Inducing Probabilistic Invertible Translation Grammars from Aligned Texts

Michael Carl
Institut für Angewandte Informationsforschung,
Martin-Luther-Straße 14
66111 Saarbrücken, Germany,
carl@iai.uni-sb.de

Abstract

This paper presents an algorithm for extracting invertible probabilistic translation grammars from bilingual aligned and linguistically bracketed text. The invertibility condition requires all translation ambiguities to be resolved in the final translation grammar. The paper examines the complexity of inducing translation grammars and proposes a number of heuristics to reduce the the theoretically exponential computation time.

1 Introduction

In the framework of corpus-based machine translation, a number of methods have been proposed for automatically inducing translation correspondences from aligned texts. Some researcher parse alignments with the aim of linking word translations and internal nodes of derivation trees in both language sides of the alignment. (Wu, 1995), for instance, proposes a bilingual stochastic parser. This parser analyses both language sides of an alignment in parallel where the leaves of the binary derivation trees are terminal symbols and all internal nodes are non-terminals. (Watanabe et al., 2000) and (Meyers et al., 1996; Meyers et al., 1998) parse each language side independently and try to find most likely node correspondences in the derivation trees. Other approaches, including this present one, try to induce a set of context-free transfer rules from the alignments.

This paper proposes an algorithm which generates and filters a translation grammar (i.e. a set of context-free transfer rules) from aligned and bracketed pieces of text. The transfer rules in this grammar are similar to Takeda's "Translation Pattern" (Takeda, 1996; Watanabe and Takeda, 1998) with the difference that an *accepted* input is necessarily also *translatable*. Translation grammars are made up of lexical transfer rules which contain only terminal symbols and translation templates - i.e. generalized transfer rules - which also contain variables.

In the framework of Example-Based Machine Translation (EBMT), a number of methods have been proposed for inducing translation grammars from aligned texts. In so-called "pure" EBMT systems, the only available knowledge resource is the aligned text itself (cf. (Block, 2000; Brown, 1997)), while in richer systems additional, linguistic knowledge resources are used to a varying degree; cf. (Somers, 1999) for a review of EBMT-systems and resources used by these systems. According to (Somers, 1999), EBMT systems differ in the number and the quality of resources used, the way this knowledge is represented, stored and used for trans-

c_1:(a)	$\leftrightarrow$	(e')	c_5:(e)	$\leftrightarrow$	(e')	c_9:	(de)	$\leftrightarrow$ (e')	c_{13}:(cde)	$\leftrightarrow$	(e')
c_2:(a)	$\leftrightarrow$	(a')	c_6:(e)	$\leftrightarrow$	(a')	c_{10}:	(de)	$\leftrightarrow$ (a')	c_{14}:(cde)	$\leftrightarrow$	(a')
c_3:(a)	$\leftrightarrow$	$(a'b')$	c_7:(e)	$\leftrightarrow$	$(a'b')$	c_{11}:	(de)	$\leftrightarrow$ $(a'b')$	c_{15}:(cde)	$\leftrightarrow$	$(a'b')$
c_4:(a)	$\leftrightarrow$	$(a'b'c')$	c_8:(e)	$\leftrightarrow$	$(a'b'c')$	c_{12}:	(de)	$\leftrightarrow$ $(a'b'c')$	c_{16}:(cde)	$\leftrightarrow$	$(a'b'c')$

Figure 1: Set of lexical transfer rules C_1 extracted from a_1

lation. EBMT systems differ also in the way generalizations are computed. In almost all cases where generalizations are induced from aligned texts, a set of two or more alignments are compared and suitable subsequences are replaced by variables. With respect to this substitution one can distinguish between methods which generalize similarities (e.g. (Block, 2000; Boström, 1999) and this present approach), methods which generalize differences (e.g. (Güvenir and Cicekli, 1998)) and methods which generalize both, similarities and - as their complement - differences (McTait and Trujillo, 1999). While there is increasing research that aims at inducing translation grammars from aligned texts, a detailed analysis of the potentials and implications of the different methods remains to be undertaken in future.

The algorithm I shall present in this paper expects as its input both language sides of the alignments to be bracketed. We currently use the shallow parser KURD (Carl and Schmidt-Wigger, 1998) for this bracketing, although knowledge-poor methods could similarly be used as, for instance, described in (Zaanen, 2000). The algorithm generates lexical transfer rules and generalizations from the bracketed alignments and assigns probabilities and weights to each of these rules. From these rules, a translation grammar is filtered. This grammar does not contain translation ambiguities. Instead, each transfer rule encodes a minimal context which makes it unique in the translation grammar. The algorithm does not require a bilingual lexicon, but a lexicon can be provided to bootstrap the system and enhance the outcome. The translation grammar is expected to re-generate the aligned text in a most compositional and complete manner. First, I describe how transfer rules are generated from the bracketed alignments. Then I describe how a set of invertible transfer rules is filtered. Last I give an example of an induced grammar.

2 Complexity of Inducing Translation Templates

The program assumes a bilingual text P which consists of n aligned pieces of text $a_1 \dots a_n$. Each alignment a_i consists of a left-hand side (lhs) e and a right-hand side (rhs) f. By means of a (shallow) parser, both language sides e and f are independently bracketed (parsed) which results in a representation similar to the following[1]:

$$a_1 : (a)\ b\ (c(d(e))) \leftrightarrow (((a')b')c')\ d'\ (e')$$

Without no further knowledge, one cannot know which of the lhs-brackets translates into which bracket in the rhs or whether a lhs-bracket has a rhs-translation at all. We therefore assume that each lhs-bracket translates with the same probability into any rhs-bracket. For an alignment a_i we can therefore extract $p \times q$ lexical transfer rules $C_i : \{c_1 \dots c_{p*q}\}$, where p is the number of lhs-brackets and q is the number of rhs-brackets. The set C_1 extracted from a_1 is shown in figure 1.

In a second step, translation templates (or generalizations) are induced from a_1 and the extracted lexical transfer rules $c_1 \dots c_{p*q}$. While a lexical transfer rule consists only of

[1] The letters "*abcde*" on the left-hand side represent lemmas of e; the letters "$a'b'c'd'e'$" those of f in the right-hand side. The brackets are also annotated with a phrasal tag and morpho-syntactic information. For the sake of simplicity, I will not consider this information here.

terminal symbols, a generalization contains variables (so-called reductions) in places where a shorter transfer rule matches a subsequence in the lhs and in the rhs. A generalization has thus at least one reduction in each language side and it has an equal number of reductions in the lhs and the rhs. Each reduction in lhs is linked exactly to one reduction in rhs.

From the transfer rule c_{16}, for instance, can be generated 4 different generalizations while from transfer rule c_{11} only one generalization can be generated. These are shown in figure 2. From the alignment a_1 can be generated 25 generalizations by substituting one or more transfer rules $c_1 \dots c_{16}$.

More generally, from a transfer rule c_j which has p bracketed sequences in its lhs and q bracketed sequences in its rhs, an exponential number of generalizations $\#G_j$ can be generated:

$$\#G_j = \sum_{i=0}^{p} \binom{p}{i} * \binom{q}{i} = \binom{p+q}{p}$$

For instance, if we assume both, p and q to be 10 and none of the 10 brackets in either language side are included in another bracket (i.e. all brackets are top-level brackets), more than 180.000 different generalizations can be generated. This is a number far too big to be computed, as 10 brackets (i.e. constituents in the parsed sentence) is not many. In fact, 50 or more brackets do appear frequently in parsed sentences, although in the experiment reported below, not all brackets are top-level. Therefore, a number of heuristics is proposed for generating from the set of possible generalizations only those achieving the highest weight.

3 Probabilistic Translation Grammars

Before introducing heuristics, I first describe how probabilities and weights are assigned to the lexical transfer rules and generalizations. While inducing the translation grammar, each possible item in the translation grammar, alignments a_i, lexical transfer rules c_j and generalizations g_k, is linked to two sets.

$$\begin{aligned} a_i &\Rightarrow \{G_i, C_i\} \\ g_k &\Rightarrow \{R_k, C_k\} \\ c_j &\Rightarrow \{G_j, A_j\} \end{aligned}$$

Each a_i is associated with a set of lexical transfer rules C_i and a set of generalizations G_i which have been generated from a_i. Each lexical transfer rule c_j is associated with a set A_j of alignments from which c_j has been extracted and a set of generalizations G_j which have been generated from c_j. Finally, each generalization g_k is associated with a set of lexical transfer rules C_k which have been replaced in the generalization (i.e. the daughters of the generalization g_k) and a set of references R_k. Since generalizations are generated from alignments and from the extracted lexical transfer rules, the set of references R_k may consist of alignments a_i and/or transfer rules c_j.

The probability of an alignment a_i is its frequency in the aligned text P divided by the number n of alignments in P.

$$p(a_i) = \frac{f(a_i)}{n} \tag{1}$$

The probability of a lexical transfer rule c_j is a function of the number of times c_j has been extracted from alignments a_i, $i = 1 \dots n$ and the cardinality $\#C_i$ of the set C_i, normalized by the size of P:

$$p(c_j) = \frac{1}{n} \sum_{c_j \in C_i} \frac{1}{\sqrt{\#C_i}} \tag{2}$$

The probability of a generalization g_k is the sum of the probabilities of the reference(s) $r \in R_k$ from which g_k has been generated.

G_i	Induced Generalization	$p(g_k)$	$w(g_k)$
$G_{11}:$	$\{\ g_1:\ (d*) \leftrightarrow (*b')$	$1/4$	$2/4$
$G_{16}:$	$g_2:\ (cd*) \leftrightarrow (*b'c')$	$1/4$	$2/4$
	$g_3:\ (cd*) \leftrightarrow (*c')$	$1/4$	$2/4$
	$g_4:\ (c*) \leftrightarrow (*b'c')$	$1/4$	$2/4$
	$g_5:\ (c*) \leftrightarrow (*c')$	$1/4$	$3/4$

Figure 2: Set of generalizations G_{11} and G_{16} induced from transfer rules c_{11} and c_{16}

$$p(g_k) = \sum_{r \in R_k} p(r) \qquad (3)$$

Based on these probabilities, a weight is computed for each a_i, c_j, g_k.
The weight $w(c_j)$ of a lexical transfer rule c_j equals the maximum weight of the generalization that has been generated from it. In case no generalization can be generated $w(c_j) = p(c_j)$.

$$w(c_j) = max\{w(g \in G_j)\} \qquad (4)$$

The weight of a generalization equals the sum of the probabilities of the references $r \in R_k$ from which it has been generated plus the sum of the weights of the lexical transfer rules (i.e. the daughters of g_k) which have been replaced in the generalization.

$$w(g_k) = \sum_{r \in R_k} p(r_i) + \sum_{c \in C_k} w(c) \qquad (5)$$

The following properties of weights in transfer rules and generalizations hold: a generalization has at least as high a weight as the transfer rule from which it was generated. A generalization has a higher weight than the daughters which have been substituted in the generalization.

$$w(r \in R_k) \leq w(g_k) > w(c \in C_k) \qquad (6)$$

Moreover, generalizations have higher weights if they contain i) more reductions or ii) if the reductions are to the highest extend compositional. We believe that these properties are suitable when using the induced probabilistic grammar for translation. As an example for achieving higher weights for more compositional generalizations, consider the following example.
As there are 16 lexical transfer rules extracted from alignment a_1 and assuming that $n = 1$, each of the rules has probability $1/4$. A sub-sequence in $c_{11}:\ (de) \leftrightarrow (a'b')$ can be substituted by rule $c_6:\ (e) \leftrightarrow (a')$. This substitution yields generalization g_1, as shown in figure 2. It is assigned the weight $2/4$ (cf. equation 5). This weight is also assigned to the transfer rule c_{11} (cf. equation 4). Subsequently, when generalizing the longer rule $c_{16}:\ (cde) \leftrightarrow (a'b'c')$, a set of four generalizations G_{16} is generated from which g_5 is assigned the highest weight due to the compositional nature of the replaced daughter $c_{11}:\ (de) \leftrightarrow (a'b')$.

4 Generating Transfer Rules

Even the quadratic effort for generating the sets C_i for all alignments a_i is too expensive in a large aligned text. Therefore, alignments $a_i,\ i = 1 \ldots n$ are generalized in a sequential manner. For each a_i, first the $p \times q$ lexical transfer rules are extracted and sorted by the length of the shorter string e or f. Transfer rules $c_j \in C_i$ are then generalized starting with the shortest rule. The crucial points in the procedure **GenerateGrammar**() are lines 4 and 9. As was shown above, extracting the set of lexical transfer requires a quadratic effort in the number of brackets, while generating G_j from c_j is exponential. To tackle this latter complexity, a version of the A* algorithm considers only a limited number of the highest weighted generalization[2].

```
1   GenerateGrammar(P)
2   begin
3       for all a_i ∈ P:
4           extract lexical transfer rules C_i from a_i;
5           for all c_j ∈ C_i :   p(c_j) += 1/(√#C_i * n);
6           add a_i as c_0 to C_i;
7           sort C_i by length of shorter e or f;
8           for each c_j ∈ C_i starting with shortest c
9               generate G_j from c_j
10              w(c_j) += max{w(g ∈ G_j)}
11          end
12      end
13  end;
```

In this way generalization of lexical transfer rules is reduced to $O(k * d)$ where d is the number of lexical transfer rules matching a subsequence in c_j and k is the number of generalizations to be generated. In addition to this, a couple of parameters can be set to reduce the number of extracted transfer rules and generalizations:

- Only transfer rules are extracted where the difference in the number of words in e and f does not exceed a pre-defined limit. This constraint reflects that generally more or less the same number of (content) words appear in both sides of a translation.

- By the same token, transfer rules and generalizations are weighted by the difference of number of words in e and f. This reflects the experience that translations are likely to contain approximately the same number of words in their source and target sides.

[2] In (Meyers et al., 1996) a similar mechanism is called "greedy heuristic".

- By means of a bilingual lexicon, transfer rules and generalizations are assigned (high) a-priori weights.

- Only a limited number of highest weighted translation rules and generalizations is generated.

- Each generalization can have up to a fixed maximum number of reductions. As reductions in generalizations are most reasonable explained to represent the arguments of the remaining unreduced token(s), a maximum of reduction might be set to four.

- A generalization can have maximum number of tokens. With increasing number of tokens, the chance to match a new sentence decreases exponentially. Note that this constraint does not apply to translation rules and generalizations from a bilingual lexicon.

- Transfer rules are only asserted if their weights are above a threshold of an already existing, ambiguous transfer rule in the database.

5 Filtering Invertible Translation Grammars

From the set P of alignments and their associated sets of generalizations $P : \{a_1 \Rightarrow G_1, \ldots, a_2 \Rightarrow G_n\}$ an invertible translation grammar is filtered in a top down fashion. The aim here is to find a most compositional set of transfer rules capable to reproduce the aligned text P in a most complete manner. To achieve this goal, translation ambiguities in the resulting grammar are avoided by including the smallest possible context which disambiguates each transfer rule. A transfer rule $e_2 \leftrightarrow f_2$ is ambiguous iff the translation grammar contains a different transfer rule $e_1 \leftrightarrow f_1$ where either e_1 equals e_2 or f_1 equals f_2. A translation grammar is invertible iff it contains no ambiguous transfer rules. In an invertible translation gram-

Induced Transfer Rules					$p(\cdot)$	$w(\cdot)$
a_1	:	(dx)	$\leftrightarrow$	$(m'n')$	1/4	2/4
g_1	:	$(d*)$	$\leftrightarrow$	$(m'*)$	1/4	2/4
c_1	:	(x)	$\leftrightarrow$	(n')	1/4	1/4
a_2	:	(de)	$\leftrightarrow$	$(a'b')$	1/8	1/4
g_2	:	$(d*)$	$\leftrightarrow$	$(*b')$	1/8	1/4
c_2	:	(e)	$\leftrightarrow$	(a')	1/8	1/8

Filtered Invertible Grammar				$p(\cdot)$	$w(\cdot)$
g_1	$:(d*)$	$\leftrightarrow$	$(m'*)$	1/4	2/4
c_1	$:(x)$	$\leftrightarrow$	(n')	1/4	1/4
a_2	$:(de)$	$\leftrightarrow$	$(a'b')$	1/8	1/4

Figure 3: Induced and Filtered Invertible Grammar

mar, therefore, each lhs-string e and each rhs-string f occurs exactly once.

The procedure to filter this grammar, **FilterGrammar()**, starts with the most frequent top-most generalizations - i.e. those generalizations of a_i with the highest weight - and recursively prints their daughters. Less frequent rules, i.e. lower weighted rules, are likely to to come along with more context. Only one generalization is filtered for each alignment. The alignments $a_i \in P$ are sorted by their weights and for each alignment starting with the highest weighted one, the function **FilterGrammar**(a_i) is called.

```
1    FilterGrammar(r_i)
2    begin
3       if G_i not empty
4          print g_k :  max{w(g_k ∈ G_i)}
5          delete all generalizations g : e ↔ f \\
              where e = lhs(g_k) or f = rhs(g_k).
6          for all c_j ∈ C_k: FilterGrammar(c_j)
7       else
8          print r_i
9          delete all rules c : e ↔ f \\
              where e = lhs(r_i) or f = rhs(r_i).
10      end
11   end;
```

The procedure is called recursively in line 6 to print the highest weighted daughters c_j of generalization g_k. To illustrate the resulting translation grammar, assume the set P contains two alignments, $\{a_1, a_2\}$ as shown in figure 3, left. Alignments a_1 and a_2 are associated with G_1 and G_2 which contain the generalizations g_1 and g_2 respectively. The daughters of g_1 and g_2 are $C_1 : \{c_1\}$ and $C_2 : \{c_2\}$. Note that the weights of a_1 and g_1 are 2/4 while the weight of a_1 and g_2 are 1/4.

The procedure **FilterGrammar**() is first called with a_1 and then with a_2. Accordingly, first generalization g_1 is printed and - in the recursion - c_1. Generalizations g_1 and g_2 are ambiguous since their lhs is identical but their rhs are different. Due to the deletion of g_2, the set G_2 is, thus, empty when calling the function with a_2. The resulting filtered invertible translation grammar is shown in figure 3, right.

6 Preliminary Experiments

The algorithm has been tested on a partially parsed text with 4997 alignments. In this first test both languages were identical, i.e. the words and the structure of their bracketings was identical in the source and target language. The aim of this experiment was to see to what extend the algorithm produces reasonable translation grammars for a pair of structurally identical languages. A reasonable assumption for translating a source language into an identical target language is that the induced transfer rules and generalizations are identical in their lhs and rhs, too. The quality of the translation grammar can then easily be measured by counting the rules which differ in their lhs and rhs.

The 4997 alignments had 45.352 words and 36.687 brackets on each language side. Dependent on the parameter setting - i.e. how many brackets were maximally considered per language side - the program took between 5 and 10 minutes on a sun work station to generate and filter the invertible grammar. The filtered invertible grammar contained 814 generalizations and 3692 lexical translation rules and alignments. 3698 of the filtered transfer rules had an identical lhs and rhs while 808 rules (18%) were different in lhs and rhs. Roughly half of these erroneous rules were lexical transfer rules and half generalizations. Simulating a lexicon covering 10% of the aligned text reduced erroneous transfer rules about 50% to 9,6% of the size of the translation grammar. Augmenting the lexicon to 20% reduced errors to 6,5% and with 50% of the aligned text covered by a lexicon produced 3.4% wrong output.

These results show that i) invertible grammars can be induced in reasonable time and ii) the proposed algorithm is scalable for the integration of further knowledge resources - such as a bilingual lexicon - which enhance the quality of the induced grammar.

In further experiments we will parse a text in different ways to see how well the algorithm can tackle different structures in both language sides. The goal for the future is to design (partial) parsers for different languages which yield similar brackets on both language sides in order to enable the algorithm to extract and filter reasonable translation grammars more easily. The induced translation grammars are supposed to be used in an EBMT system.

7 Conclusion

This paper presents an algorithm which generates and filters a translation grammar from aligned texts. The produced translation grammar consists of lexical transfer rules and generalizations. Each rule in the grammar describes an unambiguous 1-to-1 mapping from the source language to the target language. A small experiment is described to test the grammar induction performance. The algorithm shows satisfying runtime behavior and promising results.

References

Hans Ulrich Block. 2000. Example-Based Incremental Synchronous Interpretation. In *(Wahlster, 2000)*, pages 411–417.

Henrik Boström. 1999. Induction of Recursive Transfer Rules. In *Learning Language in Logic (LLL) Workshop*, Bled, Slovenia.

Ralf D. Brown. 1997. Automated Dictionary Extraction for "Knowledge-Free" Example-Based Translation. In *TMI-97*, pages 111–118.

Michael Carl and Antje Schmidt-Wigger. 1998. Shallow Postmorphological Processing with KURD. In *Proceedings of NeMLaP3/CoNLL98*, pages 257–265, Sydney.

Halil Altay Güvenir and Ilyas Cicekli. 1998. Learning Translation Templates from Examples. *Information Systems*, 23(6):353–363.

Kevin McTait and Arturo Trujillo. 1999. A Language-Neutral Sparse-Data Algorithm for Extracting Translation Pattern. In *TMI'99*.

Adam Meyers, Roman Yangarber, and Ralph Grishman. 1996. Alignment of shared forests for bilingual corpora. In *Coling'96*, Copenhagen, Denmark.

Adam Meyers, Roman Yangarber, Ralph Grishman, Catherine Macleod, and Antonio Moreno-Sandoval. 1998. Deriving transfer rules from dominance-preserving alignments. In *Computerm, First Workshop on Computational Terminology*, Montreal, Canada.

Harold Somers. 1999. Review Article: Example-based Machine Translation. *Machine Translation*, 14(2):113–157.

Koichi Takeda. 1996. Pattern-Based Machine Translation. In *COLING-96*, pages 1155–1158.

Wolfgang (ed.) Wahlster. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation.* Springer, Heidelberg.

Hideo Watanabe and Koichi Takeda. 1998. A Pattern-based Machine Translation System Extended by Example-based Processing. In *Coling 1998*.

Hideo Watanabe, Sadao Kurohashi, and Eiji Aramaki. 2000. Finding Structural Correspondences from Bilingual Parsed Corpus for Corpus-Based Translation. In *Coling 2000*.

Dekai Wu. 1995. Grammarless extraction of phrasal translation examples from parallel texts. In *TMI-95*.

Menno van Zaanen. 2000. ABL: Alignment-Based Learning. In *COLING-2000*, pages 961–967.