Proceedings of the International Multiconference on Computer Science and Information Technology pp. 67–73
ISSN 1896-7094
© 2006 PIPS

An Algorithm for Extracting Translation Rules from Scarce Bilingual Corpora

Jassem Krzysztof, Kowalski Tomasz

Abstract. We propose a method for automatical extraction of translation rules suitable for a rule-based machine translation system, by using a target language syntactic parser and scarce bilingual resources as linguistic knowledge sources.

Introduction

In recent years we observe a growing interest in Statistical Machine Translation (SMT). To a large extent this is due to the increasing volume of parallel bilingual data for major languages. Moreover, the statistical approach involves a lot of computation time to train the model but almost no manpower.

On the other hand, linguistic knowledge acquired in the training phase of a statistical model does not allow for any human tuning, which makes it difficult to correct potential errors.

In Rule-based Machine Translation (RMT), linguistic knowledge is hand-coded into the system. The translation process may be tuned according to the language-specific features. RMT systems deliver translations of better quality than SMT systems, but are much more expensive as they require manpower with both linguistic and computer programming skills in order to encode linguistic knowledge into a machine-readable form.

It would be desirable to combine the advantages of both approaches. To do so one may build an RMT system with the rules being achieved automatically – by means of statistical calculations.

The idea to require transfer rules automatically from a word-aligned corpus has been ([1, 2, 5, 6]). There, transfer units are based on subtrees in the source language parse tree.

In this paper we would like to focus on the possibility of obtaining translation rules without using a source language parser. We assume the availability of a parser for a target language and scarce bilingual resources in the form of word aligned sentences.

Galley et al. ([3]) propose a theory that gives formal semantics to word-level alignments defined over parallel corpora. They introduce a linear algorithm that can be used to derive the minimal set of syntactically motivated transformation rules from word-aligned parallel corpora. The transformation rules are extracted from the parse tree of the target sentence and the pre–determined equivalents of its nodes in the source sentence.

We modify that algorithm by using a different annotation schema for analyzed graphs and by using a binary vector algebra instead of set operations. The modified algorithm is used to extract translation rules for an RMT German-Polish system. We perform experiments with small German-Polish corpora and parse target sentences with the state-of-the-art non-statistical parser for Polish ([4]) used in the TRANSLATICA system[1].

The development of an RMT system based on this idea requires some human knowledge – it is needed for tagging the equivalence between simple components (usually words) in the parallel sentences (this may be done by just verifying the suggestions of a computer program). Such tagging, however, does not require specialized computer or linguistic skills of human operators.

Rule extraction

We extract rules later used in the translation process, from a structure called an alignment graph ([3]). An alignment graph AG is a rooted, directed (in the diagrams, direction is usually presented from top to down), connected, acyclic graph spanned over the tokens of the source sentence and the parse tree of the target sentence. We denote the set of nodes in a graph G by $V(G)$ and the set of edges by $E(G)$. We use the notation (n_1, v_2), where $n_1, n_2 \in V(G)$ to describe the edge from the node a to b in the graph G. It consists of the set of source sentence nodes S, i.e. nodes that represent words (more precisely: tokens) of the source sentence, a parse tree P of the target sentence, where $T \subset V(P)$ denotes the set of leaves

[1] TRANSLATICA (`http://www.translatica.pl`) is a commercial name of a system developed formerly under the name POLENG

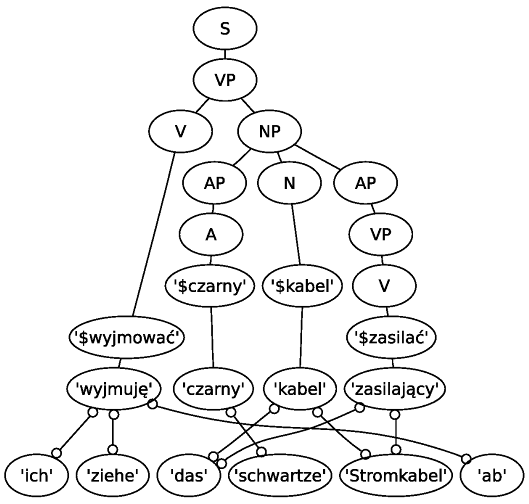


Fig. 1. An example of an alignment graph

of the parse tree, that is the target sentence nodes, and an alignment A, i.e. edges between T and S: $A = \{(t, s) \in E(AG) : t \in T, s \in S\}$. We also require that $\forall_{s \in S} \exists_{t \in T} (t, s) \in A$. Figure 1 shows an example of an alignment graph.

The idea is to extract automatically subgraphs of the alignment graph that would generate translation rules.

Definition 1. *Let n be a node of the parse tree ($n \in V(P)$), such that removing the edge between node n and its parent dissects the alignment graph into two disjoint graphs. We call the graph rooted in node n* rule–inducing *if it covers a contiguous part of the source sentence, i.e. source sentence nodes of that graph form a substring of the source sentence.*

An example of such a subgraph is shown in Figure 2.

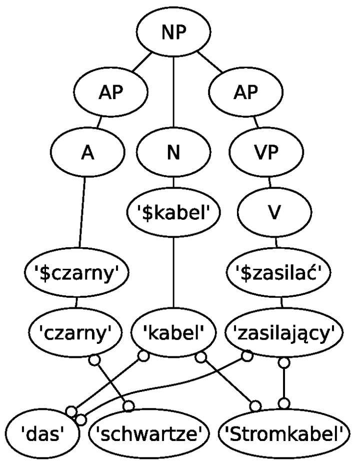
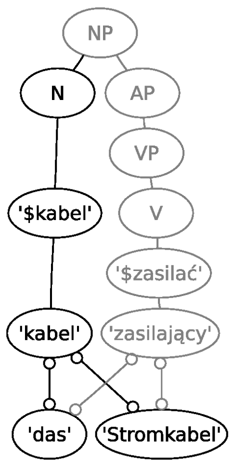


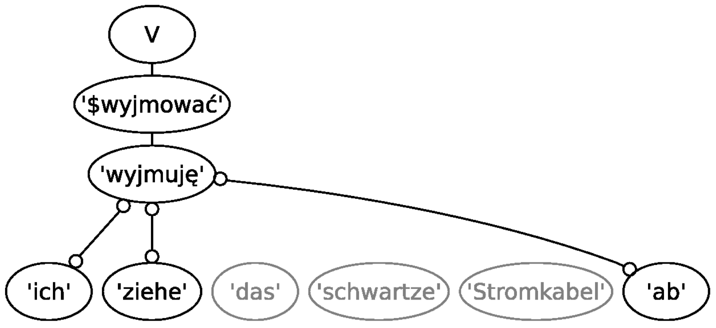
Fig. 2. An example of a rule-inducing graph

Figure 3(a) shows a subgraph that does not induce a translation rule because removing the edge between its root and the root's parent does not dissect the alignment graph into two disjoint graphs. Figure 3(b) shows a subgraph that does not induce a translation rule because it covers a non-contiguous part of the source sentence. The algorithm we describe below analyses the alignment graph in a single traversal in order to find all rule–inducing subgraphs.

First, the algorithm defines the order of the alignment edges "from left to right": number 0 is assigned to the left-most edge of the alignment, i.e. the edge that links the first token of the source sentence to



(a) The "R" node does not become the root of a graph when detached from its parents.



(b) The "C"-rooted graph does not cover a contiguous part of the source sentence.

Fig. 3. Examples of non rule-inducing graphs.

the left–most token of the corresponding target sentence tokens; $v = card(A)$ is assigned to the right-most edge of the assignment. More precisely: Let $(t_1, s_1), (t_2, s_2) \in A$. Each alignment edge is assigned a number $n((t, s) \in A) \in [1, v]$ so that the following should hold:

$$(t_1, s_1) \prec (t_2, s_2) \;\Leftrightarrow\; s_1 \underset{S}{\prec} s_2 \;\vee\; s_1 = s_2 \wedge t_1 \underset{T}{\prec} t_2 \tag{1}$$

For each node m in the alignment graph, the algorithm calculates two binary vectors, $span(m)$ and $mask(m)$ of length v equal to the number of edges in the alignment ($|span(m)| = |mask(m)| = card(A)$). For each $s \in S$ the vectors are:

$$span(s \in S) = 0 \tag{2}$$

$$mask(s \in S) = [m_v, .., m_1] : m_i = \begin{cases} 1, \exists t \in T : i = n((t, s) \in A) \\ 0, otherwise \end{cases} \tag{3}$$

For each $t \in T$ the vectors are:

$$span(t \in T) = [m_v, .., m_1] : m_i = \begin{cases} 1, \exists s \in S : i = n((t, s) \in A) \\ 0, otherwise \end{cases} \tag{4}$$

$$mask(t \in T) = \bigvee_{(t,s) \in A} mask(s) \tag{5}$$

For the remaining nodes the vectors are calculated by traversing the alignment graph bottom–up.

$$span(n \in V(P) \setminus T) = \bigvee_{(n,m) \in E(P)} span(m) \tag{6}$$

$$mask(n \in V(P) \setminus T) = \bigvee_{(n,m) \in E(P)} mask(m) \tag{7}$$

Listing 1.

```
#initializing source sentence nodes
$edge_number = 1;
foreach $s (@S)
{
  foreach $t ($s->{aligned_to})
  {
    $t->{span} |= $edge_number;
    $s->{mask} |= $edge_number;
    $edge_number << 1;
  }
  create_a_discard_rule($s)  if( 0 == $s->{mask} );
}
#initializing target sentence nodes
foreach $t (@T)
{
  $t->{mask} |= $s->{mask}  foreach $s ($t->{aligned_to});
  $seen{$n} = 1;
}
#traversing the rest of the alignment graph bottom--top
@stack = ( $r ); #root of the alignment graph
while ( $n = pop @stack )
{
    unless ( $seen{$n} )
    {
        $seen{$n} = 1;
        push @stack, $n;
        push @stack, $n->{siblings};
    }
    else
    {
        foreach $c in ($n->{siblings})
        {
            $n->{span} |= $c->{span};
            $n->{mask} |= $c->{mask};
        }
        if ( $n->{span} == $n->{mask}
           && binary_to_string($n->{span}) =~ /^0*1+0*$/
           )
        {
            convert_to_rule($n);
            @n->{siblings} = (); #detaching all siblings
        }
    }
}
```

It is claimed here that:

Lemma 1. *For any node $p \in P$, mask(p) = span(p) iff removing the edge e' between the node p and its parent dissects the alignment graph into two disjoint graphs.*

Proof. Let R be a p-rooted graph.

Suppose that $mask(p) = span(p)$ and that removing the edge e' does not result in the partition of the alignment graph into two disjoint graphs. Since P is a tree (and thus removing any edge results in the partition of the tree into two disjoint trees) there has to be an edge $e = (t, s)$ in the alignment that either $t \notin V(R) \wedge s \in V(R)$ or $t \in V(R) \wedge s \notin V(R)$. Suppose that $t \in V(R) \vee s \notin V(R)$. According to the equation 4: $span(t)_{n(e)} = 1$. Since $span(p)$, calculated according to the equation 6, is effectively

$$span(p) = \bigvee_{x \in V(R) \cap T} span(x) \tag{8}$$

and $span(t)_{n(e)} = 1$ then $span(p)_{n(e)} = 1$. We assumed that $mask(p) = span(p)$ so $mask(p)_{n(e)} = 1$. By unwinding the recursive equation 7 we get

$$mask(p) = \bigvee_{x \in V(R) \cap S} mask(x) \tag{9}$$

Therefore $\exists_{s_1 \in V(R) \cap S}\, mask(s_1)_{n(e)} = 1$. But since edges are numbered uniquely (relation 1) and $mask(s_1)_{n(e)} = 1 = mask(s)_{n(e)}$ then $s_1 = s$, which results in contradiction because we assumed that $s \in V(R)$ but $s_1 \notin V(R)$. A similar reasoning for the other case $(t \notin V(R) \vee s \in V(R))$ leads also to contradiction.

Now suppose that removing the edge e' results in partition of the alignment graph into two disjoint graphs and $mask(p) \neq span(p)$. If $mask(p) \neq span(p)$ then $\exists i = n(e) = n((t, s)) :\ mask(p)_i \neq span(p)_i$. Since $\{e'\}$ is the edge cut of the alignment graph, $t, s \in V(R)$ or $t, s \notin V(R))$. Let us assume that $span(p)_i = 0$ and $mask(p)_i = 1$. Given the equation 8 we get $1 = mask(p)_{i=n(e)=n(t,s)} = mask(s)_{n(t,s)}$. By definition of the *span* and *mask* vectors: $mask(s)_{n(t,s)} = span(t)_{n(t,s)}$. Since $s \in V(R)$, $t \in V(R)$ and the following holds: $1 = span(t)_{n(t,s)} = span(p)_{n(t,s)} = span(p)_i$.
A similar reasoning for the case $span(p)_i = 1$ and $mask(p)_i = 0$ leads also to contradiction. □

Lemma 2. *For any node $p \in P$, the p–rooted graph is rule–inducing iff*

$$span(p) = mask(p) = [\underbrace{0..0}_{a-1}, \underbrace{1..1}_{b-a+1}, \underbrace{0..0}_{v-b}],\ \textit{where}\ a \in [1..v],\ b \in [a..v]$$

Proof. According to Definition 1 a p-rooted subgraph of the alignment graph is rule-inducing iff (a) we are able to dissect the alignment graph into two disjoint graphs and (b) the leaves of the p–rooted graph form a contiguous part of the source string.

According to Lemma 1 the first condition is met when $span(p) = mask(p)$. Thus, it suffices to prove that the condition (b) is satisfied by a “0*1+0*” vector, that is a sequence of zero or more zeros, followed by one or more ones, followed by zero or more zeros.

(⇐) Let us assume that $span(p) = mask(p)$ and $mask(p)$ has the form “0*1+0*” and that the p-rooted subgraph (R) of the alignment graph does not cover a substring of the source sentence. This would imply that there has to be a sequence of source nodes of the length m such that: the first node in sequence $s_1 \in S \cap V(R)$ is followed (in terms of the source sentence) by one or more nodes $\forall_{i \in (1,m)} s_i \in S \wedge s_i \notin V(R)$, followed by a node $s_m \in S \cap V(R)$. According to equation 9:

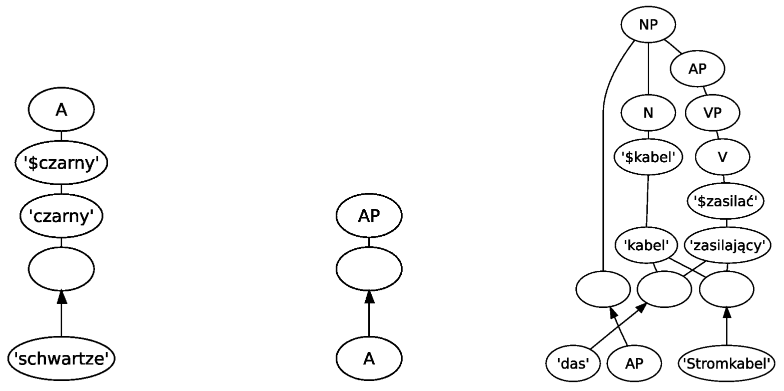
$$\begin{aligned} \forall t \in T :&\ [mask(p)]_{n_1 = n(t,s_1)} = 1 \\ \forall i \in (1, m)\ \forall t \in T :&\ [mask(p)]_{n_i = n(t,s_m)} = 0 \\ \forall t \in T :&\ [mask(p)]_{n_m = n(t,s_m)} = 1 \end{aligned}$$

But since $\forall_{i \in (1,m)} n_1 < n_i < n_m$ we observe a sequence of the form “10+1” (a single one, followed by one or more zeros, followed by a single one), which contradicts to the assumption that the *mask* vector should have a “0*1+0*” form.

(⇒) Let L be the set of leaves of the p–rooted rule–inducing graph. Since the leaves of a rule–inducing graph constitute a substring of the source sentence and the order of numbering is “left–to–right” (relation 1), the alignment edges such that $\forall_{s \in L} \exists_{t \in T} (t, s) \in A$ are labeled with subsequent numbers. Moreover for any $t \in T$ and $s \in L$ let $a = min\{n((t, s) \in A)\}$ and $b = max\{n((t, s) \in A)\}$. According to the equation 9 we get $\forall_{i \in [a,b]} [mask(p)]_i = 1$. For any source node s_i that precedes (in terms of the source sentence) any of the node in L, we have $\forall_{t \in T} n(t, s_i) \in A) < a$. For any node $s_i \notin V(R)$ we have $\forall_{i \in [0..a)} [mask(p)]_i = 0$. Similarly, for any source node s_j that succeeds' (in terms of the source sentence) any of the nodes in L, we have $\forall_{t \in T} b < n(t, s_i) \in A)$. Since any node $s_j \notin V(R)$, we have $\forall_{i \in (b..v]} [mask(p)]_i = 0$. □

Listing 1 shows the complete algorithm in the form of a Perl–like code. In order to simplify the exposition we convert the *span* vector into a string and use a regular expression to check whether it has the required form.

Once the root of a rule-inducing graph is found we convert it into a rule using the `convert_to_rule` procedure. The procedure forms the rule input of leaves of the given graph ([3]). We order the leaves according to the *mask* vector. Note that the sorted list of leaves often differs from the list obtained by in-depth traversal of the rule-inducing graph. Therefore we replace the leaves of the rule-inducing graph (now the rule body) by references to the sorted list (rule input). Figure 4 shows examples of a translation



(a) A dictionary–like rule. (b) A parser rule. (c) A translation rule with source tokens in input.

Fig. 4. Examples of translation rules.

rule formed by the `convert_to_rule` procedure. The input of a rule is shown at the bottom of every diagram with the rule's body (production) above it. Arrows between the input and the body of a rule indicate where the nodes matching the input sequence should be placed. The `form_a_discard_rule` procedure creates a special kind of translation rules. Those rules have an empty input body. They store the information that a particular token, is irrelevant to the translation process, since it was not aligned to any target token.

A rule-inducing graph may contain smaller rule-inducing graphs. If all rule-inducing graphs were converted into rules, a lot of the information contained in the rules would be stored multiple times. To avoid that overload each time a rule-inducing graph is found and converted into a rule we reduce that graph to its root in the alignment graph. Figure 4(b) shows such a rule. The rule was created after the "P"-rooted rule-inducing subgraph of the "FP"-rooted rule-inducing subgraph had been reduced to a single node.

Rule types

We observe tree types of rules acquired by the above algorithm: (a) dictionary–like rules, (b) parsing rules and (c) mixed rules. Examples of rules of each type are shown in Figure 4.

A dictionary-like rule consists of four nodes: source sentence node—source token, target sentence node—its equivalent in the target language, a node that identifies the base form of the equivalent, and a node containing grammatical information. The rule shown in Figure 4(a) was actually achieved in a post-processing step from three extracted rules, each of them consisting of only two connected nodes. It is worth noticing that it would be certainly possible to acquire a complete bilingual dictionary in this way, but it would require large corpora. Having only scarce bilingual resources, dictionary rules have to be supplied from external sources.

A parsing rule consists of P-nodes only. Those rules will most likely form only a subset of the knowledge base for the parser of target sentences because the parser may be able to deal with cases not observed in the corpora. Thus, it would be better to transfer the parser's complete knowledge base into parsing rules by some other means.

A third type of the acquired rules are mixed rules, i.e. rules consisting of source sentence nodes and parse tree nodes. We believe that rules of that type are able to capture translation nuances that a human operator (e.g. an author of translation rules for a transfer-based MT system) may not be aware of.

Actually, the parser used here takes into account also semantic features of sentence components. This shows that acquired rules may have not only syntactical but also semantic motivation.

References

1. Carbonell J., Probst K., Peterson E., Monson Ch., Lavie A., Brown R., and Levin L.: *Automatic Rule Learning for Resource-Limited MT.,* Proceedings AMTA-02. (2002)

2. Dekang L.: *A path-based transfer model for MT,* Proceedings COLING-04. (2004).
3. Galley M., Hopkins M., Knight K., Marcu D.: *What's in a translation rule?* Proceedings HLT/NAACL-04. (2004).
4. Graliński F.: *Wstępujący parser języka polskiego na potrzeby systemu POLENG, Speech and Language Technology. Volume* **6** *(2002).*
5. *Lavoie B., White M., and Korelsky T.: Learning Domain-Specific Transfer Rules: An Experiment with Korean to English Translation, Proceedings COLING-02. (2002).*
6. *Richardson S., Dolan W., Menezes A., and Pinkham J.: Achieving commercial-quality translation with example-based methods, Proceedings MT Summit VIII. (2001).*