Hierarchical Rule Generalization with
Automatically Derived Multi-level Word Classes for MT

Abstract
Hierarchical rule generalization is a central aspect of Chiang’s hierarchical decoding model. “Hierarchical substitution” is the process by which generalized rules are used in decoding. We believe that hierarchical substitutions of semantically similar words are more likely to be grammatically well-formed than hierarchical substitutions of arbitrary words. In this paper, we extend the rule generalization process to use a hierarchy of semantic bilingual word classes, which are derived automatically using a bilingual semantic similarity score. The result is that during decoding, hierarchical substitution of semantically specific classes (e.g., the class of colors) will have a higher probability than hierarchical substitution of more general classes (e.g., the class containing all words). Although we achieved a substantial increase in the number of generalized rules used during decoding, more investigation is required to determine how this will improve MT scores (BLEU) over those generated by the standard rule generalization process.

1 Introduction
This paper presents a new process of automatic derivation of a multi-level hierarchy of bilingual word pair classes, for use in generalized rule extraction. The use of bilingual semantic word classes in translation models has been explored previously, most notably in Och and Ney’s alignment template approach (2004). The clustering algorithm used in the alignment-template approach, first described in (Och, 1999), uses a maximum-likelihood approach for creating the bilingual word classes. Other methods of bilingual word clustering have also been explored. Ng (2001) used a spectral clustering algorithm to create bilingual word classes, and Zhao et al. (2005) used a variant of Ng’s algorithm to improve translation quality in an HMM-based system.

Semantic word classes have also been integrated directly into translation models developed by several researchers. Koehn’s factored translation model modifies the decoding process so that each word is not only a token, but a vector of factors that represent different levels of annotation (Koehn et al., 2007). These factors can include morphology, part-of-speech, and semantic word classes (bilingual or monolingual). This factored translation model has been added into Koehn’s Moses SMT Decoder, which natively supports the use of semantic word classes.

While our approach is similar to the Och and Ney’s use of bilingual word classes in general principle, our implementation is substantially different. The approach requires a hierarchical decoding model, and in fact can be thought of as a “natural extension” to the hierarchical rule extraction process. More importantly, we use a multi-level hierarchy of bilingual word classes, allowing us to benefit from different levels of semantic similarity. In addition, the previously discussed bilingual clustering algorithms are designed to basically create two sets of monolingual semantic classes, where each source class has exactly one corresponding target class. We instead cluster bilingual word pairs, meaning that any given source or target word can appear in any number of classes. This is very useful for our purposes, although it may not be desirable for other MT tasks.

To implement our method, we first performed a standard phrasal rule extraction (Och and Ney, 2004), and extracted a “dictionary” of word-to-word translations (e.g., ⟨coche, car⟩, ⟨hombre, man⟩, ⟨hombre, human⟩).1 We then computed a numerical similarity score between every two bilingual word pairs. Next, we performed k-means clustering of the bilingual word pairs into a predefined number of word classes (Hartigan, 1975). The classes were then grouped into a predefined number of “super-clusters,” and the super-clustering process is re-

1 Examples are given in Spanish for ease of understanding, but all experiments were performed in Chinese to English.
peated. The end result is a multi-level hierarchy of word classes.

In our experiments, we used a class hierarchy of 1000 classes → 100 classes → 10 classes → 1 class. These classes were then naturally integrated into the hierarchical generalized rule creation process by performing a simple lookup during the non-terminal replacement step. When a phrasal rule was transformed into a generalized rule by replacing part of the source and target side with non-terminals, we checked to see if the words being replaced were in at least one of our bilingual word classes. If they were, we replaced them with a non-terminal equal to the classes they were in, rather than the “generic” word classes. If the words that were replaced were in any number of our bilingual word classes. If they were, we replaced them with a non-terminal equal to the classes they were in, rather than the “generic” non-terminal. For example, if we had a phrasal rule “el coche azul, the blue car” and the word pair ⟨azul, blue⟩ was in class C1, then we would create the rule “X → ⟨el coche C1, the C1 car⟩” rather than “X → ⟨el coche X, the X car⟩.” We would also add the hierarchical rule “C1 → ⟨azul, blue⟩.”

Note that since we used a multi-level hierarchy of word classes, we would actually create these rules for every class that the word pair was in. This means that since we generally included a word class that contains all word pairs, the rule with the “generic” non-terminal would be added anyway.

Decoding can be thought of as the opposite of rule generalization, where source and target phrases are substituted back into the non-terminals of generalized rules. During normal rule generalization, there is only one non-terminal, so there are no restrictions on which phrases can be substituted into which other phrases, other than the fact that the source side must match the source input sentence. This is not necessarily a bad thing, but we expect it is beneficial to “encourage” hierarchical substitutions of words that are semantically related to the words that the non-terminals replaced during rule generalization. This encouragement is done both implicitly by the fact that \( P(s|t) \) and \( P(t|s) \) values will be higher for generalized rules that contain more specific semantic classes, and explicitly through feature weights in our log-linear model.

The next section describes the MT system used for this study, including the alignment, rule extraction, and decoding. Section 3 presents an example that illustrates the intuition behind the approach. The implementation details are then presented in section 4. Finally, results are presented along with a discussion. We conclude that it would be beneficial to explore more options in our clustering process, including clustering multi-word phrases and using syntactic relationships for clustering.

2 Overview of the MT system

We use a state-of-the-art hierarchical MT system based off of David Chiang’s Hiero (Chiang et al., 2005). The MT system can be divided 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 \( (s_1^j, t_1^j) \), we denote an alignment between these two sentences as \( a_1^j \), where the \( j \)th source word is aligned to \( (i.e., \text{translates to}) \) the \( j \)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 (Brown et al., 1994) and the HMM model (Vogel et al., 1996) 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 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 (Och and Ney, 2004).

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 used a hierarchical rule extraction process as described in Chiang et al. (2005). As a first step, all phrase translations were extracted from each training triple \( (s_k, t_k, A_k) \). We extracted each possible phrase translation \( (\hat{s}, \hat{t}) \) such that no words in \( \hat{s} \) were
aligned to any other target words (other than the ones in \( \hat{t} \)), and no words in \( \hat{t} \) were 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 that unaligned words may appear on the edges or in the center of these rules.

We converted 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 applied restrictions to the size of phrases that could be extracted and subtracted, and we also applied filtering based on the current test set. Joint and marginal counts were then summed over all the rules.

### 2.3 Decoding

For decoding, we used a hierarchical model that closely follows Chiang’s Hiero. In this model, we created a shared forest of weighted translation rules for the sentence being decoded. Since hierarchical rules are in a CFG format, the test sentence was parsed on the source side using a chart parser similar to CKY thereby creating the shared forest of target derivations. We used a log linear model to score each translation rule:

\[
w(X \rightarrow \langle s, t \rangle) = \prod_i \phi_i(X \rightarrow \langle s, t \rangle)^{\delta_i}
\]

\[
\log(w(X \rightarrow \langle s, t \rangle)) = \sum_i \delta_i \phi_i(X \rightarrow \langle s, t \rangle)
\]

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 (Papineni et al., 2001). Basic features include:

- Language model score
- \( P(s|t) \) (Koehn et al., 2003)
- \( P(t|s) \)
- Lexical probability
- Word penalty, i.e., a feature equal to the number of words in \( t \)

### 3 An example

We will now present a human-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:

\[
X \rightarrow \langle \text{el coche azul, the blue car} \rangle \quad (1)
\]

\[
X \rightarrow \langle \text{azul, blue} \rangle \quad (2)
\]

\[
X \rightarrow \langle \text{rojo, red} \rangle \quad (3)
\]

\[
X \rightarrow \langle \text{movido rapido, moved fast} \rangle \quad (4)
\]

\[
X \rightarrow \langle \text{el coche, the car} \rangle \quad (5)
\]

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:

\[
X_0 \rightarrow \langle \text{el coche } X_1, \text{the } X_1 \text{ car} \rangle \quad (7)
\]

\[
X_1 \rightarrow \langle \text{movido rapido, moved fast} \rangle
\]

That 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:

\[
\text{COL} \rightarrow \langle \text{azul, blue} \rangle \quad (8)
\]

\[
\text{COL} \rightarrow \langle \text{rojo, red} \rangle \quad (9)
\]

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

\[
X \rightarrow \langle \text{el coche COL, the COL car} \rangle \quad (10)
\]
That would disallow translation (7). Alternatively, we could create a hierarchy of increasingly large semantic classes:

\[
\begin{align*}
COL & \rightarrow \{\text{azul, blue}\} \quad (11) \\
ADJ & \rightarrow \{\text{azul, blue}\} \quad (12) \\
X & \rightarrow \{\text{azul, blue}\} \quad (13)
\end{align*}
\]

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

\[
\begin{align*}
X & \rightarrow \{\text{el coche COL, the } COL \text{ car}\} \quad (14) \\
X & \rightarrow \{\text{el coche ADJ, the } ADJ \text{ car}\} \quad (15) \\
X & \rightarrow \{\text{el coche } X, \text{ the } X \text{ car}\} \quad (16)
\end{align*}
\]

We would expect that rule (14) would “naturally” have a higher translation probability than (16), since for all non-generalized rules in the form “el coche \(w\) → the \(w\) car,” we would expect that

\[
\frac{C(w \text{ is a } COLOR)}{C(\text{words in } COLOR)} > \frac{C(w \text{ is any word})}{C(\text{total words})}
\]

Where \(C(x)\) is the count of \(x\) in the training data.

Thus, when translating “el coche rojo” the model would be encouraged to use this generalization, whereas 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.

We followed this hierarchical 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 is a semantic similarity score between each bilingual word pair, which we will denote as \(BS(\langle s_1, t_1 \rangle, \langle s_2, t_2 \rangle)\) (for “bilingual similarity”). The function is defined as:

\[
BS(\langle s_1, t_1 \rangle, \langle s_2, t_2 \rangle) = p(s_1 | t_1) * p(s_2 | t_2) * MS(s_1, s_2) * MS(t_1, t_2)
\]

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

\[
p(s | t) = \frac{C(s, t)}{\sum_{s'} C(s', t)}
\]

Where \(C(s, t)\) is the number of times that the rule \(\langle s, t \rangle\) is extracted from the aligned training data.

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

\[
MS(w_1, w_2) = \sum_s p(w_1 | S)p(w_2 | S)p(S)
\]

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 \(p(w | S) = p(w_1 | w_2, ..., w_n)\) is just the normal language model probability:

\[
P(w | S) = \frac{C(w_1, w_2, ..., w_n, w)}{\sum_{w'} C(w_1, w_2, ..., w_n, w')}
\]

and

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

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

In order to cluster the word pairs into the first level of classes we used a generic k-means clustering algorithm (Hartigan, 1975). If the number of classes a the lowest clustering level was \(m_1\), then we want to picked \(m_1\) centroids from the \(k\) word pairs. \(\text{max.sim}\) is a predefined value that designates the maximum similarity that any two centroids may have. The following pseudocode 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.  

\textbf{While} \( L \) is not empty and \( \text{size}(C) < m_1 \)  
\quad \( w \leftarrow \) Pop the top of \( L \).  
\quad \( m \leftarrow \max_{c \in C} BS(w, c) \), i.e., the max similarity between \( w \) and any existing centroid  
\quad \textbf{If} \( m \leq \max_{\text{sim}} \)  
\qquad \text{push} \( w \) onto \( C \)  

After choosing the centroids, we assigned the remaining word pairs to a class by clustering them with the centroid with which they exhibit the highest similarity with. After the initial clusters were 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 \( w = \arg \max_{w \in C} \sum_{w' \neq w \in C} BS(w, w') \).

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 pseudocode describes the algorithm:

\( S \leftarrow \) List of super clusters, sorted by total number of words  
\textbf{While} \( \text{size}(S) < m_n \)  
\quad \( s \leftarrow \) Pop the top of \( S \) (\( s \) will be \textit{smallest} super cluster)  
\quad \( r \leftarrow \arg \max_{r \in S} S\text{uperClusterSim}(s, r) \), i.e., the super cluster with the highest simialrity to \( s \)  
\quad Remove \( r \) from \( S \)  
\quad \( q \leftarrow s \cup r \) (\( s \) and \( r \) are just sets of classes).  
\quad Reinsert \( q \) into \( S \), keeping \( S \) sorted by size.

We tried different functions for \( S\text{uperClusterSim}(s, r) \), but generally used one that takes the mean similarity 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 \( R = X \rightarrow \langle s, t \rangle \) with associated translation probabilities. If we had \( n \) levels of word clustering, each \( w \) appeared in exactly \( n \) super clusters (one at each hierarchy level), denoted as \( c_1, c_2, ..., c_n \). In this context \( c_j \) represents a unique id number given to that class. Thus, after clustering was complete, we added \( n \) new translation rules for each \( w \), \( c_1 \rightarrow \langle s, t \rangle, c_2 \rightarrow \langle s, t \rangle, ... , c_n \rightarrow \langle s, t \rangle \). The probabilities associated with each of these new rules were the same as the original probabilities of \( R \). As described in the example, the higher probability rules came after generalized rule extraction. After all of the new rules were added, generalized rule creation was performed as described in Section 3 and Section 2. Hierarchical decoding was then performed as described in Section 2.

5 Results

We ran Chinese to English experiments using \textit{mt06_nist} as the test set, and a separate newswire tuning set. We had one decoding feature for each level in the class hierarchy. For example, if a particular output sentence used 4 generalized rules from the 1000 class level and 3 from the 100 class level, the feature values would simply be 4.0 and 3.0. The weights of these features were optimized automatically on the tuning set.

The results on the \textit{mt06_nist} test set using different class hierarchies are presented in Table 1.

<table>
<thead>
<tr>
<th>Cluster Levels Used</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Classes</td>
<td>35.35</td>
</tr>
<tr>
<td>1000</td>
<td>36.15</td>
</tr>
<tr>
<td>1000, 100, 10, 1</td>
<td>36.99</td>
</tr>
<tr>
<td>1</td>
<td>37.16</td>
</tr>
</tbody>
</table>

Table 1: BLEU scores on the Chinese to English \textit{mt06_nist} test set, with various clustering hierarchies

Although all configurations with classes scored better than the configuration with no classes (and therefore no generalization), the hierarchical clustering setup (4 cluster levels, with 1000, 100, 10, 1 classes) did not perform better than standard single-word rule generalization (1 class containing all words). We can also see that the 1-class experiment outperformed the 1000-class experiment, which supports our belief that allowing any hierarchical substitution is not harmful \textit{per se}.

In order to understand the underlying cause of the unexpected results above, we performed some analysis on the rules used in the different experiment
configurations, and the results are presented in Table 2.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No Classes</td>
<td>0.00%</td>
<td>2.176</td>
</tr>
<tr>
<td>1000</td>
<td>16.64%</td>
<td>2.211</td>
</tr>
<tr>
<td>1000, 100, 10, 1</td>
<td>30.86%</td>
<td>2.294</td>
</tr>
<tr>
<td>1</td>
<td>22.32%</td>
<td>2.289</td>
</tr>
</tbody>
</table>

Table 2: Analysis of rules used in decoding the test set. “Gen. Rules” is the percent of all rules used that are generalized, i.e., have non-terminals. “Mean Len.” is the mean number of non-terminals in the source side of the rules used.

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 separately and concatenating the translations. In Table 2, we can see that although the multi-level configuration uses substantially more generalized rules than the single class configuration, the mean phrase length is roughly the same.

6 Discussion

It is not clear why the class-based generalized rules had no impact on the quality of translation. The mean source phrase length did not yield a noticeable difference between the multi-level and 1-class experiments, but we don’t believe that a lack of increase in mean source phrase length necessarily accounts for a lack of increase in BLEU score. It is important to note 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 “X → ⟨el coche X, the X car⟩” rather than any arbitrary phrase). However, if errors like the one presented in the constructed example happen only very infrequently, then the hierarchical configuration may not be beneficial.

It is also possible that the semantic relationship we used for clustering is not the best choice for the task. For example, the algorithm we chose will cluster words like “doctor” and “hospital” together, but it is not clear that these words will benefit from using one another’s generalizations any more than “doctor” and any other arbitrary noun would. However, since the similarity measure was calculated based on having high overlap in language model states, it would seem likely that words in the same cluster would make good generalized rule replacements for one another.

7 Conclusion

We can conclude that our method of creating generalized rules based on a hierarchy of semantic word classes does not increase the BLEU score 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, we plan to group multi-word 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. We plan to explore more methods of clustering words and phrases, such as using purely syntactic relationships (e.g., part-of-speech tags, along the lines of work by (Sanchez-Martinez and Ney, 2007)), or a combination of syntactic and semantic relationships (e.g., grouping by semantic similarity score, but requiring that part-of-speech tags be consistent through a cluster).

References
