

# Inducing Translation Templates for Example-Based Machine Translation

Michael Carl

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

**Abstract** This paper describes an example-based machine translation (EBMT) system which relays on various knowledge resources. Morphologic analyses abstract the surface forms of the languages to be translated. A shallow syntactic rule formalism is used to percolate features in derivation trees. Translation examples serve the decomposition of the text to be translated and determine the transfer of lexical values into the target language. Translation templates determine the word order of the target language and the type of phrases (e.g. noun phrase, prepositional phrase, ...) to be generated in the target language. An induction mechanism generalizes translation templates from translation examples. The paper outlines the basic idea underlying the EBMT system and investigates the possibilities and limits of the translation template induction process.

## 1 Introduction

Until the end of the eighties MT was strongly dominated by rule-based systems which deduce translations of natural language texts based on a bilingual lexicon and a grammar. Linguistic theory and grammar formalisms have evolved and are ready at hand to support this approach. Unification and feature structures are such computational means which have influenced linguistic theories, formalisms and conception of MT systems. However, much human effort is required since, both, the grammar and the lexicon are manually entered into the machine.

In 1990 a new epoch started when Brown et al. (1990) presented a paper on statistics-based MT. Translation was no longer considered as deduction but rather as a stochastic process which relies on a number of random variables. Given a well evolved stochastic theory and powerful hardware, random variables could be computed from a sufficiently large set of example translations.

Since then machine learning technologies have evolved dramatically in such a way that not only statistics but also other learning techniques have entered the MT domain. Artificial neural networks were used for MT (McLean, 1992) or to transfer a source language parse tree into a target language parse tree (Wang and Waibel, 1995). Hiroshi et al. (1996) describe an MT system which makes use of genetic algorithms to optimize translation rules. Other approaches mix more or less shallow linguistic analysis with statistics (Knight et al., 1994) or with symbolic inductive methods (Sato and Nagao, 1990; Güvenir and Tunc, 1996; Güvenir and Cicekli, 1998; Collins and Cunningham, 1997; Collins, 1999). Knight et al. (1994) ranks semantic analyzes of an input sentence by statistical means to filter out least probable analyzes. Sato and Nagao (1990) replace subtrees in a tree bank of syntactically analyzed source-target language equivalences to obtain target language parse trees. Güvenir and Tunc (1996), Güvenir and Cicekli (1998) Collins and Cunningham (1997) and Collins (1999) induce translation templates to map a generalized input sentence into the target language. Different as they are, these approaches have in common that they use a corpus of reference translations which serves as a basis to induce (or restrict) possible translations of new unknown texts.

With inductive methods, the knowledge acquisition bottleneck largely disappears because the reference text can be prepared in a consistent way so that information extracted from it may be used directly in the translation process. However, unless the number of reference translations is overwhelming — as is the case in the experiments reported in Brown et al. (1990) and Brown et al. (1993) — it is hardly expected that all required features for the MT task can automatically be extracted from the reference translations. Even in case a large reference corpus is available, further (linguistic) knowledge resources may enhance the

Figure 1: Representational richness of EDGAR

Rich morphological representation is supported in EDGAR allowing for distributed complex disjunctions.

$$\left\{ \begin{array}{l} \text{lu=d\_art, c=w, sc=art, fu=def} \\ \text{agr=} \left\{ \begin{array}{l} \text{gen=f,} \\ \text{nb=sg,} \\ \text{case=d;g} \end{array} \right\}; \left\{ \begin{array}{l} \text{gen=m,} \\ \text{nb=sg,} \\ \text{case=n} \end{array} \right\}; \left\{ \begin{array}{l} \text{nb=plu,} \\ \text{case=g} \end{array} \right\} \end{array} \right\}; \left\{ \begin{array}{l} \text{lu=d\_rel, c=w, sc=rel, fu=np,} \\ \text{agr=} \left\{ \begin{array}{l} \text{case=n,} \\ \text{g=m,} \\ \text{nb=sg} \end{array} \right\}; \left\{ \begin{array}{l} \text{case=g;d,} \\ \text{nb=sg,} \\ \text{g=f} \end{array} \right\} \end{array} \right\}$$

translation outcome.

In this paper I shall describe the example-based machine translation system EDGAR<sup>1</sup> which makes use of reference translations, morphological knowledge and flat syntactic rules. The main feature, however, is an induction mechanism which generalizes translation templates from a corpus of reference translations. In this way four different knowledge resources, morphological analysis, syntactic rules, translation examples and induced translation templates are integrated into one MT process.

The next section gives a sketch of EDGAR. The richness of morphological representation is outlined and an example of the case base structure is shown. The way in which translation templates are induced from translation examples is briefly discussed and an example translation is given. The example translation aims to outline how the different knowledge resources interact in the translation process.

The third section describes how translation templates are induced from translation examples. Correctness criteria and the one tree criteria for translation templates induction are presented. The fourth section discusses the potentials of the induced translation templates in the translation process. It is shown that from a small set of translation examples translation templates can be induced which map structurally identical context-sensitive languages onto each other.

## 2 A sketch of EDGAR

EDGAR is an Example-Based Machine Translation (EBMT) system which integrates linguistic (i.e. morphological) knowledge, reference translations, simple syntactic rules for analysis and generation and a component which induces generalized translation templates from translation examples. EDGAR morphologically analyzes the input sentence, decomposes and generalizes it by matching it against the set of reference translations and reducing the matching chunks into

<sup>1</sup>EDGAR is an acronym for Example-based Decomposition, Generalization And Refinement.

one node. The generalized input sentence is then specified (i.e. the correct linguistic information is gathered) and ‘refined’ in the target language.

The new sentence to be translated is decomposed according to the source language parts of the examples contained in the case base (CB). On a number of levels the input sentence is thereby reduced by applying a set of reduction rules until no further generalization can be computed. The generalized input sentence is then specified and refined in the target language. Specification retrieves the target language parts of the translation examples from the CB and refines them by applying the set of refinement rules.

### 2.1 Morphological Analysis

Morphological analysis is the process of separating grammatical information and (a) stem(s) from the surface form of an input word. From an input string, lemmatization generates a basic word form that does not contain inflectional information. In theory, a lemma plus grammatical information is thus equivalent to the surface form of the word.

Currently, EDGAR makes use of MPRO morphological analyses. MPRO (Maas, 1996) is a powerful tool: it yields more than 98% correct morphological analysis and lemmas of arbitrary German and English texts. In addition to this, lemma decomposition can be carried out by MPRO. Recognition of composition and derivation yields knowledge about the internal structure of the word.

Morphological information and the value of the lemma are represented in the form of sets of attributes/values which we will refer to as feature bundles (FBs). There are no (formal) problems to include other information such as semantic and/or pragmatic features into the FBs. For (at least) two reasons this has not been the case until now. Firstly, it is hard to find a consistently broad coverage semantic analyser. Secondly the main problems we are currently facing are related to inconsistencies in reference translation examples. It does not appear, however, that such problems could be solved by means of semantic features (for an example

Figure 2: Definition of Rules

Reduction and refinement rules are used to percolate features from a matched chunk into a reduction and from the reduction into the specified chunk during refinement

---

$rule ::= name '=' description ':' action$   
 $description ::= condition_1 ',' condition_2 \dots$   
 $action ::= consequence_1 ',' consequence_2 \dots$

---

see e.g. (Carl and Hansen, 1999), in these proceedings).

The analysis of a word (a node) may consist of atomic disjunctions and complex disjunctions at a number of different levels. Which of the disjunctive representations is chosen depends on the one hand on the expressive requirements (i.e. no feature dependencies can be expressed with atomic disjunctions) and on the other hand on the linguistic assumptions of the morphological analysis.

Both types of disjunction are shown in the representation of the German article “*der*” in Figure 1. A first level of disjunction occurs at the level of the word descriptors. Different analyzes (as a determiner ( $lu=d\_art$ ) and as a relative pronoun ( $lu=d\_rel$ )) are separated by a semicolon ‘;’. The second level of disjunction occurs in the feature “*agr*”, which has a complex disjunction as its value. The case feature “*case*” in the first complex disjunctive has a local disjunction ( $g;d$  i.e. *genitive or dative*) as its value. The word “*der*” has seven different interpretations which are melted together here by means of the two different types of disjunction.

There is not need for variable binding between different attributes of the same FB<sup>2</sup>. because it is assumed that each attribute in a (morphological) FB expresses a different piece of information (it thus has a different type).

## 2.2 The Case Base

Each translation example consists of a tag-annotated source and target language part. An English-German translation example is given in 1. The English expression *A hot summer* is a translation of the German expression *Ein warmer Sommer* and both expressions

<sup>2</sup>In many theories and formalisms (e.g. HPSG, CAT2 (Sharp and Streiter, 1995)) different attributes in a FB can be forced to always have the same values by assigning the same variable as their values (they share the same structure). However, these approaches allow structure sharing and variable binding only among equal types.

are labeled with the tag *dp*. The arrow  $\leftrightarrow$  separates the source language expression and the target language expression.

1  $(A\ hot\ summer)_{dp} \leftrightarrow (Ein\ warmer\ Sommer)_{dp}$   
 2  $(in\ the\ morning)_{pp} \leftrightarrow (morgens)_{adv}$

In example 2, the English expression *in the morning* is labeled with the tag *pp* whereas its German translation *morgens* is labeled with the tag *adv*. The source language expression and the target language expression may thus be differently tagged. Throughout this paper, italic letters refer to morphological analyzes and lemmatization of the correspondent word. Subscripted italic letters denote feature values. The string *summer* thus refers to a FB carrying information on the lemma of the word “summer” and information on part of speech, number, case and gender. Analyzes of verbs include information on verb type (finite, infinite, participle), time, person, mode, etc. In case of convenience possibly indexed letters are used to denote arbitrary FB.

When compiling a CB, translation templates are induced from translation examples. In a later section I shall explain in more detail how this is automatically done, here I shall just outline the basic ideas and clarify the notation used.

A translation template differs from a translation example as it has reductions expressed by constraint variables in both the source and the target language. Translation templates are a generalization of translation examples which keep traces of properties of the reduced sequences in the their reductions. Reductions in generalizations and translation templates are represented as calligraphic letters  $\mathcal{X}$ ,  $\mathcal{Y}$  and  $\mathcal{Z}$ . The subscripted features are called the external constraints which must match an object. The superscribed numbers are called the internal constraints which keep track of (the indices of) the matched subtrees (i.e. the reduced chunks).

1t  $(A\ hot\ \mathcal{X}_{noun})_{dp} \leftrightarrow (Ein\ warmer\ \mathcal{X}_{noun})_{dp}$

The translation template 1t is a possible generalization of the translation example 1 where the source language expression *summer* and the target language expression *Sommer* have been replaced by the variable  $\mathcal{X}$ . Note that the source language reduction and the target language reduction are annotated with the part of speech *noun* of the reduced item *summer* and *Sommer* respectively. The phrase tag *dp* in the translation template is inherited from the original translation example. By means of reduction rules, features can be percolated from the matching chunk into the reduction. Thus, in case we are interested e.g. in the number of the matching *summer* chunk, by means of

reduction rules this (or other) features can be percolated into the reduction.

### 2.3 Reduction and Refinement Rules

Reduction and refinement rules make use of the KURD<sup>3</sup> formalism as described in (Carl and Schmidt-Wigger, 1998). They serve to percolate features (i.e. external constraint) between a chunk and a reduced mother node. When reducing a chunk of the input sentence into one node, reduction rules may apply in order to percolate (sets of) features from the matching chunk into the reduced mother node. After specifying an example in generation, refinement rules may apply in order to percolate features from the mother node into the specification. They may change, delete or add values, features or nodes.

Domains of refinement rules are the specified target language chunks including information available in the mother node. Domains of reduction rules are the matched chunks of the input sentence including information available in the mother node. The information available in the mother node is — among others — the type tag of the matching example.

A KURD rule essentially consists of a *description* part and an *action* part as shown in Figure 2. The *description* consists of a number of *conditions* that must match successive nodes in the chunk. Each *condition* matches the longest possible sequence of nodes and, once matched, no alternative matching is considered i.e. there is no backtracking or multiple solution generation.

While matching the *conditions* contained in the *description* part of a rule onto a chunk, nodes may be marked in order to be modified in the *action* part. The *action* part is activated if all *conditions* are satisfied. Actions of rules include the following operations:

- Unification and deletion of features.
- Concatenation and replacement of values.
- Insertion and deletion of nodes.

A rule fails if a *condition* does not match. In this case the *action* part of the rule is not activated.

All *consequences* of the *action* are executed if the *description* matches a sequence in the input sentence. The following operations are currently implemented:

- k** kills a marked node.
- u** unifies FBs into a marked node.
- r** replaces values in the marked nodes.
- d** deletes features from the marked nodes.
- i** inserts a FB before the marked node.

<sup>3</sup>KURD is an acronym for Kill Unify Replace Delete, the first operators implemented in that formalism

- a** inserts a FB after the marked node.
- c** concatenates values of features.

The concatenation operator “c” has a special syntax and performs rather powerful operations such as string concatenation (i.g. concatenation of values from different nodes) and simple arithmetic operations. It can give a unique value to a feature and it has access to some formalism internal representations such as the rule number it is part of and the number of the object it modifies. In a later section an application of this operator shall be discussed.

### 2.4 An Example Translation

A new sentence to be translated is decomposed and generalized according to the examples contained in the CB. Those sequences of the new sentence which match one (or more) example(s) are reduced into one node annotated with external constraints and internal constraints. External constraints in the generalized input sentence may stem from two different sources. Either they are copied from the matching example(s) of the CB into the reduction or they stem from the matched chunk of the input sentence. For the latter, reduction rules are used to percolate constraints from the matched chunk of the input sentence into the reductions. External constraints thus restrict the set of matching translation. In the refinement step, external constraints serve to drive the refinement process.

The generalized sentence is iteratively matched against the CB until no more reductions can be performed or the entire sentence is reduced to one single node.

Whereas the external constraints are visible in the generalization, internal constraints only serve to determine the internal structure of the target language chunk to be generated in the refinement step. Internal constraints refer to matching examples of the CB and determine the shape and content of the chunk to be specified.

To give an example, assume the CB below containing the translation examples 3 and 4 and the translation template 5. Each example is annotated with a tag which describes the type of example.

The input sentence to be translated into German is the English sentence *Every handsome man loves a pretty woman*. As shown in Figure 4 (left), in the first generalization step the sentence is decomposed into the three chunks */Every handsome man/ /loves/ and /a pretty woman/*. The generalization “ $\mathcal{X}_{dp,nom}^3 \text{ love}_{fin} \mathcal{Y}_{dp,acc}^4$ ” is computed based on the examples 3 and 4 of the CB. The reductions  $\mathcal{X}_{dp,nom}^3$  and  $\mathcal{Y}_{dp,acc}^4$  are single nodes which represent the reduced chunks */Every handsome man/* and */a pretty woman/* respectively. The reduced nodes include the external constraints



A translation template needs to hold the following template correctness criteria *TCC*.

1. A translation template contains at least two nodes in both the source and the target language side.
2. There is an equal number  $> 0$  of reductions in both the source and the target language side of the translation template.
3. Reductions in the source and the target language side are based on the same examples.

Consider the following generalizations. Template 11 violates *TCC 1* because its left hand side contains only one node. If we would permit translation template 11, generalization risks to produce unary recursive derivation trees. Such templates are therefore excluded from the CB. Translation templates 12 and 13 are not in line with *TCC 2* because the right hand side contains one more reduction than the left hand side. When specifying the node  $\mathcal{Y}$  on the right hand side we are unable to retrieve a target language example from the CB because nothing has been reduced on the left side. The example 14 is a correct translation example but it is not a template because it does not contain any reduction. Example 15 is a correct translation template. Both sides have the same number of reductions which are inverted in the source and the target language.

11	$(\mathcal{X})$	$\longleftrightarrow$	$(x \mathcal{X} y)$
12	$(ab)$	$\longleftrightarrow$	$(x \mathcal{Y} y)$
13	$(a \mathcal{X} b)$	$\longleftrightarrow$	$(x \mathcal{X} \mathcal{Y} y)$
14	$(ab)$	$\longleftrightarrow$	$(xy)$
15	$(\mathcal{X}\mathcal{Y})$	$\longleftrightarrow$	$(\mathcal{Y} x \mathcal{X})$

To illustrate the third *TCC* consider translation examples 16 and 17 which serve as a basis to generalize translation example 18. The string  $abc$  on the left hand side of example 18 can be reduced to  $a \mathcal{X}^{17}$  based on translation example 17 whereas the string  $xyz$  on the right hand side can be reduced to  $x \mathcal{X}^{16} y$  based on translation example 16. The generalizations of the left and the right hand side of example 18 as shown in 18t are corrupt generalizations of the translation example. To avoid such misleading translation templates it is checked whether the reductions on both sides are based on the same examples.

16	$(b)$	$\longleftrightarrow$	$(y)$
17	$(bc)$	$\longleftrightarrow$	$(vw)$
18	$(abc)$	$\longleftrightarrow$	$(xyz)$
18t	$(a \mathcal{X}^{17})$	$\longleftrightarrow$	$(x \mathcal{X}^{16} z)$

Taking translation example 21 as a basis to generalize translation example 22 results in the correct translation template 22t. The reductions on both sides of

the translation template refer to the same translation example 21 so that the *TCC* hold. The translation template abstracts the peculiarities of translation example 21; keeping only its general properties. The reductions' internal constraint 21 is, therefore, discarded from the translation template when stored in the CB. Instead, properties of the reduced chunk are percolated into the templates' reductions in order to select possible fillers of this slot. Since translation examples 20 and 21 have the same type, translation template 22t subsumes the translation  $ab \longleftrightarrow xy$  and  $am \longleftrightarrow xn$ . It does not, however, subsume the translation  $ao \longleftrightarrow xp$  because the reductions' external constraint  $\alpha$  does not fit the translation examples' external constraint  $\omega$ .

19	$(o)_{\omega^s}$	$\longleftrightarrow$	$(p)_{\omega^t}$
20	$(b)_{\alpha^s}$	$\longleftrightarrow$	$(y)_{\alpha^t}$
21	$(m)_{\alpha^s}$	$\longleftrightarrow$	$(n)_{\alpha^t}$
22	$(am)_{\beta^s}$	$\longleftrightarrow$	$(xn)_{\beta^t}$
22t	$(a \mathcal{X}_{\alpha^s}^{21})_{\beta^s}$	$\longleftrightarrow$	$(x \mathcal{X}_{\alpha^t}^{21})_{\beta^t}$
23	$(abc)_{\gamma^s}$	$\longleftrightarrow$	$(xyz)_{\gamma^t}$
23t	$(a \mathcal{X}_{\alpha^s} c)_{\gamma^s}$	$\longleftrightarrow$	$(x \mathcal{X}_{\alpha^t} z)_{\gamma^t}$
23tt	$(\mathcal{X}_{\beta^s} c)_{\gamma^s}$	$\longleftrightarrow$	$(\mathcal{X}_{\beta^t} z)_{\gamma^t}$

In successive induction steps, more than one translation template can be generated from one translation example. Based on translation example 20, translation example 23 can be generalized into translation template 23t. In a second step, translation template 23t can be generalized into translation template 23tt. The prefixes of the left and right hand side  $a \mathcal{X}_{\alpha^s}$  and  $x \mathcal{X}_{\alpha^t}$  match translation template 22t such that a further translation template (23tt) can be generalized. Note that the translation template can be generalized no further since according to *TCC 1* each translation template needs at least two nodes on both language sides.

### 3.2 The One Tree Principle

Translation templates are induced from translation examples by replacing the maximum of non-overlapping chunks through constraint variables. At most one translation template is induced at a time. Once a translation examples' language side has been turned into chunks, no alternative decomposition is considered to find possibly different generalizations. This induction strategy leads to one derivation tree only for each translation example. It is assumed that a target language sentence is an instantiation of only one source language generalization.

To exemplify the implications of the one tree principle consider translation example 37. Based on the translation examples 24, 25 and 26 in theory three generalizations are possible which are shown in 27t<sub>1</sub>, 27t<sub>2</sub> and 27t<sub>3</sub>. In 27t<sub>1</sub> the items  $b$  and  $y$  have been

reduced and in 27t<sub>2</sub> and 27t<sub>3</sub> the respective chunks  $ab$  and  $xy$  and  $bc$  and  $yz$  are generalized.

To obtain a maximum homogenous and consistent bi-grammar, a strict segmentation strategy is respected. Currently, decomposition proceeds from left to right and the first longest chunks are considered for reduction only. Due to this strategy, template 27t<sub>2</sub> will be induced. This processing strategy might have negative consequences but there is currently not sufficient experience to evaluate the implications. The problem is to decide which of the overlapping chunks are more appropriate and to select the most suitable of the templates 27t<sub>2</sub> or 27t<sub>3</sub>.

24	$(b)_{\alpha^s}$	$\longleftrightarrow$	$(y)_{\alpha^t}$
25	$(ab)_{\beta^s}$	$\longleftrightarrow$	$(xy)_{\beta^t}$
26	$(bc)_{\gamma^s}$	$\longleftrightarrow$	$(yz)_{\gamma^t}$
27	$(abc)_{\delta^s}$	$\longleftrightarrow$	$(xyz)_{\delta^t}$
27t <sub>1</sub>	$(a \mathcal{X}_{\alpha^s} c)_{\delta^s}$	$\longleftrightarrow$	$(x \mathcal{X}_{\alpha^t} z)_{\delta^t}$
27t <sub>2</sub>	$(\mathcal{X}_{\beta^s} c)_{\delta^s}$	$\longleftrightarrow$	$(\mathcal{X}_{\beta^t} z)_{\delta^t}$
27t <sub>3</sub>	$(a \mathcal{X}_{\gamma^s})_{\delta^s}$	$\longleftrightarrow$	$(x \mathcal{X}_{\gamma^t})_{\delta^t}$

We are, however, experimenting with weighting the translation examples and translation templates. Weights are computed in an alignment process where translation examples are automatically extracted from a bi-text. The weights assigned to the translation examples are computed based on lexical translation probabilities and their degree of ambiguity. A segmentation is to be chosen which achieves the best overall weights.

An alternative processing strategy could be to search for the overall longest chunks or to make use of chunk initial and/or chunk final item probabilities to find the most likely segmentation.

## 4 Reduction and Generalization

In this section I shall examine how information contained in different features of the input sentence's FBs may be distributed over different entries of the CB and how the information is gathered from different translation examples in the target language. I shall further show that EDGAR achieves computational power equivalent to indexing grammars (IG). IG recognize indexing languages — so-called mildly context-sensitive languages — which are known to be a subset of context-sensitive languages but are more powerful than context-free languages. According to (Partee et al., 1990) there is no phenomenon known in natural languages that falls outside indexing languages, so that they “provide us with a kind of upper bound for syntactic phenomena” of natural languages.

### 4.1 Representational Richness

Each side of a translation example is annotated with a set of features which is part of its external constraints. The number of different features/values which describe the set of examples in the CB, (i.e. the cross product of the external constraint's values) is referred to as the richness of the description language.

The completeness of a CB is independent from the representational richness of its description language. A CB is complete if it contains all translation examples required to translate a source language into a target language. As the use of language is a creative process which generates new meanings every day, for general purpose natural language translations the CB can hardly ever be complete. However, a rich description language can partially compensate the necessary incompleteness of translation examples.

To illustrate how in absence of a complete CB a partly matching input sentence can be generalized consider the following example. The English word *old* can be a noun or an adjective dependent on the context in which it occurs. In case there is only a translation example for *old* associated with the type *noun*, there would be no chance to generalize the word *old* if it occurs with its adjective interpretation.

This can be circumvented by considering one example for generalizing type information (i.e. variable features) another example for generalizing token information (i.e. fixed features). In this way the set of features matched against the CB is distributed over distinct sets of examples.

Consider a CB containing the entries 28 and 29 and the input sentence  $a_{\beta^s}$  to be translated.

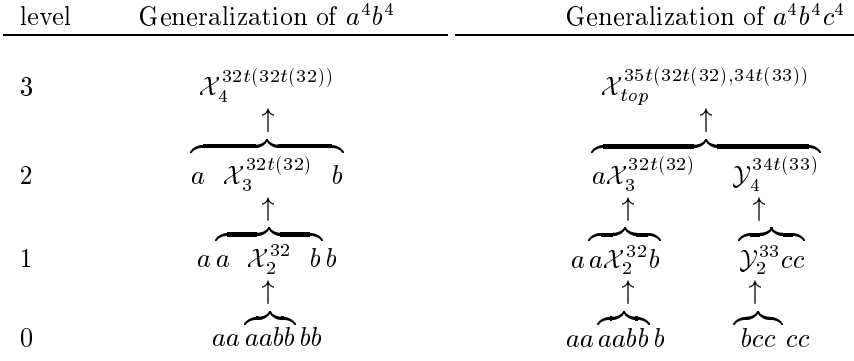
28	$(a_{\alpha^s})_{\gamma^s}$	$\longleftrightarrow$	$(x_{\alpha^t})_{\gamma^t}$
29	$(b_{\beta^s})_{\delta^s}$	$\longleftrightarrow$	$(y_{\beta^t})_{\delta^t}$

The token  $a$  matches the entry 28 whereas its type features  $\beta^s$  match entry 29. In absence of a full match, the input sentence can nevertheless be generalized into one reduction where the fixed features are taken from entry 28 and the variable features are taken from entry 29. This will yield the reduction  $\mathcal{X}_{\gamma^s + \beta^s}^{29+28}$

Here  $\gamma^s$  is the phrase tag from the matching token entry and  $\beta^s$  is percolated from the input sentence into the reduction node by means of reduction rules. The internal constraints 29 + 28 memorize the matching entries for the variable and the fixed features respectively.

When specifying the reduction in the target language the feature sets of the two entries are merged together just as they were decomposed in the source language:  $\mathcal{X}^{29+28} \rightarrow (x_{\beta^t})_{\gamma^t}$ . The token  $x$  and the phrase tag  $\gamma^t$  are taken from entry 28 and the variable features  $\beta^t$  are taken from entry 29.

Figure 5: Derivation trees of the strings  $a^4b^4$  and  $a^4b^4c^4$



## 4.2 Context-sensitive languages

Reduction rules may either copy features from the matched chunk of the input sentence into the reduction or they may interpret a feature as a stack and push or pop an item to or from it. A feature may thus work like a counter which is percolated over generalization levels by eventually adding or deleting items from it. If the feature stack is simply copied into the reduced node, the production rule works just like a context-free grammar. The context sensitivity of indexing grammars thus stems from its capacities to push and/or pop items to or from the feature stack. When applying a reduction (or refinement) rule, features may be checked to fill certain constraints. A translation template may only match if a feature stack contains a certain element.

Consider the CB containing entries 30, 32 and 32t. These bilingual mapping rules recognize the language  $a^n b^n$ ,  $n > 0$  and map it into the target language  $x^n y^n$ . A source language derivation tree of the input sentence  $a^4 b^4$  is shown in Figure 5 (left). Indices 2, 3 and 4 in the reductions are generated by reduction rules which push an item at each generalization level into a feature stack. At level 3 the whole input string is recognized and the internal constraint  $32t(32t(32))$  in the reduction  $\mathcal{X}$  memorizes the order of the invoked translation examples and translation templates. Each time translation template 32t applies for reduction, the external constraint is incremented by 1. The external constraint thus implements a counter which indicates the level of generalization.

Context-sensitive languages can equally be recognized from a finite set of translation examples and translation templates and mapped into a structurally identical target language. Translation examples 34 and 35 identify the language  $a^n b^n c^n$  and map it into the target language string  $x^n y^n z^n$  for  $n = 2$  and  $n = 3$

respectively. In order to generalize the mapping for any  $n > 3$  the complex mapping  $a^n b^n c^n \leftrightarrow x^n y^n z^n$  is simplified into the concatenation of the mapping  $a^i a b b b^i \leftrightarrow x^i x y y y^i$  and  $b c c^i \leftrightarrow y z z^i$ ,  $i > 1$ .

30	$(ab)_1$	$\longleftrightarrow$	$(xy)_1$
31	$(bc)_1$	$\longleftrightarrow$	$(yz)_1$
32	$(aabb)_2$	$\longleftrightarrow$	$(xyyy)_2$
32t	$(a \mathcal{X}_i b)_{i+1}$	$\longleftrightarrow$	$(x \mathcal{X}_i y)_{i+1}$
33	$(bcc)_2$	$\longleftrightarrow$	$(xzz)_2$
33t	$(\mathcal{Y}_j c)_{j+1}$	$\longleftrightarrow$	$(\mathcal{Y}_j z)_{j+1}$
34	$(aabbcc)_2$	$\longleftrightarrow$	$(xyyyzz)_2$
34t	$(\mathcal{Y}_j cc)_{j+2}$	$\longleftrightarrow$	$(\mathcal{Y}_j zz)_{j+2}$
35	$(aaabbcc)_3$	$\longleftrightarrow$	$(xxxxyyzzz)_3$
35t	$(a \mathcal{X}_i \mathcal{Y}_j)_{top}$	$\longleftrightarrow$	$(x \mathcal{X}_i \mathcal{Y}_j)_{top}$

Translation examples 32 to 35 are generalized into translation templates 32t to 35t according to the principles outlined in the previous sections.

Translation templates 32t and 33t recognize the two sublanguage mappings  $a^i a b b b^i \leftrightarrow x^i x y y y^i$  and  $b c c^i \leftrightarrow y z z^i$ ; translation templates 34t recognizes the language  $b c c c^{2i} \leftrightarrow y z z z^{2i}$ .

Successive generalizations of the input string  $aaaabbb-bcccc$  are shown in Figure 5 (right). At level 1, the translation examples 32 and 33 match the chunks  $aabb$  and  $bcc$  which are respectively reduced into the nodes  $\mathcal{X}_2^{32}$  and  $\mathcal{Y}_2^{33}$ . The external constraint 2 is copied from the translation examples. In successive levels of generalization, the external constraints are incremented by means of reduction rules.

Arriving at level 3, a reduction rule checks whether index  $i + 1$  in daughter nodes  $\mathcal{X}$  equals index  $j$  in daughter node  $\mathcal{Y}$ . If this is the case, the string  $a^i b^j c^j$  is recognized. The input string is thus accepted if it has entirely been recognized by any of the translation examples 34 or 35 or if it is recognized by translation template 35t with the index of the left daughter be-

ing equal to the index of the right daughter plus one. The recognized input string, then, consists of  $j$  a's, followed by  $j$  b's followed by  $j$  c's,  $j > 3$ .

The source language generalization is then expanded in the target language by specifying the internal constraints and successively refining the retrieved target language examples. However, general translation templates for a mapping of structural different languages such as  $a^n b^n \longleftrightarrow (ab)^n$  cannot be induced in EDGAR. Translating such languages into each other would require either for each  $n$  a separate translation example or for the aide of reduction and/or refinement rules to rearrange the generalization tree.

## 5 Conclusion

This paper describes an example-based machine translation system which makes use of morphological knowledge, shallow syntactic processing, translation examples and an induction mechanism which induces translation templates from the translation examples.

Induced translation templates determine a) the mapping of the word order from the source language into the target language and b) the type of sub-sentential phrases to be generated. Morphologic knowledge allows the abstraction of surface forms of the involved languages, and together with shallow syntactic processing and the percolation of constraints into reduced nodes new input sentences to be translated can be generalized. The generalized sentence is then specified and refined in the target language where refinement rules may adjust the translated chunks according to the target language context.

This paper investigates the computational power of the generalization process and describes in more detail the possibilities and limits of the translation template induction mechanism. Translation templates are seen as generalized translation examples. The template correctness criterion (TCC) formally defines the correctness of induced translation templates and the one tree principle implies a strict segmentation strategy.

Due to the induction capacities of the system the rule system remains relatively simple if the source and target language are structural similar. The conjunction of different resources allows for the analysis and generation of context-sensitive languages. Mapping of structurally different languages remains, however, beyond the capacities of the induction mechanism.

## References

Peter F. Brown, J. Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, F. Jelinek, Robert L. Mercer, and

P.S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical Machine Translation: Parameter estimation. *Computational Linguistics*, 32:263–311.

Michael Carl and Silvia Hansen. 1999. Linking Translation Memories with Example-Based Machine Translation. In *MT-Summit VII*.

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

Bróna Collins and Pádraig Cunningham. 1997. Adaptation-Guided retrieval: Approaching EBMT with caution. In *TMI-97*, pages 119–127.

Bróna Collins. 1999. *Example-Based Machine Translation: An Adaptation-Guided Retrieval Approach*. Ph.D. thesis, Trinity College, Dublin.

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

H.A. Güvenir and A. Tunc. 1996. Corpus-Based Learning of Generalised Parse Tree Rules for Translation. In *Proceedings of the 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 121–131.

Echizen-ya Hiroshi, Araki Kenli, and Yoshio Momouchi. 1996. Machine translation method using inductive learning with genetic algorithms. In *COLING-96*, pages 1020–1023.

K. Knight, Ishwar Chander, Matthew Haines, Vasileios Hatzivassiloglou, Eduard Hovy, Masayo Iida, Steve K. Luk, and Kenji Yamada. 1994. Integrating knowledge bases and statistics in MT. In *Proceedings of NeMLaP*, pages 134–141, Manchester, September.

Heinz-Dieter Maas. 1996. MPRO - Ein System zur Analyse und Synthese deutscher Wörter. In Roland Hausser, editor, *Linguistische Verifikation, Sprache und Information*. Max Niemeyer Verlag, Tübingen.

Ian J. McLean. 1992. Example-Based Machine Translation using Connectionist Matching. In *TMI-92*.

Barbara H. Partee, Alice ter Meulen, and Robert E. Wall. 1990. *Mathematical Methods in Linguistics*. Studies in Linguistics and Philosophy. Kluwer Academic Publishers, Dordrecht / Boston / London.

S. Sato and M. Nagao. 1990. Towards memory-based translation. In *COLING-90*.

Randall Sharp and Oliver Streiter. 1995. Applications in Multilingual Machine Translation. In *Proceedings of The Third International Conference and Exhibition on Practical Applications of Prolog, Paris, 4th-7th April*. URL: <http://www.iai.uni-sb.de/en/cat-docs.html>.

Ye-Yi Wang and Alex Waibel. 1995. Connectionist Transfer in Machine Translation. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, Tzigrav Chark, Bulgaria.