other source source words (other than the ones in $\hat{t}$). For the purpose of this paper, it is important to note is that unaligned words may appear on the edges or in the center of these rules.

We convert the phrase rules to hierarchical rules by first converting each phrase pair $\langle \hat{s}, \hat{t} \rangle$ to a CFG rule $X \rightarrow \langle \hat{s}, \hat{t} \rangle$, and then subtracting each phrase pair $\langle \hat{s}, \hat{t} \rangle$ from each rule in the form $X \rightarrow \langle \gamma_1 \hat{s} \gamma_2, \alpha_1 \hat{t} \alpha_2 \rangle$ to create the hierarchical rule $X \rightarrow \langle \gamma_1 X \gamma_2, \alpha_1 X \alpha_2 \rangle$. In order to reduce the number of possible rules that can be created, we apply restrictions to the size of phrases that can be extracted and subtracted, and we also apply filtering based on the current test set. Joint and marginal counts are then summed over all the rules.

2.3 Decoding

For decoding, we use a hierarchical model that closely follows Chiang's Hiero, as described in Chiang (2005). In this model, we create a shared forest of weighted translation rules for the sentence being decoded. Since hierarchical rules form a CFG, the test sentence is parsed on the source side using a chart parser similar to CKY thereby creating the shared forest of target derivations. We use a log linear model to score each translation

rule:

$$\begin{aligned} w(X \rightarrow \langle \gamma, \alpha \rangle) &= \prod_i \phi_i(X \rightarrow \langle \gamma, \alpha \rangle)^{\delta_i} \\ \log(w(X \rightarrow \langle \gamma, \alpha \rangle)) &= \sum_i \delta_i \phi_i(X \rightarrow \langle \gamma, \alpha \rangle) \end{aligned}$$

The score of a derivation (i.e. a full sentence translation) is thus the sum of the scores of each rule used in that derivation. Each rule can have an arbitrary number of features, and the feature weights are generally determined automatically through optimization towards an evaluation metric such as BLEU [5]. Basic features include:

- Language model score
- $P(\gamma|\alpha)$ [3]
- $P(\alpha|\gamma)$
- Lexical probability
- Word penalty, i.e. a feature equal to the number of words in α

3 An example

We will now present a constructed example in order to explain the intuition behind this project. Imagine that the following rules were extracted from a set of aligned Spanish to English training data. The X on the left hand side represents a non-terminal, since all rules in the hierarchical model are in CFG form. Note that in the standard generalized rule creation process, there is only one non-terminal:

$$X \rightarrow \langle \text{el coche azul}, \text{the blue car} \rangle \quad (1)$$
$$X \rightarrow \langle \text{azul}, \text{blue} \rangle \quad (2)$$
$$X \rightarrow \langle \text{rojo}, \text{red} \rangle \quad (3)$$
$$X \rightarrow \langle \text{movido rapido}, \text{moved fast} \rangle \quad (4)$$
$$X \rightarrow \langle \text{el coche}, \text{the car} \rangle \quad (5)$$

We would then create the following generalized rule from (1) and (2):

$$X \rightarrow \langle \text{el coche } X, \text{the } X \text{ car} \rangle \quad (6)$$

Now imagine that we were decoding the following sentence "el coche movido rapido". The decoder could use rules (6) and (4) in the following manner:

$$\begin{aligned} &X_0 \rightarrow \langle \text{el coche } X_1, \text{the } X_1 \text{ car} \rangle \quad (7) \\ &X_1 \rightarrow \langle \text{movido rapido}, \text{moved fast} \rangle \end{aligned}$$

Which would result in the translation "the moved fast car". Of course, the language model would disfavor this translation, but the important thing to note is that the translation

model does not "remember" that the non-terminal in (6) originally came from (2). Thus, the model does not take into account the fact that rules (4) and (2) are semantically dissimilar while (3) and (2) are semantically similar, so it may not be as reliable to use (4) with (6) when translating "el coche movido rapido" as it would be to use (3) with (6) when translating "el coche rojo".

However, instead of having only one non-terminal, we could cluster the translations into semantic classes before generalization:

$$COLOR \rightarrow \langle \text{azul}, \text{blue} \rangle \tag{8}$$
$$COLOR \rightarrow \langle \text{rojo}, \text{red} \rangle \tag{9}$$

Then, instead of creating the generalized rule (6), we would create the following generalized rule:

$$X \rightarrow \langle \text{el coche } COLOR, \text{ the } COLOR \text{ car} \rangle \tag{10}$$
$$\tag{11}$$

Which would disallow translation (7). Alternatively, we could create a hierarchy of increasingly large semantic classes:

$$COLOR \rightarrow \langle \text{azul}, \text{blue} \rangle \tag{12}$$
$$ADJ \rightarrow \langle \text{azul}, \text{blue} \rangle \tag{13}$$
$$X \rightarrow \langle \text{azul}, \text{blue} \rangle \tag{14}$$

Which would then create rules all three of the following rules:

$$X \rightarrow \langle \text{el coche } COLOR, \text{the } COLOR \text{ car} \rangle \tag{15}$$
$$X \rightarrow \langle \text{el coche } ADJ, \text{the } ADJ \text{ car} \rangle \tag{16}$$
$$X \rightarrow \langle \text{el coche } X, \text{the } X \text{ car} \rangle \tag{17}$$

We would expect that rule (15) would "naturally" have a higher translation probability than (17), since for all non-generalized rules in the form "el coche $w \rightarrow$ the w car", we would expect that

$$\frac{Count(w \text{ is a } COLOR)}{Count(\text{words in } COLOR)} > \frac{Count(w \text{ is any word})}{Count(\text{total words})}$$

Thus, when translating "el coche rojo" the model would be encouraged to use this generalization, while when translating "el coche movido rapido", it would not be. In addition to this natural probability difference, we explicitly added an optimizable feature which denoted what hierarchy level a generalized rule came from.

In this project, we followed this multi-level semantic clustering approach for creating generalized rules. The major way that the above example differs from the actual implementation is that we did not explicitly choose classes such as "color" and "adjective", instead we chose just the *number* of classes at each level and derived them automatically based on a semantic similarity score. The implementation details are described in the next section.

4 Deriving the word classes

In our experiments we chose to use a 4-level class hierarchy with 1000, 100, 10, and 1 classes at each level. The basis for our clustering algorithm was a semantic similarity score between each bilingual word pair, which we will denote as $BiSim(\langle s_1, t_1\rangle, \langle s_2, t_2\rangle)$. This function is defined as:

$$BiSim(\langle s_1, t_1\rangle, \langle s_2, t_2\rangle) = p(s_1|t_1) * p(s_2|t_2) * MonoSim(s_1, s_2) * MonoSim(t_1, t_2)$$

The probability $p(s|t)$ is defined in the normal way for our model:

$$p(s|t) = \frac{Count(s,t)}{\sum_{s'} Count(s',t)}$$

where $Count(s,t)$ is the number of times that the rule "$s \rightarrow t$" is extracted from the aligned training data.

The monolingual similarity measure $MonoSim(w_1, w_2)$ is derived from a language model in that language. We defined this function as:

$$MonoSim(w_1, w_2) = \sum_S p(w_1|S)p(w_2|S)p(S)$$

where S is a state in the language model. In other words, S is a set of word $w_1, w_2, ..., w_n$, and then $p(w|S) = p(w|w_1, w_2, ..., w_n)$ is just the normal language model probability:

$$P(w|S) = p(w|w_1, w_2, ..., w_n) = \frac{Count(w_1, w_2, ..., w_n, w)}{\sum_{w'} Count(w_1, w_2, ..., w_n, w')}$$

and

$$P(S) = \frac{\sum_{w'} Count(w_1, w_2, ..., w_n, w')}{\text{total words in monolingual corpus}}$$

Note that no smoothing is performed and we actually prune the monolingual distance tables so the bilingual similarity table is fairly sparse, i.e. $BiSim(\langle s_1, t_1\rangle, \langle s_2, t_2\rangle) = 0$ for most word pairs. We extract each word pair $\langle s, t\rangle$ from our set of non-generalized translation rules. Only one-word to one-word translations are extracted, since $BiSim()$ is only defined for single word pairs.

In order to cluster the word pairs into the first level of classes we used a simple greedy algorithm based around "centroids", which are word pairs choosen to represent each class. If the number of classes a the lowest clustering level is m_1, then we want to pick m_1 centroids from the k word pairs. max_sim is a predefined value which designates the maximum similarity that any two centroids may have. The following psuedocode describes the algorithm:

$C \leftarrow$ List of centroids, initially empty
$L \leftarrow$ Sort the word pairs by the number of other word pairs that they have a non-zeros similarity with.

while L is not empty and $size(C) < m_1$
$w \leftarrow$ Pop the top of L.
$m \leftarrow \max_{c \in C} BiSim(w, c)$, i.e. the max similarity between w and any existing centroid
if $m \leq max\_sim$
push w onto C

After choosing the centroids, we assign the remaining word pairs to a class by clustering them with the centroid that they have the highest similarity with. After the initial clusters are chosen, we performed iterative clustering by updating the centroids of each class and re-clustering. This update was performed by choosing a new centroid w in each class c such that $w = \text{argmax}_{w \in c} \sum_{w' \neq w \in C} BiSim(w, w')$.

After the initial clustering was performed, we performed "super clustering" with another simple greedy algorithm based on a "super cluster similarity measure". In the first level of super clustering we simply treated each word class as a super cluster with one word class in it. The following psuedocode describes the algorithm:

$S \leftarrow$ List of super clusters, sorted by total number of words
while $size(S) < m_n$
$s \leftarrow$ Pop the top of S (s will be *smallest* super cluster)
$r \leftarrow argmax_{r \in S} SuperClusterSim(s, r)$, i.e. the super cluster with the highest similary to s
Remove r from S
$q \leftarrow s \cup r$ (s and r are just sets of classes).
Reinsert q into S, keeping S sorted by size

We tried different functions for $SuperClusterSim(s, r)$, but the one we generally used was to take the mean similary between all of the centroids of s and the centroids of r.

Remember that each word pair $w = \langle s, t \rangle$ was actually a translation rule $R = X : s \rightarrow t$ with associated translation probabilities. If we have n levels of word clustering, each w will appear in exactly n super clusters (one at each hierarchy level), which we can denote as $c_1, c_2, ..., c_n$. In this context c_j represents a unique id number given to that class. Thus, after clustering is complete, we add n new translation rules for each w, $c_1 : s \rightarrow t$, $c_2 : s \rightarrow t$, ... , $c_n : s \rightarrow t$. The probabilities associated with each of these new rules are the same as the original probabilies of R. As described in the example, the higher probability rules come $after$ generalized rule extraction. After all of the new rules are added, generalized rule creation is performed as described in Section 3, Section 2, and Chiang (2005). Hierarchical decoding performed as described in Section 2 and Chiang (2005).

5 Results

The results of decoding with different class hierarchies are presented in Table 1. Although all configurations scored better than the baseline (which had no generalization), the hierarchical clustering setup (4 cluster levels, 1000, 100, 10, 1 classes in each level) did not perform better than standard single-word rule generalization (1 class containing all words).

Table 1: TER/BLEU scores with different clustering hierarchies

Cluster Levels Used	**TER**	**BLEU**
Baseline (No Classes)	56.38	35.35
1000	55.91	36.15
1000, 100, 10 ,1	55.17	36.99
1	55.20	37.16

We performed some analysis on the rules used in the different experiment configurations, and the results are presented in Table 2. In general, we believe that an increase in average source phrase length (the number of terminals in the source side of the translation rule) will increase translation quality, all other things being equal. The reason behind this should be fairly obvious; for example translating a three word phrase as a single unit is likely going to be more accurate than translating each word sepeartely and concatenating the translations. In Table 2, we can see that although the hierarchcial configuration uses significantly more generalized rules than the single class configuration, the mean phrase length is roughly the same.

Another thing to note is that the hierarchical class model does not introduce any "new" rules, meaning that every translation possible under under hierarchical class configuration is also possible under the single class configuration. The hierarchical configuration simply encourages the decoder to use *semantically similar* generalized rule substitutions (e.g., substitute a color in "el coche $X \rightarrow$ the X car" rather than any arbitrary phrase). However, if errors like the one presented in the constructed example do not really happen to any significant degree, then the hierarchical configuration may not be beneficial.

Table 2: Analysis of rules used in decoding

Cluster Levels Used	**Percent of Phrases Using Gen. Rules**	**Mean Source Phrase Length**
Baseline (No Classes)	0.00%	2.176
1000	16.64%	2.211
1000, 100, 10 ,1	30.86%	2.294
1	22.32%	2.289

We can conclude that creating generalized rules based on a hierarchy of semantic word classes does not increase the quality of MT output compared to the standard rule generalization method. One possible reason is that ungrammatical CFG parses, as shown in example (7), do not occur when only single word generalizations are allowed. In future work, rather than only using bilingual word pairs, we can group bilingual *phrases* into semantic classes and generalize in the same way as described above. Additionally, it is also possible that rule generalizations describe a *syntactic* relationship, while the bilingual word classes described in this experiment are grouped by a *semantic* similarity measure. Clustering based on syntactic relationships or a syntactic/semantic combination may be more beneficial.

References

[1] Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematic of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1994.

[2] David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. The hiero machine translation system: Extensions, evaluation, and analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 779–786, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.

[3] Philipp Koehn, Franz Josef Och, and Daniel. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, page 127133, 2003.

[4] Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, page 417449, 2004.

[5] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[6] Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics*, pages 836–841, Morristown, NJ, USA, 1996. Association for Computational Linguistics.