

Tackling the Complexity of Context-Free Representations in Example-Based Machine Translation

Michael Carl

Human Language Technology Center
 Hong Kong University of Science and Technology
 Clear Water Bay, Kowloon, Hong Kong
 lrac@cs.ust.hk

Abstract

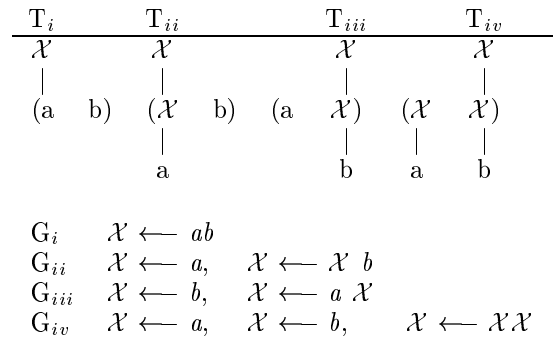
Machine Translation (MT) is seen as a mapping from a source language string into a target language string via an internal representation. I restrict the internal representations to derivation trees that can be generated from context-free grammars (CFG). It is shown that there are more than $n!$ possible representations for an input string of length $1 < n < 20$. Given this untractable complexity of internal CF-representations, all MT systems which rely on such representations implement a strategy or heuristic to reduce the search space. The paper discusses the Generalized Exemplar-Based MT-system which selects among the possible representations a subset at runtime. An analysis of the computational effort is provided.

1 Complexity of CF-Representation

I assume a universal context-free grammar (CFG) G that produces all possible representations for a given string s . I will examine the number of different representations that can be derived from the input string s given the universal grammar G . In the next section, I shall discuss the generalized exemplar-based MT-systems and analyse its way to tackle the problem of representational complexity. I shall show that this approach has the potential to generate a universal grammar.

A string of length 1 (i.e. the string consists of one word only) may either fit a grammar rule in G or else it remains unrecognized. Only in the former case the string can be reduced and an internal representation can be generated. To examine the number of different derivation trees that can be generated from a string of length $n > 1$, I shall assume that the context-free grammar G is sufficiently large such that the whole input string can be reduced into one node only. I will henceforth denote the set of possible representations of a string of length n as $R(n)$ and the cardinality of that set as $R^\#(n)$. The i th element in the set $R(n)$ is denoted by $R^i(n)$. If a special string s is under consideration I shall write $R(s)$, $R^\#(s)$ and $R^i(s)$ with the same meaning. A string s of length 1 has thus exactly

Figure 1: Four possible representations of the string ab of length 2 that a CFG can generate. Each derivation tree T_j may be generated by an appropriate grammar G_j shown in the lower part of the picture.



$R^\#(s) = R^\#(|s|) = R^\#(1) = 1$ possible representation.

The string ab has $R^\#(ab) = 4$ possible representations which are shown in the Figure 1. The grammars required to reduce the string ab into the derivation trees T_i to T_{iv} are given in the G_i to G_{iv} in Figure 1. The union of the grammars G_i to G_{iv} contains six rules and is the universal grammar for the string ab which generates all its possible representations.

A string of length $n = 3$ has $R^\#(3) = 24$ possible representations. The derivation trees for the string abc are shown in Figure 2.

In order to see how the number of possible CF-representations can be computed consider the following. In the tree representation T_{1i} the whole string fits one grammar rule and is reduced into one node. The top most node is not shown in the derivation trees in Figure 2. The three derivation trees T_{1ii} to T_{1iv} each contains one reduction, each of which reduces a chunk of length $l = 1$, while the trees T_{2i} to T_{2iii} contain two reductions each reducing one chunk of length $l = 1$ and derivation T_{2iv} contains three such similar reductions.

The remaining trees contain reductions which reduce

Figure 2: 24 possible representations for the string abc of length 3 that a CFG can generate. The upper part shows possible bracketing and the lower part shows the respective derivation trees. Each bracket in the upper part is replaced by a variable in the lower part of the figure.

	i	ii	iii	iv
1	(a b c)	((a b c))	(a (b c))	(a b (c))
2	((a) (b) c)	((a) (b) (c))	(a (b) (c))	((a) (b) (c))
3	((a b) c)	((a b) (c))	(a (b c))	((a) (b c))
4	((a) (b) c)	((a) (b) (c))	(a ((b) c))	((a) ((b) c))
5	((a (b)) c)	((a (b)) (c))	(a (b (c)))	((a) (b (c)))
6	((a) ((b) c))	((a) ((b) (c)))	(a ((b) (c)))	((a) ((b) (c)))

The derivation tree T_{1i} has depth 0, T_{1ii} to T_{3iv} have depth 1 and T_{4i} to T_{6iv} have depth 3. The top-most node is omitted. The derivation trees T_{1ii} to T_{2iv} contain variables which reduce chunks of length $l = 1$. The trees T_{3i} to T_{6iv} contain variables which reduce chunks of length $l = 2$. The shape of the partial derivation trees which reduce a chunk of length $l = 2$ are similar to the representations shown in Figure 1.

	i	ii	iii	iv
1	(a b c)	(\mathcal{X} b c) a	(a \mathcal{X} c) b	(a b \mathcal{X}) c
2	(\mathcal{X} \mathcal{X} c) a b	(\mathcal{X} b \mathcal{X}) a c	(a \mathcal{X} \mathcal{X}) b c	(\mathcal{X} \mathcal{X} \mathcal{X}) a b c
3	(\mathcal{X} c) (ab)	(\mathcal{X} \mathcal{X}) (ab) c	(a \mathcal{X}) (bc)	(\mathcal{X} \mathcal{X}) a (bc)
4	(\mathcal{X} c) (\mathcal{X} b) a	(\mathcal{X} \mathcal{X}) (\mathcal{X} b) c a	(a \mathcal{X}) (\mathcal{X} c) b	(\mathcal{X} \mathcal{X}) a (\mathcal{X} c) b
5	(\mathcal{X} c) (a \mathcal{X}) b	(\mathcal{X} \mathcal{X}) (a \mathcal{X}) c b	(a \mathcal{X}) (b \mathcal{X}) c	(\mathcal{X} \mathcal{X}) a (b \mathcal{X}) c
6	(\mathcal{X} c) (\mathcal{X} \mathcal{X}) a b	(\mathcal{X} \mathcal{X}) (\mathcal{X} \mathcal{X}) c a b	(a \mathcal{X}) (\mathcal{X} \mathcal{X}) b c	(\mathcal{X} \mathcal{X}) a (\mathcal{X} \mathcal{X}) b c

chunks of length $l = 2$. Each of the trees T_{4m} T_{6m} are variants of the respective representation T_{3m} . Thus, T_{3ii} for instance, contains two reductions: the leftmost reduction \mathcal{X} reduces the chunk ab and the rightmost reduction \mathcal{X} reduces the chunk c . The trees T_{3ii} , T_{4ii} , T_{5ii} and T_{6ii} differ with respect to how the the chunk ab is reduced where the sub-trees reducing the chunk ab are similar to the derivation trees shown in Figure 1. Any chunk of length $l =$

$i, i < n$ has thus $R^\#(i)$ different representations.

The trees T_{3i} T_{4i} , T_{5i} and T_{6i} are similar to the trees T_{3ii} to T_{6ii} , respectively, with the difference that the chunk c is not reduced. In the derivation trees T_{3ii} to T_{6ii} a different chunking takes place. Here the top-most bracketing is $(a (bc))$ and in the derivation trees T_{3iv} to T_{6iv} the bracketing corresponds to $((a) (bc))$.

The table in Figure 3 puts these considerations into

Figure 3: Calculating the number of possible representations $R^\#(3)$ for the string abc of length $l = 3$. The second column shows the decomposition of the string abc where the brackets indicate reductions on the top-most level. The first column points to the respective derivation trees plotted in Figure 2. The third column indicates the number and length of the topmost bracketed chunk and the fourth column shows for a string of length $l = 3$ how many of such chunk combinations can be generated. The fifth column shows the partial generalizations abstracting away from any particular value n .

tree	bracketing	length of chunks	$\#l$	number of chunks	
1ii	((a)bc)	$l = 1$	$\binom{3}{1}$	$\sum_{l=1}^n l \binom{n-l+1}{1}$	
1iii	(a(b)c)				
1iv	(ab(c))				
3i-6i	((ab)c)	$l = 2$	$\binom{2}{1}$		
3iii-6iii	(a(bc))				
1i	(abc)	$l = 3$	$\binom{1}{1}$		
2i	((a)(b)c)	$l_1 = 1, l_2 = 1$	$\binom{3}{2}$		$\sum_{l_1=1}^n \sum_{l_2=1}^{n-l_1} l_1 l_2 \binom{n-l_1-l_2+2}{2}$
2ii	((a)b(c))				
2iii	(a(b)(c))				
3ii-6ii	((ab)(c))	$l_1 = 2, l_2 = 1$	$\binom{2}{2}$		
3iv-6iv	((a)(bc))	$l_1 = 1, l_2 = 2$	$\binom{2}{2}$		
2iv	((a)(b)(c))	$l_1 = 1, l_2 = 1, l_3 = 1$	$\binom{3}{3}$		
				$\sum_{l_1=1}^n \sum_{l_2=1}^{n-l_1} \sum_{l_3=1}^{n-l_1-l_2} l_1 l_2 l_3 \binom{n-l_1-l_2-l_3+3}{3}$	

Figure 4: In (1) below, the number of bracketings $l(i)$ are rewritten in terms of number of representations $R^\#(i)$. To avoid recursion, the sum on the right-hand side counts from 1 to $n - 1$. The number of CF-representations for $n > 0$ can be computed according to equation 2.

$$\sum_{i=1}^n l(i) \binom{n-i+1}{1} \implies 1 + \sum_{i=1}^{n-1} R^\#(i) \binom{n-i+1}{1} \quad (1)$$

$$\begin{aligned} R^\#(n) &= 1 + \sum_{i=1}^{n-1} R^\#(i) \binom{n-i+1}{1} \\ &+ \sum_{i_1=1}^n \sum_{i_2=1}^{n-i_1} R^\#(i_1) R^\#(i_2) \binom{n-i_1-i_2+1}{2} \\ &+ \dots \\ &+ \sum_{i_1=1}^n \sum_{i_2=1}^{n-i_1} \dots \sum_{i_n=1}^{n-i_1-\dots-i_{n-1}} R^\#(i_1) R^\#(i_2) \dots R^\#(i_n) \binom{n}{n} \end{aligned} \quad (2)$$

a more formal context. For a string of length n there are $\binom{n}{1} = n$ possibilities to chose one bracketed chunk of length $l = 1$; there are $\binom{n-1}{1}$ possibilities to chose one bracketed chunk of length $l = 2$ plus there ar $\binom{n-2}{1}$ possibilities to chose one chunk of length $l = 3$ plus ... plus there is $\binom{1}{1} = 1$ possibility to chose one bracketed chunk of length $l = n$.

In brief we have

$$\begin{aligned} &\binom{1}{1} * L(l = n) + \binom{2}{1} * L(l = (n - 1)) + \\ &\quad + \dots + \\ &+ \binom{n-1}{1} * L(l = 2) + \binom{n}{1} * L(l = 1) \\ &= \sum_{l=1}^n l \binom{n-l+1}{1} \end{aligned}$$

possibilities to chose one bracketed chunk of length

Figure 5: The table puts into relation possible CF-tree representations $R^\#(n)$ for a string of length n and the number of permutations $n!$. At $n = 20$ the number of permutations exceeds the number of possible representations $R^\#(n)$.

n	$R^\#(n)$	$n!$	$R^\#(n) - n!$
1	1	1	0
2	4	2	2
3	24	6	18
4	176	24	152
5	1440	120	1320
6	12608	720	11888
7	115584	5040	110544
8	1095424	40320	1055104
9	10646016	362880	10283136
10	105522176	3628800	101893376
11	1062623232	39916800	1022706432
12	10840977408	479001600	10361975808
13	111811534848	6227020800	105584514048
14	1163909087232	87178291200	1076730796032
15	12212421230592	1307674368000	10904746862592
16	129027376349184	20922789888000	108104586461184
17	1371482141884416	355687428096000	1015794713788416
18	14656212306231296	6402373705728000	8253838600503296
19	157369985643577344	121645100408832000	35724885234745344
20	1696975718802522112	2432902008176640000	-735926289374117888

$1 \leq l \leq n$ out of a string of length n as shown in the table in Figure 3.

In addition, choosing two chunks of length l_1 and l_2 we will have the following bracketing possibilities which amount to:

$$\sum_{l_1=1}^n \sum_{l_2=1}^{n-l_1} l_1 l_2 \binom{n-l_1-l_2+2}{2}$$

To obtain the overall number of representations, these partial results are summed up as shown in equation 1 in Figure 4. Since for each chunk of length $l_i = i, 0 < i < n$ there are $R^\#(i)$ different representations, $l(i)$ can be replaced by $R^\#(i)$ in equation 2.

For each string of length n there is exactly one representation of a chunk of length $l = n$ i.e. the whole string is recognized as one chunk only, e.g. T_{1i} in Figure 2. Therefore, the expression on the left-hand side in the implication (1) (Figure 4) can be rewritten in terms of $1 + \sum_{i=1}^{n-1}$ where all $i < n$. In the recursive equation 2 we can be sure that for any $n > 0$ the number of representations $R^\#(n)$ is computed based on the number of representations $R^\#(i)$ and $i < n$.

The table in Figure 5 plots the values of $R^\#(n)$, $n!$ and $R^\#(n) - n!$ for $n = 1 \dots n = 21$. The third column shows that the number of representations

$R^\#(n) > n!$ for $0 < n < 20$.

2 Representational Complexity in Generalized Exemplar Based Machine Translation

Machine Translation can be considered as the permutation of word translations from a source language string into the target language via some internal representation. Shake-and-bake translation (Whitelock, 1992) implements this strategy in a direct manner: permutations of target language words are parsed with a target language parser and the best fitting permutation is considered the best translation.

Almost every available MT system works on a sentence level and, probably, most of the sentences to be translated in an automatic system do not exceed the length of 20 words. Assuming that any CF representation can be mapped into any target sentence permutation, we can state that machine translation has at least the complexity of the internal representations $R(n) > n!$ for an input string of length n .

Example-Based Machine Translation (EBMT) seek to infer from a corpus of reference translations a context-free bi-grammar. This bi-grammar is used to map a source language string into a target language. However, the inferred grammars differ drastically in the various approaches. The ReVerb EBMT

system (Collins, 1999; Collins and Cunningham, 1997), for instance, only computes trees of depth 1 while it allows an arbitrary number of chunks. The optimal length of each chunk is calculated by statistic means. ReVerb thus generates trees similar to $T_{1i} \dots T_{3iv}$ as depicted in Figure 2. In EDGAR (Carl, 1999), in principle, all derivation trees can be generated. The bi-grammar inference mechanism, however, seeks to generate only maximally invertible grammars (cf. (Carl and Wu, 1999)).

In this section, yet a different approach shall be examined. In (Güvenir and Cicekli, 1998; Güvenir and Tunc, 1996) translation templates i.e. bi-production rules containing non-terminals are inferred from a set of reference translations. There are no restrictions on the shape of the inferred translation templates such that for a sentence s to be translated, as many as $R(s)$ representations may be generated. I shall first outline how translation template inference is achieved in their so-called Translation Template Learner (TTL) algorithm. I will give evidence for its potential to generate a universal grammar at least for some strings. I will then discuss their proposed heuristics to tackle the problem.

An initial example translation base consists of a set of translation examples E of the form $E : a_{1:m} \leftrightarrow b_{1:n}$, where $a_{1:m}, 1 \leq m$ is a sequence of source language words and is a sequence of target language words.

For each pair of translation examples $E^g : a_{1:m}^g \leftrightarrow b_{1:n}^g$ and $E^h : a_{1:m}^h \leftrightarrow b_{1:n}^h, g \neq h$ source language sub-strings $S^1 : a_{1:k}^{g,h}, k \leq m^g, m^n$ and target language sub-strings $S^2 : b_{1:l}^{g,h}, l \leq n^g, n^h$ are sought which appear in both translation examples E^g and E^h . For instance, the two English \leftrightarrow Turkish translation examples E^1 and E^5 contain two pairs of identical substrings:

$$E^1 : \frac{\text{I took a ticket from Mary} \leftrightarrow \text{Mary'den bir bilet aldim}}{\text{Mary'den bir bilet aldim}}$$

$$E^5 : \frac{\text{I took a pen from Mary} \leftrightarrow \text{Mary'den bir kalem aldim}}{\text{Mary'den bir kalem aldim}}$$

Translation templates are inferred by substituting differences in the translation examples by non-labeled, linked variables. Thus, if a pair of translation examples E^g and E^h contain equal sub-strings S_1^1 and S_2^1 on their source language side and equal sub-strings S_1^2 and S_2^2 on the target language sides such that E^g and E^h can be re-written as follows.

$$E^g : S_1^1, D_1^g, S_2^1 \leftrightarrow S_1^2, D_2^g, S_2^2$$

$$E^h : S_1^1, D_1^h, S_2^2 \leftrightarrow S_1^2, D_2^h, S_2^2$$

where

$$D_1^g \neq D_1^h \text{ and } D_2^g \neq D_2^h$$

A translation template of the form $(S_1^1, \mathcal{X}^1, S_2^1 \leftrightarrow S_1^2, \mathcal{X}^2, S_2^2$ if $\mathcal{X}^1 \leftrightarrow \mathcal{X}^2$) can be generated and the dissimilar sub-strings $(D_1^g, D_2^g; D_1^h, D_2^h)$ are replaced by the non-labeled variables \mathcal{X}^1 and \mathcal{X}^2 . In addition to this, the translation rules $D_1^g \leftrightarrow D_2^g$ and $D_1^h \leftrightarrow D_2^h$ are extracted. In the example above this procedure yields the translation template and translation examples:

$$E^1/E^5 : \frac{\text{I took a } \mathcal{X}^1 \text{ from Mary} \leftrightarrow \text{Mary'den bir } \mathcal{X}^2 \text{ aldim}}{\text{if } \mathcal{X}^1 \leftrightarrow \mathcal{X}^2}$$

ticket \leftrightarrow bilet
pen \leftrightarrow kalem

The TTL-algorithm fails, when a different number of dissimilarities in the source and target language strings are encountered or if the number of dissimilarities is greater than one. In order to handle such translation examples, an iterative algorithm is described, which successively reduces the number of dissimilarities based on the translation examples extracted in former iterations. Provided that there is one different dissimilarity between each pair of example translations (and at least one identical substring), the algorithm generates maximally $((m-1)^2 + (m-1))/2$ non-terminal bi-production rules, where m is the number of example translations in the initial example base.

To give an example, consider the translation examples E^1 to E^7 in Figure 6. By applying the TTL algorithm to each pair of the seven translation examples, $(6^2 + 6)/2 = 21$ translation templates are generated, provided that the target language examples map in a similar way. By adding further examples to the base, one can easily see that a universal grammar can be generated. For instance, adding the translation example “You steal one pen from Mary” generates further 7 translation templates.

There is no discussion in (Güvenir and Cicekli, 1998; Güvenir and Tunc, 1996; Öz and Cicekli, 1998) on how consistent the inferred bi-production rules are and how consistency could be achieved. The authors propose a top-down translation heuristic where only the most specific bi-production rules are chosen, containing the greatest number of terminal symbols. By this heuristic, only the flattest among the possible derivation trees are selected. In another paper (Öz and Cicekli, 1998) by Öz and Cicekli it is recognized that this heuristic needs further refinement. For a

Figure 6: From the 7 translation examples E^1 to E^7 the TTL algorithm generates 21 translation templates as shown in the lower part of the picture.

E^1	I	took	a	ticket	from	Mary
E^2	You	took	a	ticket	from	Mary
E^3	I	steal	a	ticket	from	Mary
E^4	I	took	one	ticket	from	Mary
E^5	I	took	a	pen	from	Mary
E^6	I	took	a	ticket	for	Mary
E^7	I	took	a	ticket	from	John
E^1/E^2	\mathcal{X}_1	took	a	ticket	from	Mary
E^1/E^3	I	\mathcal{X}_1	a	ticket	from	Mary
E^1/E^4	I	took	\mathcal{X}_1	ticket	from	Mary
E^1/E^5	I	took	a	\mathcal{X}_1	from	Mary
E^1/E^6	I	took	a	ticket	\mathcal{X}_1	Mary
E^1/E^7	I	took	a	ticket	from	\mathcal{X}_1
E^2/E^3	\mathcal{X}_1	\mathcal{X}_2	a	ticket	from	Mary
E^2/E^4	\mathcal{X}_1	took	\mathcal{X}_2	ticket	from	Mary
E^2/E^5	\mathcal{X}_1	took	a	\mathcal{X}_2	from	Mary
E^2/E^6	\mathcal{X}_1	took	a	ticket	\mathcal{X}_2	Mary
E^2/E^7	\mathcal{X}_1	took	a	ticket	from	\mathcal{X}_2
E^3/E^4	I	\mathcal{X}_1	\mathcal{X}_2	ticket	from	Mary
E^3/E^5	I	\mathcal{X}_1	a	\mathcal{X}_2	from	Mary
E^3/E^6	I	\mathcal{X}_1	a	ticket	\mathcal{X}_2	Mary
E^3/E^7	I	\mathcal{X}_1	a	ticket	from	\mathcal{X}_2
E^4/E^5	I	took	\mathcal{X}_1	\mathcal{X}_2	from	Mary
E^4/E^6	I	took	\mathcal{X}_1	ticket	\mathcal{X}_2	Mary
E^4/E^7	I	took	\mathcal{X}_1	ticket	from	\mathcal{X}_2
E^5/E^6	I	took	a	\mathcal{X}_1	\mathcal{X}_2	Mary
E^5/E^7	I	took	a	\mathcal{X}_1	from	\mathcal{X}_2
E^6/E^7	I	took	a	ticket	\mathcal{X}_1	\mathcal{X}_2

small example base containing 488 translation examples, the TTL algorithm generates as much as 4723 translation templates. The most specific bi-production rule heuristic seems no more appropriate. Instead, bi-production rules are now weighted and translations are ordered according to their confidence value. However, this does not reduce the number of generated trees, it only gives a means to select those with the highest confidence values. In addition the authors admit that there are more than 100,000 weights (50,000 for each translation direction) to be managed, and this number is likely to increase at least linearly with the number of translation templates generated.

3 Conclusion

The paper discusses the complexity of representations which can be derived from unrestricted context-free grammars. It is shown that there are

more than $n!$ possible derivation trees for a string of length $1 < n < 20$. Given this untractable representational complexity, all computer systems which make use of such representations, need to use some heuristics to handle the amount of generated derivation trees. One basic solution to approach this problem is examined on the basis of the Generalized Exemplar-Based Machine Translation system as described in (Güvenir and Cicekli, 1998; Güvenir and Tunc, 1996; Öz and Cicekli, 1998). It is shown, that this system has the potential to generate all possible representations. Two heuristics are proposed in their papers (Güvenir and Cicekli, 1998) and (Öz and Cicekli, 1998) to tackle the problem at runtime. The soundness of this method is doubted.

Acknowledgement:

I would like to thank Jean-Luc Bonafacino and YAO Kaisheng for their patience to discuss the many questions I asked them to figure out the formula presented here.

References

- Michael Carl and Dekai Wu. 1999. Inferring maximally invertible bi-grammars for example-based machine translation. In *NLPRS 99*.
- Michael Carl. 1999. Inducing Translation Templates for Example-Based Machine Translation. In *MT-Summit VII*.
- 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 Examples. *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.
- Zeynep Öz and Ilyas Cicekli. 1998. Ordering Translation Templates by Assigning Confidence Factors. In *Third Conference of AMTA*, pages 51–62.
- Peter Whitelock. 1992. Shake-and-bake translation. In *COLING-92*, pages 784–789.