3.2: Mathematical Model of Tree Transformations

Ondřej Bojar, Martin Čmejrek

Distribution: Public

EuroMatrix
Statistical and Hybrid Machine Translation
Between All European Languages
IST 034291 Deliverable 3.2

November 30, 2007

Project funded by the European Community
under the Sixth Framework Programme for
Research and Technological Development.

Project ref no.	IST-034291
Project acronym	EuroMatrix
Project full title	Statistical and Hybrid Machine Translation Between All European Languages
Instrument	STREP
Thematic Priority	Information Society Technologies
Start date / duration	01 September 2006 / 30 Months

Distribution	Public
Contractual date of delivery	December 6, 2007
Actual date of delivery	November 30, 2007
Deliverable number	3.2
Deliverable title	Mathematical Model of Tree Transformations
Type	
Status & version	
Number of pages	32
Contributing WP(s)	3
WP / Task responsible	Jan Hajič
Other contributors	
Author(s)	Ondřej Bojar, Martin Čmejrek
EC project officer	Xavier Gros
Keywords	

The partners in EuroMatrix are:	Saarland University (USAAR)
	University of Edinburgh (UEDIN)
	Charles University (CUNI-MFF)
	CELCT
	GROUP Technologies
	MorphoLogic

Contents

Chapter 1

Introduction

First machine translation systems as well as prevailing commercial MT systems to date are based on a set of hand-crafted rules typically following the Vaquois triangle (Figure 1.1). For an input sentence represented as a string of words, some symbolic representation is constructed, possibly in several steps. This symbolic representation, with the exception of a hypothetical Interlingua, remains language dependent, so a transfer step is necessary to adapt the structure to the target language. The translation is concluded by generating target-language string of words from the corresponding symbolic representation.

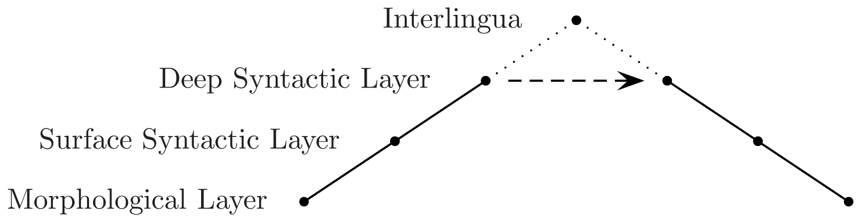


Figure 1.1: MT via tectogrammatical annotation.

In the following, we focus on one particular instance of this symbolic representation[1], namely the framework of Functional Generative Description (FGD, Sgall et al. (1986)), a formal stratificational language description that is being developed since late 1960's and recently found its implementation in the Prague Dependency Treebank (PDT, currently available in version 2.0, Hajič et al. (2006)) for Czech. An English implementation of FGD is under development and will be released as part of a parallel manually annotated Prague Czech-English Dependency Treebank (PCEDT; version 1.0 with automatic annotation is available as Čmejrek et al. (2004)).

In FGD, the three layers of language representation are called **morphological** (or m-layer), **analytical** (a-layer, corresponds to surface syntax) and **tectogrammatical** (t-layer, corresponds to deep syntax). M-layer represents the sentence as a sequence of word forms accompanied by their lemmas (base forms) and morphological tags that include part-of-speech and many other relevant categories such as case, gender, number, tense etc. A-layer and t-layer use a rooted labelled dependency tree to encode the relations between elements of the sentence. At the a-layer, nodes in the tree correspond one to one to words in the input sentence. At the t-layer, only words bearing meaning have a corresponding node while all auxiliary words are abstracted away, possibly contributing to some attributes of relevant nodes. On the other hand, t-layer includes nodes for entities that were not explicitly expressed in the sentence but

[1]Other examples of deep syntactic representation, in essence very similar to FGD, include Mel'čuk (1988), Microsoft logical form (Richardson et al., 2001) or the ideas spread across the projects PropBank (Kingsbury and Palmer, 2002), NomBank (Meyers et al., 2004) and Penn Discourse Treebank (Miltsakaki et al., 2004).

the language syntax and lexicon indicate their presence in the described situation. This is one of several other reasons that make t-layer language dependent. For an elaborated description of the t-layer for Czech refer to Deliverable 3.1 (Mikulová et al., 2007).

1.1 Motivation for Deep Syntactic Transfer

The rationale to introduce additional layers of formal language description is to bring the source and target languages closer to each other. If the layers are designed appropriately, the transfer step will be easier to implement because (among others):

- t-structures of various languages exhibit less divergences, fewer structural changes will be needed in the transfer step.
- t-nodes correspond to auto-semantic words only, all auxiliary words are identified in the source language and generated in the target language using language-dependent grammatical rules between t- and a- layers.
- t-nodes contain word lemmas, the whole morphological complexity of either of the languages is handled between m- and a- layers.
- t-layer abstracts away word-order issues. The order of nodes in a t-tree is meant to represent information structure of the sentence (topic-focus articulation). Language-specific means of expressing this information on the surface are again handled between t- and a-layers.

Overall, the design of t-layer aims at reducing data sparseness so less parallel training data should be sufficient to achieve same coverage.

Moreover, the full definition of t-layer includes explicit annotation of phenomena like co-reference to resolve difficult but inevitable issues of eg. pronoun gender selection. As tools for automatic tectogrammatical annotation improve, fine nuances could be tackled.

Chapter 2

Synchronous Tree Substitution Grammars

The idea of a *Synchronous Tree Substitution Grammar* was first sketched in (Hajič et al., 2002) and (Eisner, 2003). A rule of such a grammar has the form of a pair of so-called *little trees* (or equivalently *treelets*) with *aligned frontier nodes* that constrain both the positions, where other little trees can attach, and their type. The tree-to-tree transformation process covers the source tree by the source little trees from the rule-set, the output tree is then being constructed from the corresponding target little trees.

This chapter defines the theory of such synchronous tree substitution grammars including necessary mathematical details. It starts with the monolingual case, then extends it into the synchronous case. We also present algorithms for training the models on a corpus of parallel trees. In Chapter 3, we focus on the decoding algorithms necessary for producing translation.

We are aware of that the new theory is quite complicated. In order to help the reader to understand the new concepts, we start with the Section 2.1 giving an informal overview of the theory, jumping into the middle of the problem and trying to explain it using a "common sense". We hope that this figurative explanation makes the reading of the following pages easier.

2.1 Informal Motivation

Figure 2.1 contains an example of a tectogrammatical tree for a sample Czech sentence and an analytical tree for its English equivalent. The Figure 2.2 then contains both trees split into chunks. The chunk is usually formed by two *little trees* with filled (black) and empty (white) nodes.

The filled nodes are called *internal*, the empty nodes are called *frontier* nodes. The frontier nodes are connected by bows. The bows can be called *alignment*, *matching*, or *mapping*, and it always means the same thing.

The chunk of both trees with aligned frontier nodes is a *rule* of the *Synchronous Tree Substitution Grammar.*

The meaning of the first rule in Figure 2.2 is that the Czech *informovat nesprávně* is translated as the English *were misinformed.* The alignment between the frontier node PAT of the Czech tree and the frontier node *Sb* of the English tree means that the frontier (white) nodes must be filled at the same time by one rule. In this example, it is the rule *vedení* $\leftrightarrow$ *executives.* If a frontier node is not aligned, it means that it is not translated in the other tree.

If we follow the vertical alignment of frontier nodes and roots of the little trees below them, we get the whole "parse tree"—in another words we can find a rule that will be "plugged" into the aligned pair of frontier nodes.

The last but not least, the frontier nodes are labeled with syntactical functions[1], we call it *frontier state.* The *Probabilistic Synchronous Tree Substitution Grammar* models the probability

[1]But we could consider any reasonable labeling.

that a rule will be plugged into a given pair of matching frontier nodes with a given frontier state.

The idea of tree-to-tree transductions is so general that it can be applied to transformations between any two types of trees. In English to Czech machine translation, configurations transferring from English trees to Czech trees can operate either on the same analytical or tectogrammatical level, or they can go diagonally, e.g. from the English analytical trees to the Czech tectogrammatical trees. The tree-to-tree transductions could also be used for the "parsing" step from the analytical to the tectogrammatical representation.

2.2 Tree-to-Tree Mappings

Our goal is to describe the transformations of sentence structures that we may observe during the process of translation between two languages. Comparing the tectogrammatical tree for a sample Czech sentence "*Podle jeho názoru bylo vedení UAL o financování původní transakce nesprávně informováno.*", with the analytical tree for its English translation "*According to his opinion UAL's executives were misinformed about the financing of the original transaction.*" in Figure 2.1, we can find the corresponding groups of nodes (chunks) and list some of the mismatches that we observe:

1. The 2–1 match between the $PRED$ (predicate) of the Czech sentence *informovat nesprávně* and its English counterpart *misinformed*,
2. the elision (a 1–0 match) of the generated ACT (actor) of the Czech sentence,
3. the three 1–1 matches (*on* $\leftrightarrow$ *his; původní* $\leftrightarrow$ *original; vedení* $\leftrightarrow$ *executives*),
4. the Czech $CRIT$ (criterium) *názor* expressed by the English *Adv* (adverbial) phrase *according to ... opinion* can be either classified as a 1–3 match, or we can say that the tectogrammatical functor $CRIT$ forms the English *Adv* subtree *According to* and that the lemma *názor* matches 1–1 with *opinion*,
5. the EFF (effect) *financování* can be taken either as in a 1–3 match with *AuxP about the financing*, or we can think that the functor EFF generates the *Adv* node *about* and the lemma *financování* generates *the financing*,
6. the PAT (patient) *transakce* generates *AuxP of the ... transaction* or, in two steps, the functor PAT gives birth to *AuxP of*, the lemma *transakce* translates to *the transaction*, and there is a 1–2 match *transakce* $\leftrightarrow$ *the transaction.*

Such an informal description of observed transformations mixes lexical, functional, and structural information present in this tree-pair. In the following, we will have to proceed through several steps towards formal rules that capture all three types of information.

We can split the tree pair into corresponding chunks and number them as in Figure 2.2. A translation rule is represented by a pair of corresponding chunks. Filled nodes carry the lexical information; the other nodes are marked by their syntactical functions and can be substituted by other chunks with the same syntactical function[2]. Finally, the dashed bows between unfilled nodes indicate that the substitutions at these two nodes must proceed synchronously.

For example, the rule 1[3] formalizes our observation from item 1, i.e. that the part of the Czech tectogrammatical tree *informovat nesprávně*, preceded by some subtrees of ACT, $CRIT$, PAT, and EFF, will be translated by the part of the English analytical tree *were misinformed*, preceded by some subtrees of *Adv* and *Sb*, and followed by some *AuxP*. Rule 1 also specifies that the three pairs of subtrees of $CRIT$ and *Adv*, PAT and *Sb*, and EFF and *AuxP* will be substituted at the same time, or in other words, that these pairs of subtrees will be translations of one another. Finally, the ACT node will not have any counterpart in the English tree.

[2]The label with the syntactical function refers to both the unfilled node and the substituting chunk below it.
[3]See numbers above the root node of each chunk in the Figure 2.2.

Rule 2 corresponds to the observation 2, i.e. the generated actor is not translated into English. This rule maps the Czech chunk to a special *null* chunk on the English side.

The informal observation mentioned in item 4 is expressed by rules 3 and 4. Rule 3 says that functor $CRIT$ should be translated as *according to*, and rule 4 dictates the synchronous translation of the actual lexical information *názor* ↔ *opinion.* [4]

2.3 A Probabilistic Synchronous Tree Substitution Grammar

In this section, we describe the details of the probabilistic model of the transduction, the method of parameter estimation, and the decoding algorithm.

In our formal description of the Synchronous Tree Substitution Grammar, we stick to the symbolic markup used in (Hajič et al., 2002; Eisner, 2003) where possible.

We start with the definition of the non-synchronous Tree Substitution Grammar, and then extend to the synchronous case. As an example, we will use again the same tree pair from Figure 2.1, as in the previous section.

2.3.1 Non-synchronous Tree Substitution Grammar (TSG)

The **Tree Substitution Grammar** (TSG) is defined as follows:

1. Let Q be the set of **states**[5], and let $Start \in Q$ be the name reserved for the initial state.
2. Let L be the set of **labels** on the nodes (words) and edges (grammatical roles).
3. Let τ be the set of **little trees** or **treelets** t defined as tuples $\langle V, V^i, E, l, q, s\rangle$, where
   - V is a set of **nodes**,
   - $V^i \subseteq V$ is a subset of **internal nodes** and its complement $V^f = V - V^i$ is a set of **frontier nodes**,
   - $E \subseteq V^i \times V$ is a set of directed edges that can start from internal nodes only. The graph $\langle V, E\rangle$ must form a directed and acyclic tree.[6]
   - The function $s : V^f \rightarrow Q$ assigns a **frontier state** to each frontier node.
   - Let $r \in V$ be the **root** node of the tree, and let the **root state** q be assigned to the root.[7]
   - Let $l : (V^i \cup E) \rightarrow L$ be a function assigning a label to each internal node or edge.
4. Finally, the Tree Substitution Grammar (TSG) is defined as the tuple $\langle Q, L, \tau\rangle$.

For convenience, we will use the shorthand $t.q$ for the root state, $t.s$ for the frontier state function, and other shortcuts for all other properties of $t \in \tau$ using the same analogy.

Let $d \in V^f$ be a frontier node of t, and t' be a little tree such that $t.s(d) = t'.q$ – in other words, the frontier state of d matches the root state of t'. We may define the operation of

[4]One could object that there is no mechanism that would prevent from using rule 4 first (substituting at $CRIT$ and *Adv* frontier nodes of rule 1), so that using rule 3 would not be possible any more, and the resulting sentence (missing the words *According to*) would be ungrammatical. This can be fixed by extending the set of syntactical functions, e.g. the unfilled Czech and English nodes of rule 3 could have labels $CRIT'$ and Adv', respectively. An alternative way of fixing this problem is to consider one larger rule $CRIT$ *názor* ↔ *Adv According to ... opinion.*

[5]In our example we use the grammatical roles from the PDT

[6]We can see that the tree representing the whole sentence complies with the definition of the little tree with the empty set of V^f.

[7]If the root r is a frontier node, we can consider s such that $s(r) \neq q$.

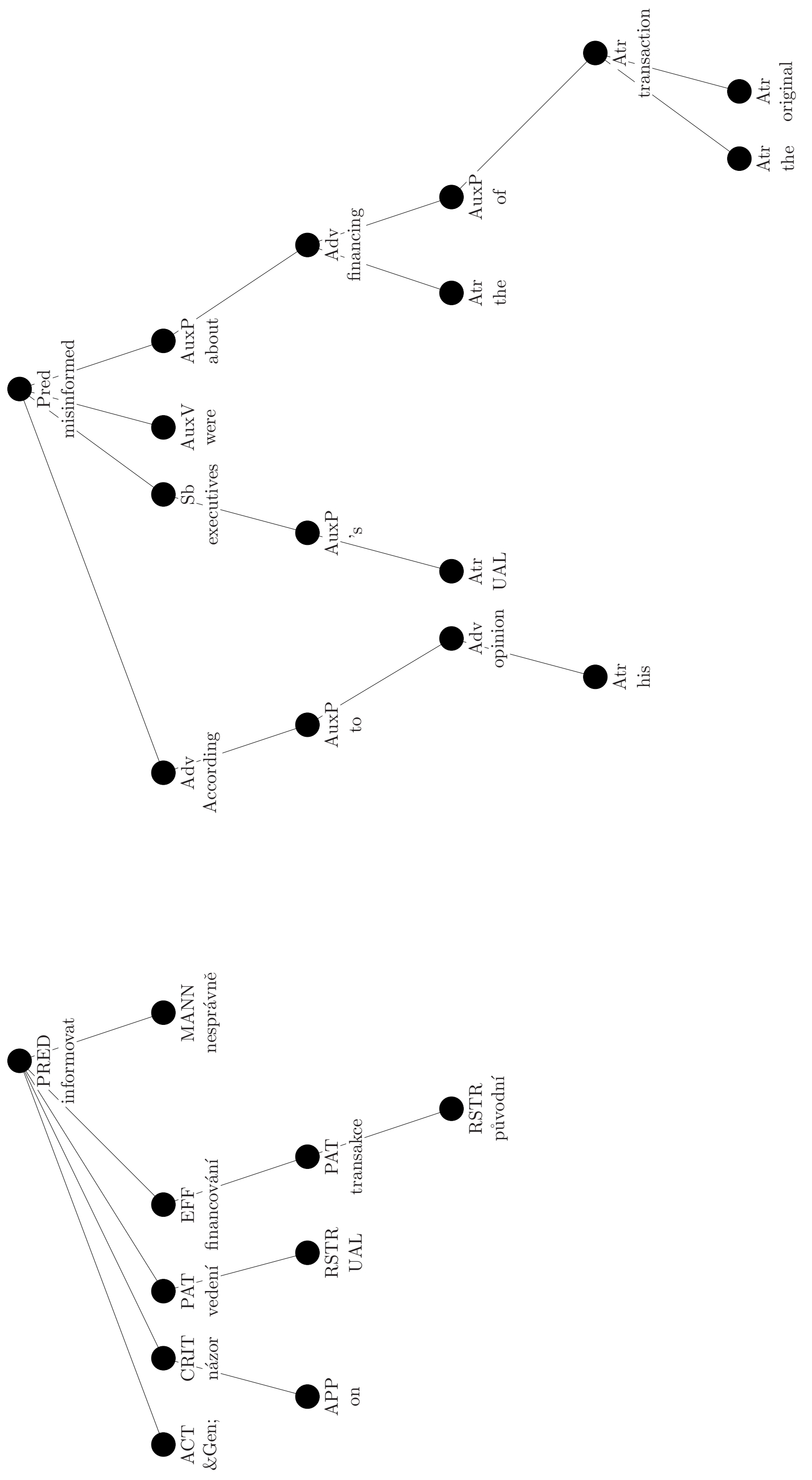


Figure 2.1: The tree pair for the tectogrammatical representation of the Czech sentence "*Podle jeho názoru bylo vedení UAL o financování původní transakce nesprávně informováno.*" and the analytical representation of the corresponding English translation "*According to his opinion UAL's executives were misinformed about the financing of the original transaction.*"

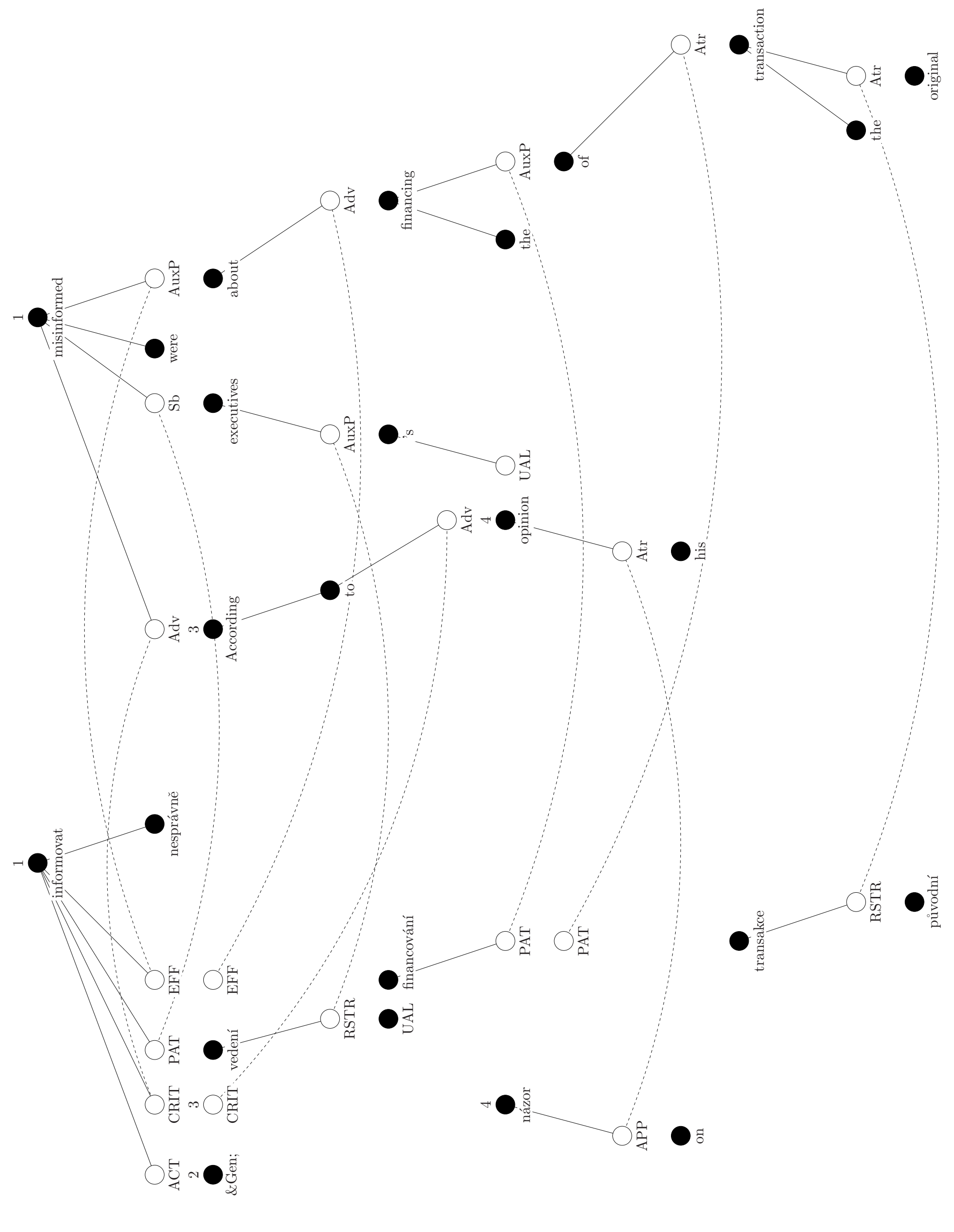


Figure 2.2: Aligned chunks of the tree structure for the tectogrammatical representation of the Czech sentence "*Podle jeho názoru bylo vedení UAL o financování původní transakce nesprávně informováno.*" and the analytical representation of the corresponding English translation "*According to his opinion UAL's executives were misinformed about the financing of the original transaction*".

substituting t at d with t'. The result of this operation is defined as little tree:

$$\begin{aligned} SUBST(t,d,t') =& \langle t.V^i \cup t'.V^i, \\ & t.V^f \cup t'.V^f - \{d\}, \\ & t.E^* \cup t'.E, \\ & t.l \cup t'.l, \\ & t.q, \\ & t.s \cup t's - \{\langle d, t'.q\rangle\}\rangle. \end{aligned} \tag{2.1}$$

We obtain $t.E^*$ from $t.E$, by "redirecting" the edge originally pointing to d to the root of t'.

The process of **derivation** from the initial state $Start$ in the TSG is described by Alg. 1:

Algorithm 1 The derivation process in TSG.

```
1.  Start with any little tree $t \in TSG$, such that $t.q = Start$.
2.  while $t.V^f \neq \emptyset$
3.      select $d \in t.V^f$
4.      select $t'$ such that $t.s(d) = t'.q$
5.      $t := SUBST(t,d,t')$
```

Line 4 of the algorithm hinted us to model the probability distribution over all possible little trees with root state q. Then the tree t' would be chosen with the probability $p(t' \mid q)$.

Thus the probability of the derivation $qt^0 \dots t^k$ starting from t_0 and using k substitutions with little trees $t_1, \dots, t_k$ can be defined as:

$$p(q, t^0, \dots, t^k) = p(t_0|q) * \prod_{i=1}^{k} p(t'_i \mid t'_i.q) \tag{2.2}$$

The probabilistic TSG does not require $t.s(d)$ to be the same as $t'.q$.

2.3.2 Inside-outside Algorithm for TSG

The probabilities $p(t \mid q)$ can be automatically obtained from a treebank using the EM algorithm. By analogy with the measures and quantities used for the training of probabilistic context-free grammars (Jelinek, 1985), we will define inside and outside probabilities, expected counts, and state the re-estimation formula.

We say that "the tree t' **fits** node d" if there is some derivation, in which t' substitutes t at d and the result of the derivation is T. Note that the root state of t' can be any of Q, since the nodes of the resulting tree T do not imply any restrictions on states used during the derivation. Thus the iteration over all little trees t' fitting d includes variants for all $q \in Q$.

The probability that the grammar TSG generates the tree T from the state $Start$ is the sum of probabilities of all possible derivations, and can be computed as the **inside probability** $\beta_{T.r}(Start)$ by the induction in Alg. 2:

Algorithm 2 The inductive algorithm for computing inside probabilities.

```
1.  for each node $c$ of $T$ in bottom-up order
2.      for each $q \in Q$, let $\beta_c(q) = 0$
3.      for each little tree $t$ that fits $c$, in a safe order
4.          increment $\beta_c(t.q)$ by $p(t \mid t.q) \cdot \prod_{d \in t.V^f} \beta_d(t.s(d))$
```

The natural-language definition of the inside probability $\beta_c(t.q)$ is the probability of generating the whole subtree of T rooted at node c with the root state q. Alg. 2 is an example of the well-known chart-parsing or dynamic-programming approach. It starts with the leaf nodes, their inside probabilities $p(t \mid t.q)$ are retrieved from the probabilistic model. Then the algorithm traverses the tree in bottom-up ordering and collects inside probabilities for the nodes higher up in the tree.

Line 3 must iterate the little trees in a *safe order*. The little trees with frontier root nodes can be selected only after all other little trees with the internal root node have been evaluated.

The **outside probabilities** $\alpha_{t.r}(q)$ can be computed by Alg. 3:

Algorithm 3 The inductive algorithm for computing outside probabilities.

```
1. for each little tree $t$ that fits $T.r$
2.     for each $q \in Q$
3.         if $q = Start$ let $\alpha_{t.r}(q) = 1$
4.         else $\alpha_{t.r}(q) = 0$
5. for each node $c$ of $T$, in top-down order
6.     for each little tree $t$ that fits $c$
7.         for each $d \in t.V^f$
8.             for each $t'$ that fits $d$
9.                 increment $\alpha_d(t'.q)$ by $p(t' \mid t.s(d)) \cdot \alpha_{t.r}(t.q) \cdot \prod_{d' \in t.V^f - d} \beta_{d'}(t.s(d'.))$
```

The natural definition of the outside probability $\alpha_d(t.q)$ is the probability of starting with the root state $T.q$, generating all parts of the tree T outside of the subtree rooted at c, and generating any subtree rooted at c with the root state $t.q$

The **expected count** $C(q,t)$ of a little tree t used in the derivation of T can be computed by Alg. 4:

Algorithm 4 The algorithm for computing expected counts.

```
1. Initialize $C(\_,\_) = 0$
2. for each node $c$ of $T$
3.     for each little tree $t$ that fits $c$
4.         Increment $C(q,t)$ by $p(t \mid q) \cdot \alpha_{t.r}(q) \cdot \prod_{d \in t.V^f} \beta_d(t.s(d))$
```

And finally, the re-estimation formula 2.3 for $p(t \mid q)$:

$$p(t \mid q) = \frac{C(q,t)}{\sum_{t'} C(q,t')} \tag{2.3}$$

In each iteration, the EM algorithm first computes the inside probabilities, the outside probabilities, the expected counts, and finally uses the re-estimation formula to obtain the new values of $p(t \mid q)$. Iterations are repeated until the $p(t \mid q)$ converges.

2.3.3 Synchronous Tree Substitution Grammar

We can extend the TSG to model the synchronous generation of a tree pair $T = (T_1, T_2)$. For this we will join two TSGs, $TSG_1 = \langle Q_1, L_1, \tau_1 \rangle$ and $TSG_2 = \langle Q_2, L_2, \tau_2 \rangle$, such that TSG_1 generates T_1 and TSG_2 generates T_2, with some restrictions on the operation of substitution.

The **synchronous tree substitution grammar** ($STSG$) is a tuple $\langle Q, L, \tau \rangle$, where

1. Q is a set of **synchronous root states**, *Start* being as before a special initial state.[8]
2. $L = L_1 \times L_2$.
3. $\tau = \tau_1 \times \tau_2$ is a set of **little tree pairs**. The little tree pair t is a tuple $\langle t_1, t_2, q, m, s \rangle$, where the little trees $t_i = \langle V_i^i, V_i^f, E_i, l_i \rangle$ have a common synchronous root state q.

   The alignment of frontier nodes m is called **matching**, and is defined as a 1-to-1 correspondence (pairing) between subsets of V_1^f and V_2^f, such that unmatched frontier nodes are mapped to *null*. For 1-0 or 0-1 mappings, we use the concept of a *null* tree that has empty sets of internal and frontier nodes[9]. The function $s : m \to Q$ assigns common frontier states to pairs of aligned frontier nodes.

[8]For convenience, we may think of $Q = Q_1 \times Q_2$, but generally, the Q can be any set of synchronous root states.

[9]Note that the concept of the *null* little tree is compliant with the rest of the definitions, except from the root of the *null* tree, and the root state of the little tree pair containing *null* tree. We leave this up to our intuition.

The operation $SUBST(t, d, t')$ of **substituting** t at d with t' for aligned node pairs $d = (d_1, d_2)$ is defined such that $d \in m$ and $t.s(d) = t'.q$. The result of this substitution is a little tree pair

$$\begin{aligned} SUBST(t, d, t') = &\langle SUBST(t_1, d_1, t'_1), \\ & SUBST(t_2, d_2, t'_2), \\ & q, \\ & t.m \cup t'.m - (d_1, d_2), \\ & t.s \cup t'.s - (d_1, d_2, t'.q)\rangle. \end{aligned} \tag{2.4}$$

The process of **derivation** from the initial state *Start* in $STSG$ is described by Alg. 5:

Algorithm 5 The derivation process in $STSG$.

```
1. Start with any little tree pair $t \in TSG$, such that $t.q = Start$.
2. while $t.m \neq \emptyset$
3.     select $d \in t.m$
4.     select $t'$ such that $t.s(d) = t'.q$
5.     $t := SUBST(t, d, t')$
```

The formula 2.2 for computing the probability of the derivation can be used for the synchronous case as well.

2.3.4 Inside-outside Algorithm for $STSG$

In order to train the probabilities $p(t \mid q)$ of the $STSG$, we have to rework Algorithms 2, 3, and 4 for the inside and outside probabilities, and expected counts, as well as the re-estimation Formula 2.3.

The definition of fitting has to be updated for the synchronous case: We say that t' **fits** node pair d, if t'_i fits d_i for $i = 1, 2$.

Alg. 6 computes the **inside probability** $\beta_{T.r}(Start)$, in other words, the probability that the $STSG$ generates a tree pair T from the initial symbol *Start*.

Algorithm 6 The inductive algorithm for computing inside probabilities for $STSG$.

```
1. for each node $c_1$ of $T_1$, in bottom-up order
2.     for each node $c_2$ of $T_2$, in bottom-up order
3.         for each $q \in Q$, let $\beta_{c_1,c_2}(q) = 0$
4.         for each little tree $t_1$ that fits $c_1$
5.             for each little tree $t_2$ that fits $c_2$
6.                 for each probable matching $m$ of frontier nodes of $t_1$ and $t_2$
7.                     construct $t$ from $q$, $t_1$, $t_2$, and $m$
8.                     increment $\beta_c(q)$ by $p(t \mid t.q) \cdot \prod_{d \in m} \beta_d(t.s(d))$
```

Lines 4 and 5 must iterate little trees fitting a node pair c in a *safe order.* First, we have to evaluate the pairs with the *null* little tree, then the little tree pairs with internal root nodes, and finally the little tree pairs with a frontier root nodes.

Alg. 7 computes the **outside probability** $\alpha_{t.r}(q)$:

Algorithm 7 The inductive algorithm for computing outside probabilities for $STSG$.

```
1.  for each node $c_1$ of $T_1$, in top-down order
2.      for each node $c_2$ of $T_2$, in top-down order
3.          for each $q \in Q$
4.              if $q = Start$ let $\alpha_c(q) = 1$
5.              else $\alpha_c(q) = 0$
6.          for each little tree $t_1$ that fits $c_1$
7.              for each little tree $t_2$ that fits $c_2$
8.                  for each probable matching $m$ of frontier nodes $t_1$ and $t_2$
9.                      construct $t$ from $q$, $t_1$, $t_2$, and $m$
10.                     for each pair of matching frontier nodes $f \in m$
11.                         increment $\alpha_f(t.s(f))$ by $p(t \mid q) \cdot \alpha_{t.r}(q) \cdot \prod_{d \in m-\{f\}} \beta_d(t.s(d))$
```

The expected counts $C(q, t)$ are computed using Alg. 8:

Algorithm 8 The algorithm for computing expected counts for $STSG$.

```
1.  Initialize $C(\_, \_) = 0$
2.  for each node $c_1$ of $T_1$
3.      for each node $c_2$ of $T_2$
4.          for each little tree $t_1$ that fits $c_1$
5.              for each little tree $t_2$ that fits $c_2$
6.                  for each probable matching $m$ of frontier nodes $t_1$ and $t_2$
7.                      construct $t$ from $q$, $t_1$, $t_2$, and $m$
8.                      Increment $C(q, t)$ by $p(t \mid q) \cdot \alpha_{t.r}(q) \cdot \prod_{d \in m} \beta_d(t.s(d))$
```

Finally, the probabilities are re-estimated using Formula 2.3, but this time the tree pairs t and t' iterate through all possible t_1, t_2, and m.

2.4 Conclusion

We have presented the details of the probabilistic Synchronous Tree Substitution Grammars, a new method for learning tree-to-tree transformations between non-isomorphic trees.

Chapter 3

STSG in Machine Translation

This chapter describes how STSG can be applied in the process of machine translation. As defined in Section 2.3.3, given a starting **synchronous state** $Start_{1:2} \in Q_1 \times Q_2$, a **synchronous derivation** $\delta = \{t^0_{1:2}, \ldots, t^k_{1:2}\}$ constructs a pair of dependency trees (T_1, T_2) by:

- attaching treelet pairs $t^0_{1:2}, \ldots, t^k_{1:2}$ at corresponding frontier nodes, and
- ensuring that the root states $t^0_{1:2}.q, \ldots, t^k_{1:2}.q$ of the attached treelets pairs $t^0_{1:2}, \ldots, t^k_{1:2}$ match the frontier states of the corresponding frontier nodes.

Note that in this section we require (1) each treelet to contain at least one internal node and (2) all frontier nodes in a treelet pair to be mapped, i.e. the left and right treelets must contain the same number of frontier nodes. These two additional requirements ensure that the translation procedure (1) will not loop (by generating output treelets while not consuming anything from the input tree) and (2) will not skip any subtree of the input tree.

For the purposes of further explanation, we define source-side projection $\mathbf{1}(\delta)$ and target-side projection $\mathbf{2}(\delta)$ of a derivation δ as the trees T_1 and T_2 constructed by δ, respectively. Given a source tree T_1, we denote $\Delta(T_1) = \{\delta \mid \mathbf{1}(\delta) = T_1\}$, the set of derivations δ yielding T_1 on the source side.

Note that given a tree T, not all subtrees $t \subseteq T$ can be considered as part of (one side of) a valid (synchronous) derivation because STSG derivations have no adjunction operation. We say that a subtree t of a tree T satisfies **STSG property**, if for every internal node $n \in t$ all immediate dependents of n in T are included in t as well, either as internal or as frontier nodes. In other words, we assume no tree adjunction operation was necessary to over any children of n in T.

3.1 Translation as Probability Maximization

Our goal is to translate a source sequence of words s_1 into a target sequence of words $\hat{s_2}$, where $\hat{s_2}$ is the most likely translation out of all possible translations s_2:

$$\hat{s_2} = \operatorname*{argmax}_{s_2} p(s_2 \mid s_1) \tag{3.1}$$

We introduce the source and target dependency trees T_1 and T_2 as hidden variables to the maximization, assuming no other dependencies except those along the pipeline indicated in Figure 1.1:

$$\hat{s_2} = \operatorname*{argmax}_{s_2, T_1, T_2} p(T_1 \mid s_1) \cdot p(T_2 \mid T_1) \cdot p(s_2 \mid T_2) \tag{3.2}$$

Rather than searching the joint space, we break the search into three independent steps: parsing (3.3), tree transduction (3.4) and generation (3.5):

$$\hat{T_1} = \operatorname*{argmax}_{T_1} p(T_1 \mid s_1) \tag{3.3}$$

$$\hat{T_2} = \operatorname*{argmax}_{T_2} p(T_2 \mid \hat{T_1}) \tag{3.4}$$

$$\hat{s_2} = \operatorname*{argmax}_{s_2} p(s_2 \mid \hat{T_2}) \tag{3.5}$$

We mention the tools used for parsing and generation in Chapter 4 below. STSG is used to find the most likely target tree $\hat{T_2}$ given T_1. Applying the Viterbi approximation we search for the most likely derivation $\hat{\delta}$ instead and take its target-side projection.[1]

$$\hat{T_2} = \operatorname*{argmax}_{T_2} p(T_2 \mid T_1) \doteq \oldstylenums{2}(\hat{\delta}) = \oldstylenums{2}\Big(\operatorname*{argmax}_{\delta \in \Delta(T_1)} p(\delta)\Big) \tag{3.6}$$

In other words, we consider all decompositions of T_1 into a set of treelets $t_1^0, \ldots, t_1^k$, expand each treelet t_1^i into a treelet pair $t_{1:2}^i$ using a treelet pair dictionary and consider the probability of the synchronous derivation $\delta = \{t_{1:2}^0, \ldots, t_{1:2}^k\}$. Having found the most likely $\hat{\delta}$, we return the right-hand-side tree constructed by $\hat{\delta}$.

3.2 Log-linear Model

Following Och and Ney (2002) we further extend 3.6 into a general log-linear framework that allows us to include various features or **models**:

$$\hat{\delta} = \operatorname*{argmax}_{\delta \in \Delta(T_1)} \exp\Big(\sum_{m=1}^{M} \lambda_m h_m(\delta)\Big) \tag{3.7}$$

Each of the M models $h_m(\delta)$ provides a different score aimed at predicting how good the derivation δ is. The weighting parameters λ_m, $\sum_1^M \lambda_m = 1$, indicate relative importance of the various features and are tuned on an independent dataset.

To facilitate efficient decoding (see Section 3.3 below), we require most feature functions $h_m(\delta)$ to decompose in lockstep with the derivation, i.e. to take the form:

$$h_m(\delta) = \sum_{i=0}^{k} h_m(t_{1:2}^i) \tag{3.8}$$

[1]Here is a step-by-step explanation of the approximation:

$$\begin{aligned}
\hat{T_2} &= \operatorname*{argmax}_{T_2} p(T_2 \mid T_1) && \text{marginalize over derivations } \delta \\
&= \operatorname*{argmax}_{T_2} \sum_{\delta} p(T_2, \delta \mid T_1) && \text{apply chain rule} \\
&= \operatorname*{argmax}_{T_2} \sum_{\delta} p(T_2 \mid \delta, T_1) \cdot p(\delta \mid T_1) && p(T_2 \mid \delta, T_1) = 1 \text{ because } T_2 = \oldstylenums{2}(\delta) \\
&= \operatorname*{argmax}_{T_2} \sum_{\delta} p(\delta \mid T_1) && \text{apply Fundamental Law} \\
&= \operatorname*{argmax}_{T_2} \sum_{\delta} \frac{p(\delta, T_1)}{p(T_1)} && \text{ignore } p(T_1)\text{, constant in maximization} \\
&= \operatorname*{argmax}_{T_2} \sum_{\delta} p(\delta, T_1) && p(\delta, T_1) = \Big\langle \begin{matrix} p(\delta) & \text{if } \delta \in \Delta(T_1) \text{ because } T_1 = \oldstylenums{1}(\delta) \\ 0 & \text{otherwise} \end{matrix} \\
&= \operatorname*{argmax}_{T_2} \sum_{\delta \in \Delta(T_1)} p(\delta) && \text{approximate sum by only the largest element} \\
&\doteq \operatorname*{argmax}_{T_2} \max_{\delta \in \Delta(T_1)} p(\delta) && \text{Viterbi approximation to search for } \delta \text{ instead of } T_2 \\
&\doteq \oldstylenums{2}(\operatorname*{argmax}_{\delta \in \Delta(T_1)} p(\delta))
\end{aligned}$$

3.2.1 STSG Model

One of the most basic features is based on the STSG probability of the synchronous derivation, as defined in Eq. 2.2 and reprinted here for the synchronous case:

$$p(\delta) = p(t^0_{1:2} \mid Start_{1:2}) * \prod_{i=1}^{k} p(t^i_{1:2} \mid t^i_{1:2}.q) \tag{3.9}$$

To incorporate this probability into the log-linear model, we take the log of it, defining the STSG model:

$$h_{STSG}(\delta) = \log(p(\delta)) = \log(p(t^0_{1:2} \mid Start_{1:2})) + \sum_{i=1}^{k} \log(p(t^i_{1:2} \mid t^i_{1:2}.q)) \tag{3.10}$$

Note that if $h_{STSG}(\cdot)$ were the only feature used, the log-linear model reduces to the straightforward maximization of $p(Start_{1:2}, \delta)$:

$$\hat{\delta} = \operatorname*{argmax}_{\delta \in \Delta(T_1)} \exp\big(h_{STSG}(\delta)\big) = \operatorname*{argmax}_{\delta \in \Delta(T_1)} p(Start_{1:2}, \delta) \tag{3.11}$$

3.2.2 Reverse and Direct Treelet Models

The STSG model assumes the choice of a treelet pair $t_{1:2}$ depends only on the synchronous state q of the two frontiers where $t_{1:2}$ is attached.

Inspired by the common practice of statistical machine translation, we include the channel model ("reverse") and "direct" conditional probabilities:

$$h_{direct}(t^i_{1:2}) = \log\big(p(t^i_2 \mid t^i_1)\big) \tag{3.12}$$

$$h_{reverse}(t^i_{1:2}) = \log\big(p(t^i_1 \mid t^i_2)\big) \tag{3.13}$$

The reverse model is justified by Bayes decomposition of $p(target|source)$ while the direct model empirically proves as a comparably valuable source (see e.g. Och (2002)).

3.2.3 N-gram Language Models

A probabilistic target-language model used to promote coherent hypotheses is a very important predictor of translation quality (see e.g. Och (2002)).

In the canonical mode, an STSG decoder is expected to produce an output dependency tree and thus cannot directly employ pervasive n-gram language models. However, if no structure is needed at the output (e.g. when translating to a-trees and directly reading off node labels), we can safely destroy all target-side tree structure, representing T_2 as a sequence of output words $w_1, \ldots, w_J$. Naturally, until the complete target hypothesis is constructed, we have to keep track of exact positions of yet-to-expand frontiers within the sequence of output words.

In this special case, the traditional sequence (language) model can be used, with a bit of careful delayed computation around unexpanded frontiers:

$$h_{\mathrm{LM}_n}(\delta) = \log \prod_{j=1}^{J} p(w_j | w_{j-1} \ldots w_{j-n+1}) \tag{3.14}$$

We assume w_j to be set to a special out-of-sentence symbol for $j < 1$.

3.2.4 Binode Tree Language Model

Given the output dependency tree structure, a more natural language model estimates sentence probability based on edges in the tree. As documented eg. by Charniak (2001), such models can improve parsing accuracy.

We define binode probability of the target tree T_2 as the multiplication of probabilities of all the edges $e \in T_2$. Given the governor $g(e)$ and the child $c(e)$ of e, we can define three different probabilies, "direct", "reverse" and "joint", leading to three separate models:

$$h^{biLM}_{direct}(\delta) = \log \prod_{e \in T_2} p(g(e) \mid c(e)) \tag{3.15}$$

$$h^{biLM}_{reverse}(\delta) = \log \prod_{e \in T_2} p(c(e) \mid g(e)) \tag{3.16}$$

$$h^{biLM}_{joint}(\delta) = \log \prod_{e \in T_2} p(c(e), g(e)) \tag{3.17}$$

3.2.5 Additional Features

Following common practice in phrase-based machine translation (e.g. Koehn (2004a) or Zens et al. (2005)), we include penalties to consider the number of treelets and words used to construct a derivation:

$$h_{treelet\ penalty}(\delta) = -|\delta| \tag{3.18}$$

$$h_{word\ penalty}(\delta) = -\sum_{i=0}^{k} |t_2^i| \tag{3.19}$$

where $|t_2^i|$ denotes the number of words in target treelet t_2^i.

3.3 Decoding Algorithms for STSG

The search space of all possible decompositions of input tree multiplied by all possible translations of source treelets is too large to be explored in full, efficient approximation algorithms have to be designed.

3.3.1 Top-Down Beam Search

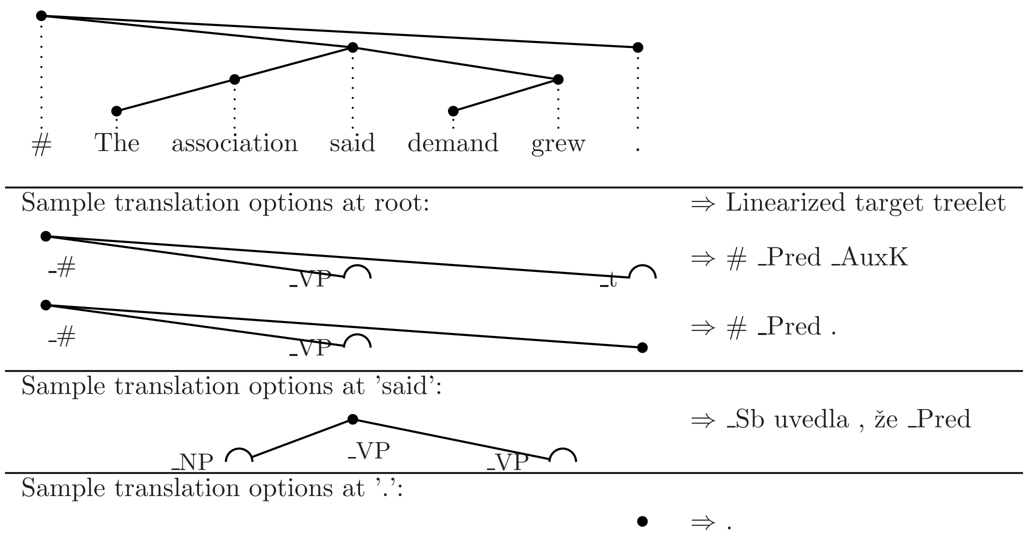
The current version of our decoder implements a beam search inspired by the strategy of phrase-based decoder Moses (Koehn et al., 2007). While Moses constructs partial hypotheses in a left-to-right fashion (picking source phrases in arbitrary order), our partial hypotheses are constructed top-to-bottom along with the source tree T_1 being covered from top to bottom. The algorithm, in essence very similar to the one described recently by Huang et al. (2006) but dating back to Aho and Johnson (1976), is outlined in Alg. 9. The main difference is that we tackle the exponetial search space of tree decompositions using a pre-processing phase while Huang et al. (2006) use memoization.

The first step is the construction of "translation options". For each input node $x \in T_1$, all possible treelets rooted at the node are examined and if a translation of a treelet is found, it is stored as one of the translation options for x. Figure 3.1 illustrates sample translation options for the auxiliary root ("#"), the main verb "said" and the full stop ".". For conciseness, the target treelet structure is omitted in the picture as if the target output tree was directly linearized.

Figure 3.2 illustrates the gradual expansion of a hypothesis using translation options constructed in the first step. Once all input nodes are covered (and thus no frontiers are left in

Algorithm 9 Top-down beam-search STSG decoding algorithm.

```
1.  For input tree $T_1$ of $n$ nodes, prepare translation options table:
2.      For each source node $x \in T_1$
3.          Construct all possible treelet pairs $t_{1:2}$ where $t_1$ is rooted at $x$
            and covers a subtree of $T_1$.
4.              The subtree has to satisfy the STSG property:
5.                  If $y \in T_1$ is covered with an internal node of $t_1$, all dependents of $y$
                    have to be covered by $t_1$ as well.
6.          Record only $\tau$ best possible treelet pairs rooted at $x$.
7.  Create stacks $s_0, \ldots, s_n$ to hold partial hypotheses, stack $s_i$ for hypotheses covering
    exactly $i$ input nodes.
8.  Insert initial hypothesis (a single frontier node) into $s_0$.
9.  for $i \in 0 \ldots n-1$
10.     foreach hypothesis $h \in s_i$
11.         Expand $h$ by attaching one of possible translation options at a pair of pending frontiers,
12.             extending the set of covered words and adding output words.
13.         Insert the expanded $h'$ ($j$ words covered) to $s_j$, pruning $s_j$ to contain at most $\sigma$ hypotheses.
14. Output top-scoring $h^*$ from $s_n$.
```



#
The
association
said
demand
grew
.
Sample translation options at root:
⇒ Linearized target treelet
-#
_VP
_.
⇒ # _Pred _AuxK
-#
_VP
⇒ # _Pred .
Sample translation options at 'said':
⇒ _Sb uvedla , že _Pred
_NP
-VP
_VP
Sample translation options at '.':
⇒ .

Figure 3.1: Translation options example.

the partial output), the output hypothesis is returned. In practice, we beam-search the space of derivations, studying σ best-scoring partial hypothesis of the same number of covered input nodes at once. Note that each expansions is guaranteed to cover at least one more input node, so the algorithm cannot loop.

3.3.2 Bottom-up Dynamic-Programming Decoding Algorithm

Čmejrek (2006) presents another possible method of searching for the most probable translation T_2 of a given input tree T_1.

The most probable derivation is computed by a dynamic-programming style Alg. 10. For each node $c_1 \in T_1$ and each synchronous state $q \in Q$, we find and store the root treelet pair $t_{1:2}$ of the most probable derivation $\hat{\delta_{c_1}}$ that covers the whole subtree of T_1 rooted at c_1 and has q as the root synchronous state. The treelets are stored in arrays $A_{c_1}(q)$ and the corresponding (inside) probabilities of $\hat{\delta_{c_1}}$ are stored in $\beta_{c_1}(q)$.

The final derivation $\hat{\delta}$ covering whole T_1 is constructed by starting from $t^0_{1:2} = A_{T_1.r}(Start_{1:2})$ and recursively including all treelet pairs $t^i_{1:2} = A_{f^i_1}(q^i)$ to cover all frontiers f^i_1 (respecting the synchronous states q^i) of previously included treelets $t^0_{1:2}, \ldots, t^{i-1}_{1:2}$.

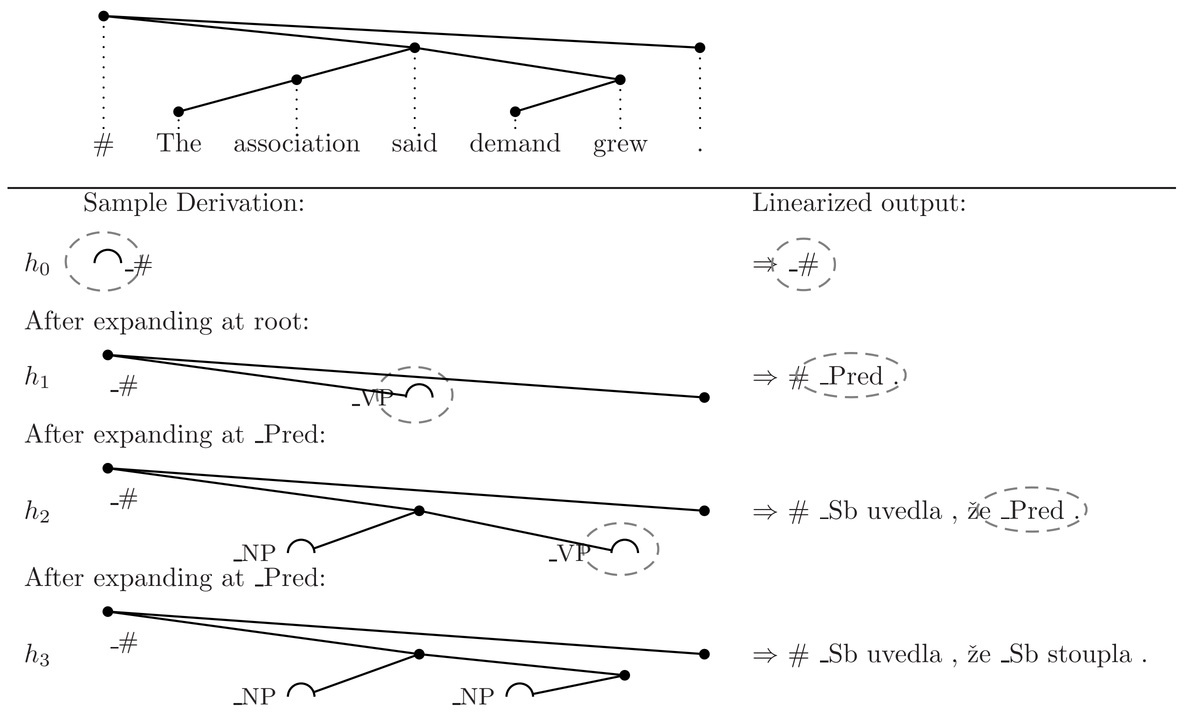


Figure 3.2: Top-down hypothesis expansion using translation options from Figure 3.1.

Algorithm 10 Bottom-up decoding algorithm for *STSG*.

```
1. for each node $c_1 \in T_1.V$ in bottom-up order
2.     for each $q \in Q$ let $\beta_{c_1}(q) = -\infty$
3.     for each little tree $t_1$ that fits $c_1$ in a safe order
4.         while $t_{1:2}$=proposeNewTreeletPair($t_1$) // we have to try all possible $t_2, q, m, s$
5.             let $prob = p(t_{1:2} \mid t_{1:2}.q) \cdot \prod_{(d_1,d_2)\in m} \beta_{d_1}(t_{1:2}.s((d_1,d_2)))$
6.             if $\beta_{c_1}(q) < prob$ // found a higher scoring derivation
7.             then let $\beta_{c_1}(q) = prob$ and $A_{c_1}(q) = t_{1:2}$
```

As opposed to computing the inside probabilities (see Alg. 2), we do not have to fit nodes of T_2, and therefore no restrictions are put on the choice of t_2. That is also why the inside probabilities are indexed by c_1 only.

3.4 Heuristic Estimation of STSG Model Parameters

Given a sentence-parallel treebank, we can use the expectation-maximization algorithm described in Section 2.3.4 to obtain treelet-to-treelet alignments and estimate STSG derivation probability as defined in Eq. 2.2. Our plan is to soon adopt this method, but for the time being we restrict our training method to a heuristic based on GIZA++ (Och and Ney, 2000) word alignments. So instead of treelet-to-treelet alignments, we base our probability estimates on node-to-node alignments only.

For each tree pair in the training data, we first read off the sequence of node labels and use GIZA++ tool to extract a possibly N-N node-to-node-alignment.[2] In the next step, we extract all treelet pairs from each node-aligned tree pair such that all the following conditions are satisfied:

- each treelet may contain at most 5 internal and at most 7 frontier nodes (the limits are fairly arbitrary),

[2]GIZA++ produces asymmetric 1-N alignments, we follow standard practices to combine 1-N and N-1 alignments from two GIZA++ runs.

- each internal node of each treelet, if aligned at all, must be aligned to a node in the other treelet,
- the mapping of frontier nodes has to be a subset of the node-alignment,
- each treelet must satisfy STSG property.

All extracted treelet pairs contribute to our maximum likelihood probability estimates. In general, given a left treelet t_1, a right treelet t_2 and their respective root states q_1 and q_2, we estimate three separate models: "stsg", "direct" and "reverse":

$$h_{stsg}(t_{1:2}) = \log \frac{\text{count}(t_1, q_1, t_2, q_2)}{\text{count}(q_1, q_2)} \tag{3.20}$$

$$h_{direct}(t_{1:2}) = \log \frac{\text{count}(t_1, q_1, t_2, q_2)}{\text{count}(t_1, q_1, q_2)} \tag{3.21}$$

$$h_{reverse}(t_{1:2}) = \log \frac{\text{count}(t_1, q_1, t_2, q_2)}{\text{count}(t_2, q_1, q_2)} \tag{3.22}$$

3.5 Methods of Back-off

As expected, and also pointed out by Čmejrek (2006), the additional structural information boosts data-sparseness problem. Many source treelets in the test corpus were never seen in our training data. To make things worse, our heuristic treelet extraction method constrains the set of extractable treelet pairs by three rigid structures: source tree, target tree and the word alignment. A single error in the word alignment or parsing prevents our method from learning a treelet pair. We thus have to face not only natural divergence of sentence structures but also divergence caused by random errors in any of the automatically obtained annotations.[3]

To tackle the problem, our decoder utilizes a sequence of back-off models, i.e. a sequence of several methods of target treelet construction and probability estimation. Each subsequent model is based on less fine-grained description of the input treelet and constructs the target treelet on the fly from independent components.

The order and level of detail of the back-off methods is fixed but easily customizable in a configuration file.

3.5.1 Preserve All

The most straightforward method is to preserve all information in an observed treelet pair. This includes:

- left and right treelet structure, including all frontiers and internals and preserving the linear order of the nodes
- full labels of left and right internals
- state labels of left and right frontiers

An example of a complete treelet pair is given in Figure 3.3.

3.5.2 Drop Frontiers

One of significant limitations of STSG is the lack of adjunction operation. In order to handle input treelets with branching that was not seen in the training data, we collect treelet pairs while ignoring any frontiers. An example of such treelet pair is given in Figure 3.4.

[3]For example, **?**) attempt to loosen the rigidity of structures by defining quasi-synchronous (monolingual) grammar for target language that prefers to analyze or generate target-side sentence in alignment with the source-side tree but is not restricted to do so.

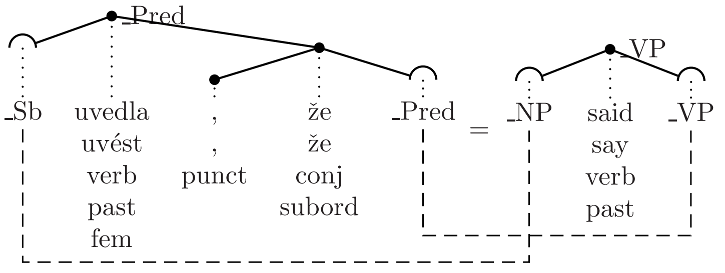


Figure 3.3: A treelet pair with all information preserved.

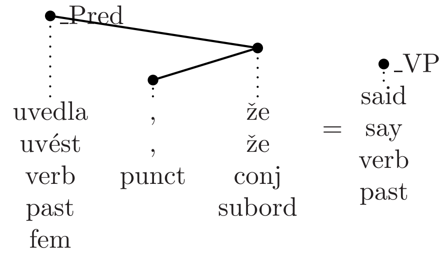


Figure 3.4: A treelet pair with no frontiers.

Once the translation using this model is attempted, we remove all frontiers from the source treelet, map the "skeleton" to the target treelet and attach the required number of frontier nodes to the target tree. The position and state label of the frontiers can be chosen based on a separate probabilistic model.

As a further refinement, one might think of dropping only frontiers representing adjuncts but preserving frontiers for complements. Either a valency lexicon would supply the distinction between argument and adjuncts, or we could use some heuristic such as suggested by Bojar (2004).

In the current implementation, we employ this method of back-off only in cases where the output is directly linearized. Therefore, the governing node for a frontier has not to be determined when attaching the frontier and we can use a simple model to "zip" the sequence of target internals and the sequence of target frontiers (we do not allow any reordering of the frontiers). The target label of a frontier is chosen based on the label of the source frontier.

3.5.3 Translate Word by Word

The technique of dropping frontiers cannot be used when producing output trees, unless we design a frontier re-attachment model. However, we still need to overcome the no-adjunction limitation of STSG. A simple solution is possible, if we restrict treelet size to one internal only.

If the source treelet contains exactly one internal node, the structure of the treelet is known: the internal node is the root of the treelet and its immediate dependents are all frontiers of the treelet, see eg. Figure 3.5.

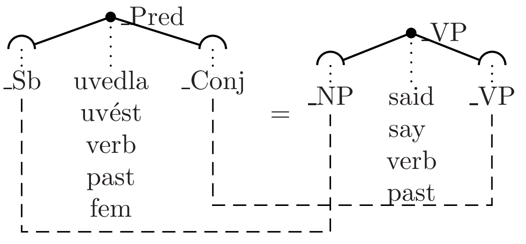


Figure 3.5: A treelet pair with one internal node in each treelet.

We can easily decompose such treelets and translate independently: 1. the label of the internal node, 2. each of the frontier labels. Again, we could consider reordering of the nodes but unless a satisfactory reordering model is designed, we keep the order intact.

A clear disadvantage of this back-off method is that the number of nodes cannot change in the process of translation. This poses a significant problem for transfer at a-layer, but for transfer at t-layer, preserving tree structure is a viable approximation (Čmejrek et al., 2003).

3.5.4 Keep Word Non-Translated

In cases where a word was never seen in the training data, the methods described so far would not provide any translation for the word, so the translation of the whole sentence would fail producing no output. Much more useful approach is to keep the unknown word not translated and try to translate the rest of the sentence.

Technically, we achieve this by adding a special rule that preserves treelet structure, copies internal labels and independently translates each of frontier labels. In practice, we prefer to restrict this method to treelets containing one internal only.

3.5.5 Factored Input Nodes

As described in Deliverable 3.1 (Mikulová et al., 2007), and also indicated in Figure 3.3, internal node labels are usually not atomic values. For example, an a-node usually bears the value of word form, lemma, morphological tag (all inherited from m-layer) and analytical function (afun) label. For t-nodes, the set of attributes is significantly larger, as attributes explicitly encode linguistic features such as verbal tense, modality, iterativeness, person, nominal gender, negation and many others.

Treating node labels as atomic and thus relying on all attributes to exactly match the input leads to severe sparse data problem. We allow to specify only a subset of input attributes ("factors") to be taken into account while searching for a treelet translation. In practice, we usually use a sequence of models, each depending on fewer and fewer input factors. For example, a back-off model for "preserve all" as illustrated in Figure 3.3 could be based on source lemmas only. See Figure 3.6 for a hypothetical rule for Czech-to-English transfer.

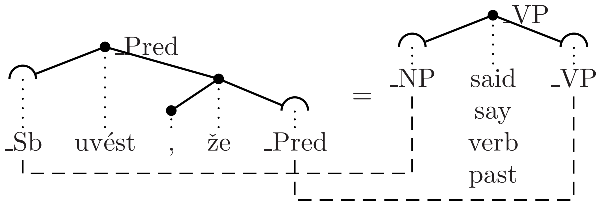


Figure 3.6: A treelet pair with all source lemmas only.

3.5.6 Factored Output Nodes

Ignoring some attributes of input nodes is not sufficient as a back-off method alone. For output factors, however, we have no option and each node has to eventually provided with all attributes. We use the idea of "mapping" and "generation" steps from factored phrase-based translation (Koehn and Hoang, 2007).

Currently, our implementation of factored models is limited to treelets containing exactly one internal. We will soon extend this to treelets of any size. However, the size and shape of the treelet (chosen according to a subset of input factors) will probably remain fixed while additional output factors are constructed.

Figure 3.7 illustrates a sequence of five **decoding steps**: three **mapping steps** that convert source factors to target factors and two **generation steps** that ensure coherence of output factors. Many other configurations are possible.

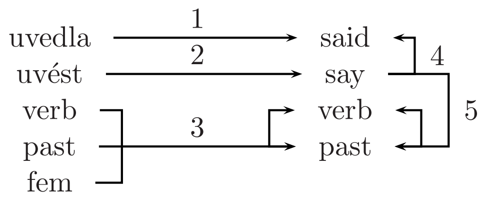


Figure 3.7: Sample decoding steps in word-for-word factored translation.

In setups with multiple output factors, we apply also the language models described in Section 3.2.3 and Section 3.2.4 several times using various subsets of output factors to provide a back-off for probability estimation. For instance, even if a node pair was never seen in the exact configuration constructed in a sequence of decoding steps, the pair of node lemmas may be very common so we wish to score it with a non-zero probability.

3.6 Remarks on Implementation

The STSG decoder called `treedecode` is being implemented in Mercury (Somogyi et al., 1995)[4] and currently consists of about 17,000 lines of code.

Supported features, apart from methods described in previous sections, include:

- parallel execution (both training and translation phases) on Sun Grid Engine[5],
- efficient storage of translation tables using `tinycdb`[6],
- binding to IrstLM (Federico and Cettolo, 2007) for n-gram language modelling,
- disk caching of various steps of computation to speed up consecutive startups and reuse partial results upon failure (similar effects can be achieved using the technique of "checkpointing"),
- basic debugging output in Scalable Vector Graphics (SVG),
- preliminary support for minimum error-rate training using two approaches, (Och, 2003) and (Smith and Eisner, 2006a).

The source code is currently available upon request, future versions will be freely accessible on a website. If interested, please contact Ondřej Bojar (`bojar@ufal.mff.cuni.cz`).

[4]`http://www.cs.mu.oz.au/research/mercury/`
[5]`http://gridengine.sunsource.net/`
[6]`http://www.corpit.ru/mjt/tinycdb.html`

Chapter 4

Empirical Evaluation

At the current stage of development, only preliminary results of an end-to-end evaluation can be presented. Nevertheless, we try to cover a wide range of experimental settings when translating from English to Czech, as illustrated in Figure 4.1, which is a refinement of Figure 1.1.

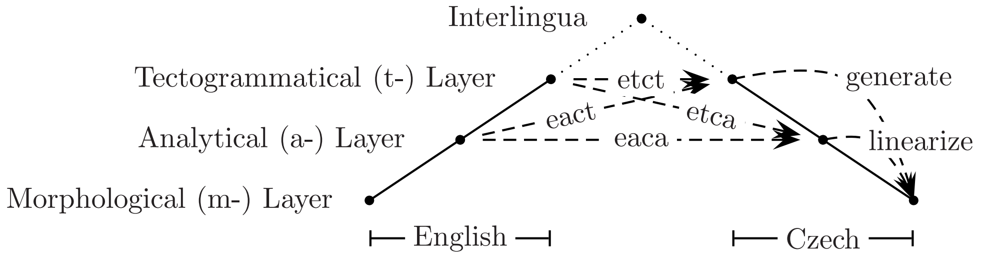


Figure 4.1: Experimental settings of syntactic MT.

Our main focus is the translation from English t-layer to Czech t-Layer (etct). The general applicability of STSG to any dependency trees allows us to test the same model also for analytical translation (eaca) or across the layers (etca and eact). For each configuration, we extract treelet pairs using the heuristics described in Section 3.4, possibly employing some of the back-off techniques from Section 3.5. The EM training procedure as described in Section 2.3.4, though implemented, was not yet incorporated into our training process.

4.1 Data and Pipeline Setup

Apart from our STSG decoder, we use several additional tools along the training and translation pipeline, as summarized in Table 4.1.

Step	Tool Used
English morphological analysis (text→m)	Minnen et al. (2001)
English tagging (text→m)	Ratnaparkhi (1996)
English constituency parsing (text→phrase structure)	Collins (1996)
English dependencies (phrase structure→a)	hand-written rules
English tectogrammatical parsing (a→t)	rules similar to Čmejrek et al. (2003)
Czech morphological analysis (text→m)	Hajič (2004)
Czech dependency parsing (m→a)	McDonald et al. (2005)
Czech tectogrammatical parsing (a→t)	Klimeš (2006)
Czech tectogrammatical generation (t→text)	Ptáček and Žabokrtský (2006)

Table 4.1: Tools used in training data preparation and in end-to-end evaluation.

We train and test our system on the News Commentary corpus as available for the ACL 2007 workshop on machine translation (WMT), respecting the standard sectioning of the data.[1]

4.2 Experimental Results

Table 4.2 reports the BLEU (Papineni et al., 2002) scores of several configurations of our system, higher scores suggesting better MT quality. We report single-reference lowercased BLEU[2,3].

The values in column LM Used indicate the type of language model used in the experiment. An n-gram model can be applied to the output sequence of words. For setups where the final sequence of words is constructed using the generation component by Ptáček and Žabokrtský (2006) with no access to a language model, we use at least a binode LM to improve output tree coherence.

Method of Transfer	LM Used	BLEU
epcp	n-gram	10.9±0.6
eaca	n-gram	8.8±0.6
epcp	none	8.7±0.6
eaca	none	6.6±0.5
etca	n-gram	6.3±0.6
etct factored, preserving structure	binode	5.6±0.5
etct factored, preserving structure	none	5.3±0.5
eact, no output factors	binode	3.0±0.3
etct, vanilla STSG (no factors), all node attributes	binode	2.6±0.3
etct, vanilla STSG (no factors), all node attributes	none	1.6±0.3
etct, vanilla STSG (no factors), just t-lemmas	none	0.7±0.2
Phrase-based as reported by Bojar (2007)		
Vanilla	n-gram	12.9±0.6
Factored to improve target morphology	n-gram	14.2±0.7

Table 4.2: Preliminary English-to-Czech BLEU scores for syntax-based MT evaluated on DevTest dataset of ACL 2007 WMT shared task.

For the sake of comparison, we report also results achieved by Bojar (2007) using the phrase-based decoder Moses on the same dataset. To a certain extent, our tree-based decoder can simulate phrase-based decoding if we replace the dependency structure of an a-tree with a simple left-to-right chain of words ("linear tree"). The results obtained using this approach are labelled "epcp". Our phrase-based approximation epcp is bound to work worse than Moses because we strictly follow the left-to-right order prohibiting any phrase reordering, and because we still do not use proper minimum error-rate training algorithms.

4.3 Discussion and Future Research

At the first sight, our preliminary results support common worries that with a more complex system it is increasingly difficult to obtain good results. However, we are well aware of many limitations of our current experiments as discussed below.

Withing the scope of our main focus, the tectogrammatical transfer ("etct"), we see dramatic improvement from BLEU 1.6 to BLEU 5.6. The score 1.6 is achieved using the very baseline

[1]`http://www.statmt.org/wmt07/`

[2]For methods using the direct t→text generation system by Ptáček and Žabokrtský (2006), we tokenize the hypothesis and the reference using the rules from the official NIST `mteval-v11b.pl` script. For methods that directly produce sequence of output tokens, we stick to the original tokenization.

[3]The reported ± bounds indicate empirical 95% confidence intervals obtained using bootstrapping method by Koehn (2004b).

of STSG translation: nodes including all attributes are treated as atomic units, only the STSG probability (Section 3.2.1) is used and no language model is applied. Our best "etct" result scoring 5.6 uses various back-off methods, including factored input and output nodes and two binode models (one less fine-grained, again as a means of back-off). We do not discuss the many aspects of the configuration, because we feel the space of possible configurations was not yet fully explored to draw any final conclusions.

4.3.1 BLEU Favours n-gram LMs

BLEU is known to favour methods employing n-gram based language models. An empirical evidence supporting claim that can be observed in Table 4.2: an n-gram LM gained 2 BLEU points for both "eaca" and "epcp".

In future experiments we plan to attempt two ways to tackle the problem: employing some LM-based rescoring even after the generation component (Ptáček and Žabokrtský, 2006), as well as using other automatic metrics of MT quality instead of BLEU to avoid the bias.

4.3.2 Error Cumulation

All components in our setup deliver only the single best candidate. Any errors will therefore accumulate over the whole pipeline. This primarily hurts the "etct" scenario where all our tools are employed.

In future, we would like to pass and accept several candidates, allowing each step in the calculation to do any necessary rescoring.

4.3.3 Conflict of Structures

Our current heuristic method of treelet extraction (Section 3.4) crucially depends on the quality of both English and Czech trees as well as the node alignment between them. A single error in any of the rigid sources may prevent to extract a treelet pair, not to mention natural divergence between the sentence and its translation. Precisely this reason explains the loss of performance of "eaca" compared to "epcp".

We hope that using the EM procedure (Section 2.3.4) will gain some recall. The current heuristic method can be also modified to accept a certain level of structure divergence, such as a certain portion of node-alignments leading out of the treelet pair. Another option is to try other tree-alignment methods such as proposed by (Smith and Eisner, 2006b).

4.3.4 Sentence Generation Tuned for Manual Trees

The rule-based generation system (Ptáček and Žabokrtský, 2006) was designed to generate Czech sentences from full-featured manual Czech tectogrammatical trees from the (monolingual) PDT.

Our target-side training trees are the result of an automatic analytical and tectogrammatical parsing procedure as implemented by McDonald et al. (2005) and Klimeš (2006), resp. Further noise is added during the tree transfer, so our final input to the generation component contains random errors in tree structure as well as missing or bad attribute values.

As the manual annotation of PCEDT proceeds, we may be able to train the transfer system on manual Czech trees. Simultaneously, we are improving the generation component to be more robust towards malformed input.

4.3.5 Errors in Source-Side Analysis

For the purposes of source-side English analysis, we still rely on very simple rules similar to those used by Čmejrek et al. (2003) to convert Collins (1996) parse trees to analytical and tectogrammatical dependency trees.

We hope to improve the English-side pipeline soon using recent taggers and parsers. Furthermore, the tectogrammatical analysis of English will be improved as manual English t-trees become available during PCEDT annotation, in progress.

In the meantime, we will perform a thorough analysis of our automatic trees to judge how much information relevant for translation is accidentally lost due to omissions in our rules. Alternatively, we might include some attributes based directly on a-trees in the source t-trees. This would serve as a back-off in case the a→t rules fail to provide all necessary information.

4.4 Conclusion

We have described a mathematical model of tree transformations and methods to automatically estimate model parameters from a parallel treebank. Two decoding algorithms to search for the most probable translation of an input tree were outlined and a preliminary version of the decoder was implemented. Several methods of back-off have been proposed and included in our implementation of the decoder.

Despite the empirical results of our approach so far being well below the state-of-the-art baseline, the pipeline of our process is complete and allows for an end-to-end evaluation. We have indicated several reasons for the poor performance and in the following research we will attempt to resolve the issues.

Chapter 5

Appendices

5.1 Acknowledgment

The work on this project was supported by the grant FP6-IST-5-034291-STP (EuroMatrix). We would like to thank Zdeněk Žabokrtský for his rules performing automatic annotation of English t-layer.

5.2 References

A. V. Aho and S. C. Johnson. 1976. Optimal code generation for expression trees. *J. ACM*, 23(3):488–501.

Ondřej Bojar. 2004. Problems of Inducing Large Coverage Constraint-Based Dependency Grammar for Czech. In *Constraint Solving and Language Processing, CSLP 2004*, volume LNAI 3438, pages 90–103, Roskilde University, September. Springer.

Ondřej Bojar. 2007. English-to-Czech factored machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 232–239, Prague, Czech Republic, June. Association for Computational Linguistics.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Meeting of the Association for Computational Linguistics*, pages 116–123.

Martin Čmejrek, Jan Cuřín, and Jiří Havelka. 2003. Czech-English Dependency-based Machine Translation. In *EACL 2003 Proceedings of the Conference*, pages 83–90. Association for Computational Linguistics, April.

Martin Čmejrek, Jan Cuřín, Jiří Havelka, Jan Hajič, and Vladislav Kuboň. 2004. Prague Czech-English Dependecy Treebank: Syntactically Annotated Resources for Machine Translation. In *Proceedings of LREC 2004*, Lisbon, May 26–28.

Martin Čmejrek. 2006. *Using Dependency Tree Structure for Czech-English Machine Translation*. Ph.D. thesis, ÚFAL, MFF UK, Prague, Czech Republic.

Michael Collins. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), Companion Volume*, pages 205–208, Sapporo, July.

Marcello Federico and Mauro Cettolo. 2007. Efficient handling of n-gram language models for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 88–95, Prague, Czech Republic, June. Association for Computational Linguistics.

Jan Hajič. 2004. *Disambiguation of Rich Inflection (Computational Morphology of Czech)*. Nakladatelství Karolinum, Prague.

Jan Hajič, Martin Čmejrek, Bonnie Dorr, Yuan Ding, Jason Eisner, Daniel Gildea, Terry Koo, Kristen Parton, Gerald Penn, Dragomir Radev, and Owen Rambow. 2002. Natural Language Generation in the Context of Machine Translation. Technical report. NLP WS'02 Final Report.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský, and Magda Ševčíková Razímová. 2006. Prague Dependency Treebank 2.0. LDC2006T01, ISBN: 1-58563-370-4.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical Syntax-Directed Translation with Extended Domain of Locality. In *Proc. of 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA.

Frederick Jelinek. 1985. Markov Source Modeling of Text Generation. In J. K. Skwirzinski, editor, *Impact of Processing Techniques on Communications*, pages 569–598, Dordrecht: Nijhoff. NATO Advanced Study Institute.

Paul Kingsbury and Martha Palmer. 2002. From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, Las Palmas, Spain.

Václav Klimeš. 2006. *Analytical and Tectogrammatical Analysis of a Natural Language.* Ph.D. thesis, ÚFAL, MFF UK, Prague, Czech Republic.

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *Proc. of EMNLP*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn. 2004a. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In Robert E. Frederking and Kathryn Taylor, editors, *AMTA*, volume 3265 of *Lecture Notes in Computer Science*, pages 115–124. Springer.

Philipp Koehn. 2004b. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT/EMNLP 2005*, October.

Igor A. Mel'čuk. 1988. *Dependency Syntax - Theory and Practice.* Albany: State University of New York Press.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In A. Meyers, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Marie Mikulová, Allevtina Bémová, Jan Hajič, Eva Hajičová, Jiří Havelka, Veronika Kolářová, Lucie Kučová, Markéta Lopatková, Petr Pajas, Jarmila Panevová, Magda Ševčíková, Petr Sgall, Jan Štěpánek, Zdeňka Urešová, Kateřina Veselá, and Zdeněk Žabokrtský. 2007. Annotation on the tectogrammatical level in the Prague Dependency Treebank. Project Euromatrix - Deliverable 3.1, ÚFAL, Charles University.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The Penn Discourse TreeBank. In *Proceedings of Fourth International Conference on Language Resources and Evaluation, LREC 2004*.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of the 17th conference on Computational linguistics*, pages 1086–1090. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*, pages 295–302.

Franz Joseph Och. 2002. *Statistical Machine Translation: From Single-Word Models to Alignment Templates*. Ph.D. thesis, RWTH Aachen University.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of the Association for Computational Linguistics*, Sapporo, Japan, July 6-7.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL 2002, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Jan Ptáček and Zdeněk Žabokrtský. 2006. Synthesis of Czech Sentences from Tectogrammatical Trees. In *Proc. of TSD*, pages 221–228.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania, May.

Stephen D. Richardson, William B. Dolan, Arul Menezes, and Monica Corston-Oliver. 2001. Overcoming the customization bottleneck using example-based mt. In *Proceedings of the workshop on Data-driven methods in machine translation*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence and Its Semantic and Pragmatic Aspects*. Academia/Reidel Publishing Company, Prague, Czech Republic/Dordrecht, Netherlands.

David A. Smith and Jason Eisner. 2006a. Minimum-risk annealing for training log-linear models. In *Proceedings of the International Conference on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL), Companion Volume*, pages 787–794, Sydney, July.

David A. Smith and Jason Eisner. 2006b. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30, New York, June.

Zoltan Somogyi, Fergus Henderson, and Thomas Conway. 1995. Mercury: an efficient purely declarative logic programming language. In *Proceedings of the Australian Computer Science Conference*, pages 499–512, Glenelg, Australia, February.

R. Zens, O. Bender, S. Hasan, S. Khadivi, E. Matusov, J. Xu, Y. Zhang, and H. Ney. 2005. The RWTH Phrase-based Statistical Machine Translation System. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, pages 155–162, Pittsburgh, PA, October.