Bimorphism Machine Translation

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR PHILOSOPHIAE
(Dr. phil.)

im Fachgebiet

Informatik

Vorgelegt

von Diplom-Sprachwissenschaftler Daniel Quernheim

geboren am 23. Februar 1985 in Soest

Die Annahme der Dissertation wurde empfohlen von

1. Professor Dr. Alexander Koller (Saarbrücken)
2. Professor Dr. Andreas Maletti (Leipzig)

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 10. April 2017 mit dem Gesamtprädikat
magna cum laude

Contents

List of Figures

List of Tables

List of Equations

Chapter 1

Introduction

The field of statistical machine translation has made tremendous progress due to the rise of statistical methods, making it possible to obtain a translation system automatically from a bilingual collection of text. Some approaches do not even need any kind of linguistic annotation, and can infer translation rules from raw, unannotated data. However, most state-of-the art systems do linguistic structure little justice, and moreover many approaches that have been put forward use ad-hoc formalisms and algorithms. This inevitably leads to duplication of effort, and a separation between theoretical researchers and practitioners.

In order to remedy the lack of motivation and rigor, the contributions of this dissertation are threefold:

1. After laying out the historical background and context, as well as the mathematical and linguistic foundations, a rigorous algebraic model of machine translation is put forward. We use regular tree grammars and bimorphisms as the backbone, introducing a modular architecture that allows different input and output formalisms.

2. The challenges of implementing this bimorphism-based model in a machine translation toolkit are then described, explaining in detail the algorithms used for the core components.

3. Finally, experiments where the toolkit is applied on real-world data and used for diagnostic purposes are described. We discuss how we use exact decoding to reason about search errors and model errors in a popular machine translation toolkit, and we compare output formalisms of different generative capacity.

Let us motivate these three aspects using an example. Consider the English sentence $\mathbf{e}$:[1]

[1] Adapted from "A Conservative Europe" by Michael Howard (https://www.project-syndicate.org/commentary/a-conservative-europe) with slight changes. The original spelling is: "We've heard that empty promise before". The German translation is: "Dieses leere Versprechen haben wir schon einmal gehört."

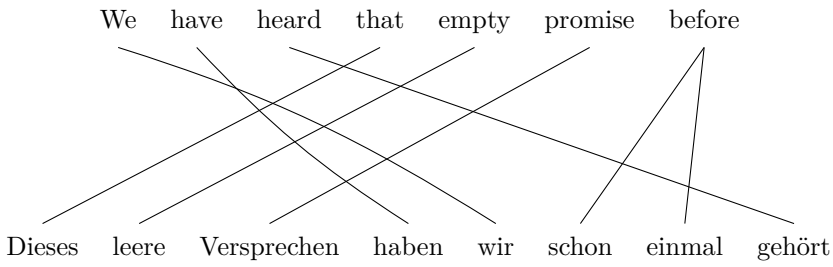


Figure 1. Word alignment between **e** and **g**

(**e**) *We have heard that empty promise before.*

How does a machine translation algorithm obtain a German translation? In this case, a professional human translator gave the rendering **g**, which is a perfectly acceptable translation:

(**g**) *Dieses leere Versprechen haben wir schon einmal gehört.*

What if **g** was the original sentence, and **e** is a translation? Or maybe both **g** and **e** are translations of a corresponding sentence in yet another third language? For the moment, let us assume that these questions are irrelevant. Let us instead assume that **e** and **g** came into existence simultaneously. This poses a number of interesting questions.

The close relationship between **e** and **g** motivates the question how a process generating the pair $(\mathbf{e}, \mathbf{g})$ at the same time might look. Imagine an abstract thought t whose "meaning" is **e** or **g**, depending on which language t is to be rendered in. Then we may want to write $e(t) = \mathbf{e}$ and $g(t) = \mathbf{g}$ to indicate that **e** is the English rendering of t, and **g** is the German rendering of t.

However, sentences are not monolithic. In fact, not only is **e** a translation of **g**, but also some parts of **e** are translations of parts of **g**. For instance, *"promise"* clearly corresponds to *"Versprechen"*. Similarly, the string of words *"that empty promise"* is a translation of *"Dieses leere Versprechen"*, and, of course, the entire English sentence is a translation of the German sentence. Let us write $a \mathrel{\widehat{=}} b$ to indicate that a is a translation of b. In general, these correspondences can be many-to-many words. Figure 1 illustrates the correspondences on the word level.

We thus arrive at a *word alignment* (we might want to treat it like a set of axioms), which can be generalized to a *phrase alignment*. We use the term *phrase* to denote any contiguous string of words. For instance, *"Dieses leere Versprechen"* is a translation of *"that empty promise"* because *"Dieses"* is a translation of *"that"*, *"leere"* is a translation of *"empty"*, and *"Versprechen"* is a translation of *"promise"*. We observe that words and phrases can be concatenated to form bigger phrases; we also call the structure of **e** and **g** a *hierarchical*

phrase structure (Chomsky, 1957). The formation of a translational correspondence can be written like

$$\frac{\text{that} \mathrel{\widehat{=}} \text{dieses} \quad \text{empty} \mathrel{\widehat{=}} \text{leere} \quad \text{promise} \mathrel{\widehat{=}} \text{Versprechen}}{\text{that empty promise} \mathrel{\widehat{=}} \text{Dieses leere Versprechen}}.$$

(Here, and in the following, we write the translational correspondence rules in the style of inference rules, with premises over the line, and the conclusion under the line.)

However, this inference rule is very specific. Assuming that there is a productive pattern, we may want to replace the words by variables, such as:

$$\frac{x_1 \mathrel{\widehat{=}} y_1 \quad x_2 \mathrel{\widehat{=}} y_2 \quad x_3 \mathrel{\widehat{=}} y_3}{x_1 x_2 x_3 \mathrel{\widehat{=}} y_1 y_2 y_3}. \tag{1}$$

Unfortunately, phrase alignment is not always linear like this. For instance, *"We have"* is a translation of *"haben wir"* because *"We"* is aligned to *"wir"*, and *"have"* is aligned to *"haben"*. This gives rise to another formation:

$$\frac{x_1 \mathrel{\widehat{=}} y_1 \quad x_2 \mathrel{\widehat{=}} y_2}{x_1 x_2 \mathrel{\widehat{=}} y_2 y_1}, \tag{2}$$

where x_1 was instantiated with *"We"*, x_2 with *"have"*, y_1 with *"wir"* and y_2 with *"haben"*. Note how the order differs between languages. This is called a *reordering*.

Using these translational correspondences, we can try to explain the generation of two corresponding sentences at the same time by assembling them from smaller units that have been generated simultaneously, and doing so recursively down to the word level.

However, things are more complicated. Unfortunately, rules like the above do not generalize well. For instance, even though *"Versprechen"* $\mathrel{\widehat{=}}$ *"promise"* and *"Dieses"* $\mathrel{\widehat{=}}$ *"that"*, arguably *"Versprechen Dieses"* is not a good translation of *"that promise"* because it is not a well-formed string of words in German. Clearly, Equation (2) cannot be applied in every context. Instead, reordering has to take into account the environment. Systematic reordering is found across many language pairs. For instance, in French, an adjective usually follows the noun it modifies, whereas in English and German it is the other way around. In German, the main verb of a subordinate clause is usually put at the end, which is not the case in English. Furthermore, there is optional reordering as exhibited in **e** and **g**. A version of **g** that more closely resembles **e** on the word level would be:

(**g**′) *Wir haben dieses leere Versprechen schon einmal gehört.*

Note how a word alignment between **e** and **g**′ exhibits significantly fewer crossings (compare Figures 1 and 2). Both **g**′ and **g** are valid translations of **e**; it is a matter of personal preference to choose one.

Also consider the English phrase *"we have heard"* in **e**. Unless we assume that it should not be translatable as a unit (and there is currently no reason to

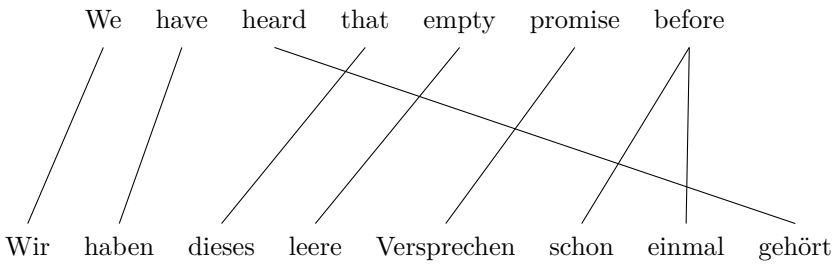


Figure 2. Word alignment between **e** and **g**′

assume so), we have to match it up with a discontiguous German phrase *"haben wir ... gehört"* in **g**. English–German translation exhibits a large number of discontiguities. Unfortunately, when we try to formulate a translational correspondence that only involves contiguous phrases, we run into a problem. In order to match *"we have heard"* with *"haben wir ... gehört"*, we also have to incorporate *"schon einmal"* on the German side, making it necessary to include *"before"* on the English side. This in turn requires to add *"that empty promise"* to form a contiguous phrase. However, then we end up with a cumbersome rule:

$$\frac{x_1 \mathrel{\widehat{=}} y_1 \quad x_2 \mathrel{\widehat{=}} y_2 \quad x_3 \mathrel{\widehat{=}} y_3 \quad x_4 \mathrel{\widehat{=}} y_4}{x_1x_2x_3x_4 \mathrel{\widehat{=}} y_3y_1y_4y_2},$$

inferred from the instantiation of x_1 with *"we have"*, x_2 with *"heard"*, x_3 with *"that empty promise"* and x_4 with *"before"*, as well as y_1 with *"haben wir"*, y_2 with *"gehört"*, y_3 with *"Dieses leere Versprechen"* and y_4 with *"schon einmal"*.

Therefore, it seems natural to allow translational correspondence rules to use discontiguous phrases, like so:

$$\frac{x_1 \mathrel{\widehat{=}} y_1 \quad x_2 \mathrel{\widehat{=}} y_2}{x_1x_2 \mathrel{\widehat{=}} (y_1, y_2)}. \tag{3}$$

This rule can be instantiated more easily, e.g., x_1 with *"we have"*, x_2 with *"heard"*, y_1 with *"haben wir"* and y_2 with *"gehört"*. We can furthermore instantiate x_3 with *"before"* and y_3 with *"schon einmal"* in the following translational correspondence rule to obtain a non-contiguity on the English side:

$$\frac{x_1x_2 \mathrel{\widehat{=}} (y_1, y_2) \quad x_3 \mathrel{\widehat{=}} y_3}{(x_1x_2, x_3) \mathrel{\widehat{=}} y_1y_3y_2}.$$

This shows how non-contiguity in the premise can lead to non-contiguity or contiguity in the conclusion.

Another problem is that nothing stops us from using rules with phrases for which they are not suitable. For instance, it is perfectly fine for auxiliary verb and participle to be split in German, but less so for adjective and noun. One possibility to remedy this is to restrict reordering and discontiguity to

be triggered only by specific words. This is the route chosen by word-based and phrase-based machine translation models. However, these models do not usually have a notion of hierarchical phrases as mentioned previously. In word- and phrase-based machine translation, a sentence is simply a sequence of words (or phrases).

In order to overcome this limitation, it seems natural to condition the type of applicable translational correspondence rules on the types of phrases involved. The grouping of phrases into categories should be neither too coarse nor too fine-grained, so that we are able to generalize well without overgenerating, i.e., generating many bad or ungrammatical translations. Linguistic theory provides a way to categorize phrases by type, namely *constituency parsing*. Figure 3 shows constituency parse trees for **e** and **g** that visualize the hierarchical structure of the sentences, along with the word alignment that we established already.[2]

We can now formulate translational correspondences by taking the phrase labels into account. For instance, we may condition the above-mentioned concatenation correspondence (Equation (1)) on the root labels of the subtrees involved, thereby treating phrases that are assigned the same label as equivalent:

$$\frac{x_1[\text{DT}] \mathrel{\widehat{=}} y_1[\text{PDAT}] \quad x_2[\text{JJ}] \mathrel{\widehat{=}} y_2[\text{ADJA}] \quad x_3[\text{NN}] \mathrel{\widehat{=}} y_3[\text{NN}]}{x_1x_2x_3[\text{NP}] \mathrel{\widehat{=}} y_1y_2y_3[\text{NP-OA}]}, \tag{4}$$

and we also observe that $x_1x_2x_3$ is of type NP, and $y_1y_2y_3$ is of type NP-OA. Formally, we introduce an equivalence relation $\equiv$ with equivalence classes defined by the root label of the subtree. We can also formulate a rule that uses tree sequences to explain the above-mentioned discontiguous correspondence (Equation (3)) conditioned on the labels involved:

$$\frac{x_1[\text{VBN}] \mathrel{\widehat{=}} y_1[\text{VVPP}] \quad x_2[\text{NP}] \mathrel{\widehat{=}} y_2[\text{NP-OA}]}{x_1x_2[\text{VP}] \mathrel{\widehat{=}} (y_2[\text{NP-OA}], y_1[\text{VVPP}])}. \tag{5}$$

We might be tempted to only consider translational correspondence between constituents. However, as we have seen, this may be too strict a restriction. Many collocations, such as *"there is"*, are not constituents in the linguistic sense, but translatable units.

Let us for the moment assume the restriction that we can only treat constituents as basic units in the input sentence, but (possibly non-contiguous) sequences of contituents in the output sentence. With this restriction, we can try to derive both **e** and **g** in a possible generative process, as seen in the deduction diagram in Figure 4. Another way to visualize this process is in the shape of a tree, as in Figure 5. Upon closer inspection, we note how this tree mimics the shape of the parse tree associated with **e** (cf. Figure 3). Every node in this derivation tree corresponds to a node in the parse tree of **e**, while this is not true for the parse tree of **g**. In fact, there is exactly one derivation step for every node in the parse tree of **e** that does not have exactly one child. We

[2]These parse trees have been generated using EGRET and BITPAR. See Section 3.2.2 for details.

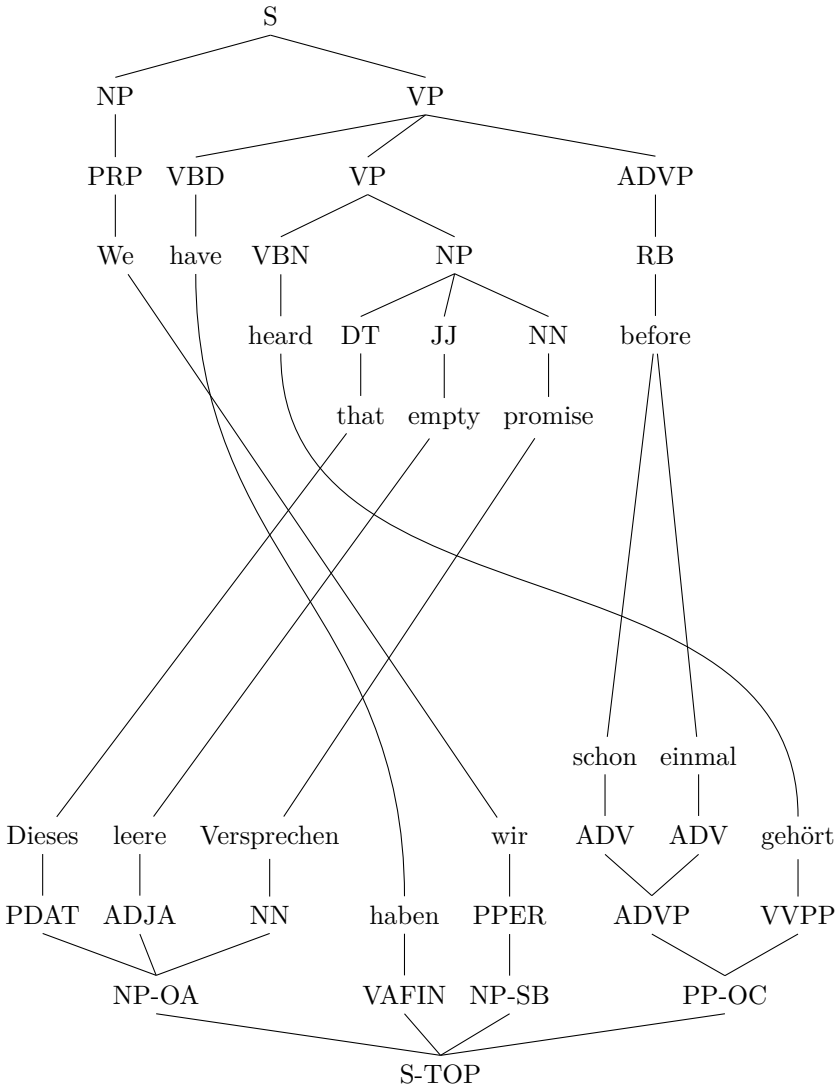


Figure 3. Constituent parse trees for **e** and **g** with word alignment

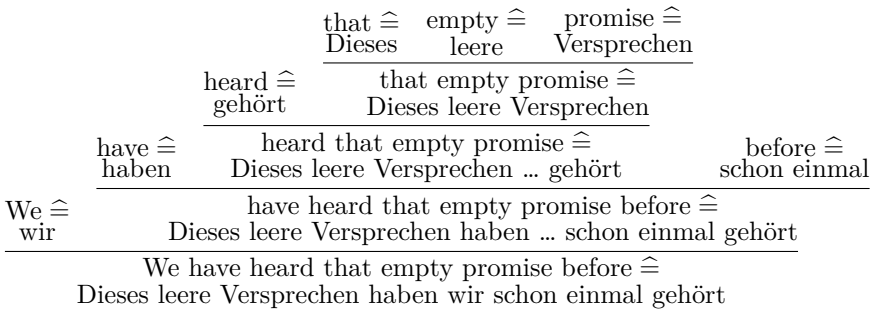


Figure 4. A possible derivation for **e** and **g**

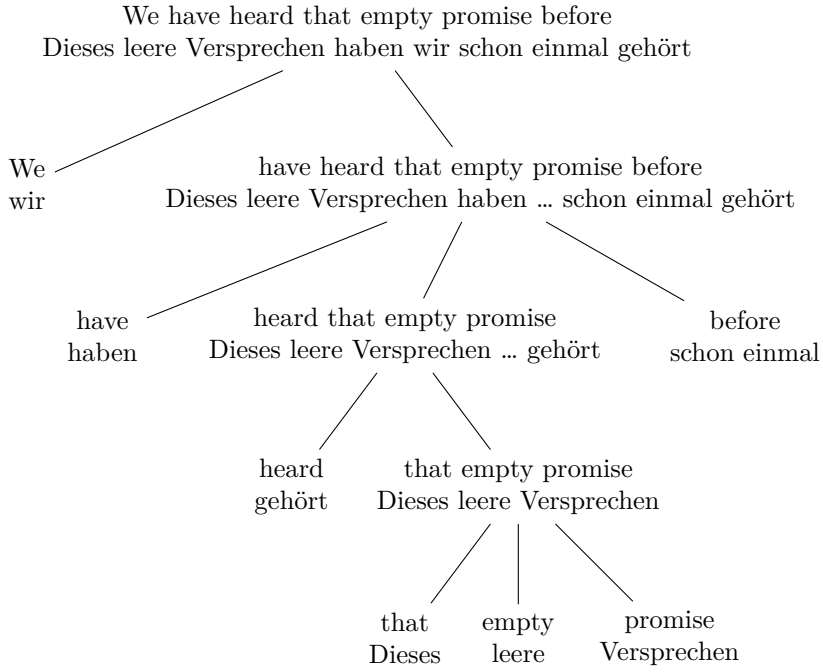


Figure 5. A possible derivation tree for **e** and **g**

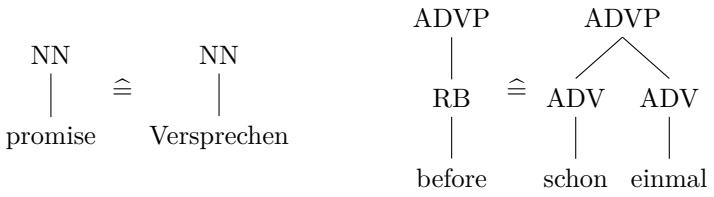


Figure 6. Word alignment lifted to trees

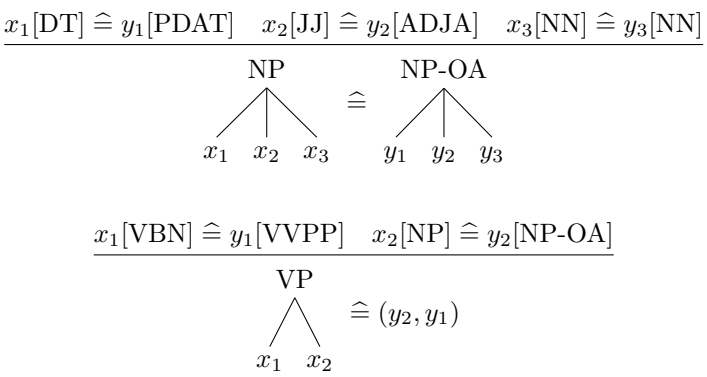


Figure 7. Translational correspondence rules on trees

do not rule out alternative derivations yet, but we assume that this derivation is the canonical one for the underlying parse trees.

This again exemplifies output discontiguities, which can be modeled by the *multi bottom-up tree transducer* formalism (Engelfriet et al., 2009) that will play a major role later in this dissertation. Furthermore, we use the notion of derivation trees to motivate the choice of *bimorphisms* (Arnold and Dauchet, 1982) as the backbone of our translation framework. Bimorphisms allow us to separate the constraints on how derivation steps can be combined, and how derivation steps are realized in the input and output language. The formation of translational correspondences using concatenation can be seen as the precursor of a bimorphism that uses concatenation for the input and output realization. After we introduced parse trees, we can now lift the word alignments to trees, as shown in Figure 6. We can also lift correspondence formation rules to work on trees and sequences of trees, as a precursor of a bimorphism operating with trees. An example is shown in Figure 7. In fact, these rules can also be extracted automatically from bilingual data.

The stateful behavior of a bimorphism determines how rules can be composed. In our example, this was determined by categories of phrases, motivated by linguistic syntax. Using appropriate generalizations, translation rules extracted from a large bilingual corpus can be used to translate sentences that have never been encountered.

With the theory of bimorphism machine translation in mind, we can identify three major questions that we will deal with in this dissertation:

- Given translations $\mathbf{f}$ and $\mathbf{e}$ (or a bilingual dataset of translations), what is a good formalism F that is suited to explain the observations? In particular, how expressive do the input and output realization mappings have to be? We already addressed some of the aspects that are relevant in this introduction, and we will need to find a balance between expressiveness and computational complexity.
- Given translations $\mathbf{f}$ and $\mathbf{e}$, for a given formalism F, how can we find a set of translational correspondences D that can be formulated in F to explain the observations? Depending on the formalism that has been chosen, starting from word alignments, admissible rules are extracted from training data. Furthermore, rules can be scored according to their relative usefulness, so that alternative derivations can be ranked.
- Given a sentence $\mathbf{f}$ and a set of translational correspondences D in a given formalism F, how can we find the best output sentence $\mathbf{e}$? This is the actual translation problem. The approach advocated in this dissertation is to compute the set of derivation trees such that for every derivation tree t, the source evaluation of the inference rules in t yields $\mathbf{f}$; this can be achieved by reversing the input mapping. Then the target evaluation of every such derivation tree t is a possible translation candidate.

More specific questions that arise from this theory of translation are manifold:

1. How should the generative process be modeled? This includes questions such as:
    - Do we need discontiguities on the input side as well as on the output side? Should discontiguities be restricted, or should an arbitrary number of components be allowed in the discontiguous constituents?
    - Do we need operations that are more complicated than concatenation and possibly reordering for the realization of translational correspondences, and which operations do we need on parse trees?
    - How do we restrict the way in which translational correspondences can be combined? How fine-grained should the linguistic annotation on the constituent labels be?
    - How do we model the generative process in a probabilistic way, to account for the fact that there is usually a choice in translation, but some translations are more likely than others?
2. Given an input sentence and a translation model, how do we obtain a translation? This includes questions such as:
    - How can we efficiently compute the set of derivations that explain the input sentence, and what is the output of these derivations?

- How can we incorporate other knowledge sources into this process?
- How can we restrict the possible derivations to those licenced by a parse tree or a set of parse trees for the input sentence, i.e., how can we incorporate a parser into the pipeline?

3. How does this theory hold up on real data?

   - Is it possible to build a translation model automatically from parallel data?
   - Can training data in the order of millions of sentences and the resulting models be handled?
   - Can large amounts of input be handled in a reasonable amount of time, and are the results competitive?
   - Does the implementation allow us to identify weaknesses or offer support for other theories?

We will turn to each of these questions in due time. The structure of this dissertation is as follows:

In Chapter 2, we will recall mathematical preliminaries, mainly some basic algebra and basic probability theory. In Chapter 3, we will introduce string and tree language theory and recall some basic linguistic theory, mainly context-free parsing, as well as the basic foundations of statistical machine translation, including a historical overview.

In Chapter 4, we will replace the "noisy channel model" and the linguistically uninformed non-hierarchical translation models by a unified theory of tree-based bimorphism translation in Section 4.1.2. Intuitively speaking, a bimorphism allows us to specify exactly how translational correspondence rules can be combined, and how they are realized in the input and output language.

In Chapter 5, we show how to obtain and to use a translation model. First, in Section 5.1, we show how to infer translation rules "from scratch", i.e., from a parallel corpus that has been parsed and equipped with word alignments (cf. Figure 3). In Section 5.2, we will address the problem of efficiently translating using a bimorphism representation of the translation model, which essentially involves the computation of an inverse mapping (to obtain the possible derivation trees) and the integration of a language model (Section 5.3). We will go into detail about crucial aspects of the algorithms that we implemented. Furthermore, in Section 5.4 we will explain how to set good rule and parameter weights, and how to optimize these by tuning on real data.

In Chapter 6, we will present three experiments that illustrate different use cases of the theory of this dissertation. The first experiment will investigate questions of engineering versus theory, including a comparison of output models of different power. The second experiment will investigate shortcomings of common approaches due to pruning (i.e., limiting the search space to speed up decoding at the cost of exactness). Finally, the third experiment will study the feasibility of our approach to decoding on realistic amounts of data.

Chapter 2

Preliminaries

Contents

This chapter is entirely self-contained. It explains basic concepts that are used throughout the remainder of this dissertation. In order to ease the cognitive load on the reader's mind, more advanced concepts will only be introduced when they are needed for the first time. In particular, strings, trees, languages over strings and trees, as well as automata are the subject of Section 3.1. Tree homomorphisms and bimorphisms will be introduced in Section 4.1.1.

An overview of frequently used symbols can be found in Table 2 for reference. These symbols are usually augmented by indices. For instance, t_1 will in all likelihood stand for a tree. An overview of frequently used notation can be found in Table 1, and common mappings and relations defined in this dissertation can be found in Table 3.

2.1 Set theory

We will use the standard logical connectors $\wedge$ 'and', $\vee$ 'or', $\implies$ 'implies' and $\iff$ 'if and only if', as well as the quantifiers $\exists$ 'there exists' and $\forall$ 'for all'.

A *set* S is any (finite or infinite) unordered collection of distinguishable objects, called elements. We write $s \in S$ to say that s is an *element* of S, and $s \notin S$ to say that s is not an element of S. We can define a set by listing its items like

$$\text{Suits} = \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}$$

or by characterizing the property that its elements satisfy (*set comprehension*):

$$\text{BlackSuits} = \{s \in \text{Suits} \mid s \text{ is black}\} = \{\clubsuit, \spadesuit\}.$$

notation	type of object
$A, B, \ldots, S, T, \ldots$	set
$a, b, \ldots, s, t, \ldots$	element
$i, j, k, \ldots$	natural number
R	relation
$A, B, \ldots$	alphabet
$a, b, \ldots$	symbol
$u, v, w, \ldots$	string
$\Sigma, \Gamma, \Delta, \ldots$	ranked alphabet
$\sigma, \gamma, \delta, \ldots$	ranked symbol
t, u	tree
p, w	position in a tree
L	(tree) language
F	weighted forest (tree series)
ϑ, ζ, ξ	substitution
$\mathcal{G}, \mathcal{D}$	automaton (or grammar)
Q	set of states
$p, q, \ldots$	state
P	set of transitions (or productions)
Ω	elementary events
$\mathcal{F}$	events
X, Y, Z	random variables
$\mathcal{L}$	set of all natural languages
$\mathcal{U}$	alphabet of all UNICODE® symbols
$E, F, \ldots$	alphabet (including tags) of natural language
$E^{(0)}, F^{(0)}$	alphabet (words only) of natural language
$\mathbf{E}, \mathbf{F}, \ldots$	natural languages, also as formal languages
$\mathbf{f}, \mathbf{e}$	source (input) and target (output) sentence
f, e	source (input) and target (output) tree
λ	feature weight vector
ϕ	feature function

Table 1. Notation used in this dissertation

symbol	meaning
$a \wedge b$, $a \vee b$	a and b, a or b
$a \implies b$, $a \iff b$	a implies b, a if and only if b
$\exists a$	there exists a
$\forall a$	for all a
$\{a, b, c\}$	set with elements a, b, c
$a \in A$	a is an element of A
$A \subseteq B$	A is a subset of B
$A \cup B$, $A \cap B$	union of A and B, intersection of A and B
$A - B$	difference of A and B
$\emptyset$	empty set
$\lvert S\rvert$, 2^S	cardinality of S, powerset of S
$\mathbb{N}, \mathbb{R}$	natural numbers, real numbers
$[n]$	$\{1, \ldots, n\}$
$A \times B$	Cartesian product of A and B
A^*	sequences (or strings) over A
$(), \varepsilon$	empty sequence, empty string
$v \sqsubseteq w$	v is a prefix of w
$v \leq w$	v precedes w lexicographically
$\pi_i(a)$	i-th element of sequence a
a_i^j	subsequence (or substring) from symbol i to j
$S^{\leq}$	set S as an ordered sequence
$R\,;S$	relation composition of R and S
$(x_j \mid j \in J)$	J-indexed family
$f : A \to B$	mapping from A to B
B^A	set of mappings from A into B
$[a]_\sim$	equivalence class of a in $\sim$
$L_1 L_2$	concatenation of L_1 and L_2
(Σ, rk)	ranked alphabet
$T_\Sigma(V)$	V-indexed trees over Σ
$t(w)$	label of t at position w
$t\lvert_w$	subtree of t at position w
$t[u]_w$	tree obtained by inserting u at position w in t
$\vartheta(t)$	substitution ϑ applied to t
$X : \Omega \to A$	random variable over A
$\Pr(X = x)$	probability of random variable X assuming value x
$\perp$	end-of-string marker
$x \mathrel{\widehat{=}} y$	x and y are translational correspondences
$x \equiv y$	x and y are (linguistically) interchangeable
$P \bowtie P'$	P and P' are consistently aligned
$(B, C) \smile (B', C')$	(B, C) and (B', C') define a unit of translation

Table 2. Symbols used in this dissertation

name	type	explanation
$\mathsf{supp}:$	$\mathbb{R}^A \to 2^A$	support of $F : A \to \mathbb{R}$
$\mathsf{index}:$	$S \to \mathbb{N}$	index of symbol in ordered set
$\mathsf{rk}:$	$\Sigma \to \mathbb{N}$	rank of symbol in (Σ, rk)
$\mathsf{pos}_L:$	$T_\Sigma(V) \to 2^{\mathbb{N}^*}$	positions of a tree with label in L
$\max \mathsf{pos}:$	$T_\Sigma(V) \to 2^{\mathbb{N}^*}$	frontier positions of a tree
$\mathsf{yield}:$	$T_\Sigma(V) \to \left(\Sigma^{(0)} \cup V\right)^*$	yield of a tree
$\mathsf{lower}:$	$\mho \to \mho$	lowercaser
$\mathsf{tokenize}_{\mathbf{L}}:$	$\mho^* \to \left(L^{(0)}\right)*$	tokenizer
$\mathsf{grams}_n:$	$A^* \to 2^{A^n}$	set of n-grams
$\#_b:$	$A^* \to \mathbb{N}$	number of occurrences of a string in b
$\mathsf{prec}_N:$	$\mathbf{E}^* \times \mathbf{E}^* \to \mathbb{R}$	clipped precision
$\mathsf{bp}:$	$\mathbf{E}^* \times \mathbf{E}^* \to \mathbb{R}$	brevity penalty
$\mathsf{bleu}_N:$	$\mathbf{E}^* \times \mathbf{E}^* \to \mathbb{R}$	BLEU-N score
$\bigwedge_U:$	$2^U \to \max U$	span of position set
$\bigtriangleup_U:$	$\max U \to 2^U$	closure of frontier position set
$\arg\max:$	$\mathbb{R}^S \to S$	element of S maximizing the function
$\alpha_{\mathcal{G}}:$	$Q_{\mathcal{G}} \to \mathbb{R}$	outside weight
$\beta_{\mathcal{G}}:$	$Q_{\mathcal{G}} \to \mathbb{R}$	inside weight
$\gamma_{\mathcal{G}}:$	$P_{\mathcal{G}} \to \mathbb{R}$	contribution of production

Table 3. Mappings and relations defined in this dissertation

Let S and T be sets. We say that S is a *subset* of T, written $S \subseteq T$, if $s \in T$ for all $s \in S$. Two sets are equal, written $S = T$, if and only if $S \subseteq T$ and $T \subseteq S$. We define *union*, *intersection* and *difference*:

$$\begin{aligned} S \cup T &= \{s \mid s \in S \text{ or } s \in T\} && \text{(union)}, \\ S \cap T &= \{s \mid s \in S \text{ and } s \in T\} && \text{(intersection)}, \\ \text{and } S - T &= \{s \mid s \in S \text{ and } s \notin T\} && \text{(difference)}. \end{aligned}$$

The *empty set* is denoted by $\emptyset$; $a \notin \emptyset$ for any a. Sets S and T are *disjoint* if $S \cap T = \emptyset$. We write 2^S for the *powerset* of S, defined by $2^S = \{S' \mid S' \subseteq S\}$. Note that $\emptyset \in 2^S$ and $S \in 2^S$ for every S. If S is finite, the *cardinality* $|S|$ of S is the number of elements in S; the cardinality of 2^S is $2^{|S|}$.

The set of *natural numbers*, i.e. nonnegative integers $\{0, 1, 2, \dots\}$, is denoted by $\mathbb{N}$. We write $\mathbb{N}^+$ for $\mathbb{N} - \{0\}$. For every $n \in \mathbb{N}^+$, we define $[n] = \{i \in \mathbb{N}^+ \mid 1 \leq i \leq n\}$. The set of *real numbers* is denoted by $\mathbb{R}$. We write $\mathbb{R}^+$ for $\{r \in \mathbb{R} \mid 0 \leq r\}$, and $[a, b]$ for $\{r \in \mathbb{R} \mid a \leq r \leq b\}$.

Let $A_1, \dots, A_n$ be sets. We define the generalized *Cartesian product*

$$\bigtimes_{i=1}^{n} A_i = \{(a_1, \dots, a_n) \mid \forall i \in [n] : a_i \in A_i\}.$$

The members of $\bigtimes_{i=1}^{n} A_i$ are called *sequences*, and we define the *length* of a

sequence $a \in \bigtimes_{i=1}^{n} A_i$ to be $|a| = n$. We furthermore define the special case $A^0 = \{()\}$, i.e., the set containing only the *empty sequence* (), and $|()| = 0$.

Instead of $\bigtimes_{i=1}^{n} A_i$, we can also write $A_1 \times \cdots \times A_n$. For a given set A, we write A^n for $\bigtimes_{i=1}^{n} A$ and $A^* = \bigcup_{n \in \mathbb{N}} A^n$. Note that $() \in A^*$ for every set A.

Let a be a sequence, and $i, j \in \mathbb{N}$. We write a_i for the i-th item of a, i.e., $a = (a_1, \ldots, a_n)$. Then a_i^j is shorthand notation for the *sub-sequence* $(a_{i'}, \ldots, a_{j'})$ where $i' = i$ if $i \geq 1$ and $i' = 1$ otherwise, and $j' = j$ if $j \leq n$ and $j' = n$ otherwise.

2.2 Relations and mappings

Let A, B, C be sets. A *relation* R from A into B is a subset of $A \times B$. We write $a\,R\,b$ for $(a, b) \in R$, and we call $\mathsf{dom}(R) = \{a \mid a\,R\,b\}$ the *domain* and $\mathsf{ran}(R) = \{b \mid a\,R\,b\}$ the *range* of R. For every subset $A' \subseteq A$, the *application* $R(A')$ is defined to be the set $\{b \mid \exists a \in A' : a\,R\,b\}$. For every sequence $s = (a_1, \ldots, a_k) \in A^*$, the application $R(s)$ is defined as $(R(a_1), \ldots, R(a_k))$. The *inverse* R^{-1} of R is the relation from B into A given by $R^{-1} = \{(a, b) \mid b\,R\,a\}$.

The relation R is a *mapping* (or *function*) if $a\,R\,b \wedge a\,R\,b' \implies b = b'$ for every $a \in A$ and $b, b' \in B$. We denote a mapping R from A into B by $R : A \to B$ ($A \to B$ is called the *type* of R), and write $R(a) = b$ instead of $R(\{a\}) = \{b\}$. If $a = (a_1, \ldots, a_k)$ is a sequence, we may drop the parentheses and write $R(a_1, \ldots, a_k)$ instead of $R((a_1, \ldots, a_k))$. The set of mappings from A into B is denoted by B^A. A mapping R is *injective* if $R(a) = R(a')$ implies $a = a'$ for every $a, a' \in A$, and it is *surjective* if $R^{-1}(\{b\}) \neq \emptyset$ for every $b \in B$. It is *bijective* if it is injective and surjective.

Let $R \subseteq A \times B$ and $R' \subseteq B \times C$ be relations. Then the *composition* $R\,;R'$ of R and R' is a relation from A into C defined by $R\,;R' = \{(a, c) \mid \exists b : a\,R\,b \wedge b\,R'\,c\}$. We generalize composition to sets of relations by letting $\mathcal{R}\,;\mathcal{R}' = \{R\,;R' \mid R \in \mathcal{R}, R' \in \mathcal{R}'\}$.

We identify a set A with its *identity relation* $\mathsf{id}(A) = \{(a, a) \mid a \in A\}$. A is *countable* if there exists an injective mapping $m : A \to \mathbb{N}$. A mapping $R : A^n \to A$ can be called an n-ary *operation* on A. A subset $A' \subseteq A$ is said to be *closed* under R if $R(a') \in A'$ for every $a' \in (A')^n$.

A binary relation $R \subseteq A \times A$ is *reflexive* if $a\,R\,a$ for every $a \in A$, *symmetric* if $a\,R\,b$ implies $b\,R\,a$, *antisymmetric* if $a\,R\,b$ and $b\,R\,a$ implies $a = b$, and *transitive* if $a\,R\,b$ and $b\,R\,c$ implies $a\,R\,c$ for all $a, b, c \in A$. An *equivalence relation* on A is a relation $\sim$ that is reflexive, symmetric and transitive. For every $a \in A$, we call the set $\{b \in A \mid a \sim b\}$ the *equivalence class* of a, denoted $[a]_\sim$. The set of equivalence classes of A is denoted $A/\!\sim$. Every mapping $f : A \to B$ induces an equivalence relation $\sim_f$ on A, given by $a \sim_f b$ if $f(a) = f(b)$. Hence, we have $[a]_{\sim_f} = f^{-1}(\{f(a)\})$.

A *partial order* on A is a relation $\leq \subseteq A \times A$ that is reflexive, antisymmetric and transitive. If $a \leq b$ or $b \leq a$, then a and b are *comparable* with regard to $\leq$. Every partial order $\leq$ has an inverse $\geq = \leq^{-1}$ that is also a partial order. A *linear order* is a partial order that is total, i.e., all elements are comparable.

Every order $\leq$ on A can be turned into a *strict order* $< = \leq - \mathsf{id}(A)$. Let $\leq$ be a partial order on A. For a given subset $S \subseteq A$, we define the *minimal* and *maximal* set by

$$\min_{\leq} S = \{s \in S \mid \forall a \in A : a \leq s \implies a = s\} \quad \text{and}$$
$$\max_{\leq} S = \{s \in S \mid \forall a \in A : a \geq s \implies a = s\}\,.$$

We drop the subscript when the order is clear from the context.

Let $\leq$ be a linear order on A. For a given finite subset $S \subseteq A$, we write $S^{\leq}$ for the sequence $(s_1, \ldots, s_n)$ such that $s_1 < \cdots < s_n$ and $S = \{s_1, \ldots, s_n\}$, and $\mathsf{index} : S \to [n]$, defined by $\mathsf{index}(s_i) = i$ for all $i \in [n]$. Furthermore, for $S \neq \emptyset$, $|\min_{\leq} S| = |\max_{\leq} S| = 1$, i.e., minimal and maximal elements are uniquely defined.

Finally, let J and X be sets. A (J-indexed) *family* is a mapping $f : J \to X$. We may denote $f(j)$ by x_j and f by $(x_j \mid j \in J)$. J is called an *index set*.

2.3 Probability theory

A discrete *probability space* is a triple $(\Omega, \mathcal{F}, \Pr)$ consisting of

- a non-empty countable set Ω, the set of *elementary events*;
- the set $\mathcal{F} = 2^{\Omega}$, called *events*;
- a mapping $\Pr : \mathcal{F} \to [0, 1]$ such that:
  - $\Pr(\Omega) = 1$; and
  - for every mapping $I : \mathbb{N} \to \mathcal{F}$ such that $I(i) \cap I(j) = \emptyset$ for every $i \neq j$, we have:

$$\Pr\left(\bigcup_{n \in \mathbb{N}} I(n)\right) = \sum_{n \in \mathbb{N}} \Pr(I(n))\,.$$

We call $\Pr(\omega)$ the *probability* of the event $\omega \in \mathcal{F}$ occurring.

Let A be a non-empty countable set. A (discrete) *random variable* X over A is a mapping $X : \Omega \to A$. For $a \in A$, we set

$$\Pr(X = a) = \Pr(X^{-1}(a))\,, \tag{6}$$

i.e., the probability of X assuming the value a is the probability of the associated event $X^{-1}(a)$. Then Pr is called a *probability distribution* over A. When the random variable is understood from context, we can write a instead of $X = a$, and $\Pr(a)$ instead of $\Pr(X = a)$.

Let X, Y be random variables over A. We will write x for $X = x$, and y for $Y = y$ to simplify the presentation. The *joint probability* $\Pr(x, y)$ is defined by

$$\Pr(x, y) = \Pr(X^{-1}(x) \cap Y^{-1}(y))\,. \tag{7}$$

In other words, it is the probability of an event occurring that satisfies both $X = x$ and $Y = y$. The *conditional probability* $\Pr(y|x)$ of $Y = y$ given $X = x$ is defined by:

$$\Pr(y|x) = \begin{cases} \frac{\Pr(x,y)}{\Pr(x)} & \text{if } \Pr(x) \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Using a reformulation of Equation (8), the joint probability can be expressed using conditional probabilities by the *chain rule*:

$$\Pr(x, y) = \Pr(x) \cdot \Pr(y|x)\,. \tag{9}$$

Another useful equation is *Bayes' rule* (after mathematician Thomas Bayes):

$$\Pr(y|x) = \frac{\Pr(x|y) \cdot \Pr(y)}{\Pr(x)}\,. \tag{10}$$

Chapter 3

Background

Contents

This chapter will explain the linguistic background needed in this dissertation, as well as a formalization of strings, trees and languages. Some computational linguistic techniques that are used for the processing of the linguistic data in the experiments are introduced. Most notions of this chapter involving trees, tree languages and tree automata are further explained in Engelfriet (2015), Gécseg and Steinby (2015), and Comon et al. (2007).

3.1 Strings, trees, languages

An *alphabet* is any nonempty finite set. The elements of an alphabet are called *symbols*. If A is an alphabet, we call elements of A^* *strings*. For simplicity, we will write a for the string (a) that consists of one symbol only, and $a_1 \cdots a_n$ for the string $(a_1, \ldots, a_n)$. The *empty string* () will be denoted by ε. The *concatenation* of two strings v and w will be denoted by $v.w$ or just by juxtaposition: vw. It is defined by $(v_1, \ldots, v_k)(w_1, \ldots, w_m) = (v_1, \ldots, v_k, w_1, \ldots, w_m)$.

Let $a, b \in A^*$. We call a an *n-gram* if $|a| = n$; if $n = 1$ (or $2, 3$), we call a a *unigram* (or *bigram*, *trigram*). We say that a is *substring* of b if there exist

$c, d \in A^*$ such that $b = cad$. If $|a| = n$, we may also say that a is an n-gram of b. The *substring count* $\#_b : A^* \to \mathbb{N}$ is defined by:

$$\#_b(v) = |\{(u, w) \in A^* \times A^* \mid uvw = b\}|.$$

Furthermore, a is a *prefix* of b, written $a \sqsubseteq b$, if there exists $c \in A^*$ such that $ac = b$. The relation $\sqsubseteq$ is a partial order.

A linear order $\leq$ on A extends to a linear order on A^*, called the *lexicographic order* extending $\leq$ such that:

- $\varepsilon \leq w$ for all $w \in A^*$,
- $v \leq w$ implies $av \leq aw$ for every $a \in A$ and $v, w \in A^*$, and
- $a \leq b$ implies $av \leq bw$ for every $a, b \in A$ and $v, w \in A^*$.

It is also a linear order.

A subset L of A^* is called a (string) *language* over A. A sequence $T = (t_1, \ldots, t_k)$ of strings $t_1, \ldots, t_k \in L$ is called a *text* in L. We extend $\#$ to texts such that $\#_T(v) = \sum_{i=1}^{k} \#_{t_i}(v)$. We extend concatenation to languages by defining $L_1.L_2 = \{vw \mid v \in L_1, w \in L_2\}$ for every $L_1, L_2 \subseteq A^*$.

String languages are commonly classified according to their expressivity. A particularly well-known characterization is the *Chomsky hierarchy* (Hopcroft and Ullman, 1979). In this dissertation, we will deal with Chomsky type 3 languages (*regular languages*) and Chomsky type 2 languages (*context-free languages*). In particular, a language $L \subseteq A^*$ is regular if and only if the equivalence relation $\equiv$ on A^*, given by $x \equiv y$ if and only if $xz \in L \iff yz \in L$ for every $x, y, z \in A^*$, has only finitely many equivalence classes. This fundamental theorem is known as the *Myhill-Nerode theorem* (Hopcroft and Ullman, 1979). Context-free string languages will briefly be mentioned again in Section 3.2.2.

A *ranked alphabet* is a pair (Σ, rk) where Σ is an alphabet and $\mathsf{rk} : \Sigma \to \mathbb{N}$ assigns a *rank* (or *arity*) to every symbol. If $\mathsf{rk}(\sigma) = 0$ $(1, 2, 3)$, we call σ a *nullary* (*unary*, *binary*, *ternary*) symbol. We define $\Sigma^{(k)} = \{\sigma \in \Sigma \mid \mathsf{rk}(\sigma) = k\}$. To simplify presentation, we will often not explicitly mention the rank function and only write Σ for the ranked alphabet. We will also sometimes introduce an element $\sigma \in \Sigma^{(k)}$ together with its rank as $\sigma^{(k)}$. A string over a ranked alphabet Σ is any element of $(\Sigma^{(0)})^*$. In this way, every alphabet A can be extended to a ranked alphabet $(A, A \times \{0\})$.

Example 1 (*Ranked alphabet.*)
Let $\Sigma = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{PP}, \mathrm{Det}, \mathrm{N}, \mathrm{V}, \mathrm{P}, \mathrm{the}, \mathrm{cat}, \mathrm{sat}, \mathrm{on}, \mathrm{mat}\}$ with $\mathsf{rk} : \Sigma \to \mathbb{N}$ given by:

$$\begin{aligned} \mathsf{rk}(\mathrm{S}) = \mathsf{rk}(\mathrm{NP}) = \mathsf{rk}(\mathrm{VP}) = \mathsf{rk}(\mathrm{PP}) &= 2, \\ \mathsf{rk}(\mathrm{Det}) = \mathsf{rk}(\mathrm{N}) = \mathsf{rk}(\mathrm{V}) = \mathsf{rk}(\mathrm{P}) &= 1, \\ \mathsf{rk}(\mathrm{the}) = \mathsf{rk}(\mathrm{cat}) = \mathsf{rk}(\mathrm{sat}) = \mathsf{rk}(\mathrm{on}) = \mathsf{rk}(\mathrm{mat}) &= 0. \end{aligned}$$

Then (Σ, rk) is a ranked alphabet. The string "$(\mathrm{the}, \mathrm{cat}, \mathrm{sat}, \mathrm{on}, \mathrm{the}, \mathrm{mat})$" (written as a sequence) is a string over Σ.

Let Σ be a ranked alphabet and V a set. We define $\Sigma(V) = \{\sigma^{(k)}(v_1, \ldots, v_k) \mid v_1, \ldots, v_k \in V, \sigma \in \Sigma\}$. We write $T_\Sigma(V)$ for the set of *V-indexed Σ-trees*, defined

as the smallest set T such that $V \subseteq T$ and $\Sigma(T) \subseteq T$. We then call elements of V *variables*. We write T_Σ for $T_\Sigma(\emptyset)$, the set of *ground Σ-trees*, i.e., variable-free Σ-trees. A *tree language* over Σ is any subset of T_Σ.

For each tree $t \in T_\Sigma(V)$ we identify *nodes* by *positions*. Positions are strings over $\mathbb{N}$. The set of positions of a tree is inductively defined by the function $\mathsf{pos} : T_\Sigma(V) \to 2^{\mathbb{N}^*}$:

$$\mathsf{pos}(t) = \begin{cases} \{\varepsilon\} & \text{if } t \in V \\ \{\varepsilon\} \cup \bigcup_{1 \leq i \leq k} (\{i\}.\, \mathsf{pos}(t_i)) & \text{if } t = \sigma(t_1, \ldots, t_k) \end{cases}$$

Note that the set of positions is prefix-closed, i.e., $vw \in \mathsf{pos}(t)$ with $v, w \in \mathbb{N}^*$ implies $v \in \mathsf{pos}(t)$. The set of *frontier* positions of t is $\max_{\sqsubseteq} \mathsf{pos}(t)$, i.e., the set of all positions that are not a prefix of any other position. We write $t(w)$ for the *label* (taken from $\Sigma \cup V$) of t at position $w \in \mathsf{pos}(t)$, defined by:

$$t(w) = \begin{cases} t & \text{if } w = \varepsilon \text{ and } t \in V \\ \sigma & \text{if } w = \varepsilon \text{ and } t = \sigma(t_1, \ldots, t_k) \\ t_i(w') & \text{if } w = iw', i \in [k], w' \in \mathbb{N}^* \text{ and } t = \sigma(t_1, \ldots, t_k) \end{cases}$$

Example 2
Let Σ be the ranked alphabet defined in Example 1. This is a Σ-tree:

$$t = \mathrm{S}(\mathrm{NP}(\mathrm{Det}(\mathrm{the}), \mathrm{N}(\mathrm{cat})), \\ \mathrm{VP}(\mathrm{V}(\mathrm{sat}), \mathrm{PP}(\mathrm{P}(\mathrm{on}), \mathrm{NP}(\mathrm{Det}(\mathrm{the}), \mathrm{N}(\mathrm{mat})))))).$$

Figure 8 shows the graphical representation of t, where positions have been annotated at the nodes. For instance, $t(11) = t(2221) = \mathrm{Det}$, and $t(\varepsilon) = \mathrm{S}$.

We will use this kind of visualization (usually without position annotations) throughout this dissertation, with the root of the tree at the top, and subtrees sorted left-to-right. Occasionally, we might visualize trees with the root at the bottom, growing upwards, as in Figure 9. Since trees have one unique root, this should not cause any confusion.

We use $t|_w$ to address the *subtree* of t that is rooted in position w:

$$t|_w = \begin{cases} t & \text{if } w = \varepsilon \\ t_i|_{w'} & \text{if } w = iw' \text{ and } t = \sigma(t_1, \ldots, t_k), i \in [k] \end{cases}$$

We write $t[u]_w$ to represent the tree that is obtained from replacing the subtree $t|_w$ at w by $u \in T_\Sigma(V)$:

$$t[u]_w = \begin{cases} u & \text{if } w = \varepsilon \\ \sigma(t_1, \ldots, t_i[u]_{w'}, \ldots, t_k) & \text{if } w = iw' \text{ and } t = \sigma(t_1, \ldots, t_k), i \in [k] \end{cases}$$

If $(w_1, \ldots, w_k) \in \mathsf{pos}(t)^k$ is a sequence of pairwise incomparable positions with regard to $\sqsubseteq$, and $(u_1, \ldots, u_k) \in T_\Sigma(V)^k$, then

$$t[(u_1, \ldots, u_k)]_{(w_1, \ldots, w_k)} = (\cdots t[u_1]_{w_1} \cdots)[u_k]_{w_k}\,.$$

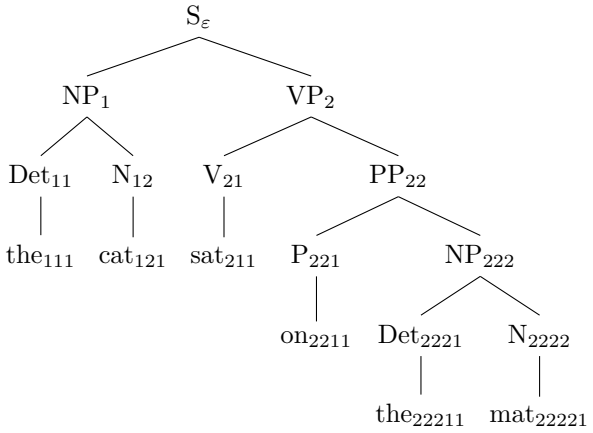


Figure 8. Graphical representation of a tree with positions annotated

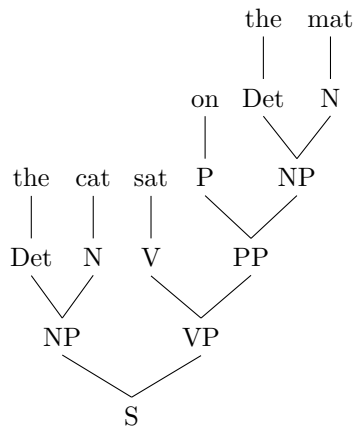
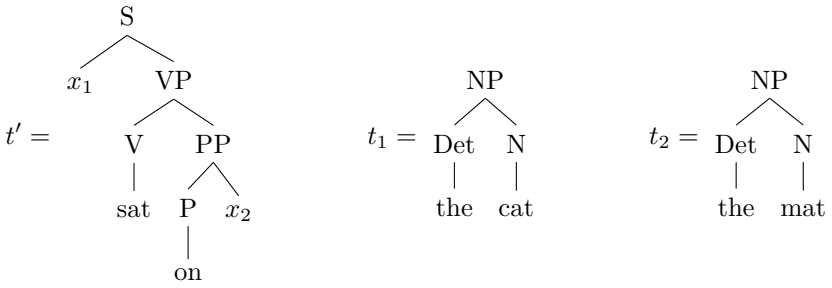


Figure 9. Graphical representation of a tree with the root at bottom, growing upwards

Example 3
Recall t from Example 2. We have $t = (t'[t_1]_1)[t_2]_{222}$ with



For a given set $L \subseteq \Sigma \cup V$ of labels, we let

$$\mathsf{pos}_L(t) = \{w \in \mathsf{pos}(t) \mid t(w) \in L\}$$

be the set of all positions labeled with a symbol or variable from L. Furthermore, we define the (string) *yield* of a tree as the concatenation of its frontier symbols and variables by $\mathsf{yield} : T_\Sigma(V) \to \left(\Sigma^{(0)} \cup V\right)^*$:

$$\mathsf{yield}(t) = \begin{cases} t & \text{if } t \in \Sigma^{(0)} \cup V \\ \mathsf{yield}(t_1) \cdots \mathsf{yield}(t_k) & \text{if } t = \sigma(t_1, \ldots, t_k) \end{cases}$$

Example 4
The yield of the tree t in Example 2 is the string (written as a sequence)

$$\mathsf{yield}(t) = (\text{the}, \text{cat}, \text{sat}, \text{on}, \text{the}, \text{mat})\,.$$

A *substitution* is a function $\vartheta : V \to T_\Sigma(V)$. We extend ϑ to trees as follows:

$$\vartheta(t) = \begin{cases} \vartheta(t) & \text{if } t \in V \\ \sigma(\vartheta(t_1), \ldots, \vartheta(t_k)) & \text{if } t = \sigma(t_1, \ldots, t_k) \end{cases}$$

If $T = (t_1, \ldots, t_k)$ is a sequence of trees, then $\vartheta(T) = (\vartheta(t_1), \ldots, \vartheta(t_k))$. A *ground substitution* is a function $\vartheta : V \to T_\Sigma$.

Example 5
Recall t, t', t_1 and t_2 from Example 3. With the substitution ϑ such that $\vartheta(x_1) = t_1$ and $\vartheta(x_2) = t_2$, we have $\vartheta(t') = t$.

3.1.1 Weighted languages and relations

Let A be an alphabet, and Σ a ranked alphabet. A *weighted language* is any mapping $F : A^* \to \mathbb{R}$. A *weighted tree language* (or *tree series*, or *weighted*

forest) is a mapping $F : T_\Sigma \to \mathbb{R}$. We then call the elements of $\mathbb{R}$ weights[3]. We define the *support* of a weighted language F as $\mathsf{supp}(F) = F^{-1}(\mathbb{R} - \{0\})$.

Let F_1 and F_2 be two weighted (tree) languages over Σ. We extend $+$ and $\cdot$ in the natural way to weighted languages by:

- $(F_1 + F_2) : T_\Sigma \to \mathbb{R}$ with $(F_1 + F_2)(s) = F_1(s) + F_2(s)$;
- $(F_1 \cdot F_2) : T_\Sigma \to \mathbb{R}$ with $(F_1 \cdot F_2)(s) = F_1(s) \cdot F_2(s)$.

We may also write $F_1 \cup F_2$ for $F_1 + F_2$ and $F_1 \cap F_2$ for $F_1 \cdot F_2$. For every $F \subseteq T_\Sigma$, we may implicitly treat F like its *characteristic function* $F_\mathbb{R} : T_\Sigma \to \mathbb{R}$ with $F_\mathbb{R}(s) = 1$ if $s \in F$ and $F_\mathbb{R}(s) = 0$ otherwise.

Let A, B, C be sets. A *weighted relation* from A into B is a mapping $R : A \times B \to \mathbb{R}$. The inverse of R is $R^{-1} : B \times A \to \mathbb{R}$ with $R^{-1}(b, a) = R(a, b)$ for all $a \in A$ and $b \in B$. The weighted composition of two weighted relations $R_1 : A \times B \to \mathbb{R}$ and $R_2 : B \times C \to \mathbb{R}$ is $R_1 ; R_2 : A \times C \to \mathbb{R}$ with

$$R_1 ; R_2(a, c) = \sum_{b \in B} R_1(a, b) \cdot R_2(b, c) .$$

Please note that this summation might be infinite if $\{b \in B \mid R_1(a, b) \neq 0\}$ is infinite. Fortunately, we will generally only consider weighted relations where any element is only related to finitely many elements with non-zero weight. Like above, the characteristic functions of unweighted relations may be used, and moreover, identity relations of sets or (weighted or unweighted) languages may be used.

Let A and B be sets, $F : A \to \mathbb{R}$, and $h : A \to B$. Then the *application* of h to F is defined as the mapping $h(F) : B \to \mathbb{R}$ such that $h(F)(b) = \sum_{a \in h^{-1}(b)} F(a)$ for all $b \in B$. (Again, we will generally only consider cases where this summation is finite.) The *inverse application* of h to $G : B \to \mathbb{R}$, written $h^{-1}(F) : A \to \mathbb{R}$, is defined by $h^{-1}(G)(b) = G(h(a))$ for all $a \in A$.

3.1.2 Tree automata

Let Σ be a ranked alphabet. Let $X = (x_i \mid i \in \mathbb{N})$ be a family of variables. A *weighted Σ-tree automaton* $\mathcal{G}$ is a triple (Q, I, P) where[4]

- Q is a finite set of *states*,
- $I : Q \to \mathbb{R}$ is the *initial weight* mapping, and
- $P : Q \times \Sigma(Q) \to \mathbb{R}$ is a *transition weight* mapping.

We may define I and P partially, then $I(q) = 0$ and $P(q, \sigma(q_1, \ldots, q_k)) = 0$ when not specified otherwise.

[3] In this dissertation, we will only consider weighted languages over $\mathbb{R}$. Most of the constructions will work for any commutative semiring, i.e., a structure $(S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ such that $(S, \oplus, \mathbf{0})$ and $(S, \otimes, \mathbf{1})$ are commutative monoids, $\mathbf{0} \otimes s = s \otimes \mathbf{0} = \mathbf{0}$ for all $s \in S$, and $\otimes$ distributes over $\oplus$. Definitions for these terms can be found in Droste and Kuich (2009), and the theory of semiring-weighted languages can be found in Droste et al. (2009).

[4] Please note that tree automata are regular tree grammars in normal form. We will only use tree automata and not general regular tree grammars in this dissertation.

Let $t \in T_\Sigma(X)$ be a tree. A *run* of $\mathcal{G}$ on t is a mapping $r : \mathsf{pos}(t) \to Q$. The weight of a run r is

$$\mathsf{wt}(r) = \prod_{p \in \mathsf{pos}_\Sigma(t)} P(r(p), r'(p))\,,$$

where $r'(p) = t(p)(r(p1), \ldots, r(pk))$ with $t(p) \in \Sigma^{(k)}$. The set of all runs of $\mathcal{G}$ on t is $\mathsf{runs}_\mathcal{G}(t)$. Note that $\mathsf{runs}_G(t)$ is finite for every $t \in T_\Sigma$. The *weighted tree language* (also called *formal tree series*) recognized by $\mathcal{G}$ is $L_\mathcal{G} : T_\Sigma \to A$, defined by

$$L_\mathcal{G}(t) = \sum_{r \in \mathsf{runs}_\mathcal{G}(t)} I(r(\varepsilon)) \cdot \mathsf{wt}(r)\,.$$

$\mathcal{G}$ is *unambiguous* if $|\{r \in \mathsf{runs}_\mathcal{G}(t) \mid \mathsf{wt}(r) \neq 0\}| \leq 1$ for every $t \in T_\Sigma$. $\mathcal{G}$ is *deterministic* if $P(q, \sigma(q_1, \ldots, q_k)) \neq 0$ and $P(q', \sigma(q_1, \ldots, q_k)) \neq 0$ implies $q = q'$. If $\mathcal{G}$ is deterministic, then $\mathcal{G}$ is unambiguous. If $P(q, \sigma(q_1, \ldots, q_k)) \in \{0, 1\}$ and $I(q) \in \{0, 1\}$ for all $\sigma^{(k)} \in \Sigma$ and $q, q_1, \ldots, q_k \in Q$, then $\mathcal{G}$ is a trivially weighted or simply *unweighted tree automaton*. $\mathcal{G}$ is *acyclic* if there is no tree $t \in T_\Sigma$ and run $r : \mathsf{pos}(t) \to Q$ with $r(p) = r(q)$ for $p \sqsubseteq q$ and $p \neq q$ such that $\mathsf{wt}(r) \neq 0$.

Example 6 (*Tree automaton.*)
Let $\Sigma = \{\alpha^{(2)}, \beta^{(0)}\}$ be a ranked alphabet, and $\mathcal{K} = (\{q\}, I, P)$ a weighted tree automaton with $I(q) = 1$ and P specified by: $P(q, \alpha(q, q)) = 1$ and $P(q, \beta()) = 2$. $\mathcal{K}$ is unambiguous and deterministic, but not acyclic. It computes the function $L_\mathcal{K}(t) = 2^n$ for every tree $t \in T_\Sigma$, where $n = |\,\mathsf{pos}_{\{\beta\}}(t)|$.

The class of weighted tree languages over Σ recognized by weighted Σ-tree automata is called the class of *weighted regular tree languages* over Σ, denoted $\mathsf{REC}(\Sigma)$. Let $L_1, L_2 \in \mathsf{REC}(\Sigma)$. Then:

$$\begin{aligned} &L_1 \cup L_2 \in \mathsf{REC}(\Sigma) && (\mathsf{REC}(\Sigma) \text{ is closed under union}),\\ &L_1 \cap L_2 \in \mathsf{REC}(\Sigma) && (\mathsf{REC}(\Sigma) \text{ is closed under intersection}). \end{aligned}$$

These *closure properties* can be used to reason about tree languages by building them up from simpler tree languages, thus simplifying proofs and facilitating modelling and engineering tasks. A well-known construction for intersection in the unweighted and weighted case is the *product construction* (see, e.g., Berstel and Reutenauer (1982)). Let $\mathcal{G}_1 = (Q_1, I_1, P_1)$ and $\mathcal{G}_2 = (Q_2, I_2, P_2)$ be weighted tree automata over the same ranked alphabet Σ. We construct $\mathcal{G}_1 \cap \mathcal{G}_2 = (Q_1 \times Q_2, I_\cap, P_\cap)$ where

$$\begin{aligned} I_\cap(q, q') &= I_1(q) \cdot I_2(q) && \text{and}\\ P_\cap((q, q'), \sigma((q_1, q_1'), \ldots, (q_k, q_k'))) &= P_1(q, \sigma(q_1, \ldots, q_k))\\ &\quad \cdot P_2(q', \sigma(q_1', \ldots, q_k'))\,. \end{aligned}$$

Then $L_{\mathcal{G}_1 \cap \mathcal{G}_2} = L_{\mathcal{G}_1} \cap L_{\mathcal{G}_2}$.

3.2 Linguistic techniques

For the remainder of this dissertation, we will assume that $\mathcal{L}$ stands for the set of natural languages, and identifiers of natural languages are written in boldface, e.g., **E** for English, **F** for French, **G** for German etc. A natural language **L** coincides with its extension, the set of all sentences in **L**. In this way, **L** denotes a natural language, and its characterization as a formal language, i.e., a set of strings over a yet to be defined alphabet.

Data-driven computational linguistics makes extensive use of annotated as well as raw linguistic data. The most widely used linguistics resources are *corpora*. A monolingual corpus is simply a text, i.e., a sequence of sentences. However, we will continue to say corpus when we mean a resource, to distinguish it from other types of text, such as input text to be translated, translation output, test and development sets, etc. Monolingual corpora are widely used to train n-gram language models.

A *parallel corpus* is a sequence of m-tuples $(s_1, \ldots, s_\ell)$ of length ℓ. It is understood that for every $s_i = (a_1, \ldots, a_m)$, the sentences $a_1, \ldots, a_m$, are related, i.e., translations of each other, so $a_1 \in \mathbf{L}_1, \ldots, a_m \in \mathbf{L}_m$, for $\mathbf{L}_1, \ldots, \mathbf{L}_m \in \mathcal{L}$. The most commonly used type of parallel corpus, and the only one used in the remainder of this dissertation, is a bilingual parallel corpus ($m = 2$). Parallel corpora are widely used to train translation models.

In the experiments in Section 6, the popular EuroParl (Koehn, 2005) parallel corpus is used. It was extracted from parliament proceedings of the European parliament translated into 21 European languages[5]. The proceedings were stripped of formatting, timestamps and similar data, and then preprocessed to determine sentence boundaries and to align them across languages.

Sentence alignment is not a trivial problem. While it is rather easy to split a text into sentences, in a parallel corpus there might not be a 1-to-1 mapping, as human translators sometimes sacrifice faithfulness for fluency. Heuristics are needed to determine the most likely correspondence. For Europarl, the algorithm of Gale and Church (1993) was used, which matches sentences by length, merging them if needed. In the context of this thesis, we will neglect the problem that a sentence in one language might map to two sentences in the other language, or more complicated cases. We will always assume a 1-to-1-mapping.

Let us now discuss the alphabet of a natural language. Consider a sentence like the following:

```
The cat sat on the mat.
```

Superficially, this looks like a string of letters, whitespace and punctuation symbols, and indeed this is how it is represented. For simplicity, we will assume that our raw textual data is stored as Unicode strings. Let $\mho$ be the alphabet of all Unicode® symbols, that is letters in various scripts, whitespace, punctuation,

[5] Romanic (French, Italian, Spanish, Portuguese, Romanian), Germanic (English, Dutch, German, Danish, Swedish), Slavic (Bulgarian, Czech, Polish, Slovak, Slovene), Finno-Ugric (Finnish, Hungarian, Estonian), Baltic (Latvian, Lithuanian), and Greek. The corpus is available at http://www.statmt.org/europarl/.

numbers, and other symbols as laid out by the Unicode® standard.[6] Our text is therefore a string $t \in \mathcal{U}^*$:

$$t = \texttt{The␣cat␣sat␣on␣the␣mat.}$$

First, let us transform this string into lowercase. We define the function $\mathsf{lower} : \mathcal{U} \to \mathcal{U}$ such that it maps every uppercase letter to its lowercase version and every other symbol on itself. We can extend lower as usual to operate on strings. Then we have:

$$\mathsf{lower}(t) = \texttt{the␣cat␣sat␣on␣the␣mat.}$$

However, this is not what we intend when we write sentences in natural language. Whitespace enables us to see that `cat` and `sat` are to be treated as words, i.e., symbols of another alphabet E of words of the English language, punctuation symbols, numbers etc. Whitespace, however, should not be a part of this alphabet. We should thus represent this sentence as:

$$\underbrace{\texttt{the}}_{\texttt{␣}} \cdot \underbrace{\texttt{cat}}_{\texttt{␣}} \cdot \underbrace{\texttt{sat}}_{\texttt{␣}} \cdot \underbrace{\texttt{on}}_{\texttt{␣}} \cdot \underbrace{\texttt{the}}_{\texttt{␣}} \cdot \underbrace{\texttt{mat}}_{\texttt{␣}} \cdot \texttt{.}$$

To keep the presentation general and language-independent, we assume for every language $\mathbf{L}$ a subset $L^{(0)} \subseteq \mathcal{U}^*$ of Unicode strings that represent its words, called the *lexicon*. We assume that $L^{(0)} \subseteq L$ is the nullary part of a ranked alphabet L that also contains non-nullary symbols called *tags*, defining the internal structure of sentences. For instance, we write E for the set of words and tags of English and $E^{(0)}$ for the set of words of English (or, for historical reasons, the target language of machine translation). The exact nature of these alphabets can be debated. It would not be unreasonable to represent natural language as strings of morphemes instead of words, but this is a minor concern from a theoretical point of view.

Tokenization is the task of transforming text in a language $\mathbf{L}$, written as a string over $\mathcal{U}$ into a string over $L^{(0)}$, the lexicon of $\mathbf{L}$, for further processing. In other words, a tokenizer is an $\mathcal{L}$-indexed family $(\mathsf{tokenize}_{\mathbf{L}} \mid \mathbf{L} \in \mathcal{L})$ of functions $\mathsf{tokenize}_{\mathbf{L}} : \mathcal{U}^* \to \left(L^{(0)}\right)^*$. Note that we make no restriction on the mapping. In real-world scenarios, we will usually have a very close resemblancy of a sentence t in a language $\mathbf{L}$ and the concatenation of $\mathsf{tokenize}_{\mathbf{L}}(t)$. This, however, will be considered an implementation detail. For instance, there is no reason a tokenizer could not have a built-in transliteration component, e.g., mapping Cyrillic or Arabic to Latin text representations. A tokenizer may also perform other tasks such as normalization of punctuation symbols.

A closely related problem is *segmentation* for languages that do not usually put whitespace between words, such as Chinese. We treat this as a special case of tokenization, where word boundaries must be guessed. In order to keep the presentation coherent, we will assume that $\mathsf{tokenize}_{\mathbf{L}}$ is a weighted regular string relation (also called rational string series, Droste and Kuich (2009)).

[6] Unicode is a registered trademark of Unicode, Inc. in the United States and other countries. "The Unicode Standard is a character coding system designed to support the worldwide interchange, processing, and display of the written texts of the diverse languages and technical disciplines of the modern world. In addition, it supports classical and historical texts of many written languages." (http://unicode.org/standard/standard.html)

3.2.1 Morphology and morphosyntax

Morphology is the study of the internal structure of words, and the relations between the units that words are composed of. These units are commonly called *morphemes*, and a word can be composed of one morpheme, or a string of morphemes. For instance, the word *"cat"* is also a morpheme, while *cats* is composed of the morphemes *"cat"* and *"-s"*, which indicates plural. In this case, *"cat"* is a *free* morpheme, while *"-s"* is a bound morpheme because it cannot occur on its own. Morphemes are generally acknowledged to be the smallest units of syntax, i.e., the smallest units of language that have grammatical function or carry meaning. For an introduction to morphology, see Aronoff and Fudeman (2005).

Words in natural languages are traditionally grouped into categories called *parts of speech*. For instance, *"awesome"* is an adjective, while *"in"* is a preposition. The process of assigning words to these categories is called part-of-speech tagging. We adopt the view that a part-of-speech tagger maps the tokenized input to a simplified alphabet, e.g., the Penn treebank tagset[7] or the STTS tagset[8]. However, a word is not always the same part-of-speech. For instance, *"rock"* can be either a noun or a verb.

Part-of-speech tagging can be implemented using weighted finite-state string transducers (Knight and May, 2009). Thus, they can be easily integrated with other finite-state or context-free tools, e.g., a parser that assumes part-of-speech tagged input. We will skip a detailed description of part-of-speech tagging and assume it is part of morphological preprocessing or parsing (see the following section). The part-of-speech tags will then be treated as "pre-frontier", unary nodes of the parse tree.

Morphosyntax is the branch of morphology that is concerned with the interplay of morphology and syntax. For instance, the subject "*cats*" requires a plural verb since English exhibits agreement in number of subject and verb. Languages have varying degrees of morphosyntax. For instance, in German determiners have to agree in number, gender and case with nouns (see Figure 10). A similar phenomenon cannot be found in English.

Morphological preprocessing is relevant because it allows to infer more general rules than just on the surface. For languages with many bound morphemes, it allows to infer translation rules for individual morphemes, rather than strings of morphemes. Morphological preprocessing is often implemented using finite-state string transducers, i.e., as a weighted regular string relation. This makes a morphology component easy to integrate in our finite-state workflow. Again, we will skip a detailed description of morphology and assume it is part of parsing. We assume that morphological information will be encoded in the "pre-frontier" nodes of the parse tree, by using a more fine-grained tagset.

[7] The Penn treebank tagset was used in the syntactic annotations of the Penn treebank (Marcus et al., 1993) on which the Berkeley parser and Egret were trained (see Section 3.2.2).

[8] The STTS tagset was used in the syntactic annotations of the Tiger treebank (Brants et al., 2004) on which BitPar was trained (see Section 3.2.2).

3.2.2 Syntactic parsing

The basic unit of a parallel corpus is a sentence pair[9]. Therefore, we end our description of linguistic techniques at the sentence level. Parsing is the subdiscipline of computational linguistics that deals with assigning a structure to every sentence of a given natural language.

Usually, parsing is preceded by tokenization (and possibly part-of-speech tagging and/or morphological preprocessing). Therefore, we define a *parser* for a language $\mathbf{L}$ to be a function that maps a string $s \in \mathbf{L}$ to a *derivation tree*. This is a very broad definition, and we make a distinction between the derivation tree (an abstract tree encoding the derivation process) and the *derived tree*, or *parse tree*, which is obtained from the derivation tree by a mapping.[10]

There is no universal consensus on which formalism is best suited to describe natural language. The research on the structure of natural language goes as far back as Pāṇini, who formalized the rules of Vedic Sanskrit (around 500 BCE)[11]. However, early efforts disregarded the complexity of human language, i.e., how powerful the underlying formalism needs to be. The first rigorous treatment of the complexity of natural language was published by Chomsky (1957), in which he proves that some phenomena exhibited by English and other languages are not regular languages.

This has lead to the widely accepted notion that context-free languages are a natural fit, and a large number of formalization attempts have been made, as well as many implementations. In particular, in the case of context-free parsing, the derivation tree is the derived tree, and just called a parse tree. We call subtrees of a parse tree *constituents* and assume a natural equivalence relation $t \equiv t'$ if and only if $t(\varepsilon) = t'(\varepsilon)$. This way, the ranked alphabet of every language defines natural equivalence classes of constituents called *categories*, including part-of-speech tags. For instance, we call all subtrees rooted in an NP symbol *"noun phrases"* and assume that they are (to a certain degree) exchangeable, meaning that if $u[t]_w$ is well-formed, then $u[t']_w$ is, and vice versa for any tree u and position $w \in \mathsf{pos}(u)$. The equivalence relation of categories can be made more fine-grained by adding morphological annotations and subcategorization information, i.e., information about how a constituent may combine with other constituents. Figure 10 shows a context-free parse tree generated by BitPar[12] (Schmid, 2004, 2006) that sports rich morphological annotations for case, number and gender, as well as subcategorization information.

The equivalence relation $\equiv$ naturally leads to a formulation of parsing using tree automata. Formally, we will assume that for a given language $\mathbf{E}$, there exists a ranked alphabet E such that the input is a tokenized string over $E^{(0)}$. This

[9] This assumption is over-simplifying because it disregards supra-sentential phenomena such as coreference. Sentences cannot usually be properly understood or translated without context.

[10] For more on mappings between derivation trees and derived trees, see Section 4.2, where a range of (synchronous) grammar formalisms is discussed.

[11] Pāṇini, *Aṣṭādhyāyī*.

[12] Courtesy of Helmut Schmid, freely available for research and education from http://www.cis.uni-muenchen.de/~schmid/tools/BitPar/.

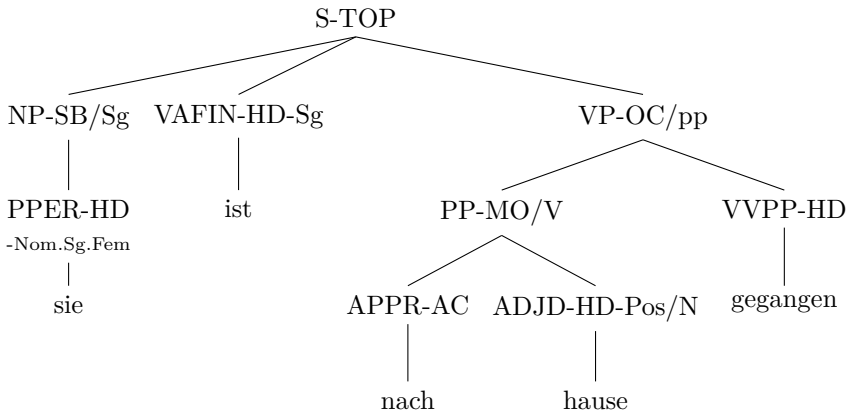


Figure 10. Parse tree with rich morphological annotations

entails that if any part-of-speech tagging is to be modeled, this will implicitly be part of the parser. In the tree formulation of the Myhill-Nerode theorem, for a language $L \subseteq T_E$, and any two trees $t, u \in T_E$, we have $t \equiv u$ if and only if $c[t]_w \in L \iff c[u]_w \in L$ for any $c \in T_E$ and $w \in \mathsf{pos}(c)$. This enables us to represent categories in the stateset of a tree automaton.

A parser for $\mathbf{E}$ is then defined by a weighted regular tree language $P : T_E \to [0, 1]$, and parsing of a sentence $\mathbf{e}$ is the task of computing a parse forest

$$P\langle\mathbf{e}\rangle = \mathsf{yield}^{-1}(\mathbf{e}) \cap P\,. \tag{11}$$

A notable exponent of this approach is the Berkeley Parser[13], which employs a technique called state splitting (Petrov et al., 2006; Petrov and Klein, 2007) that makes it essentially a weighted regular tree language. The parse tree can be obtained from the derivation tree by a simple delabeling (i.e., a function mapping each symbol of a ranked alphabet to a symbol of the same rank in another ranked alphabet) removing the state annotations. Unfortunately, the Berkeley Parser only computes the highest-scoring tree(s), not the full parse forest. A re-implementation of this system is Egret[14], which can also output parse forests.

Recent research suggests that some constructions in languages like Dutch and Swiss German cannot be modelled by context-free languages either. Typically, these counterexamples involve crossing dependencies between substrings. The claim proceeds by arguing that lexical material that is generated in different steps of the derivation should be independent (hence, *context-free*), and therefore context-free grammars cannot account for certain structures (Shieber, 1985). Some noteworthy non-context-free formalisms are dependency grammars (which

[13] Available from https://github.com/slavpetrov/berkeleyparser.

[14] Courtesy of Hui Zhang, available under the Apache 2.0 and LGPL 3.0 open source licences from https://sites.google.com/site/zhangh1982/egret.

construct dependency trees that may not be trees in the way we defined trees, since they may have crossing edges), and tree-adjoining grammars (TAG), which are more powerful than context-free grammars. Both have successfully been used in parsing and machine translation. Synchronous TAG will be discussed in Section 4.2.

In this dissertation, while we acknowledge that context-free string structure is a crude approximation of natural language capacity, we will work with context-free parsers and the resulting structures only. This allows for some simplifying assumptions. The techniques described in this dissertation however might be also applicable to more powerful formalisms as long as the set of derivation trees is a regular tree language. We will revisit this assumption in Section 4.2.

3.3 Translation as decoding

Machine translation (MT) approaches can be roughly classified along two axes: statistical versus hand-crafted, and by how much linguistic information they incorporate in the translation process. The latter can be captured in the illustration called *Vauquois triangle* (Vauquois, 1968) in Figure 11. Translation of an input sentence (lower left) can proceed on a straight line to the output sentence (lower right) on the string level. This approach is taken by word- or phrase-based approaches. On the other hand, approaches that attempt to transform the input into a language-independent representation are called *Interlingua* approaches.

There is some middle ground, though. If the input sentence is parsed first and then mapped to an output string, the approach is called *tree-to-string* translation. Similarly, mapping the input string to an output tree (and then reading off the yield) is called *string-to-tree* translation. Parsing the input sentence, and mapping the input parse tree to an output tree by a syntactic transfer is called *tree-to-tree* translation. In this dissertation, we will not cover any representation beyond syntactic parse trees, in particular no semantic representation.

Our principal approach is syntax-based, with an intermediate representation. This intermediate, language-independent representation will be a regular tree language, and we use a source and target mapping to obtain the input and output trees (whose yields will be the input and output string). Together, this will be formalized as a bimorphism. In the next section, we will introduce statistical approaches to machine translation, culminating in our bimorphism machine translation theory.

The first MT systems consisted of hand-crafted sets of rules. With the rise of computing power in mind, and the successful application of cryptography in World War II, the idea of *translation as decoding* was developed.

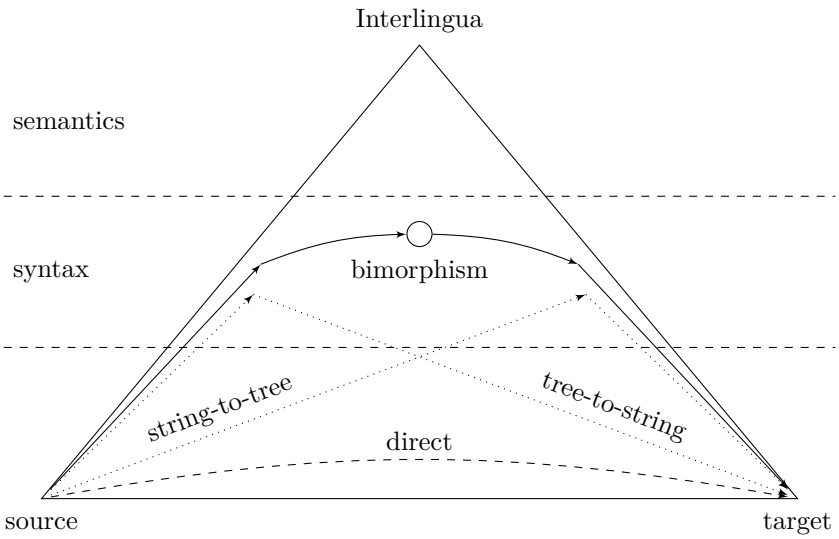


Figure 11. The Vauquois triangle

> When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."
>
> —Warren Weaver, in a letter to Norbert Wiener (1947)

This quote by one of the pioneers of statistical machine translation is remarkable. It re-states the problem of translation as a decoding problem, implying that:

1. There is a source text written in English;
2. This source text has been encrypted and is Russian on the surface;
3. The underlying English source can be recovered by reversing the encryption, i.e., breaking the cypher.

In particular, just like a cypher can be cracked if one has access to source and target surface strings by studying the patterns, the idea was put forward that translation could be automatically learned from large quantities of bilingual data. However, it took almost half a century before these ideas were picked up again.

3.3.1 Noisy channel model

The foundations of statistical machine translation (SMT) were established by an IBM research group (Brown et al., 1990, 1993). The authors present successively refined models that are therefore colloquially known as *IBM Models 1, 2, 3, 4 and 5*. We will present these models here not only for the historic perspective,

but also because many ideas are still in use or have been applied to other, more sophisticated models. In particular, the IBM models revisit the ideas of Warren Weaver and Claude Shannon. The basic assumptions are:

1. that every sentence in one language is a possible translation of any sentence in the other;
2. that every pair of sentences $(\mathbf{f}, \mathbf{e})$ can be assigned a conditional probability $\Pr(\mathbf{e}|\mathbf{f})$ to be interpreted as the probability that a translator, when given the source sentence $\mathbf{f}$, produces the target sentence $\mathbf{e}$.[15]

This process is formalized in the *noisy channel model* (Shannon, 1948) (also called *source-channel model*, Och and Ney (2002)). The underlying assumption is that the text was initially written in, say, English (we will denote the set of English sentences by $\mathbf{E}$), but distorted by transmission through a noisy channel, i.e., translated into French (we will denote the set of French sentences by $\mathbf{F}$). In this way, we can only observe the distorted message $\mathbf{f} \in \mathbf{F}$, the French text. The task is now to recover the most probable English text $\mathbf{e} \in \mathbf{E}$, i.e., to maximize the probability $\Pr(\mathbf{e}|\mathbf{f})$, of the underlying event $\mathbf{e}$ given the observation $\mathbf{f}$.

We can now formulate the translation problem by assuming a probability space $(\Omega, 2^\Omega, \Pr)$ with $\Omega = \mathbf{F} \times \mathbf{E}$, and random variables $S_\mathbf{F} : \Omega \to \mathbf{F}$ and $S_\mathbf{E} : \Omega \to \mathbf{E}$, defined by $S_\mathbf{F}(\mathbf{f}, \mathbf{e}) = \mathbf{f}$ and $S_\mathbf{E}(\mathbf{f}, \mathbf{e}) = \mathbf{e}$. For convenience, we will write $\Pr(\mathbf{f})$ for $\Pr(S_\mathbf{F} = \mathbf{f})$ and $\Pr(\mathbf{e})$ for $\Pr(S_\mathbf{E} = \mathbf{e})$.

In order to stay consistent with existing literature, we define the arg max operation as follows: Let S be a set. We define $\arg\max_S : \mathbb{R}^S \to S$ such that $\arg\max_S(f) \in f^{-1}(\max f(S))$, and undefined if $\max f(S) = \emptyset$. The specific implementation (i.e., which member of S to choose in the case of a tie) does not matter. We usually write $\arg\max_{s \in S} f(s)$ instead of $\arg\max_S(f)$, and implicitly assume that it is defined.

We can rewrite the search for the best translation of $\mathbf{f}$, using Bayes' rule, as follows:

$$\begin{aligned} \hat{\mathbf{e}} &= \arg\max_{\mathbf{e} \in \mathbf{E}} \Pr(\mathbf{e}|\mathbf{f}) \\ &= \arg\max_{\mathbf{e} \in \mathbf{E}} \frac{\Pr(\mathbf{f}|\mathbf{e}) \cdot \Pr(\mathbf{e})}{\Pr(\mathbf{f})} \\ &= \arg\max_{\mathbf{e} \in \mathbf{E}} \left(\Pr(\mathbf{f}|\mathbf{e}) \cdot \Pr(\mathbf{e})\right) . \end{aligned} \tag{12}$$

The last rewriting step is justified since $\Pr(\mathbf{f})$ is constant for all $\mathbf{e}$. The task of *decoding* is then to find the target language sentence $\hat{\mathbf{e}}$ that maximizes $\Pr(\mathbf{e}) \cdot \Pr(\mathbf{f}|\mathbf{e})$.

Decomposing the problem in this way allows us to model the noisy channel using two distinct submodels, a model for $\Pr(\mathbf{f}|\mathbf{e})$ which we call the *translation model*, and a model for $\Pr(\mathbf{e})$ which we call the *language model*. Figure 12 gives us an intuitive illustration of this decomposition. The sender generates $\mathbf{e}$ with

[15] We follow the traditional custom of using $\mathbf{f}$ for `French` or `foreign` to denote the source or input language (i.e., the language from which we want to translate), and $\mathbf{e}$ for `English` to denote the target or output language (i.e., the language into which we want to translate).

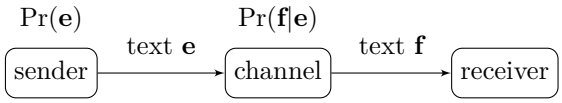


Figure 12. Noisy channel model

probability Pr(**e**). Then **e** is distorted by the noisy channel, and the receiver will observe **f** with probability Pr(**f**|**e**).[16]

Brown et al. (1990) then proceed to explain how probabilities for Pr(**e**) and Pr(**f**|**e**) can be estimated from a bilingual corpus. Note that the translation model now models the probability of the source **f** given the target **e**. Language models are usually implemented using n-gram models. Besides machine translation, they are (and were much earlier) used in speech recognition. We will describe n-gram language models in Section 5.3.

For the translation model, a broad spectrum of different formalisms has been put forward. The IBM models are *word-based*, i.e., the probability of Pr(**f**|**e**) is estimated by relating single words (*alignments*), including parameters for words that do not have a correspondence (that are left *unaligned*). These models are rather crude, but they remain in wide use as a way to generate *word alignments* that can be used to bootstrap more sophisticated models.

IBM model 1 can be characterized as follows: For a sentence pair $(\mathbf{e}, \mathbf{f})$ with $\mathbf{e} = (\mathbf{e}_1, \ldots, \mathbf{e}_m)$ and $\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_n)$, the lexical translation scores $t(\mathbf{f}_i|\mathbf{e}_i)$ can be obtained from the alignments between **f** and **e** (a mapping $a : [n] \to [m] \cup \{0\}$). However, the alignments are also unknown. We could obtain the most probable alignment for a given sentence pair from the lexical translation scores, however, these are unknown. To overcome this "chicken and egg" problem, EM (expectation-maximization) training can be used. We will revisit EM in our experiment chapter, Section 5.4.3.

Then the translation probability of **f** given **e** with word alignment a is modelled as:

$$\Pr(\mathbf{f}, a|\mathbf{e}) = \frac{\varepsilon}{(|\mathbf{e}|+1)^n} \cdot \prod_{i=1}^{n} t(\mathbf{f}_i|\mathbf{e}_{a(i)}) \,. \tag{13}$$

Higher-order IBM models successively refine Model 1. Model 2 adds an absolute distortion model $d(j|i, |\mathbf{f}|, |\mathbf{e}|)$, i.e., a probability that an English word at position j translates to a foreign word at position i.

Model 3 adds a fertility model $n(\phi|\mathbf{e}_j)$ to model the probability that the English word $\mathbf{e}_j$ translates into ϕ foreign words. Model 4 remedies the sparse and unreliable absolute distortion model by adding a relative reordering model, while Model 5 fixes a deficiency problem. (A model is deficient if it assigns non-zero probability to impossible outcomes. Models 3 and 4 are deficient because

[16] Compared to other applications of the noisy channel model (e.g., signal processing, speech recognition, error correction), where the "noise" is rather random and the part of the model that we would rather not want to model, in machine translation the noise is the interesting part.

they can reorder all words to the same position, which should be disallowed.)

The usual training pipeline is to start with Model 1. After a few EM iterations, good lexical translation scores are obtained. These are transferred to Model 2, which yields a good absolute reordering model after a few iterations. Proceeding in this way, we will finally obtain a trained Model 5.

3.3.2 Log-linear models

Recall the rewritten Noisy channel model from Equation (12):

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}\in\mathbf{E}} \left(\Pr(\mathbf{f}|\mathbf{e}) \cdot \Pr(\mathbf{e})\right) .$$

The optimal translation $\hat{\mathbf{e}}$ is the translation that has the highest probability, decomposed into $\Pr(\mathbf{f}|\mathbf{e})$ for the translation model and $\Pr(\mathbf{e})$ for the language model.

Unfortunately, while this is mathematically well-motivated, there are some problems with this approach, as identified by Och and Ney (2002). In particular, the noisy channel model cannot easily accomodate additional sources of information, and it cannot explain why replacing $\Pr(\mathbf{f}|\mathbf{e})$ by $\Pr(\mathbf{e}|\mathbf{f})$ or a different weighting of language model and translation model can give comparable or better results.

Motivated by these objections, Och and Ney (2002) propose to model $\Pr(\mathbf{e}|\mathbf{f})$ directly in the *maximum entropy* framework. In this framework, the translation model is usually further decomposed into a number of feature functions (or *features* for short) that score different aspects of the translation. We say "score" because now we are deviating from probability theory. From now on, the scores will not necessarily be probabilities. The hopeful assumption is that the scores "behave like" the actual probabilities in a way that translations with higher probabilities receive a better score; and that the translation with the highest score will be the translation with the highest probability. Functions and scores will not be mathematically well-founded, but motivated by engineering experience. In this sense, everything that follows will still be inspired by the noisy channel model, but the analogy ends here.

Features may be represented as a family of m functions

$$(\phi_i : \mathbf{F} \times \mathbf{E} \to \mathbb{R} \mid i \in [m]) .$$

We associate with each feature function a parameter λ_i to model the relative importance of this feature. Together, these parameters may be represented by the parameter vector $\lambda = (\lambda_1, \ldots, \lambda_m)$. Thus, the equation for the maximum entropy model becomes:

$$\begin{aligned} \Pr(\mathbf{e}|\mathbf{f}) &\propto \prod_{i=1}^{m} \phi_i(\mathbf{f}, \mathbf{e})^{\lambda_i} \\ &= \exp\left(\sum_{i=1}^{m} \lambda_i \cdot \log \phi_i(\mathbf{f}, \mathbf{e})\right) . \end{aligned} \tag{14}$$

Note how we are "misusing" the symbol $\propto$ to denote the rather complex relation "behaves like".

The logarithmic transformation justifies the name *log-linear model*. Since 'arg max' can be defined so that it is not affected by logarithmic transformation, we may in fact rewrite Equation (12) into:

$$\begin{aligned}\hat{\mathbf{e}} &= \arg\max_{\mathbf{e}\in\mathbf{E}} \Pr(\mathbf{e}|\mathbf{f}) \\ &\propto \arg\max_{\mathbf{e}\in\mathbf{E}} \sum_{i=1}^{m} \lambda_i \cdot \log \phi_i(\mathbf{f},\mathbf{e})\,.\end{aligned} \tag{15}$$

While the logarithmic transformation is preferable for numerical stability (and applied in the toolkit discussed in the implementation section of this dissertation), we will not use it in the theoretical discussion. We prefer the "original" formulation:

$$\hat{\mathbf{e}} \propto \arg\max_{\mathbf{e}\in\mathbf{E}} \prod_{i=1}^{m} \phi_i(\mathbf{f},\mathbf{e})^{\lambda_i}\,. \tag{16}$$

This choice is personal preference only, since the real numbers are arguably more intuitive than logarithms. All of the following is equally true after a logarithmic transformation.

In this symmetric formulation, features have access to both source and target even if they use only one of them. Source features are typically scores on the parse trees of the input sentence, translation features are scores on the translation rules, and target features are typically language model scores, sentence length, and similar features. For instance, a typical phrase-based MT system (Koehn et al., 2003) will consist of a phrase translation model, a reordering model to account for the variation in syntax between the languages, and a target language model. Note that the noisy channel model is a special case of the maximum entropy model if the two features $\phi_1(\mathbf{f},\mathbf{e}) = \Pr(\mathbf{e})$ and $\phi_2(\mathbf{f},\mathbf{e}) = \Pr(\mathbf{f}|\mathbf{e})$, and $\lambda_1 = \lambda_2 = 0.5$ are used. To find good values for the parameter vector λ, they are usually optimized with regard to an automatic evaluation metric such as BLEU (cf. Section 5.4.1).

Phrase-based models (Koehn et al., 2003) extend word-based models by allowing substrings of arbitrary (but usually bounded) length to be treated as *phrases* and assigning phrase translation probabilities. Thus, the basic translation unit is a phrase pair, and a phrase decomposition probability distribution is added to the model. The table of phrase pairs is obtained from alignments obtained by training the IBM models. However, these alignments are mappings, i.e., one English word is always mapped to at most one foreign word. For many-to-many alignments, the pipeline is usually run in both directions, and the resulting alignments are then merged. A notable exponent of the phrase-based approach is the alignment template system (Och and Ney, 2004). The alignment template system has also been implemented using a cascade of weighted finite-state string transducers (Kumar and Byrne, 2003), proving the value of automata and formal language theory for application use.

Chapter 4

Theory

Contents

This chapter will develop a theory of bimorphism machine translation. We move to hierarchical models and replace the noisy channel model and the log-linear model by a symmetric generative process on the basis of bimorphisms, motivated by inference rules. Towards the end of the chapter, a variety of synchronous grammar formalisms are discussed, and our choice of multi bottom-up tree transducers as the default formalism in the rest of this dissertation is backed up.

4.1 Bimorphism machine translation

The models discussed so far are centered around the notion that sentences are essentially sequences of words or phrases. However, it is widely accepted that words can be grouped into phrases, phrases can be grouped into bigger phrases, which in turn can be grouped in a hierarchical fashion until we arrive at a "phrase" spanning the entire sentence. In this way, the internal structure of a sentence can be represented by a parse tree (see Section 3.2.2). We will distinguish between "pseudo-syntax based" models that introduce a tree structure that does not necessarily match linguistic analysis, and "syntax-based models" whose input and output trees coincide with parse trees.

Most hierarchical machine translation systems implement a *synchronous grammar* formalism for the translation model. Synchronous grammars capture the intuition of a generative process that derives both the source and the target sentence simultaneously, using the same set of production rules. Every rule thus contains a source and a target part, as well as some kind of relation between source and target part.

When hierarchical phrases are modelled without incorporating linguistic information, this is usually a string-based synchronous grammar. A common choice is *synchronous context-free grammar* (SCFG, Satta and Peserico (2005)), but restricted forms of SCFG are also widely used. HIERO (Chiang, 2007) uses only one equivalence class X (and an additional equivalence class S for concatenation), while *inversion transduction grammar* (Wu, 1997) and *inversion-invariant transduction grammar* (Wu, 1995) put restrictions on the form of the production rules.

4.1.1 From inference rules to bimorphisms

We will formalize the intuitive notion of inference rules put forward in the introduction. To this end, we will decouple the language-dependent realizations from the mechanism that generates well-formed structures of inference rules. We will use a weighted regular tree language to model the way in which inference rules can be combined, and we will use functions that operate on derivation trees to map into arbitrary structures (in particular, trees) instead of only strings.

To this end, let Σ be a ranked alphabet, and let A and B be sets. A family of mappings $(h_k : \Sigma^{(k)} \to A^{(A^k)} \mid k \in \mathbb{N})$ determines a *homomorphism* $h : T_\Sigma \to A$ by

$$h(t) = h_k(\sigma)(h(t_1), \ldots, h(t_k)) \tag{17}$$

for every $t = \sigma^{(k)}(t_1, \ldots, t_k) \in T_\Sigma$. Intuitively, h_k assigns a k-ary operation over A to every $\sigma \in \Sigma^{(k)}$.

A *weighted bimorphism* is a weighted relation of the form $f^{-1}\,;L\,;g$ where $f : T_\Sigma \to A$ and $g : T_\Sigma \to B$ are homomorphisms, and $L \in \mathsf{REC}(\Sigma)$. Note that L is weighted, but f and g are unweighted. We may also write (f, L, g), especially if we want to highlight the components instead of the resulting relation. For given classes X, Y of homomorphisms on T_Σ, we write

$$\mathcal{B}(X, Y) = \{f^{-1}\,;L\,;g \mid L \in \mathsf{REC}(\Sigma), f \in X, g \in Y\}\,. \tag{18}$$

Bimorphisms were first defined by Arnold and Dauchet (1982). In particular, we will now consider the case where Σ is a ranked alphabet of inference rules, L is a weighted language that scores these inference rules and their combinations, and $f : T_\Sigma \to T_F$ or $f : T_\Sigma \to \mathbf{F}$, and $g : T_\Sigma \to T_E$ or $g : T_\Sigma \to \mathbf{E}$. In this way, we can reason about translation spaces $T_F \times T_E$ of trees, or mixed models for tree-to-string $(T_F \times \mathbf{E})$ or string-to-tree $(\mathbf{F} \times T_E)$ translation.

Example 7 (*Bimorphism for an SCFG.*)
In this example (adapted from Chiang (2006)), we specify an SCFG between

English (**E**) and Japanese that was transliterated[17] (**J**) as a bimorphism $\widehat{=} = (\mathsf{in}, L_{\mathcal{G}}, \mathsf{out})$, where $\mathcal{G} = (Q, P, I)$ is a (trivially) weighted tree automaton over the ranked alphabet of inference rules $\Sigma = \{1^{(2)}, 2^{(2)}, 3^{(0)}, 4^{(0)}, 5^{(0)}\}$ with

$$\begin{aligned} Q &= \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}, \mathrm{V}\} \\ I(q) &= \begin{cases} 1 & \text{if } q = \mathrm{S} \\ 0 & \text{otherwise} \end{cases} \\ P &= \{(\mathrm{S}, 1(\mathrm{NP}, \mathrm{VP}), 1), (\mathrm{VP}, 2(\mathrm{V}, \mathrm{NP}), 1), \\ &\quad (\mathrm{NP}, 3(), 1), (\mathrm{NP}, 4(), 1), (\mathrm{V}, 5(), 1)\} \end{aligned}$$

In addition to $\widehat{=}$, this implicitly specifies relations $\widehat{=}_{\mathrm{V}}$, $\widehat{=}_{\mathrm{VP}}$, $\widehat{=}_{\mathrm{NP}}$, and $\widehat{=}_{\mathrm{S}} = \widehat{=}$ because of the states in Q. Let $\mathsf{in} : T_\Sigma \to \mathbf{E}$ and $\mathsf{out} : T_\Sigma \to \mathbf{J}$ be defined by:

$$\begin{aligned} \mathsf{in}(1)(x_1, x_2) &= x_1 x_2 & \mathsf{out}(1)(x_1, x_2) &= x_1 x_2 \\ \mathsf{in}(2)(x_1, x_2) &= x_1 x_2 & \mathsf{out}(2)(x_1, x_2) &= x_2 x_1 \\ \mathsf{in}(3)() &= \text{i} & \mathsf{out}(3)() &= \text{watashi wa} \\ \mathsf{in}(4)() &= \text{the box} & \mathsf{out}(4)() &= \text{hako wo} \\ \mathsf{in}(5)() &= \text{open} & \mathsf{out}(5)() &= \text{akemasu} \end{aligned}$$

Note how the word order differs in rule 2. A synchronous application of in and out to the derivation tree $t = 1(3, 2(5, 4))$ yields the following string pair in $\widehat{=}$:

$$\dfrac{\dfrac{}{\text{i} \mathrel{\widehat{=}_{\mathrm{NP}}} \text{watashi wa}}\ (3) \quad \dfrac{\dfrac{}{\text{open} \mathrel{\widehat{=}_{\mathrm{V}}} \text{akemasu}}\ (5) \quad \dfrac{}{\text{the box} \mathrel{\widehat{=}_{\mathrm{NP}}} \text{hako wo}}\ (4)}{\text{open the box} \mathrel{\widehat{=}_{\mathrm{VP}}} \text{hako wo akemasu}}\ (2)}{\text{i open the box} \mathrel{\widehat{=}} \text{watashi wa hako wo akemasu}}\ (1)$$

Let us introduce an important class of homomorphisms. To this end, let Δ and Σ be ranked alphabets, and let $X_n = \{x_i \mid i \in [n]\}$ be a set of variables for every $n \in \mathbb{N}$. A family f of mappings $\big(f_k : \Delta^{(k)} \to T_\Sigma(X_k)\big)$ determines a homomorphism $h_f : T_\Delta \to T_\Sigma$, called *tree homomorphism*, given by

$$h_f(\delta(t_1, \ldots, t_k)) = \xi(f_k(\Delta)) \tag{19}$$

for every $\delta \in \Delta^{(k)}$, where $\xi : X_k \to T_\Sigma$ is the ground substitution defined by $\xi(x_i) = h_f(t_i)$. If $|\,\mathsf{pos}_{\{x\}}(f_k(\delta))| \leq 1$ for every $x \in X_k$ and $\delta^{(k)} \in \Delta$, then h is *linear*. If $|\,\mathsf{pos}_{\{x\}}(f_k(\delta))| \geq 1$ for every $x \in X_k$ and $\delta^{(k)} \in \Delta$, then h is *nondeleting* (or *complete*). The class of tree homomorphisms will be denoted HOM, and the class of linear and nondeleting tree homomorphisms, which coincides in expressive power with the class $\mathsf{lnh\text{-}BOT}$ of tree transformations computed by linear nondeleting homomorphic bottom-up tree transducers (Engelfriet et al., 2009) will be denoted $\mathsf{ln\text{-}HOM}$.

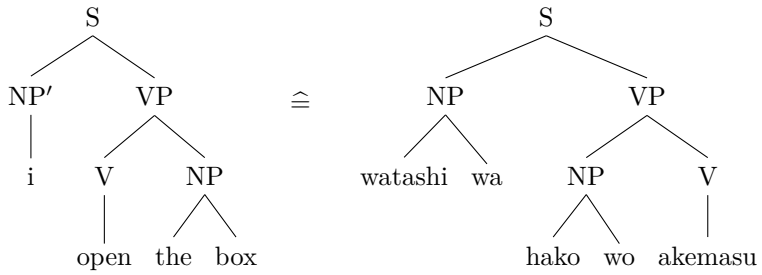
[17] The original spelling of the derived Japanese sentence is: 私は箱を開けます。

Let $h : T_\Delta \to T_\Sigma$ be a tree homomorphism. If L is a weighted regular tree language over Σ, then $h^{-1}(L)$ is a weighted regular tree language over Δ. A *functional relabeling* (or *delabeling*) is a mapping $r : \bigcup_{k\in\mathbb{N}} \Delta^{(k)} \to \Sigma^{(k)}$. It defines a tree homomorphism h_r determined by the family $\big(r_k : \Delta^{(k)} \to T_\Sigma(X_k)\big)$ where $r_k(\delta) = r(\delta)(x_1, \ldots, x_k)$ for every $\delta^{(k)} \in \Delta$. We denote the class of delabelings by REL.

Example 8 (*Tree homomorphisms*)
Recall the bimorphism $(\mathsf{in}, L_\mathcal{G}, \mathsf{out})$ from Example 7. Let us now replace the string concatenation operations by suitable tree-building operations. We define tree homomorphisms $\mathsf{in}' : T_\Delta \to T_E$ and $\mathsf{out}' : T_\Delta \to T_J$:

$$\begin{aligned}
\mathsf{in}'(1)(x_1, x_2) &= \mathrm{S}(x_1, x_2) & \mathsf{out}'(1)(x_1, x_2) &= \mathrm{S}(x_1, x_2)\\
\mathsf{in}'(2)(x_1, x_2) &= \mathrm{VP}(x_1, x_2) & \mathsf{out}'(2)(x_1, x_2) &= \mathrm{VP}(x_2, x_1)\\
\mathsf{in}'(3)() &= \mathrm{NP}'(\mathrm{i}) & \mathsf{out}'(3)() &= \mathrm{NP}(\text{watashi, wa})\\
\mathsf{in}'(4)() &= \mathrm{NP}(\text{the, box}) & \mathsf{out}'(4)() &= \mathrm{NP}(\text{hako, wo})\\
\mathsf{in}'(5)() &= \mathrm{V}(\text{open}) & \mathsf{out}'(5)() &= \mathrm{V}(\text{akemasu})
\end{aligned}$$

Both in' and out' are linear and nondeleting. We obtain the following tree pair $(\mathsf{in}'(t), \mathsf{out}'(t)) \in (\mathsf{in}', L_\mathcal{G}, \mathsf{out}')$:



(Note that we introduced NP' to avoid rank conflicts.)

4.1.2 A generative story

In the previous section, we showed how to create a bimorphism translation model based on intuitive inference rules for translational correspondence. The intuition behind this was that we are using a bimorphism $(\mathsf{in}, L, \mathsf{out})$ with $L : T_\Delta \to [0, 1]$ and homomorphisms $\mathsf{in} : T_\Delta \to T_F$ and $\mathsf{out} : T_\Delta \to T_E$ to model translation between **F** and **E**. We will now make the connection to statistical machine translation again. Let us go back to the original problem of modelling the conditional probability (cf. Equation (12)):

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}\in\mathbf{E}} \Pr(\mathbf{e}|\mathbf{f})\,.$$

We will now introduce trees to the probabilistic model. We still assume that both **f** and **e** are generated simultaneously, but furthermore we assume that **f**

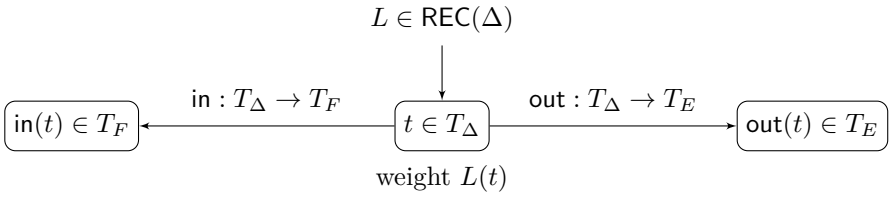


Figure 13. Decoder bimorphism

and $\mathbf{e}$ are only the surface representation of a hidden derivation tree over some ranked alphabet Δ of langugage-independent symbols by means of a bimorphism $\mathcal{D} = (\mathsf{in}, L, \mathsf{out})$. Then, for an observation of a string pair $(\mathbf{f}, \mathbf{e})$, there exists $t \in T_\Delta$ such that $f = \mathsf{in}(t)$ and $e = \mathsf{out}(t)$, and $\mathsf{yield}(f) = \mathbf{f}$ and $\mathsf{yield}(e) = \mathbf{e}$. A graphical representation of this generic bimorphism model is shown in Figure 13.

In the following, we assume a probability space $(\Omega, 2^\Omega, \Pr)$ with $\Omega = T_\Delta$, i.e., we assume that our elementary events are derivation trees. Furthermore, we assume two tree homomorphisms $\mathsf{in} : T_\Delta \to T_F$ and $\mathsf{out} : T_\Delta \to T_E$, and we introduce four random variables $T_\mathbf{F} : \Omega \to T_F, T_\mathbf{E} : \Omega \to T_E, S_\mathbf{F} : \Omega \to \mathbf{F}$ and $S_\mathbf{E} : \Omega \to \mathbf{E}$ such that

$$\begin{aligned} T_\mathbf{F}(t) &= \mathsf{in}(t) & S_\mathbf{F}(t) &= \mathsf{yield}(\mathsf{in}(t)) \\ T_\mathbf{E}(t) &= \mathsf{out}(t) & S_\mathbf{E}(t) &= \mathsf{yield}(\mathsf{out}(t)) \end{aligned}$$

for every $t \in \Omega$. For convenience, we will write $\Pr(f)$ for $\Pr(T_\mathbf{F} = f)$, $\Pr(e)$ for $\Pr(T_\mathbf{E} = e)$, $\Pr(\mathbf{f})$ for $\Pr(S_\mathbf{F} = \mathbf{f})$ and $\Pr(\mathbf{e})$ for $\Pr(S_\mathbf{E} = \mathbf{e})$. Then $\arg\max_{\mathbf{e} \in \mathbf{E}} \Pr(\mathbf{e}|\mathbf{f})$ can be rewritten:

$$\begin{aligned} \hat{\mathbf{e}} &= \arg\max_{\mathbf{e} \in \mathbf{E}} \Pr(\mathbf{e}|\mathbf{f}) \\ &= \arg\max_{\mathbf{e} \in \mathbf{E}} \frac{\Pr(\mathbf{f}, \mathbf{e})}{\Pr(\mathbf{f})} \\ &= \arg\max_{\mathbf{e} \in \mathbf{E}} \Pr(\mathbf{f}, \mathbf{e}) \\ &= \arg\max_{\mathbf{e} \in \mathbf{E}} \sum_{t \in T_\Delta} \Pr(\mathbf{f}, t, \mathbf{e})\,. \end{aligned}$$

We note that $\Pr(\mathbf{f}, t, \mathbf{e}) = 0$ if $\mathsf{yield}(\mathsf{in}(t)) \neq \mathbf{f}$ or $\mathsf{yield}(\mathsf{out}(t)) \neq \mathbf{e}$. Therefore, we can restrict t accordingly, and we can then simplify the joint probability as follows:

$$\begin{aligned} \hat{\mathbf{e}} &= \arg\max_{\mathbf{e} \in \mathbf{E}} \sum_{t \in T_\Delta} \Pr(\mathbf{f}, t, \mathbf{e}) \\ &= \arg\max_{\mathbf{e} \in \mathbf{E}} \sum_{\substack{\mathsf{yield}(\mathsf{in}(t)) = \mathbf{f} \\ \mathsf{yield}(\mathsf{out}(t)) = \mathbf{e}}} \Pr(t)\,. \end{aligned}$$

We will now split $\Pr(t)$ into three parts and introduce model weights. Let $\lambda_1, \lambda_2, \lambda_3 \in [0,1]$ such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Then

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e} \in \mathbf{E}} \sum_{\substack{\mathsf{yield}(\mathsf{in}(t)) = \mathbf{f} \\ \mathsf{yield}(\mathsf{out}(t)) = \mathbf{e}}} \Pr(t)^{\lambda_1} \cdot \Pr(t)^{\lambda_2} \cdot \Pr(t)^{\lambda_3} .$$

Now we will decompose $\Pr(t)$ in two different ways. First, let $\mathsf{in}(t) = f$ and $\mathsf{yield}(\mathsf{in}(t)) = \mathbf{f}$. Note that $T_{\mathbf{F}}^{-1}(f) \subseteq S_{\mathbf{F}}^{-1}(\mathsf{yield}(f))$, and therefore

$$\begin{aligned} \Pr(t) &= \Pr(t, f) \\ &= \Pr(f) \cdot \Pr(t|f) \\ &= \Pr(f, \mathbf{f}) \cdot \Pr(t|f) . \end{aligned}$$

Second, let $\mathsf{out}(t) = e$ and $\mathsf{yield}(\mathsf{out}(t)) = \mathbf{e}$. Since $T_{\mathbf{E}}^{-1}(e) \subseteq S_{\mathbf{E}}^{-1}(\mathsf{yield}(e))$,

$$\begin{aligned} \Pr(t) &= \Pr(t, e) \\ &= \Pr(t|e) \cdot \Pr(e) \\ &= \Pr(t|e) \cdot \Pr(e, \mathbf{e}) \\ &= \Pr(t|e) \cdot \frac{\Pr(e, \mathbf{e})}{\Pr(\mathbf{e})} \cdot \Pr(\mathbf{e}) . \end{aligned}$$

We therefore obtain

$$\begin{aligned} \Pr(t) &= \Pr(t)^{\lambda_1} \cdot \Pr(t)^{\lambda_2} \cdot \Pr(t)^{\lambda_3} \\ &= \Big(\underbrace{\Pr(f, \mathbf{f})}_{\text{parser}} \cdot \underbrace{\Pr(t|f)}_{\text{forward}} \Big)^{\lambda_1} \cdot \underbrace{\Pr(t)}_{\text{symm.}}{}^{\lambda_2} \cdot \left(\underbrace{\Pr(t|e)}_{\text{backward}} \cdot \underbrace{\frac{\Pr(e, \mathbf{e})}{\Pr(\mathbf{e})}}_{\text{synLM}} \cdot \underbrace{\Pr(\mathbf{e})}_{\text{LM}} \right)^{\lambda_3} . \end{aligned} \tag{20}$$

This decomposition allows us to model each of the six factors separately, in the hopeful assumption that individual weaknesses of sub-models will be balanced by other sub-models. Furthermore, the variation of $\lambda_1, \lambda_2, \lambda_3$ allows us to give the models different weight according to how well they approximate the actual probability distributions. Again, this can be modeled as a specific instance of the maximum entropy framework. We choose the following approximations to model the individual factors:

- $\Pr(f, \mathbf{f})$ is the probability of the tree f and the sentence $\mathbf{f}$ occuring, or simply the probability of the tree f. This is modeled by the score that a parser $P_F : T_F \to [0,1]$ assigns to f, assuming that P_F models a probability distribution on trees. We will refer to our approximation by a weighted regular tree language $\mathcal{F}_1$.
- $\Pr(t|f)$ is the probability of derivation t given tree f. This is called *forward translation probability* and can be modeled as an aspect of the generative process by the weighted regular tree language $\mathcal{T}_1$.

- $\Pr(t)$ is the probability of the elementary event t, i.e., the derivation t. It corresponds to the *symmetric translation probability* and can be modeled as part of the generative process. In the bimorphism formulation of the generative process, it is modeled by the weighted regular tree language $\mathcal{T}_2$.

- $\Pr(t|e)$ is the probability of derivation t given tree e. This is called *backward translation probability* and can be modeled as an aspect of the generative process by the weighted regular tree language $\mathcal{T}_3$.

- $\Pr(e, \mathbf{e}) \cdot \Pr(\mathbf{e})^{-1}$ is the probability of the tree e, divided by the probability of the string $\mathbf{e}$, i.e., all trees with yield $\mathbf{e}$. The numerator can again be modeled by a parser $P_E : T_E \to [0, 1]$, while the denominator can be obtained as the language weight of $P_E \cap \mathsf{yield}^{-1}(\mathbf{e})$. Together, this factor may be interpreted as a *syntactic language model*, and is modeled by the weighted regular tree language $\mathcal{E}_1$.

- $\Pr(\mathbf{e})$ is the probability of the string $\mathbf{e}$. An alternative (and arguably more reliable) way to estimate $\Pr(\mathbf{e})$ is to use an n-gram *language model*. We will model it by the weighted regular tree language $\mathcal{E}_2$.

We can therefore write the generative process $\mathcal{D}$ in weighted bimorphism form, i.e.,

$$\mathcal{D} = (\mathsf{in}, L, \mathsf{out})\,, \tag{21}$$

where

$$\begin{aligned} L &= (\mathcal{F}_1 \cdot \mathcal{T}_1)^{\lambda_1} \cdot \mathcal{T}_2^{\lambda_2} \cdot (\mathcal{T}_3 \cdot \mathcal{E}_1 \cdot \mathcal{E}_2)^{\lambda_3} \\ &= \underbrace{\mathcal{F}_1^{\lambda_1}}_{=\mathcal{F}} \cdot \underbrace{\mathcal{T}_1^{\lambda_1} \cdot \mathcal{T}_2^{\lambda_2} \cdot \mathcal{T}_3^{\lambda_3}}_{=\mathcal{T}} \cdot \underbrace{\mathcal{E}_1^{\lambda_3} \cdot \mathcal{E}_2^{\lambda_3}}_{=\mathcal{E}}\,. \end{aligned} \tag{22}$$

The weighted regular tree language $L = \mathcal{F} \cdot \mathcal{T} \cdot \mathcal{E}$ will assign scores to the derivation trees, i.e., $L : T_\Delta \to \mathbb{R}$. Since the derivation trees implicitly contain all the information about the input and output sentence, the sub-models can be evaluated directly on the derivation trees as well. In this way, we can score not only the translation, but also incorporate the language model. Furthermore, we are able to score syntactic annotations on the input language or use scores from a probabilistic parser. In a similar way as explained in Section 3.3.2, we could also deviate and formulate a maximum entropy model instead. See Section 6.2 for an experiment that compares this approach with a maximum entropy model.

The set of all derivation trees for a natural language tree pair $(f, e) \in T_E \times T_F$ is

$$\mathcal{D}(f, e) = \{t \in T_\Delta \mid \mathsf{in}(t) = f \wedge \mathsf{out}(t) = e\}\,.$$

Hence, the best translation tree e for an input tree f is approximated by:

$$\hat{e} \propto \arg\max_e \sum_{t \in \mathcal{D}(f,e)} L(t)\,. \tag{23}$$

However, we have to account for the fact that the input is given as a string, and the output should also be a string, thus:

$$\hat{\mathbf{e}} \propto \arg\max_{\mathbf{e}} \sum_{t \in \mathcal{D}(\mathbf{f},\mathbf{e})} L(t)\,, \tag{24}$$

where $\mathcal{D}(\mathbf{f}, \mathbf{e}) = \{t \in T_\Delta \mid \mathsf{yield}(\mathsf{in}(t)) = \mathbf{f} \wedge \mathsf{yield}(\mathsf{out}(t)) = \mathbf{e}\}$.

Before we return to the question of how to compute $\hat{\mathbf{e}}$, we will discuss what kind of bimorphism, i.e., what kind of tree homomorphisms to use. Our task now is to find a suitable formalism to model the tree transformations that appear in natural language translation. It has been a subject of debate what formalism is best suited. In the following sections, we will review some candidates from the computational and the linguistic angle.

4.2 Synchronous grammar formalisms

Recall that our translation model assumes a bimorphism $\mathcal{D} = (\mathsf{in}, L, \mathsf{out})$ with $L : T_\Delta \to \mathbb{R}$, $\mathsf{in} : T_\Delta \to T_F$ and $\mathsf{out} : T_\Delta \to T_E$. We can elegantly simulate the mechanism of a synchronous grammar by identifying each rule ρ with a ranked symbol in Δ, and the source and target rule components with the homomorphic images $\mathsf{in}(\rho)$ and $\mathsf{out}(\rho)$, respectively.

Let us therefore describe a few common formalisms as bimorphisms. Recall that the power of the relation computed by $\mathcal{D} = (\mathsf{in}, L, \mathsf{out})$ depends on in and out. By restricting the homomorphisms to certain classes, we can define classes of bimorphisms. A broad range of synchronous grammars can be decomposed by placing appropriate restrictions on the homomorphisms. The application of bimorphisms in the context of synchronous grammars is not new; see for instance Shieber (2006, 2014). Fortunately, the bimorphism decomposition saves us the trouble of defining grammar formalisms, tree transducers or similar devices for some types of synchronous grammars; we will now report some results from the literature about relevant formalisms.

First, note that we usually assume $\mathsf{in} : T_\Delta \to T_F$ and $\mathsf{out} : T_\Delta \to T_E$, i.e., both homomorphisms map derivations to trees. In addition to synchronous grammars generating strings (building structure along the derivation but not explicitly deriving trees), many formalisms have been proposed that generate syntax trees on either the input or output side. For instance, a tree-to-string system transforms parsed input sentences into strings in the output language, while a string-to-tree system will take the input and build up a syntax tree in the output language. In order to explain tree-to-string, string-to-tree and string-to-string synchronous grammars, we can also define homomorphisms $\mathsf{in}' : T_\Delta \to \mathbf{F}$ or $\mathsf{out}' : T_\Delta \to \mathbf{E}$, i.e., string-generating homomorphisms. In particular, we can consider the compositions $\mathsf{in}\,;\mathsf{yield}$ and $\mathsf{out}\,;\mathsf{yield}$. The string relation $\mathsf{yield}(R)$ associated with a tree relation R is $\{(\mathsf{yield}(t), \mathsf{yield}(u)) \mid (t, u) \in R\}$. For a bimorphism $B = (h, L, h')$, this means that $\mathsf{yield}(B) = (h\,;\mathsf{yield}, L, h'\,;\mathsf{yield})$. Figure 14 illustrates the relation between bimorphisms and the tree relations and string relations they represent. We say that two formalisms are *strongly*

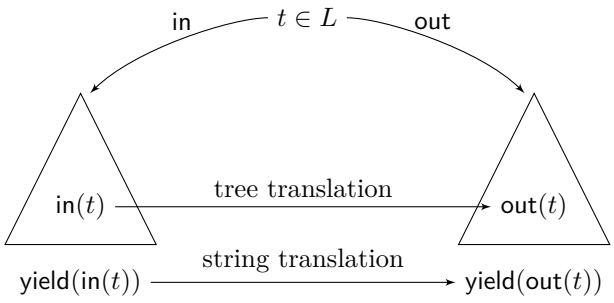


Figure 14. Bimorphism tree and string translation

equivalent if the classes of tree relations they define coincide. Two formalisms are *weakly equivalent* if the classes of string relations associated with their tree relations coincide.

Let us start with simple tree transducers, first introduced by Rounds (1970) and Thatcher (1970). For unrestricted *bottom-up tree transducers* (defining the class BOT), we find the following result (Engelfriet, 1975):

$$\mathsf{BOT} = \mathcal{B}(\mathsf{REL}, \mathsf{HOM})\,. \tag{25}$$

We can restrict the output homomorphism to be linear and non-deleting, and obtain a bimorphism characterization of the linear non-deleting bottom-up transducer:

$$\mathsf{ln\text{-}BOT} = \mathcal{B}(\mathsf{REL}, \mathsf{ln\text{-}HOM})\,. \tag{26}$$

By lifting the relabeling restriction from the input side, the linear nondeleting *extended top-down tree transducer* is obtained. We denote its class of transformations by ln-XTOP, and note that it is equally powerful as the class of transformations computed by linear nondeleting *extended bottom-up tree transducers* (Fülöp et al., 2011), written ln-XBOT. A decomposition result can be found in Engelfriet et al. (2009):

$$\mathsf{ln\text{-}XTOP} = \mathsf{ln\text{-}XBOT} = \mathcal{B}(\mathsf{ln\text{-}HOM}, \mathsf{ln\text{-}HOM})\,. \tag{27}$$

The attribute "extended" refers to the fact that when written as a transducer, a device computing a relation in ln-XTOP can process more than one symbol at a time, which corresponds to the inverse of the homomorphism that is used on the input side, as opposed to a relabeling. The class ln-XTOP computes the same relations as $\mathsf{REL}^{-1}\,;\mathsf{STSG}\,;\mathsf{REL}$ (Shieber, 2004; Fülöp et al., 2010), i.e., a *synchronous tree substitution grammar* (Eisner, 2003) with relabelings on both sides. Another way to describe this relationship is that ln-XTOP is STSG with states (Fülöp et al., 2010). A weaker formalismus is SDTS which can be decomposed using two quasi-alphabetic homomorphisms (qA) (Steinby and Tîrnăucă, 2009).

However, as already mentioned in the introduction, we find that tree homomorphisms cannot deal with discontiguities. We will therefore generalize tree homomorphisms to sequences of trees. To this end, let Δ and Σ be ranked alphabets, and $m \in \mathbb{N}^+$. Let $X_n^m = \{x_{(i,j)} \mid i \in [n], j \in [m]\}$ be a set of variables for every $n \in \mathbb{N}$. A family f of mappings $\left(f_k : \Delta^{(k)} \to T_\Sigma(X_k^m)^m\right)$ determines a homomorphism $h_f : T_\Delta \to T_\Sigma^m$, called *tree m-morphism*, given by

$$h_f(\delta(t_1, \ldots, t_k)) = \xi(f_k(\delta)) \tag{28}$$

for every $\delta \in \Delta^{(k)}$, where $\xi : X_k^m \to T_\Sigma$ is the ground substitution defined by $\xi(x_{(i,j)}) = (h_f(t_i))_j$.

Tree m-morphisms were initially defined by Arnold and Dauchet (1982). We will write m-MM for the class of tree m-morphisms, and $\mathsf{MM} = \bigcup_k k\text{-}\mathsf{MM}$. Let π_1 be defined such that $\pi_1(s)$ is the first item of the sequence s. In this way, $\{h\,;\pi_1 \mid h \in \mathsf{MM}\}$ (i.e., taking only the first tree of the multi-tree sequence) coincides with the class h-MBOT of tree transformations computed by homomorphic multi bottom-up tree transducers (Engelfriet et al., 2009). We note that $\mathsf{HOM} = \text{1-}\mathsf{MM}\,;\pi_1$, which emphasizes the notion that m-morphisms generalize homomorphisms.

Let $h : T_\Delta \to T_\Sigma$ be a tree m-morphism. If L is a weighted regular tree language over Σ, then $h^{-1}(L')$ is a weighted regular tree language over $\Delta \cup \{\bot^{(0)}\}$, where $L' = \{(\underbrace{t, \bot, \ldots, \bot}_{m \text{ components}}) \mid t \in L\}$. The main decomposition result using m-morphisms used in this dissertation is the following decomposition (Engelfriet et al., 2009):

$$\begin{aligned} \text{ln-XMBOT} &= \mathcal{B}(\text{ln-HOM}, \text{h-MBOT}) \\ &= \mathcal{B}(\text{ln-HOM}, \mathsf{MM}\,;\pi_1)\,. \end{aligned} \tag{29}$$

Note that $\text{h-MBOT} = \mathsf{MM}\,;\pi_1$, i.e., h-MBOT computes tree m-morphisms, taking the first component of the m-tree sequence (Engelfriet et al., 2009). Furthermore, $\text{ln-HOM} \subseteq \text{h-MBOT}$, and therefore $\text{ln-XTOP} \subseteq \text{ln-XMBOT}$. Moreover, $\text{ln-XMBOT} = \text{l-XMBOT}$ (Engelfriet et al., 2009), i.e., deletion does not give extra power to this class.

When m-morphisms are also allowed on the input side, we obtain *synchronous forest-substitution grammars* (Maletti, 2013), defining the class

$$\mathsf{SFSG} = \mathcal{B}(\mathsf{MM}\,;\pi_1, \mathsf{MM}\,;\pi_1)\,. \tag{30}$$

The class SFSG is essentially the class of *non-contiguous synchronous tree-sequence substitution grammars* (Zhang et al., 2008; Sun et al., 2009) with states (Raoult, 1997; Radmacher, 2008; Maletti, 2013).

Another popular synchronous formalism is *synchronous tree-adjoining grammar* (Shieber and Schabes, 1990; Büchse et al., 2011; Maletti, 2010a), which has a bimorphism decomposition using a special class of tree transducers called embedded transducers (ln-E). Nederhof and Vogler (2012) introduce *synchronous context-free tree grammars* (SCFTG) that can simulate all the above-mentioned

Formalism	Bimorphism	Source
ln-BOT =	$\mathcal{B}(\mathsf{REL}, \mathsf{ln\text{-}HOM})$	Fülöp et al. (2011)
BOT =	$\mathcal{B}(\mathsf{REL}, \mathsf{HOM})$	Fülöp et al. (2011)
SDTS =	$\mathcal{B}(\mathsf{qA}, \mathsf{qA})$	Steinby and Tîrnăucă (2009)
ln-XTOP =	$\mathcal{B}(\mathsf{ln\text{-}HOM}, \mathsf{ln\text{-}HOM})$	Engelfriet et al. (2009)
ln-XMBOT =	$\mathcal{B}(\mathsf{ln\text{-}HOM}, \mathsf{MM}\,;\pi_1)$	Engelfriet et al. (2009)
SFSG =	$\mathcal{B}(\mathsf{MM}\,;\pi_1, \mathsf{MM}\,;\pi_1)$	Raoult (1997)
STAG =	$\mathcal{B}(\mathsf{ln\text{-}E}, \mathsf{ln\text{-}E})$	Shieber (2006)
SCFTG =	$\mathcal{B}(\mathsf{MAC}, \mathsf{MAC})$	Nederhof and Vogler (2012)

Table 4. Bimorphism decomposition of selected synchronous formalisms

formalisms. Their bimorphism decomposition makes use of *macro tree transducers* (MAC). Table 4 summarizes bimorphism decompositions of selected formalisms.

We now turn our attention to string-generating synchronous grammars. Hiero (Chiang et al., 2005; Chiang, 2007) extends phrase-based translation by introducing linked variables. This implements a weaker version of the popular *synchronous context-free grammar* (SCFG) formalism.

Another restriction of SCFG that has been proposed is *inversion transduction grammar* (Wu, 1995, 1997). For completeness, we also mention positive and bottom-up non-erasing binary *range concatenation grammars* that, in turn, are an extension of inversion transduction grammars, proposed by Søgaard (2008). Unrestricted synchronous context-free grammar SCFG is weakly equivalent to SDTS, STSG and ln-XTOP, despite their varying capabilities. The string-generating power of ln-XMBOT has been studied by Gildea (2012), yielding that ln-XMBOT is weakly equivalent to a restricted form of *synchronous linear context-free rewriting systems* (SLCFRS, Kaeshammer (2013)). Furthermore, SFSG is weakly equivalent to unrestricted SLCFRS as well as *generalized multitext grammars* (GMTG, Melamed et al. (2004)).

In the next sections, we will discuss desirable qualities of formalisms for translation models, both from an empirical and algorithmic point of view, keeping in mind the tradeoff between explanatory power and computational complexity.

4.2.1 Empirical adequacy

Linguistic expressiveness is a crucial concern: can the model handle empirically observed data, and can it do so in a straightforward, elegant manner? Moreover, is it possible to efficiently represent patterns, and is the description succinct? We will also report in this section whether the formalisms in question have already been successfully applied to machine translation tasks.

In this section, we again assume a translation system from $\mathbf{F}$ into $\mathbf{E}$, implemented as a bimorphism $\mathcal{D} = (\mathsf{in}, L, \mathsf{out})$ such that $L : T_\Delta \to \mathbb{R}$ and $\mathsf{in} : T_\Delta \to T_F$ and $\mathsf{out} : T_\Delta \to T_E$ are homomorphisms. Let $\mathbf{f} \in \mathbf{F}$ and $\mathbf{e} \in \mathbf{E}$.

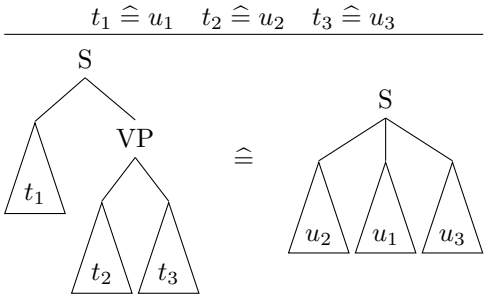


Figure 15. Example of a local rotation

We will write $\mathbf{f} \mathrel{\widehat{=}} \mathbf{e}$ if $\mathbf{e}$ is a possible translation of $\mathbf{f}$, i.e., there exists a derivation $t \in T_\Delta$ with $L(t) \neq 0$ such that $\mathbf{f} = \mathsf{yield}(\mathsf{in}(t))$ and $\mathbf{e} = \mathsf{yield}(\mathsf{out}(t))$. We will also write $\mathsf{in}(t) \mathrel{\widehat{=}} \mathsf{out}(t)$ to indicate that the translation is possible on the tree level. This enables us to study the capability of generalizing systematic translation patterns.

Let us introduce a number of transformations that need to be handled. We distinguish structural divergences that are purely syntactical, and lexical-semantic divergences. The topic of lexical-semantic divergences in translation has been extensively studied by Dorr (1994). She identifies a number of typical divergences between English, Spanish and German. What the examples in her classification have in common is that they require local reordering. Fox (2002) studies reordering between English and French and concludes that few systematic reorderings can account for the majority of divergences.

Let $t_1, t_2, t_3 \in T_{\mathbf{F}}$, and $u_1, u_2, u_3 \in T_{\mathbf{E}}$ such that $t_1 \mathrel{\widehat{=}} u_1$, $t_2 \mathrel{\widehat{=}} u_2$ and $t_3 \mathrel{\widehat{=}} u_3$. Then Figure 15 shows a transformation that occurs in English-to-Arabic translation practice (Knight, 2008) and should therefore be handled.

First, let us remark that this type of transformation can be called a *local rotation* or reordering even though the size of the subtrees t_i and u_i is potentially unbounded. However, only constituents that are separated by a bound degree of parent and sibling relationships need to be reordered. This enables tree-based translation models to handle rotations more easily than word-based or phrase-based translation models, whose reordering capabilities are usually restricted. In particular, word-based or phrase-based models with fixed reordering limit cannot express these reorderings.

It turns out that a top-down tree transducer (a stateful tree homomorphism (Yamada and Knight, 2001, 2002)) can handle the local rotation in Figure 15 only if it is neither non-deleting nor linear. However, linearity and non-deletion are two desirable properties. Moreover, the solution using deletion and copying is rather cumbersome. Neither can Figure 15 be expressed by a synchronous context-free grammar (SCFG). The class of transformations represented by this transformation is the class of *flattenings*. The property that is needed to handle flattenings, and that both SCFG and top-down tree transducers are lacking is a property called deep attachment of variables, as pointed out

by Maletti et al. (2009).

This inability to handle certain local rotations illustrates that top-down tree transducers are not symmetric, and furthermore that they are generally inadequate for linguistic tasks (Shieber, 2004). Both the flattening and its inverse can however be expressed by STSG and ln-XTOP. Mind that SCFG is only weakly equivalent to these formalisms.

Moreover, SCFG is also weakly equivalent to SDTS, which is less powerful than STSG because it does not have the power of deep attachment of variables. It turns out that SDTS is also not able to handle the flattening of Figure 15. We conclude this example by dismissing top-down tree transducers and SDTS, and stating that STSG and more powerful formalisms have the ability to realize local reorderings. Formalisms similar to ln-XTOP have been implemented in tree-to-string and string-to-tree models (Huang et al., 2006; Graehl et al., 2008; Galley et al., 2004; Neubig, 2014).

Next, let us turn to the question of *non-contiguity*. Recall that by assigning a tree structure to an input sentence, a context-free parser also groups the sentence into substrings that are the yields of the parse tree's subtrees. We call every subtree (and by extension its yield) a *constituent* of the input sentence, with the intuition that every constituent forms a meaningful linguistic unit. Constituents are usually classified by the label of their root.

Any constituent whose yield is a contiguous substring of the input sentence is called a contiguous constituent, and any constituent whose yield is not a contiguous substring of the input sentence is called a non-contiguous constituent. We can represent non-contiguous constituents by sequences of subtrees of the parse tree. Even though context-free parse trees, by definition, only exhibit contiguous constituents, we can see evidence of non-contiguous constituents as soon as we study parallel text. This has already been observed by Shieber and Schabes (1990).

To illustrate this point, Figure 16 shows a sentence-aligned pair of English and German sentences. Both sentences have been parsed by Egret and BitPar, respectively, and a word alignment has been generated. The alignments between words are shown in dashed and dotted lines. We can see that *'went'* is aligned to *'ist'* and *'gegangen'*, which are separated by *'nach hause'*. This alignment is evidence that the English constituent 'VBD(went)' corresponds to the German non-contiguous constituent '(VAFIN-HD-Sg(ist), (VVPP-HD(gegangen))'. We call this type of alignment a *non-contiguous alignment*, and we extend translational correspondence $\widehat{=} \subseteq \overline{(T_F)}^* \times (T_E)^*$ to sequences of trees between languages **F** and **E** such that we can write

$$(\text{VBD(went)}) \mathrel{\widehat{=}} (\text{VAFIN-HD-Sg(ist)}, \text{VVPP-HD(gegangen)})\,.$$

It is natural to also extend the notion of constituency to include sequences of constituents. Like in Section 3.2.2, where we defined an equivalence relation on constituents based on their root label, we can say that sequences of constituents are in the same equivalence class whenever their root labels match, i.e., $(t_1, \ldots, t_k) \equiv (t'_1, \ldots, t'_k)$ if $t_i(\varepsilon) = t'_i(\varepsilon)$ for every $i \in [k]$ with trees

$t_1, \ldots, t_k, t'_1, \ldots, t'_k \in T_L$ for some language $\mathbf{L}$. Note how this connects to the Myhill-Nerode theorem discussed earlier.

Other phenomena that exhibit non-contiguity appearing unbounded on the string level, but local on the tree level are topicalization (Maletti, 2015) and long-distance dependencies, such as agreement between constituents that are separated by arbitrary material or German particle verbs. All of these can be elegantly modelled in ln-XMBOT (see also Section 5.1 about rule extraction). Other formalisms that are equipped to model non-contiguity are STAG (Shieber and Schabes, 1990; Shieber, 2004, 2006, 2007; DeNeefe, 2011) and (the non-contiguous variant of) synchronous tree-sequence substitution grammar (Sun et al., 2009), or its stateful version SFSG (Maletti, 2013).

The importance of non-contiguous translation in tree-based translation has been argued by (Sun et al., 2009), who show that allowing discontiguities in either source or target has a beneficial effect on translation quality. In fact, their results seem to indicate that it is enough to only allow discontiguities on one side. In string-to-string translation, Kaeshammer (2015) implements a translation system based on LCFRS (weakly equivalent to SFSG and GMTG). Moreover, a variant of ln-XMBOT has been implemented in the popular Moses statistical machine translation toolkit (Braune et al., 2013). This contribution adds *shallow multi bottom-up tree transducers*, i.e., devices that are effectively restricted to local tree grammars. String-to-tree ln-XMBOT translation has been explored by Seemann et al. (2015b) and produced good results. A systematic evaluation of ln-XMBOT in machine translation can be found in (Seemann et al., 2015a).

4.2.2 Theoretical properties

In the previous section, we established that at least the power of ln-XTOP or STSG is needed to handle most common transformations in natural language. However, the ability to handle discontiguities is a strong argument in favor of more powerful formalisms. We will now consider theoretical and algorithmic properties of selected powerful formalisms, making sure to stay in the realm of tractability.

First, let us summarize what requirements we have from the theoretical point of view. For any given bimorphism class $\mathcal{C}$ of bimorphisms and for every $\mathcal{D} = (\mathsf{in}, L, \mathsf{out})$ such that $\mathsf{in}^{-1}\,;L\,;\mathsf{out} \in \mathcal{C}$, we consider the following properties:

- L is a weighted regular tree language
  We require $L \in \mathsf{REC}(\Delta)$, which in fact is true by the definition of bimorphisms. This property allows us to use closure properties along the pipeline, and furthermore it allows us to easily use the EM algorithm to optimize the weights of L. We will explain this in detail in Section 5.4.3.

- in^{-1} preserves regularity
  We require that the inverse of the input homomorphism preserves regularity, i.e., $\mathsf{in}^{-1}(L) \in \mathsf{REC}(\Delta)$ for every $L \in \mathsf{REC}(F)$. This is a desirable property because the output of a parser P_F will be a weighted regular

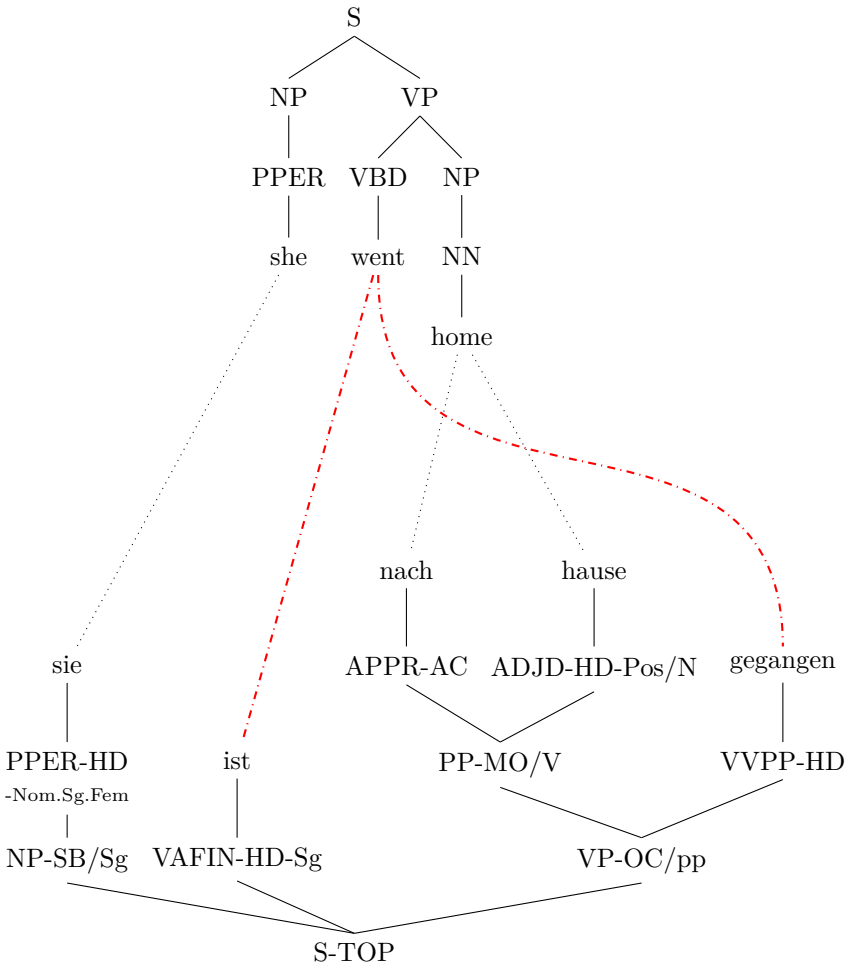


Figure 16. Bilingual pair of parse trees with word alignment exhibiting a non-contiguous alignment (red, dashdotted lines)

tree language, and a regular representation of $\mathsf{in}^{-1}(P_F\langle\mathbf{f}\rangle)$ allows us to use closure properties, like closure under intersection, for the remaining part of the pipeline. We will explain this in detail in Section 5.2.1. Both Equation (27) and (29) share the property that they use a tree homomorphism as their first component. This has the algorithmic advantage that the inverse image of a regular tree language under a tree homomorphism is easy to compute. Therefore, this makes a compelling argument for In-XTOP and In-XMBOT, while the latter is preferred (Maletti, 2011b).

- $\mathsf{in}^{-1}\,;L\,;\mathsf{out}$ preserves regularity
  We deem this property (application of $\mathcal{D}$ preserves regularity) secondary, as long as in^{-1} preserves regularity. While preservation of regularity is a useful property because it enables bucket-brigade approaches (May et al., 2010), this is more of a concern of individual parts of the pipeline, not the full pipeline.

- out^{-1} preserves regularity
  We require that the inverse of the output homomorphism preserves regularity, i.e., $\mathsf{out}^{-1}(L) \in \mathsf{REC}(\Delta)$ for every $L \in \mathsf{REC}(E)$. This allows us to incorporate a string-based language model in a theoretically clean way. We will explain this in detail in Section 5.3.

- $\mathsf{out}^{-1}\,;L\,;\mathsf{in}$ preserves regularity
  Again, we deem this property (inverse application of $\mathcal{D}$ preserves regularity) secondary.

- $\mathcal{C}$ is closed under composition
  In some settings, composition is useful because it allows to model a cascade of components that can then be composed into a monolithic model. Furthermore, a translation model $\mathcal{D}_1 : T_F \to T_E$ may be composed with a translation model $\mathcal{D}_2 : T_E \to T_G$ to yield $\mathcal{D}_1\,;\mathcal{D}_2 : T_F \to T_G$. In this way, the language $\mathbf{E}$ serves as a *pivot* language for translation between $\mathbf{F}$ and $\mathbf{G}$. Unfortunately, the class $\mathcal{B}(\mathsf{In\text{-}HOM}, \mathsf{In\text{-}HOM})$ is not closed under composition. However, the class $\mathcal{B}(\mathsf{In\text{-}HOM}, \mathsf{MM}\,;\pi_1)$ is closed under composition and contains all compositions of $\mathcal{B}(\mathsf{In\text{-}HOM}, \mathsf{In\text{-}HOM})$ which provides a compelling argument for the use of In-XMBOT over In-XTOP.

- $\mathcal{C}$ is symmetric
  This property is somewhat important from a practical point of view, as it allows us to use the same set of algorithms for the inverse of any bimorphism in $\mathcal{C}$. However, we deem this property secondary, since also our probabilistic model (20) is not entirely symmetric due to the different features for source and target.

Table 5 summarizes the relevant linguistic and theoretical properties of some formalisms. Many of these can be found, e.g., in Maletti (2010b).

	ROT	DIS	TCH	REG	REG^{-1}	CMP	SYM
ln-TOP	no	no	?	yes	yes	yes	no
TOP	yes	no	yes	no	yes	no	no
ln-XTOP	yes	no	yes	yes	yes	no	yes
ln-XMBOT	yes	yes	yes	no	yes	yes	no
SFSG	yes	yes	yes	no	no	no	yes
STAG	yes	yes	yes	no	no	no	yes

ROT = handles rotations
DIS = handles discontiguity
TCH = efficiently trainable
REG = preservation of regularity
REG^{-1} = preservation of regularity of the inverse
CMP = closure under composition
SYM = symmetry

Table 5. Linguistic and theoretical properties of selected synchronous formalisms

The formalism of choice in this dissertation is the linear nondeleting multi bottom-up tree transducer (ln-XMBOT), which is the variant of multi bottom-up tree transducer that has been deemed most suitable for machine translation (Engelfriet et al., 2009). It subsumes synchronous tree substitution grammar and synchronous tree insertion grammar (Maletti, 2011b), but still has some favorable properties from an algorithmic point of view. In particular, ln-XTOP $\subseteq$ ln-XMBOT, and moreover every "sensible" transformation in XTOP is also contained in ln-XMBOT (Maletti, 2012). In conclusion, we choose ln-XMBOT as our default formalism because (1) its bimorphism decomposition has favorable computational properties; and (2) it is expressive enough to handle many syntactic phenomena elegantly.

It should be mentioned that ln-XTOP preserves regularity in both directions (Fülöp et al., 2010), while only the inverse of ln-XMBOT preserves regularity. However, as long as the derivation tree language is recognizable, this does not seem to constitute a drawback. For the expressive power of the regularity-preserving subclass of XMBOT, see Maletti (2015).

4.2.3 Related work

The idea of using bimorphism formulations of synchronous grammar formalisms in the context of machine translation was first discussed by Tîrnăucă (2016). Moreover, this dissertation draws inspiration from, and has goals very similar to the dissertation of Büchse (2015), who develops an algebraic specification of a decoder. However, he limits the presentation to SCFG and STSG.

Many synchronous grammar formalisms can be explained in the overarching

framework of interpreted regular tree grammars (Koller and Kuhlmann, 2011) (iRTG). In this approach, a regular tree language models the permissible derivation trees (i.e., trees over the basic units of translation), and for both source and target language, these trees are mapped to an algebra term via a tree homomorphism. Depending on the algebra, this term is then realized as a string, tree, or other kind of structure. Our bimorphism approach is very similar. The difference is that we do not use intermediate homomorphisms, rather we map the derivation trees to source and target objects directly using arbitrary homomorphisms. This way, the intermediate homorphism and the realization algebra can be simulated by an appropriately chosen homomorphism. On the other hand, by choosing an appropriate algebra, an iRTG can simulate any bimorphism. All constructions in this dissertation using bimorphisms could be stated using iRTG as well.

Chapter 5

Implementation

Contents

This chapter explains how to obtain a translation model and how to decode using the models obtained this way.

5.1 Rule extraction

We will now describe how to obtain a weighted regular tree language of derivation trees as well as tree homomorphisms in and out from data. Rule extraction is the task of automatically generating a set of translation rules, i.e., basic translation units, from a sentence-aligned, possibly word-aligned and bi-parsed parallel corpus. Rule extraction algorithms have been described for various formalisms, such as the extended tree-to-string transducer (Galley et al. (2004), followed up

by Galley et al. (2006)), where only one half of the parallel corpus is parsed. The algorithm proposed in this section is essentially the rule extraction for SFSG by Sun et al. (2009). A similar algorithm that extracts non-contiguous rules, but for the string-generating SLCFRS, has been described by Kaeshammer (2015). As a special case, we include the rule extraction algorithm for multi bottom-up tree transducers put forward by Maletti (2011a). The rule extraction implementation included in our toolkit extracts basic translation units for the multi bottom-up tree transducer formalism by default. This is, to the author's knowledge, the first implementation of the algorithm proposed by Maletti (2011a). By restricting the target to contiguous constituents, basic translation units for synchronous tree substitution grammar (Zhang et al., 2006) can be extracted.

Consider the word-aligned bilingual pair of parse trees (f, e, a) in Figure 17, where $f \in T_F$ is the upper tree, $e \in T_E$ is the lower tree, and the word alignment $a \subseteq \max \mathsf{pos}(f) \times \max \mathsf{pos}(e)$ is given by the dotted lines[18]. For convenience, and following standard practice in natural language processing, we will use symbols with implicit rank. For instance, $e(3)$ and $e(33)$ are both labeled NP but have rank 3 and 2, respectively. This is in conflict with the definition of ranked alphabets, but can easily be solved by replacing the symbols with NP-3 and NP-2, making the rank explicit.

We now explain an extension of the rule extraction algorithms proposed by Maletti (2011a), Zhang et al. (2008) and Sun et al. (2009). We will present a two-step algorithm that first identifies in which way basic units of translations might have been composed, and then deduces inference rules from there. We use the intuition that whenever sequences of trees in f (or single trees for multi bottom-up tree transducer rules) are consistently aligned to sequences of trees in e, a translational correspondence can be established between them.

First, we need some definitions. Let $U \subseteq \mathbb{N}^*$ be a finite prefix-closed position set, i.e., $pi \in U \implies p \in U$ for all $p \in \mathbb{N}^*$ and $i \in \mathbb{N}$. In particular, $\mathsf{pos}(t)$ is such a set for every tree t. For every position $p \in U$, we define the *span* of p, by

$$\bigwedge_U p = \{p' \in \max U \mid p \sqsubseteq p'\}\,. \tag{31}$$

If $U = \mathsf{pos}(t)$ for some tree t, the span of a position p is the set of all frontier positions in the subtree rooted in p. We generalize $\bigwedge_U$ in the obvious way to subsets $P \subseteq U$ by

$$\bigwedge_U P = \bigcup_{p \in P} \bigwedge_U p\,. \tag{32}$$

Furthermore, for a subset of frontier positions $P \subseteq \max U$, we define the *closure* of P:

$$\bigtriangleup_U P = \Big\{p \in U \mid \bigwedge_U p \subseteq P\Big\}\,. \tag{33}$$

[18] Adapted from "A Failed Global Recovery" by Stephen S. Roach https://www.project-syndicate.org/commentary/a-failed-global-recovery with slight changes. The original phrasing is: "Most pundits dismiss the possibility of a double-dip recession". The German translation is: "Die meisten Beobachter schließen die Möglichkeit einer W-förmigen Rezession aus."

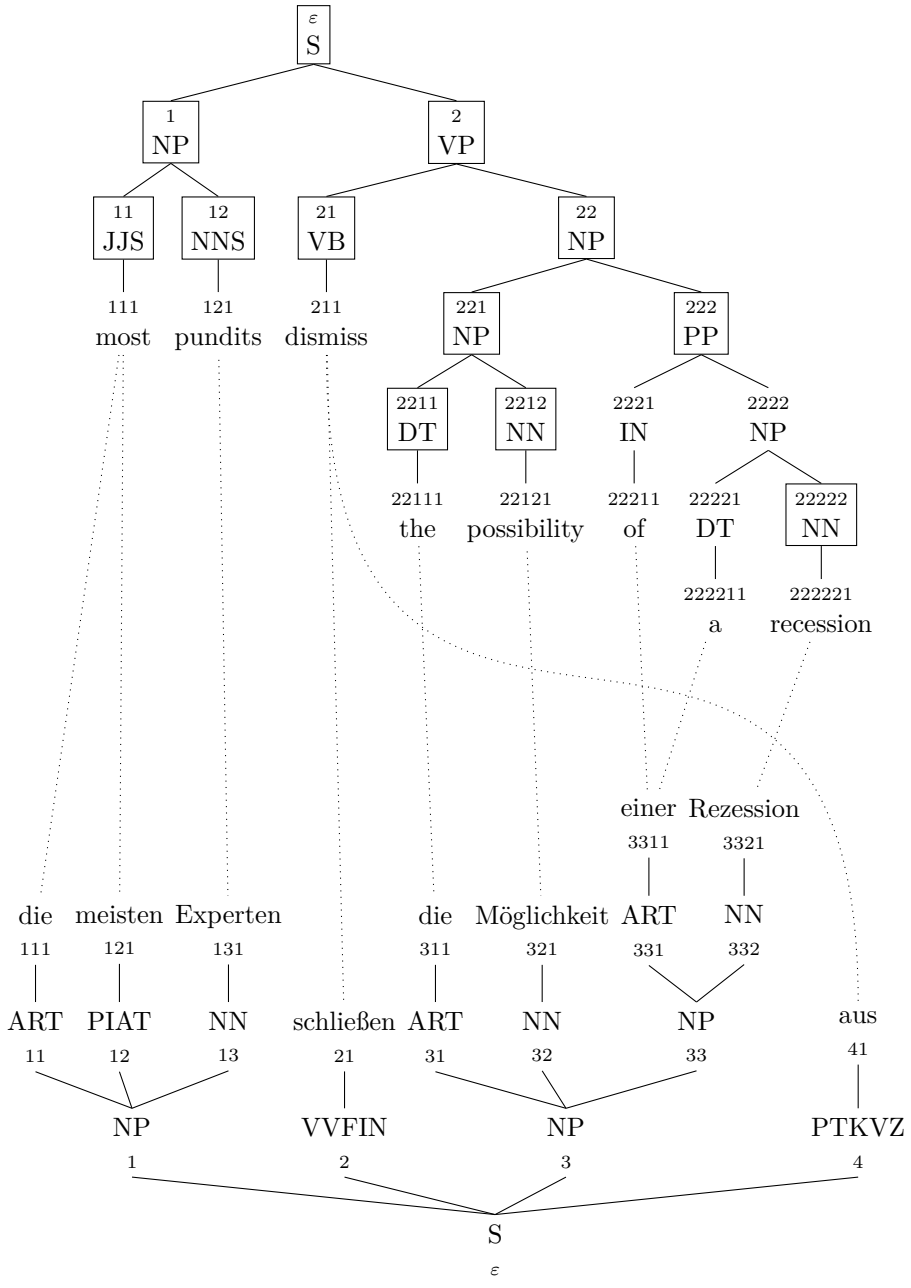


Figure 17. Bilingual pair of parse trees with word alignment (f, e, a)

Example 9
Consider f in Figure 17. We have $\bigwedge_{\overline{\mathsf{pos}(f)}} \{1\} = \{111, 121\}$ and $\bigwedge_{\overline{\mathsf{pos}(f)}} \{21\} = \{211\}$. Therefore,

$$\bigwedge_{\mathsf{pos}(f)} \bigwedge_{\overline{\mathsf{pos}(f)}} \{1, 21\} = \bigwedge_{\mathsf{pos}(f)} \{111, 121, 211\} = \{1, 11, 111, 12, 121, 21, 211\}\,.$$

We will drop the subscript whenever it is clear from the context. Recall that $\sqsubseteq$ is a partial order on positions, therefore we can make use of the min operation for a given subset $P \subseteq U$:

$$\min P = \{p \in P \mid \forall p' \in P : p' \sqsubseteq p \implies p' = p\}\,.$$

Intuitively, this is the set of highest positions in P. For each subset of frontier positions $P' \subseteq \max U$, the set $\min \bigwedge_U P'$ is called the *minimal* position set of P'.

Example 10
Consider f in Figure 17. We have

$$\min \bigwedge_{\mathsf{pos}(f)} \bigwedge_{\overline{\mathsf{pos}(f)}} \{1, 21\} = \min\{1, 11, 111, 12, 121, 21, 211\} = \{1, 21\} \quad \text{and}$$
$$\min \bigwedge_{\mathsf{pos}(f)} \bigwedge_{\overline{\mathsf{pos}(f)}} \{11, 12\} = \min\{1, 11, 111, 12, 121\} = \{1\}\,.$$

Lemma 1
Let $U \subseteq \mathbb{N}^*$ be a finite prefix-closed position set. For all $P, P' \subseteq \max U$, we have

$$P = P' \iff \bigwedge P = \bigwedge P' \iff \min \bigwedge P = \min \bigwedge P'\,.$$

Proof. $P = P'$ trivially implies $\bigwedge P = \bigwedge P'$, and $\bigwedge P = \bigwedge P'$ trivially implies $\min \bigwedge P = \min \bigwedge P'$. To complete the circle, let $p \in P$. Then $p \in \bigwedge P$ and thus there exists $p' \in \min \bigwedge P$ such that $p' \sqsubseteq p$. Consequently, $p' \in \bigwedge P'$, which yields that $p \in P'$ because $p \in \bigwedge_U p' \subseteq P'$. This proves $P \subseteq P'$, and $P' \subseteq P$ can be proved in the same manner.

□

Now suppose we have (f, e, a) like in Figure 17, with the source tree $f \in T_F$ and the target tree $e \in T_E$, as well as an alignment relation $a \subseteq \max \mathsf{pos}(f) \times \max \mathsf{pos}(e)$. Let $U_f = \{p \in \mathsf{pos}(f) \mid \exists p' \in \mathsf{dom}(a) : p \sqsubseteq p'\}$ and $U_e = \{p \in \mathsf{pos}(e) \mid \exists p' \in \mathsf{ran}(a) : p \sqsubseteq p'\}$. Intuitively, this removes unaligned frontier nodes and positions that do not dominate at least one aligned frontier node.

We can now formalize the set of minimal *consistently aligned* position sets by defining that $B \subseteq U_f$ and $B' \subseteq U_e$ are consistently aligned, written $B \bowtie_{(f,e,a)}$

B', if there exist $P \subseteq \mathsf{dom}(a)$ and $P' \subseteq \mathsf{ran}(a)$ such that

$$\begin{aligned} a(P) = P' \text{ and } a^{-1}(P') = P \qquad &\text{and} \\ B = \min \bigwedge_{U_f} P \text{ and } B' = \min \bigwedge_{U_e} P' \,. \end{aligned} \tag{34}$$

We drop the subscript on $\bowtie$ whenever it is clear from the context. Intuitively, $B \bowtie B'$ defines a unit of translation, i.e., the tree sets $\{f|_b \mid b \in B\}$ and $\{e|_{b'} \mid b' \in B'\}$ can be synchronously generated in one derivation step.

We can use a shortcut to check for a given position set $B \subseteq U_f$ whether there exists $B' \subseteq U_e$ such that $B \bowtie B'$.

Lemma 2

$$\min \bigwedge_{U_f} a^{-1}\left(a\left(\underline{\bigwedge_{U_f}} B\right)\right) = B \implies B \bowtie \min \bigwedge_{U_e} a\left(\underline{\bigwedge_{U_f}} B\right) \quad \text{and}$$

$$\min \bigwedge_{U_f} a^{-1}\left(a\left(\underline{\bigwedge_{U_f}} B\right)\right) \neq B \implies B \notin \mathsf{dom}(\bowtie) \,.$$

Proof. For clarity, we drop the subscript U_f. We observe that $\underline{\bigwedge} B \subseteq a^{-1}(a(\underline{\bigwedge} B))$, which yields $\min \bigwedge \underline{\bigwedge} B \subseteq \min \bigwedge a^{-1}(a(\underline{\bigwedge} B)) = B$ using the assumption. Together with $B \subseteq \bigwedge \underline{\bigwedge} B$ and $\min B = B$ we obtain

$$\begin{aligned} \min \bigwedge a^{-1}\left(a\left(\underline{\bigwedge} B\right)\right) &= \min \bigwedge \underline{\bigwedge} B \\ a^{-1}\left(a\left(\underline{\bigwedge} B\right)\right) &= \underline{\bigwedge} B \\ a^{-1}\left(a\left(P\right)\right) &= P \,. \end{aligned}$$

by Lemma 1. For the second implication, $B \in \mathsf{dom}(\bowtie)$ implies that there exists $P \subseteq \mathsf{dom}(a)$ such that $\min \bigwedge P = B$, and furthermore $P = a^{-1}(P') = a^{-1}(a(P))$, which implies $\min \bigwedge a^{-1}\left(a\left(P\right)\right) = B$. From $\min \bigwedge P = B$, it follows that $\underline{\bigwedge} B = P$, which implies $\min \bigwedge a^{-1}\left(a\left(\underline{\bigwedge} B\right)\right) = B$.

□

Example 11
In Figure 17, $U_f = \mathsf{pos}(f)$ and $U_e = \mathsf{pos}(e)$ because a is total and surjective. All positions $p \in \mathsf{pos}(f)$ such that $\min \bigwedge a^{-1}\left(a\left(\underline{\bigwedge}\{p\}\right)\right) = \{p\}$ have been marked with a box. These are all singleton position sets in $\mathsf{pos}(f)$ that are consistently

aligned to position sets in $\mathsf{pos}(e)$. The relation $\bowtie$ is given by:

$$\begin{array}{ccc} \{\varepsilon\} \bowtie \{\varepsilon\} & \{1\} \bowtie \{1\} & \{11\} \bowtie \{11, 12\} \\ \{12\} \bowtie \{13\} & \{2\} \bowtie \{2, 3, 4\} & \{21\} \bowtie \{2, 4\} \\ \{22\} \bowtie \{3\} & \{221\} \bowtie \{31, 32\} & \{2211\} \bowtie \{31\} \\ \{2212\} \bowtie \{32\} & \{222\} \bowtie \{33\} & \{22222\} \bowtie \{332\}\,. \end{array}$$

Recall that the relation $\bowtie$ is the set of all pairs of consistently aligned position sets. We extend the partial order $\sqsubseteq$ to $\mathsf{dom}(\bowtie)$ such that:

$$B_1 \sqsubseteq B_2 \iff \forall b_2 \in B_2 : \exists b_1 \in B_1 : b_1 \sqsubseteq b_2\,.$$

Now let us define a relation $\smile$ denoting *units of translation* by extending $\bowtie$ to larger structures. Let $B \in \mathsf{dom}(\bowtie)$, and $C \subseteq \mathsf{dom}(\bowtie)$ such that $B \bowtie B'$ and $C \bowtie C'$. Then a unit of translation is given by $(B, C) \smile (B', C')$ if

- $B \sqsubset C_i$ for all $C_i \in C$, and
- for all different $C_i, C_j \in C$, all $b_i \in C_i$ and $b_j \in C_j$ are incomparable with regard to $\sqsubseteq$ (implying that for all different $C'_i, C'_j \in C'$, all $b'_i \in C'_i$ and $b'_j \in C'_j$ are incomparable with regard to $\sqsubseteq$).

(Note that this condition implies that all different $C_i, C_j \in C$ (and $C'_i, C'_j \in C'$) are disjoint, and furthermore that $\bigwedge C_i$ and $\bigwedge C_j$, and $\bigwedge C'_i$ and $\bigwedge C'_j$ are disjoint.)

Let $M \subseteq \bowtie$. The unit of translation $(B, C) \smile (B', C')$ is an M-unit of translation if $(B, B') \in M$ and $(C_i, C'_i) \in M$ for every $C_i \in C$ such that $C_i \bowtie C'_i$. It is *M-minimal* if there are no M-units $(B, C_1) \smile (B', C'_1)$ and $(B_1, C_2) \smile (B'_1, C'_2)$ of translation such that $B_1 \in C_1$ (and $B'_1 \in C'_1$) and $C = C_1 - \{B_1\} \cup C_2$.

5.1.1 From parallel corpus to bimorphism

We can now infer a bimorphism from units of translation found in a parallel corpus. Our intuition is that units of translations can be composed if and only if their nonterminal symbols in the root and frontier positions match. This is otherwise known as *locality*, since the derivation tree language will be local. We will be using pairs of sequences of labels as states, using the $\widehat{=}$ symbol to appeal to the intuition about translational correspondences from the introduction.

Let $(B, C) \smile (B', C')$ be a unit of translation. In order to extract a canonical production rule, we notice that the position sets in C can be ordered lexicographically, and because $C_i \cap C_j = \emptyset$ for all different $C_i, C_j \in C$, we have either $C_i^{\leq} \leq C_j^{\leq}$ or $C_j^{\leq} \leq C_i^{\leq}$, and therefore this again defines a linear order. From now on, for convenience, we assume that $C^{\leq} = (C_1, \ldots, C_k)$ and $C_i \bowtie C'_i$ for every $i \in [k]$. We define a way to turn sets of positions into sequences of symbols in canonical order by letting $t(P) = (t(p_1), \ldots, t(p_n))$ where $P^{\leq} = (p_1, \ldots, p_n)$, i.e., the sequence of labels in lexicographic order, for every tree t over any ranked alphabet and $P \subseteq \mathsf{pos}(t)$. Similarly, we write $t|_P$ for $(t|_{p_1}, \ldots, t|_{p_n})$.

In order to motivate the construction that follows, let us first examine the intuitive inference rule described by $(B, C) \smile (B', C')$ which can be represented by:

$$\frac{f|_{C_1} \mathrel{\widehat{=}} e|_{C'_1} \quad \cdots \quad f|_{C_k} \mathrel{\widehat{=}} e|_{C'_k}}{f|_B \mathrel{\widehat{=}} e|_{B'}}.$$

Arguing that the type of a subtree is defined by its root position (recall the argument about equivalence classes of trees from Section 3.2.2, and equivalence classes of sequences of trees from Section 4.2.1), we will replace the positions by symbols, and use sequences of symbols as states in the tree automaton that we will construct. Then $(B, C) \smile (B', C')$ can be represented by a state and a sequence of states (which for illustrative purposes will still be written in the style of an inference rule)

$$\frac{f(C_1) \mathrel{\widehat{=}} e(C'_1) \quad \cdots \quad f(C_k) \mathrel{\widehat{=}} e(C'_k)}{f(B) \mathrel{\widehat{=}} e(B')},$$

and partial mappings $\mathsf{in} : (T_F^*)^k \to T_F^*$ and $\mathsf{out} : (T_E^*)^k \to T_E^*$ such that

$$\begin{aligned} \mathsf{in}(f|_{C_1}, \ldots, f|_{C_k}) &= f|_B \quad \text{and} \\ \mathsf{out}(e|_{C'_1}, \ldots, e|_{C'_k}) &= e|_{B'}\,. \end{aligned}$$

Let $X = \{x_{(i,j)} \mid i, j \in \mathbb{N}\}$ be a set of variables, and let us define two mappings $v_f : \bigcup_{i=1}^k C_i \to X$ and $v_e : \bigcup_{i=1}^k C'_i \to X$ such that:

$$\mathsf{index}(C_i)(p) = j \implies v_f(p) = x_{(i,j)} \quad \text{for all } p \in \bigcup_{i=1}^k C_i \text{ and}$$

$$\mathsf{index}(C'_i)(p') = j \implies v_e(p') = x_{(i,j)} \quad \text{for all } p' \in \bigcup_{i=1}^k C'_i\,.$$

These mappings are unambiguously defined because $C_i \cap C_j = \emptyset$ and $C'_i \cap C'_j = \emptyset$ for all $i \neq j$. Now we can define in and out as follows:

$$\begin{aligned} \mathsf{in}(f_1, \ldots, f_k) &= \xi(f'|_B) \quad \text{and} \\ \mathsf{out}(e_1, \ldots, e_k) &= \xi'(e'|_{B'})\,, \end{aligned}$$

where $\xi(x_{(i,j)}) = (f_i)_j$ and $\xi'(x_{(i,j)}) = (e_i)_j$ are ground substitutions, and f' and e' are obtained by chained replacement of subtrees by variables as follows:

$$\begin{aligned} f' &= (\ldots f[v_f(C_1^{\leq})]_{C_1^{\leq}} \ldots)[v_f(C_k^{\leq})]_{C_k^{\leq}} \quad \text{and} \\ e' &= (\ldots e[v_e(C_1'^{\leq})]_{C_1'^{\leq}} \ldots)[v_e(C_k'^{\leq})]_{C_k'^{\leq}}\,. \end{aligned}$$

Note that in and out are determined by $f'|_B$ and $e'|_{B'}$, respectively.

Additionally, we may keep information about the lexical alignments by restricting a to positions in $\bigwedge B$ and $\bigwedge B'$ that are not covered by any position set in $\bigwedge C$ or $\bigwedge C'$, adjusting the positions as needed. For every $(b_i, b'_j) \in B \times B'$, let $((i,p),(j,p')) \in a'$ if $(b_i.p, b'_j.p') \in a$ and $c \not\sqsubseteq b_i.p$ for any $c \in \bigcup C$ and $c' \not\sqsubseteq b'_j.p'$ for any $c' \in \bigcup C'$.

Example 12 (*Rule excision.*)
Let us excise an inference rule for the unit of translation given by

$$\begin{gathered}(\{2\},\{\{21\},\{221\},\{222\}\}) \smile (\{2,3,4\}\,\{\{2,4\},\{31,32\},\{33\}\})\\ \{2\} \bowtie \{2,3,4\}\\ \{21\} \bowtie \{2,4\}\\ \{221\} \bowtie \{31,32\}\\ \{222\} \bowtie \{33\}\,.\end{gathered}$$

The relevant sequences of symbols are:

$$\begin{aligned} f(\{2\}) &= (\mathrm{VP}) & e(\{2,3,4\}) &= (\mathrm{VVFIN},\mathrm{NP},\mathrm{PTKVZ})\\ f(\{21\}) &= (\mathrm{VB}) & e(\{2,4\}) &= (\mathrm{VVFIN},\mathrm{PTKVZ})\\ f(\{221\}) &= (\mathrm{NP}) & e(\{31,32\}) &= (\mathrm{ART},\mathrm{NN})\\ f(\{222\}) &= (\mathrm{PP}) & e(\{33\}) &= (\mathrm{NP})\,. \end{aligned}$$

Then the state and state sequence are

$$\frac{(\mathrm{VB}) \mathrel{\widehat{=}} (\mathrm{VVFIN},\mathrm{PTKVZ}) \quad (\mathrm{NP}) \mathrel{\widehat{=}} (\mathrm{ART},\mathrm{NN}) \quad (\mathrm{PP}) \mathrel{\widehat{=}} (\mathrm{NP})}{(\mathrm{VP}) \mathrel{\widehat{=}} (\mathrm{VVFIN},\mathrm{NP},\mathrm{PTKVZ})}\,.$$

The mappings v_f and v_e are defined by:

$$\begin{aligned} v_f(21) &= x_{(1,1)} & v_f(221) &= x_{(2,1)} & v_f(222) &= x_{(3,1)}\\ v_e(2) &= x_{(1,1)} & v_e(4) &= x_{(1,2)} & v_e(31) &= x_{(2,1)}\\ v_e(32) &= x_{(2,2)} & v_e(33) &= x_{(3,1)}\,. \end{aligned}$$

We use v_f and v_e to replace positions by variables in f and e as follows:

$$f' = f[\underbrace{x_{(1,1)}}_{v_f(21)}]_{21}\ [\underbrace{x_{(2,1)}}_{v_f(221)}]_{221}\ [\underbrace{x_{(3,1)}}_{v_f(222)}]_{222} \qquad \text{and}$$

$$e' = e\,\underbrace{[x_{(1,1)}]_2}_{v_e(2)}\ \underbrace{[x_{(1,2)}]_4}_{v_e(4)}\ \underbrace{[x_{(2,1)}]_{31}}_{v_e(31)}\ \underbrace{[x_{(2,2)}]_{32}}_{v_e(32)}\ \underbrace{[x_{(3,1)}]_{33}}_{v_e(33)}\,.$$

Then $\mathsf{in}(f_1,f_2,f_3) = \xi(f'|_{\{2\}})$, and $\mathsf{out}(e_1,e_2,e_3) = \xi'(e'|_{\{2,3,4\}})$ because $\{2\} \bowtie \{2,3,4\}$, with ξ and ξ' defined as above. Figure 18 shows a graphical representation, with in given by $f'|_{\{2\}}$ and out given by $e'|_{\{2,3,4\}}$.

Now let $\mathcal{C} = ((f_1,e_1,a_1),\ldots,(f_n,e_n,a_n))$ be a parallel, parsed, word-aligned corpus of length n where $a_i \subseteq \max \mathsf{pos}(f_i) \times \max \mathsf{pos}(e_i)$. For every $i \in [n]$, let $\smile_i$ be the set of M_i-units of translation of (f_i,e_i,a_i), where $M_i \subseteq \bowtie_{(f_i,e_i,a_i)}$. Recall that for every unit of translation $(B,C) \smile_i (B',C')$, we extracted a quintuple $(q,(q_1,\ldots,q_k),f'|_B,e'|_{B'},a')$ of

- a state $q \in F^m \times E^n$,
- a state sequence $(q_1,\ldots,q_k) \in (F^* \times E^*)^k$,

- a tree sequence $f'|_B \in T_F(X)^m$,
- a tree sequence $e'|_{B'} \in T_E(X)^n$, and
- a terminal alignment relation

$$a' \subseteq \Big(\bigcup_{i \in [m]} \{i\} \times \max \mathsf{pos}((f'|_B)_i) \Big) \times \Big(\bigcup_{j \in [n]} \{j\} \times \max \mathsf{pos}((e'|_{B'})_j) \Big) .$$

Let us define a mapping ρ that maps every unit of translation in $\bigcup_{i \in [n]} \smile_i$ to such a quintuple by the procedure laid out above. Then let Δ be a ranked alphabet such that every $\rho((B, C) \smile_i (B', C')) = (q, (q_1, \ldots, q_k), f'|_B, e'|_{B'}, a')$ is a k-ary symbol in Δ. Now let $\mathcal{C} = (Q, I, P)$ be a weighted tree automaton such that

$$\begin{aligned} Q &= \{q \mid (q, (q_1, \ldots, q_k), f'|_B, e'|_{B'}, a') \in \Delta\} \\ P(q, \delta(q_1, \ldots, q_k)) &= \begin{cases} 1 & \text{if } \delta = (q, (q_1, \ldots, q_k), f'|_B, e'|_{B'}, a') \in \Delta \\ 0 & \text{otherwise} \end{cases} \\ I(q) &= \begin{cases} 1 & \text{if } q = f_i(\varepsilon) \mathbin{\widehat{=}} e_i(\varepsilon) \text{ for some } i \in [n] \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{35}$$

Finally, we observe that in and out as defined above extend to a homomorphism (more precisely, an m-morphism) over Δ, and therefore $(\mathsf{in}\,;\pi_1, L_{\mathcal{C}}, \mathsf{out}\,;\pi_1)$ is a bimorphism in the class $\mathcal{B}(\mathsf{MM}\,;\pi_1, \mathsf{MM}\,;\pi_1) = \mathsf{SFSG}$. This way, we subsume:

- Rule extraction for non-contiguous synchronous tree sequence substitution grammar (STSSG, Sun et al. (2009)), where $M = \bowtie$, and m-morphisms are used on both the source and the target side.

- Rule extraction for linear and nondeleting extended multi bottom-up tree transducers (ln-XMBOT), where

$$M = \bowtie \cap \Big(\big\{ \{p\} \mid p \in \mathsf{pos}(f) \big\} \times 2^{\mathsf{pos}(e)} \Big) ,$$

  such that only tree homomorphisms are needed on the source side. An ln-XMBOT rule $(q, (q_1, \ldots, q_k)), f'|_B, e'|_{B'}, a')$ is traditionally represented by $f'' \to e'|_{B'}$ (where f'' is the only tree in $f'|_B$), with variables replaced by the corresponding label. More precisely, a variable $x_{(i,j)}$ is replaced in f'' by the j-th component of the left-hand side of q_i, and in $e'|_{B'}$ by the j-th component of the right-hand side of q_i. The states are implicitly represented by links between these symbols, and the variables can be easily reconstructed using the lexicographic ordering of positions in f''. Dotted lines indicate the terminal alignment a'. Figure 19 shows the more common representation of the rule from Figure 18.

- Extraction of minimal rules for ln-XMBOT (Maletti (2011a)): like above, but only extracting M-minimal rules. All minimal rules that can be extracted from the example (f, e, a) are given in Figures 20, 21 and 22.

$$\frac{(\mathrm{VB}) \mathrel{\widehat{=}} (\mathrm{VVFIN}, \mathrm{PTKVZ}) \quad (\mathrm{NP}) \mathrel{\widehat{=}} (\mathrm{ART}, \mathrm{NN}) \quad (\mathrm{PP}) \mathrel{\widehat{=}} (\mathrm{NP})}{(\mathrm{VP}) \mathrel{\widehat{=}} (\mathrm{VVFIN}, \mathrm{NP}, \mathrm{PTKVZ})}$$

$$\mathsf{in}(f_1, f_2, f_3) = \xi\Big(\; x_{(1,1)} \quad \overset{\mathrm{VP}}{\underset{\mathrm{NP}}{\overset{\diagup\diagdown}{\underset{x_{(2,1)} \quad x_{(3,1)}}{\diagup\diagdown}}}} \; \Big)$$

$$\mathsf{out}(e_1, e_2, e_3) = \xi'\Big(\; x_{(1,1)} \; , \quad \overset{\mathrm{NP}}{\overset{\diagup\diagdown\diagup\diagdown}{x_{(2,1)} \quad x_{(2,2)} \quad x_{(3,1)}}} \quad , \; x_{(1,2)} \; \Big)$$

Figure 18. Non-minimal rule

Figure 20 shows all minimal rules that have no nonterminal leaves ("lexical" rules). Figure 21 shows all minimal rules that have no terminal leaves ("structural" rules). Figure 22 shows a rule that has both terminal and nonterminal leaves ("hybrid" rules). In practice, this distinction is not important.

- Rule extraction for synchronous tree substitution grammar (STSG) or extended top-down tree transducers (ln-XTOP), where only homomorphisms are allowed on the source and the target side:

$$M = \bowtie \cap \Big(\big\{ \{p\} \mid p \in \mathsf{pos}(f) \big\} \times \big\{ \{p'\} \mid p' \in \mathsf{pos}(e) \big\} \Big) .$$

Extensions of this rule extraction algorithm come to mind. For instance, we only explained rule extraction for context-free parse trees. Rules for grammar formalisms that make a difference between derivation tree and derived tree (for instance, tree-adjoining grammar) might be extracted from the derivation trees. Word alignments should be put on the leaf symbols of the derived tree corresponding to the production rules that introduced these words. The mapping between derivation trees and derived trees will then be part of the input and output homomorphisms. Furthermore, rule extraction from forests instead of single trees has been suggested (Mi and Huang, 2008). We note that an extension of our rule extraction to forests (represented as weighted tree automata) is possible, but not trivial because in the presence of discontiguities, it should be ensured that any given sequence of states used in a rule can be obtained as part of a non-zero run of the forest.

Another possibility to extend rule extraction are weighted alignments. Assume a *weighted* alignment relation $\max \mathsf{pos}(f) \times \max \mathsf{pos}(e) \to \mathbb{R}$. Then a weighted version of $\bowtie$ may be defined, licensing the extraction of rules whose

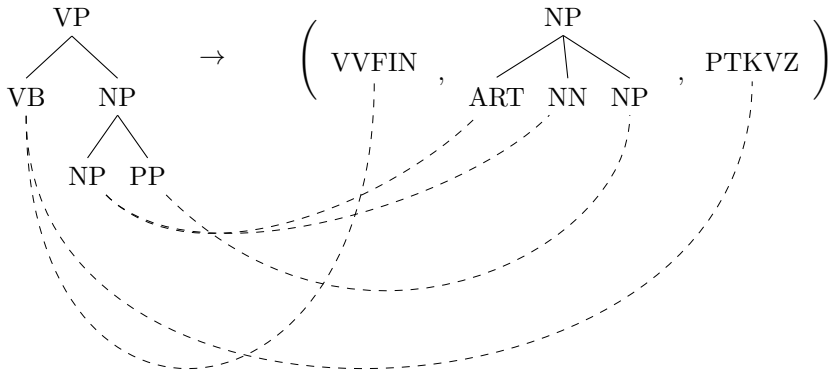


Figure 19. MBOT rule representation

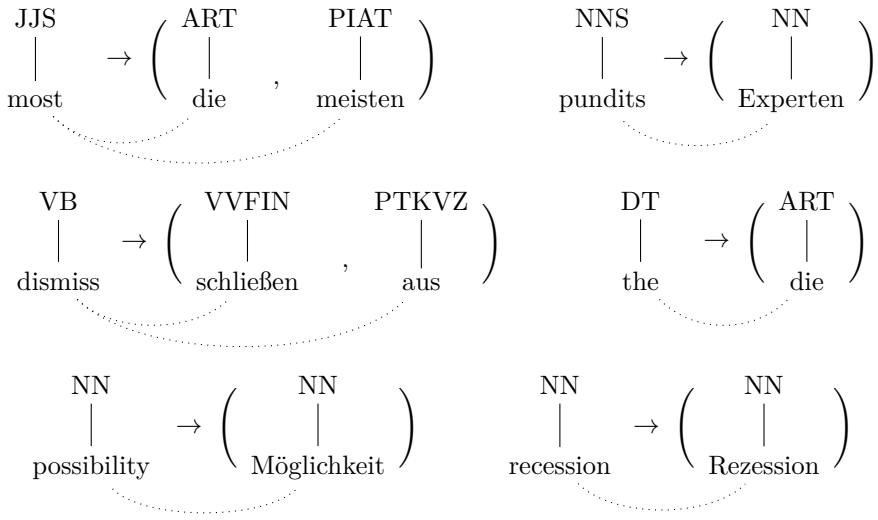


Figure 20. Minimal, purely lexical rules

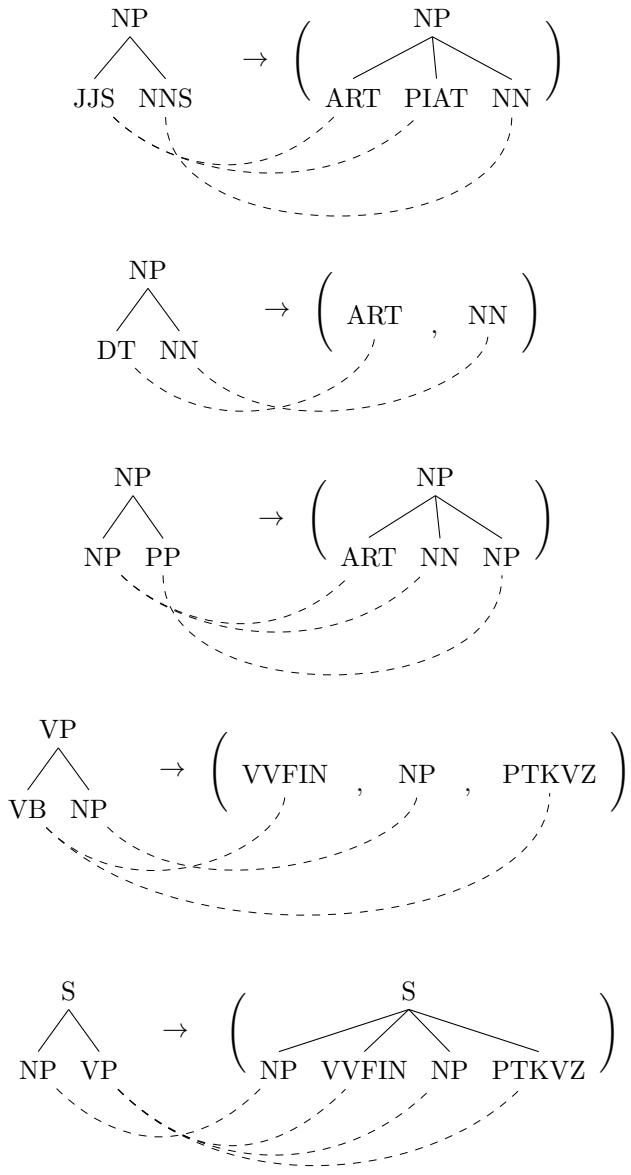


Figure 21. Minimal, purely structural rules

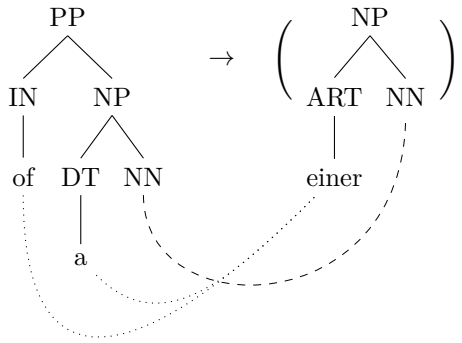


Figure 22. Minimal, hybrid rule

alignments have non-zero weight, but assigning to every extracted rule the weight of its alignment for rule scoring purposes. We will not discuss this method of assigning weights, however we will discuss how to assign weights based on relative frequency of the extracted patterns.

5.1.2 Relative frequency estimation

Next, we will describe how to assign weights to the productions of the tree automaton constructed in the previous section. Recall that, according to the construction in Equation (35), there is exactly one non-zero production for every symbol of the ranked alphabet Δ of translation rules, and therefore we can specify a weight function $w : \Delta \to \mathbb{R}$ to define the production weights. This extends to a feature function $\phi_w : T_\Delta \to \mathbb{R}$ in the natural way, i.e., $\phi_w(t) = \prod_{p \in \mathsf{pos}(t)} w(t(p))$. (This means that ϕ_w can be written as a homomorphism.)

Recall the mapping ρ from the previous section, and let $\#(\delta) = |\rho^{-1}(\delta)|$. This mapping associates to every translation rule its *count*, i.e., how often it was extracted. We can normalize counts by grouping rules that share a feature, e.g., all rules that share the same left-hand (relative-frequency estimation), and we can use lexical weighting. Relative-frequency estimation is a computationally cheap way of obtaining rule scores. Essentially, we are making the simplifying assumption that the sentence pair (f, e) has a canonical derivation, namely the derivation that used exactly the rules that we obtained from it. Other possible derivations are disregarded. In Section 5.4.3, we will discuss a method of obtaining rule scores that takes into account all possible derivations that explain (f, e).

To normalize counts among competing rules, we assume an equivalence re-

lation $\sim$ on Δ. Then for every $\delta \in \Delta$,

$$\phi_{\sim}(\delta) = \frac{\#(\delta)}{\sum_{\delta' \in [\delta]_{\sim}} \#(\delta')} \tag{36}$$

is the score that a feature function $\phi_{\sim}$ defined by $\sim$ assigns to δ. Assume $\delta = (q, (q_1, \ldots, q_k), f, e, a)$ and $\delta' = (q', (q'_1, \ldots, q'_{k'}), f', e', a')$. Let $q = \ell \mathrel{\widehat{=}} r$, $q' = \ell' \mathrel{\widehat{=}} r'$, and $q_i = \ell_i \mathrel{\widehat{=}} r_i$ as well as $q'_j = \ell'_j \mathrel{\widehat{=}} r'_j$ for every $i \in [k]$ and $j \in [k']$. Typical equivalence classes for feature functions are:

- $\delta \sim \delta'$ if $(q_1, \ldots, q_k) = (q'_1, \ldots, q'_{k'})$, or if $q = q'$; this can be used to model the symmetric translation probability.
- $\delta \sim \delta'$ if $\ell = \ell'$, or $k = k'$ and $\ell_i = \ell'_i$ for every $i \in [k]$, or if $f = f'$; this can be used to model the forward translation probability.
- $\delta \sim \delta'$ if $r = r'$, or $k = k'$ and $r_i = r'_i$ for every $i \in [k]$, or if $e = e'$; this can be used to model the backward translation probability.

All of these somehow convey the notion that δ and δ' are "competitors".

Example 13
Let Δ be the set of rules in Figures 20, 21 and 22, and $\sim$ be an equivalence relation on Δ determined by the left-hand side component of their parent state. We find the following equivalence classes:

$$\Delta/\sim = \{\mathrm{JJS}, \mathrm{NNS}, \mathrm{VB}, \mathrm{DT}, \mathrm{NN}, \mathrm{NP}, \mathrm{VP}, \mathrm{S}, \mathrm{PP}\}\,.$$

All rules have been observed exactly once, and every root symbol has been observed once, except NN which has been observed twice, and NP which has been observed three times. Therefore, for every $\delta \in \Delta$, the relative frequency is

$$\phi_{\sim}(\delta) = \begin{cases} 1/2 & \text{if } \delta \in \mathrm{NN} \\ 1/3 & \text{if } \delta \in \mathrm{NP} \\ 1 & \text{otherwise.} \end{cases}$$

Bidirectional lexical weighting is a way to assign scores to rules based on the lexical translation scores of their leaf nodes (Koehn et al., 2003). Let $\delta = (q, (q_1, \ldots, q_k), f', e', a)$. Recall that a is a relation between $F^{(0)}$-labeled leaves in f' and $E^{(0)}$-labeled leaves in e'. (In practice, a is augmented such that every unaligned word is aligned to a special NULL token.) Suppose we have lexical weights $w(\mathbf{e}|\mathbf{f})$ and $w(\mathbf{f}|\mathbf{e})$, e.g., from the IBM model parameters that generated the word alignments (see Section 3.3.1) for all $\mathbf{f} \in F^{(0)}$ and $\mathbf{e} \in E^{(0)}$. We can define the lexical weight for tree sequences $f = (f_1, \ldots, f_m)$ and $e = (e_1, \ldots, e_n)$ by:

$$\mathsf{lex}(f|e, a) = \prod_{(i,p) \in \mathsf{dom}(a)} \frac{1}{|a(i,p)|} \sum_{(j,p') \in a(i,p)} w(f_i(p)|e_j(p'))\,, \tag{37}$$

and we obtain the direct and inverse lexical weighting feature functions

$$\begin{aligned} \phi_{\mathsf{lex}}(\delta) &= \mathsf{lex}(e'|f', a) \qquad \text{and} \\ \phi_{\mathsf{lex}^{-1}}(\delta) &= \mathsf{lex}(f'|e', a^{-1})\,. \end{aligned} \tag{38}$$

Taking count feature functions and lexical weighting feature functions together, we arrive at a family of mappings $(\phi_i : \Delta \to \mathbb{R}^+ \mid i \in [n])$ that scores each rule according to feature functions, with the intuition that the feature score of a derivation tree can be computed as the product of its rule scores. These can be used to score productions in $\mathcal{C}$ to obtain a sequence of translation models $(\mathcal{T}_i \mid i \in [n])$, where each $\mathcal{T}_i$ is a weighted tree automaton over Δ that is obtained as the product of an unweighted tree automaton $\mathcal{T}'$ and the extension of ϕ_i to trees. This will be explained in more detail in the next section.

5.2 Decoding

We have explained how translation can be decomposed into distinct components that score various aspects of the translation, and how a translation model can be obtained from data. It remains to implement the relevant automata-theoretic constructions. Furthermore, a functioning decoder is necessary to implement MERT tuning (see Section 5.4.2) or EM training (see Section 5.4.3).

In this chapter, let us assume that a translation model is given as a bimorphism $\mathcal{D} = (\mathsf{in}, L, \mathsf{out})$, and the input text is given as a text $T = (\mathbf{f}_1, \ldots, \mathbf{f}_n)$, i.e., a sequence of strings $\mathbf{f}_i$, and each string needs to be translated independently. We parse T with a parser $P_F : T_F \to [0, 1]$ to obtain parse forests $P_F\langle\mathbf{f}_1\rangle, \ldots, P_F\langle\mathbf{f}_n\rangle$ as laid out in the linguistics section. We then apply the translation model to each $P_F\langle\mathbf{f}_i\rangle$ to obtain a weighted language of translation candidates. We summarize the decoding pipeline in Figure 23 to illustrate the steps carried out for every sentence $\mathbf{f}_i$.

Recall that decoding is the task of finding, for a given input sentence $\mathbf{f}$, the best output sentence $\hat{\mathbf{e}}$, approximated by (cf. Equation (24))

$$\hat{\mathbf{e}} \propto \arg\max_{\mathbf{e}} \sum_{t \in \mathcal{D}(\mathbf{f},\mathbf{e})} L(t)\,,$$

where $\mathcal{D}(\mathbf{f}, \mathbf{e}) = \{t \in T_\Delta \mid \mathsf{yield}(\mathsf{in}(t)) = \mathbf{f} \wedge \mathsf{yield}(\mathsf{out}(t)) = \mathbf{e}\}$. In practice, for efficiency's sake, we may approximate the summation over all derivations trees by finding the single highest-scoring derivation (the Viterbi derivation), i.e.,

$$\hat{t} \propto \arg\max_{t \in \mathcal{D}(\mathbf{f})} L(t)\,, \tag{39}$$

where $\mathcal{D}(\mathbf{f}) = \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f}))$. Thus, $\hat{\mathbf{e}}$ is approximated by $\mathsf{yield}(\mathsf{out}(\hat{t}))$. Recall the decomposition of $L = \mathcal{F} \cdot \mathcal{T} \cdot \mathcal{E}$ into separate components, as laid out in Equation (22), where

- $\mathcal{F} = \mathcal{F}_1^{\lambda_1}$ scores aspects of the input only;

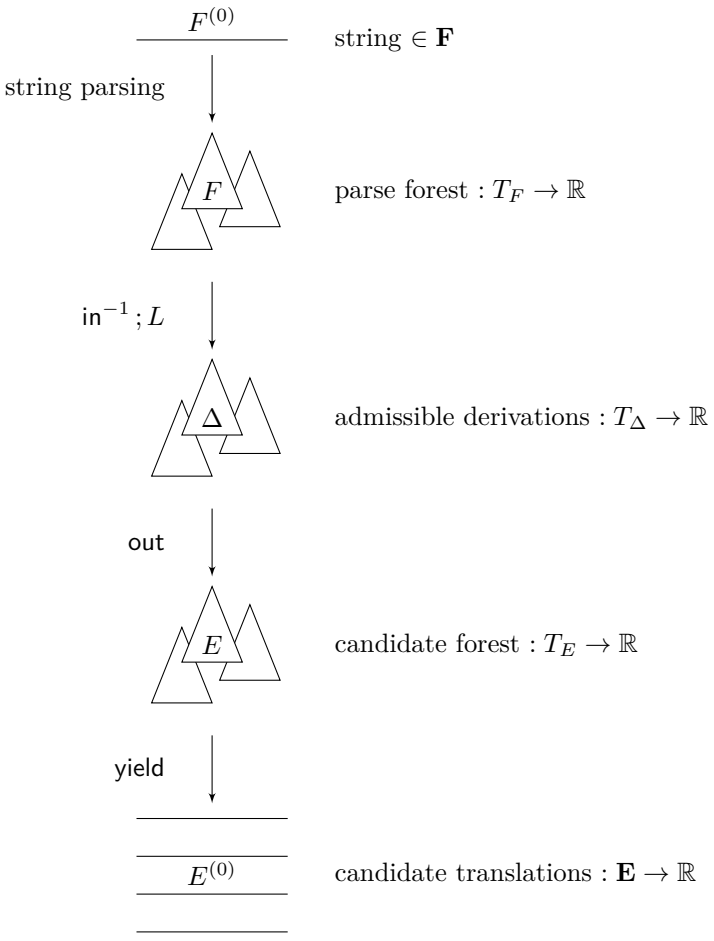


Figure 23. Bimorphism decoding pipeline

- $\mathcal{T} = \mathcal{T}_1^{\lambda_1} \cdot \mathcal{T}_2^{\lambda_2} \cdot \mathcal{T}_3^{\lambda_3}$ scores aspects of the translation (*translation model*); and
- $\mathcal{E} = \mathcal{E}_1^{\lambda_3} \cdot \mathcal{E}_2^{\lambda_3}$ scores aspects of the output only (*language model*).

In order to correctly use the weight vector λ, we introduce a family of scaling operations

$$\left(\cdot^a : (\mathbb{R}^+)^{T_\Delta} \to (\mathbb{R}^+)^{T_\Delta}\right)$$

with $L^a(t) = L(t)^a$ for every $a \in \mathbb{R}$, $t \in T_\Delta$ and $L : T_\Delta \to \mathbb{R}^+$.

If $L \in \mathsf{REC}(\Delta)$ is represented by an unambiguous weighted tree automaton, then L^a can be obtained by simply applying the exponentiation operation to each transition weight, since exponentiation distributes over multiplication. Thus, $L^a \in \mathsf{REC}(\Delta)$. We therefore assume that all $\mathcal{F}_i, \mathcal{T}_i, \mathcal{E}_i$ can be represented by unambiguous weighted tree automata. We then obtain:

$$L = \underbrace{\mathcal{F}_1^{\lambda_1}}_{\text{input}} \cdot \underbrace{\mathcal{T}_1^{\lambda_1} \cdot \mathcal{T}_2^{\lambda_2} \cdot \mathcal{T}_3^{\lambda_3}}_{\text{translation}} \cdot \underbrace{\mathcal{E}_1^{\lambda_3} \cdot \mathcal{E}_2^{\lambda_3}}_{\text{output}}, \tag{40}$$

and $\mathcal{F}, \mathcal{T}, \mathcal{E}, L \in \mathsf{REC}(\Delta)$ because of closure under intersection.

The weighted forest of scored derivation trees for a given input sentence $\mathbf{f}$ can thus be computed as:

$$\begin{aligned}
\mathcal{D}(\mathbf{f}) \cdot L &= \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot L \\
&= \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot (\mathcal{F} \cdot \mathcal{T} \cdot \mathcal{E}) \\
&= \Bigg(\underbrace{\Bigg(\underbrace{\Big(\underbrace{\mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot \mathcal{F}}_{\text{input}} \Big)}_{\text{input + translation}} \cdot \mathcal{T} \Bigg)}_{\text{input + translation + output}} \cdot \mathcal{E} \Bigg) .
\end{aligned} \tag{41}$$

The last representation of $\mathcal{D}(\mathbf{f}) \cdot L$ allows us to model the bimorphism in a cascade. In order to approximate $\hat{\mathbf{e}}$ by $\mathsf{yield}(\mathsf{out}(\hat{t}))$, we will compute

$$\begin{aligned}
\hat{t} &= \arg\max_{t \in \mathcal{D}(\mathbf{f})} L(t) \\
&= \arg\max_{t \in \mathcal{D}(\mathbf{f})} (\mathcal{F} \cdot \mathcal{T} \cdot \mathcal{E})(t) \\
&= \arg\max \mathcal{D}(\mathbf{f}) \cdot \mathcal{F} \cdot \mathcal{T} \cdot \mathcal{E} .
\end{aligned} \tag{42}$$

5.2.1 Input and translation models

First, we explain how to obtain $\mathcal{D}(\mathbf{f}) \cdot \mathcal{F} = \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot \mathcal{F}$. Assume that $P_F : T_F \to [0, 1]$ is a weighted regular tree language implementing a probabilistic parser. Then $\mathcal{F}_1 : T_\Delta \to \mathbb{R}^+$ is the corresponding weighted tree language scoring

derivation trees, defined by $\mathcal{F}_1 = \mathsf{in}^{-1}(P_F) = \mathsf{in}\,;P_F$, i.e., for every tree $t \in T_\Delta$, $\mathcal{F}_1(t) = P_F(\mathsf{in}(t))$. Then $\mathcal{F}_1$ is a weighted regular tree language because of closure under inverse homomorphism.

Lemma 3
Suppose that the input sentence $\mathbf{f}$ was parsed, obtaining a forest $P_F\langle\mathbf{f}\rangle$. Then

$$\mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot \mathcal{F}_1 = \mathsf{in}^{-1}(P_F\langle\mathbf{f}\rangle)\,. \tag{43}$$

Proof. According to the definition of a parser in Equation (11), we can rewrite

$$\begin{aligned}\mathsf{in}^{-1}(P_F\langle\mathbf{f}\rangle) &= \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f}) \cdot P_F)\\ &= \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot \mathsf{in}^{-1}(P_F)\\ &= \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot \mathcal{F}_1\,.\end{aligned}$$

□

Essentially, we therefore want to compute $\mathsf{in}^{-1}(P_F\langle\mathbf{f}\rangle)$, i.e., the $\mathcal{F}_1$-weighted set of derivation trees in T_Δ that map to a tree in the input parse forest. Let us assume that $\mathcal{F}_1$ is the only input feature function, and there exists a scaling parameter λ_1 such that $\mathcal{F} = \mathcal{F}_1^{\lambda_1}$ as mentioned above. We will directly compute

$$\begin{aligned}\mathsf{in}^{-1}(P_F\langle\mathbf{f}\rangle^{\lambda_1}) \cdot \mathcal{T} &= \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot \mathcal{F}_1^{\lambda_1} \cdot \mathcal{T}\\ &= \mathsf{in}^{-1}(\mathsf{yield}^{-1}(\mathbf{f})) \cdot \mathcal{F} \cdot \mathcal{T}\\ &= \mathcal{D}(\mathbf{f}) \cdot \mathcal{F} \cdot \mathcal{T}\end{aligned} \tag{44}$$

by a product construction. Both the closure under inverse homomorphism and intersection are well-known (Fülöp and Vogler, 2009). Translating packed forests is beneficial (Liu et al., 2009; Mi et al., 2008; Mi and Huang, 2008; Neubig and Duh, 2014). Parsers like BitPar and Egret can easily generate a packed representation of all parse trees in an unambiguous regular tree grammar. The set of derivations for a parse forest of the input sentence is then computed according to the following product construction:

Let Δ be the ruleset of the translation model. Recall that $\mathsf{in} \in \mathsf{HOM}$ with $\mathsf{in} : T_\Delta \to T_F$ and $\mathcal{T} = (Q_\mathcal{T}, I_\mathcal{T}, P_\mathcal{T})$ is a $\mathbb{R}$-weighted tree automaton over Δ. Let $\mathcal{P} = (Q_\mathcal{P}, I_\mathcal{P}, P_\mathcal{P})$ be a $\mathbb{R}$-weighted tree automaton over F representing the input parse forest, i.e., $P_F\langle\mathbf{f}\rangle^{\lambda_1} = L_\mathcal{P}$. We construct the $\mathbb{R}$-weighted tree automaton $\mathcal{A} = (Q_\mathcal{T} \times Q_\mathcal{P}, I, P)$ over Δ such that

- $I((p,q)) = I_\mathcal{P}(q) \cdot I_\mathcal{T}(p)$ for every $q \in Q_\mathcal{P}$ and $p \in Q_\mathcal{T}$;
- for every

$$\begin{gathered}\delta^{(k)} \in \Delta,\\ q, q_1, \ldots, q_k \in Q_\mathcal{P} \text{ and}\\ p, p_1, \ldots, p_k \in Q_\mathcal{T},\end{gathered}$$

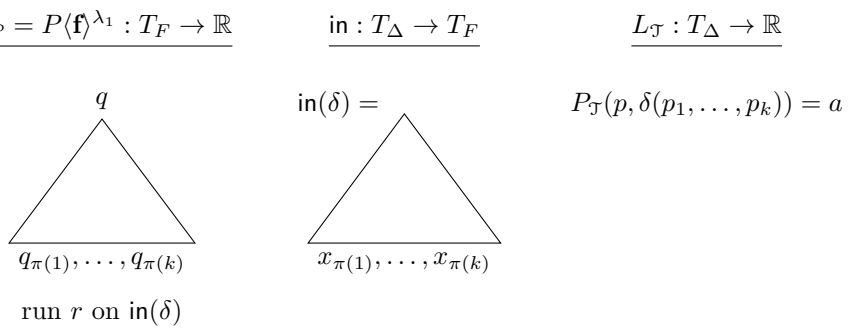


Figure 24. Product construction

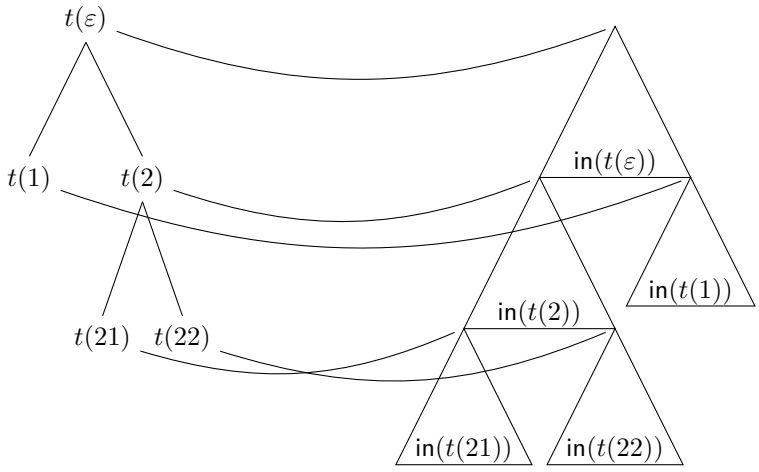


Figure 25. Run mapping

let R be the set of all runs r of $\mathcal{P}$ on $\mathsf{in}(\delta)$ such that $r(\varepsilon) = q$ and for all $w \in \mathsf{pos}(\mathsf{in}(\delta))$ such that $\mathsf{in}(\delta)(w) = x_i$, we have $r(w) = q_i$. Then

$$P((p,q), \delta((p_1,q_1), \ldots, (p_k,q_k))) = \sum_{r \in R} \mathsf{wt}(r) \cdot P_{\mathcal{T}}(p, \delta(p_1, \ldots, p_k)) \,.$$

The construction is schematically depicted in Figure 24. The permutation $\pi : [k] \to [k]$ is only needed in this illustration because the order of x_i and p_i need not match.

Lemma 4
$L_{\mathcal{A}} = \mathsf{in}^{-1}(L_{\mathcal{P}}) \cdot L_{\mathcal{T}} = \mathsf{in}^{-1}(P\langle \mathbf{f} \rangle^{\lambda_1}) \cdot L_{\mathcal{T}}$.

Proof. Let $t \in T_\Delta$. We then have

$$L_{\mathcal{A}}(t) = \sum_{r \in \mathsf{runs}_{\mathcal{A}}(t)} I(r(\varepsilon)) \cdot \mathsf{wt}(r) = \sum_{\substack{r \in \mathsf{runs}_{\mathcal{A}}(t) \\ (p,q) = r(\varepsilon)}} I_{\mathcal{T}}(p) \cdot I_{\mathcal{P}}(q) \cdot \mathsf{wt}(r)\,.$$

First, note that $\mathsf{in}^{-1}(L_{\mathcal{P}})(t) = L_{\mathcal{P}}(\mathsf{in}(t))$ by definition. Also, there is a mapping of positions $\psi : \mathsf{pos}(t) \to \mathsf{pos}(\mathsf{in}(t))$ such that ψ maps every position w to the position of the root of the homomorphic image of $t(w)$. An example is shown in Figure 25. Let us define an equivalence relation $\equiv$ on the runs of $\mathcal{P}$ on t such that $r \equiv r'$ if $r(w) = r'(w)$ for every $w \in \mathsf{ran}(\psi)$.

We now define two mappings $\psi_{\mathcal{T}}$ and $\psi_{\mathcal{P}}$ mapping runs of $\mathcal{A}$ to runs of $\mathcal{T}$, and to equivalence classes of runs of $\mathcal{P}$, respectively. Then ϕ, defined by $\phi(r) = (\psi_{\mathcal{T}}(r), \psi_{\mathcal{P}}(r))$, will be a bijection. Let $r \in \mathsf{runs}_{\mathcal{A}}(t)$. Then we define $\psi_{\mathcal{T}}(r)$ and $\psi_{\mathcal{P}}(r)$ such that for every position $w \in \mathsf{pos}(r)$ with $r(w) = (p, q)$, we have $\psi_{\mathcal{T}}(r)(w) = p$ and every run $r' \in \psi_{\mathcal{P}}(r)$ has $r'(\psi(w)) = q$. With these mappings in place, it remains to prove that

$$\mathsf{wt}(r) = \mathsf{wt}(\psi_{\mathcal{T}}(r)) \cdot \sum_{r' \in \psi_{\mathcal{P}}(r)} \mathsf{wt}(r')\,,$$

for every $r \in \mathsf{runs}_{\mathcal{A}}(t)$, which will yield

$$\begin{aligned}
L_{\mathcal{A}}(t) &= \sum_{\substack{r \in \mathsf{runs}_{\mathcal{A}}(t) \\ (p,q) = r(\varepsilon)}} I_{\mathcal{T}}(p) \cdot I_{\mathcal{P}}(q) \cdot \mathsf{wt}(r) \\
&= \sum_{\substack{r \in \mathsf{runs}_{\mathcal{A}}(t) \\ (p,q) = r(\varepsilon)}} \left(I_{\mathcal{T}}(p) \cdot I_{\mathcal{P}}(q) \cdot \mathsf{wt}(\psi_{\mathcal{T}}(r)) \cdot \sum_{r' \in \psi_{\mathcal{P}}(r)} \mathsf{wt}(r') \right) \\
&= \sum_{r \in \mathsf{runs}_{\mathcal{T}}(t)} I_{\mathcal{T}}(r(\varepsilon)) \cdot \mathsf{wt}(r) \cdot \sum_{r \in \mathsf{runs}_{\mathcal{P}}(t)} I_{\mathcal{P}}(r(\varepsilon)) \cdot \mathsf{wt}(r) \\
&= L_{\mathcal{T}}(t) \cdot L_{\mathcal{P}}(t)\,.
\end{aligned}$$

Let $R_{\mathcal{P}}(w)$ be the set of sub-runs r' of $\mathcal{P}$ on $\mathsf{in}(t(w))$ such that $r'(\varepsilon) = \pi_2(r(w))$ and for all $w' \in \mathsf{pos}(\mathsf{in}(t(w)))$ such that $\mathsf{in}(t(w))(w') = x_i$, we have $r'(w') = \pi_2(r(wi))$. Then, for every run $r \in \mathsf{runs}_{\mathcal{A}}$ and run $r' \in \psi_{\mathcal{P}}(r)$, we obtain

$$\mathsf{wt}(r') = \prod_{w_i \in \mathsf{pos}(r)} \mathsf{wt}(r'_i)$$

such that $r'_i \in R_{\mathcal{P}}(w_i)$ coincides with r' on the segment covered by both, and by applying the law of distributivity, we obtain

$$\sum_{r' \in \psi_{\mathcal{P}}(r)} \mathsf{wt}(r') = \prod_{w \in \mathsf{pos}(r)} \sum_{r' \in R_{\mathcal{P}}(w)} \mathsf{wt}(r')\,.$$

Note that

$$\mathsf{wt}(\psi_{\mathcal{T}}(r)) = \prod_{\substack{w \in \mathsf{pos}(r) \\ k = \mathsf{rk}(r(w))}} P_{\mathcal{T}}(\psi_{\mathcal{T}}(r)(w), t(w)(\psi_{\mathcal{T}}(r)(w1), \ldots, \psi_{\mathcal{T}}(r)(wk)))\,.$$

We then obtain

$$\begin{aligned}
\mathsf{wt}(r) &= \prod_{\substack{w \in \mathsf{pos}(r) \\ k = \mathsf{rk}(r(w))}} \left(\sum_{r' \in R_{\mathcal{P}}(w)} \mathsf{wt}(r') \right) \\
&\qquad \cdot P_{\mathcal{T}}(\psi_{\mathcal{T}}(r)(w), t(w)(\psi_{\mathcal{T}}(r)(w1), \ldots, \psi_{\mathcal{T}}(r)(wk))) \\
&= \mathsf{wt}(\psi_{\mathcal{T}}(r)) \cdot \prod_{w \in \mathsf{pos}(r)} \sum_{r' \in R_{\mathcal{P}}(w)} \mathsf{wt}(r') \\
&= \mathsf{wt}(\psi_{\mathcal{T}}(r)) \cdot \sum_{r' \in \psi_{\mathcal{P}}(r)} \mathsf{wt}(r')
\end{aligned}$$

by construction of $\mathcal{A}$. □

In order to avoid computing the whole product automaton, we only explore reachable, non-zero-weighted productions. Furthermore, we store the source model of the translation rules in a hypertree structure (Zhang et al., 2009). In this structure, trees are represented as strings, and prefixes are shared. This allows us to compute the matching portions of the homomorphic tree images of all translation rules simultaneously.

To this end, we define, for every $\mathsf{in}(\delta)$, a monadic tree (string) over sequences of strings over alphabet symbols via an intermediate representation that extends the position set of $\mathsf{in}(\delta)$ by filling short paths with $\perp$ placeholders. We define the extended position mapping $P : \mathbb{N}^* \to F \cup X \cup \{\perp\}$ as a partial mapping such that $P(p) = \mathsf{in}(\delta)(p)$ for all $p \in \mathsf{pos}(\mathsf{in}(\delta))$; and for every $p \in \mathbb{N}^*$, if $P(p)$ is defined and $p1 \notin \mathsf{pos}(\mathsf{in}(\delta))$, if there exists $p' \in \mathsf{pos}(\mathsf{in}(\delta))$ with $|p'| > |p|$, then $P(p1) = \perp$.

Then a string $(w_0, w_1, \ldots, w_n)$ can be defined such that $w_0 = (P(\varepsilon))$, $n = \max\{|p| \mid p \in \mathsf{dom}(P)\}$ and for all $i \in [n]$, we let $w_i = (c(p_1), \ldots, c(p_k))$ where $(p_1, \ldots, p_k) = \{p \in \mathsf{dom}(P) \mid |p| = i - 1\}^{\leq}$ and $c(p) = P(\{(p.j) \mid j \in \mathbb{N}, p.j \in \mathsf{dom}(P)\}^{\leq})$. Intuitively, this arranges labels with the same distance from the root in a sequence, concatenating immediate siblings. Figure 26 shows an example homomorphic tree image $\mathsf{in}(\delta)$ and its conversion into a string depicted as a monadic tree. We can then build a prefix-sharing tree ("trie") from the string representations of $\mathsf{in}(\delta)$ for every $\delta \in \Delta$ to speed up matching.

We can further exploit the structure of $\mathcal{T}$ to delay assigning weights. Recall the rule extraction and scoring procedure from Section 5.1.2. We obtained an unweighted tree automaton $\mathcal{T}'$ and a family of feature functions represented as homomorphisms $(\phi_i \mid i \in [n])$ given by $(w_i \mid i \in [n])$ with $w_i : \Delta \to \mathbb{R}$ extended to weighted languages $\phi_i : T_\Delta \to \mathbb{R}$ with $\phi_i(t) = \prod_{p \in \mathsf{pos}(t)} w_i(t(p))$ such that

$$\mathcal{T} = \mathcal{T}' \cdot (\phi_1 \cdots \phi_j)^{\lambda_1} \cdot (\phi_{j+1} \cdots \phi_k)^{\lambda_2} \cdot (\phi_{k+1} \cdots \phi_n)^{\lambda_3}\,. \tag{45}$$

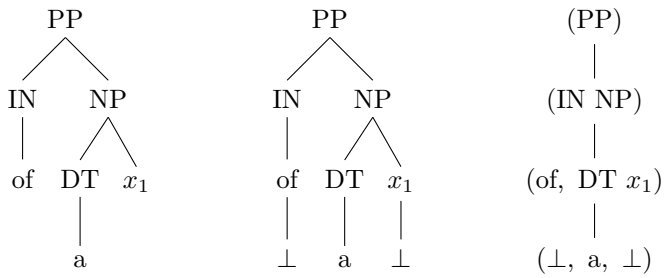


Figure 26. Transformation of a tree into a string

Instead of computing $\mathcal{A}$ such that $L_{\mathcal{A}} = \mathsf{in}^{-1}(L_{\mathcal{P}}) \cdot L_{\mathcal{T}} = \mathcal{D}(\mathbf{f}) \cdot \mathcal{F} \cdot \mathcal{T}$, we can compute $\mathcal{A}'$ such that $L_{\mathcal{A}'} = \mathsf{in}^{-1}(L_{\mathcal{P}}) \cdot \mathcal{T}'$, and compute $L_{\mathcal{A}'} \cdot (\phi_1 \cdot \dots \cdot \phi_j)^{\lambda_1} \cdot (\phi_{j+1} \cdot \dots \cdot \phi_k)^{\lambda_2} \cdot (\phi_{k+1} \cdot \dots \cdot \phi_n)^{\lambda_3}$ later.

The advantage is that the derivation forest can be stored and evaluated symbol by symbol, and does not need to be recomputed when production weights or the parameter vector λ change. We will explain in Section 5.4 how to optimize weights without changing the unweighted structure of $\mathcal{T}'$.

5.2.2 k-best derivations

So far, we have shown how to compute $\mathcal{D}(\mathbf{f}) \cap \mathcal{F} \cap \mathcal{T}$. It remains to integrate $\mathcal{E}$ to obtain Equation (42). At the heart of our implementation is a performant algorithm to find the best-scoring trees in a weighted forest of derivations. While finding the single best-scoring derivation tree is relatively easy, care has to be taken to produce a correct list of k best-scoring derivation trees for a given k, efficiently. Huang and Chiang (2005)[19] present an algorithm that lazily computes a k-best list, starting from the *Viterbi derivation*.

First, we need to define the notion of k-best list. We will define it generally because it is a useful concept. Let S be a set and $f : S \to \mathbb{R}$ be a mapping such that $\max \mathsf{ran}(f)$ exists. The mapping f may be partial, but $f(n-1)$ needs to be defined for every $n > 1$ if $f(n)$ is defined. Then $f^{\geq} : \mathbb{N}^+ \to S$ can be defined as an injective mapping such that $n \leq n'$ if and only if $f(f^{\geq}(n)) \geq f(f^{\geq}(n'))$ for every $n, n' \in \mathbb{N}^+$. Intuitively, $f^{\geq}$ lists the elements of S according to the value that f assigns them. Just like for 'arg max', the specific choice of mapping does not matter. For every $k \in \mathbb{N}^+$, we call the sequence $(f^{\geq}(1), \dots, f^{\geq}(k))$ a *k-best list*.

Let $\mathcal{A} = (Q, I, P)$ be an acyclic weighted tree automaton representing a derivation forest, i.e., $L_{\mathcal{A}} = \mathsf{in}^{-1}(L_{\mathcal{P}}) \cap L_{\mathcal{T}} = \mathcal{D}(\mathbf{f}) \cap \mathcal{F} \cap \mathcal{T}$. The set of *derivations* of $\mathcal{A}$ is the set $D_{\mathcal{A}}$ of pairs (t, r) of a tree $t \in T_\Delta$ and a run $r : \mathsf{pos}(t) \to Q$.

[19] The original paper contained a mistake which has been corrected. The revised version can be obtained from the author's homepage: http://www.cis.upenn.edu/~lhuang3/huang-iwpt-correct.pdf.

The set of derivations of a state q is

$$D_{\mathcal{A}}(q) = \{(t, r) \in D_{\mathcal{A}} \mid r(\varepsilon) = q\}\,. \tag{46}$$

Since by construction of $\mathcal{A}$, every ranked symbol $\delta \in \Delta$ has at most one corresponding non-zero production, and therefore every tree $t \in T_\Delta$ has at most one non-zero run, finding the best tree can be replaced by finding the best derivation. Following Huang and Chiang (2005), we assume that for every state q, the set of derivations is ordered by

$$(t, r) \leq (t', r') \iff \mathsf{wt}(r) \leq \mathsf{wt}(r')\,,$$

and therefore we can define $\mathbf{D}_{\mathcal{A}}(q)$ such that $\mathbf{D}_{\mathcal{A}}(q) = f^{\geq}$ for $f : D_{\mathcal{A}}(q) \to \mathbb{R}$ with $f(t, r) = \mathsf{wt}(r)$.

A derivation (t, r) with $t = \delta^{(0)}(t_1, \ldots, t_k)$ such that $P(q, \delta(q_1, \ldots, q_k)) \neq 0$ can then be represented by a pair $(\delta, (j_1, \ldots, j_k)) \in \Delta \times (\mathbb{N}^+)^k$ such that $\mathbf{D}_{\mathcal{A}}(q_i)(j_i) = (t|_i, r_i)$ for every $i \in [k]$. In the terminology of Huang and Chiang (2005), this (recursive) representation is called a *derivation with backpointers* (each j_i is a backpointer to the i-th best derivation of q_i), and obviously the set of derivations with backpointers is ordered by $\leq$ as well. Then the Viterbi derivation of a state q can be given as a derivation with backpointers by:

$$V_{\mathcal{A}}(q) = \mathbf{D}_{\mathcal{A}}(q)(1) \in \max\left\{(\delta, \underbrace{(1, \ldots, 1)}_{m \text{ components}}) \mid \delta \in \Delta^{(m)}\right\},$$

i.e., by choosing the optimal subderivations and then maximizing the root symbol $\delta \in \Delta$. Similarly, the problem of finding the best k derivations can be reduced to an operation that picks the maximum derivation from a set of candidates, and an operation that assembles these candidates from subderivations, represented by backpointers. This strategy is correct as long as the optimal substructure property is fulfilled (it is fulfilled here because of the properties of $\mathbb{R}$):

$$a_1 \leq b_1, \ldots, a_k \leq b_k \implies f(a_1, \ldots, a_k) \leq f(b_1, \ldots, b_k)\,.$$

Here, f is the operation that assembles candidates from subderivations, and it is just multiplication of real numbers; as soon as we pick an inferior subderivation, the entire derivation will become inferior. For the k-best case, we notice that we can find the k-th best derivation by computing for all productions, and the states involved, the k best derivations:

$$\mathbf{D}_{\mathcal{A}}(q)(k) \in \left\{(\delta, (j_1, \ldots, j_k)) \mid \delta \in \Delta^{(m)}, j_1, \ldots, j_k \in [k]\right\}.$$

The authors then proceed how to use various optimization techniques to minimize runtime, such as a lazy enumeration of k-best lists for individual symbols $\delta \in \Delta$. As an example, consider Figure 27, depicting the lazy enumeration of a k-best list in two dimensions, i.e., for a binary production. Starting from the best derivation (marked with 1), the set of candidates, i.e., derivations that should be explored next, is shown with dashed lines. In the first step, these are

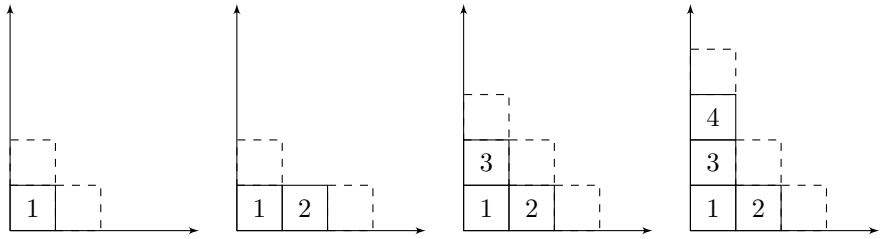


Figure 27. Enumeration of a k-best list

only those derivations where one of the child derivations is replaced by the next best derivation in that position. It is guaranteed that no other derivation is better than the best among these. After identifying the next item on the k-best list, it is replaced by its neighbors, yielding a new set of candidates. This is continued whenever a new item needs to be generated.

For a more detailed presentation of the algorithm, the reader is referred to Huang and Chiang (2005). The k-best list can then be mapped to trees in the output language by applying the tree m-morphism. Obviously (since k implies a finite set), the resulting tree language is again regular. It can therefore be represented by a regular tree automaton. It has been shown (May and Knight, 2006a) that alternative derivations representing the same tree should be merged (and their scores added). This can be achieved by determinization of the tree automaton, or by simple "crunching", i.e., enumerating the trees and adding the scores for each equivalence class. Furthermore, the yield language (which is context-free) can be obtained for language model scoring. In addition, this merges trees that have the same yield, i.e., alternative structures for the same surface string.

5.3 Language model scoring

The language model's task is to assign a score (probability) to a sentence in the target language. Our forest approach enables us to estimate the probability of a syntactic tree, but also allows the more popular n-gram language models. Recall Equation (20):

$$\begin{aligned}\Pr(u) &= \Pr(u)^{\lambda_1} \cdot \Pr(u)^{\lambda_2} \cdot \Pr(u)^{\lambda_3} \\ &= \Big(\underbrace{\Pr(f, \mathbf{f})}_{\text{parser}} \cdot \underbrace{\Pr(u|f)}_{\text{forward}}\Big)^{\lambda_1} \cdot \underbrace{\Pr(u)}_{\text{symm.}}{}^{\lambda_2} \cdot \left(\underbrace{\Pr(u|e)}_{\text{backward}} \cdot \underbrace{\frac{\Pr(e, \mathbf{e})}{\Pr(\mathbf{e})}}_{\text{synLM}} \cdot \underbrace{\Pr(\mathbf{e})}_{\text{LM}}\right)^{\lambda_3}.\end{aligned}$$

We consider two approximations:

$$\mathcal{E}_1(u) \approx \frac{\Pr(e, \mathbf{e})}{\Pr(\mathbf{e})} \quad \text{and}$$
$$\mathcal{E}_2(u) \approx \Pr(\mathbf{e})\,.$$

$\mathcal{E}_1$ is a syntactic language model because it makes use of the parse tree e, while $\mathcal{E}_2$ only scores the sentence $\mathbf{e}$, and is usually realized by an n-gram language model.

5.3.1 Syntactic language models

Let us approximate $\Pr(e, \mathbf{e}) \cdot \Pr(\mathbf{e})^{-1}$ using a parser $P_E : T_E \to [0, 1]$. Recall that $P_E\langle\mathbf{e}\rangle = \mathsf{yield}^{-1}(\mathbf{e}) \cap P_E$, which is a good approximation of $\Pr(e, \mathbf{e})$ as explained earlier. Furthermore, we can use the language weight of $P_E\langle\mathbf{e}\rangle$ to approximate $\Pr(\mathbf{e})$:

$$\sum_{t \in T_E} P_E\langle\mathbf{e}\rangle(t) \approx \Pr(\mathbf{e})\,.$$

The language weight of $P_E\langle\mathbf{e}\rangle$ can be efficiently computed using the inside weights of all states of $P_E\langle\mathbf{e}\rangle$. Since the notion of inside weights is useful in other contexts as well, we give a general description.

Let $\mathcal{G} = (Q, I, P)$ be a weighted tree automaton over Δ. Then the *inside weight* $\beta_\mathcal{G}(q)$ of a state $q \in Q$ is defined by:

$$\beta_\mathcal{G}(q) = \sum_{(q, \delta(q_1, \ldots, q_k)) \in \mathsf{dom}(P)} \left(P(q, \delta(q_1, \ldots, q_k)) \cdot \prod_{i=1}^{k} \beta_\mathcal{G}(q_i) \right) . \tag{47}$$

Intuitively, $\beta_\mathcal{G}(q)$ is the sum of all trees generated by state q, and therefore

$$\sum_{q \in Q} I(q) \cdot \beta_\mathcal{G}(q) = \sum_{t \in T_\Delta} L_\mathcal{G}(t)\,.$$

Note that the recursive definition does not cause any problems if $\mathcal{G}$ has no cycles, i.e., for every non-zero weighted run r of $\mathcal{G}$, $r(p) = r(p')$ and $p \sqsubseteq p'$ implies $p = p'$ for all $p, p' \in \mathsf{pos}(r)$. Even if $\mathcal{G}$ has cycles, as long as every cycle has a weight less than 1, the inside weight is always well-defined (Graehl et al., 2008).

5.3.2 n-gram language models

Consider the probability space $(\mathbf{E}, 2^\mathbf{E}, \Pr)$. The elementary events are English sentences (i.e., strings of English words, whether "grammatical" or not). We are interested in the probability of the event $\{w\}$, the occurrence of the sentence $w = w_1 \cdots w_m \in \mathbf{E}$. Let us assume a family of random variables $X_n : \mathbf{E} \to E \cup \{\bot\}$ for all $n \in \mathbb{N}^+$. The symbol $\bot$ is used to denote the end of w, i.e., $X_i(w) = \bot$ for all $i > m$. The joint probability of w, i.e., the joint probability

of $X_1 = w_1, X_2 = w_2, \ldots, X_m = w_m$, can be decomposed, using conditional probabilities, as follows:

$$\begin{aligned}\Pr(w) &= \Pr(w_1) \cdot \Pr(w_2|w_1) \cdot \Pr(w_3|w_1, w_2) \cdot \ldots \cdot \Pr(w_m|w_1, \ldots, w_{m-1}) \\ &= \Pr(w_1) \cdot \prod_{i=2}^{m} P\left(w_i \middle| w_1^{i-1}\right) .\end{aligned} \tag{48}$$

To account for the $\perp$ symbol marking the end of the sentence, we decompose the joint probability of $w_1, \ldots, w_m, \perp$ in the same fashion:

$$\Pr(w\perp) = \Pr(w_1) \cdot \left(\prod_{i=2}^{m} P\left(w_i \middle| w_1^{i-1}\right)\right) \cdot \Pr(X_{m+1} = \perp \,|\, w) . \tag{49}$$

However, in practice, there is not enough data to estimate all the conditional probabilities that are used in Equation (48). A common simplification is to impose an upper bound n on the length of the history used in the conditional probabilities. The underlying assumption, also called *Markov assumption* (after mathematician Andrey Markov), is:

$$P\left(w_i \middle| w_1^{i-1}\right) \approx P\left(w_i \middle| w_{i-(n-1)}^{i-1}\right) . \tag{50}$$

Using Equations (49) and (50), we can approximate the probability of a sentence by multiplying n-gram probabilities as follows:

$$\Pr(w\perp) \approx \left(\prod_{i=1}^{m} P\left(w_i \middle| w_{i-(n-1)}^{i-1}\right)\right) \cdot \Pr(\perp | w_{m-(n-1)}^{m}). \tag{51}$$

This means that the probabilities of infinitely many sentences can be approximated by a finite number of probabilities (since there are finitely many n-grams over every alphabet), which can more easily be observed.

We can then proceed with maximum likelihood estimation. Let C be the training corpus. Then

$$\Pr(w|w') \approx \frac{\#_C(w'w)}{\#_C(w)} \tag{52}$$

assigns the probabilities based on counting n-grams in C. To account for the probability of n-grams that have not been observed, smoothing techniques are used. This however is outside of the scope of this dissertation.

All of the above calculations can be realized in the realm of weighted regular languages (Hanneforth and Würzner, 2009). Let us therefore assume that, even if it is not typically represented as such, a language model is simply a $\mathbb{R}$-weighted regular language over E, in other words a function $\mathsf{LM} : \mathbf{E} \to [0, 1]$. We will present two methods for integrating a language model. Both are theoretically sound and guarantee the optimal solution, but only one is currently feasible and has been implemented.

5.3.3 Integration by product construction

In the previous section, we concluded that a language model is essentially a weighted regular string language $\mathsf{LM} : \mathbf{E} \to [0,1]$. It is well-known that $\mathsf{yield}^{-1}(L)$ is a regular tree language for every regular string language L, even in the weighted case (Bar-Hillel construction, see Maletti and Satta (2009)). Given that out is a linear and non-deleting m-morphism, it is also an input-ε-free multi bottom-up tree transducer, and therefore its inverse application preserves recognizability (Maletti, 2011b). We can therefore represent $\mathsf{out}^{-1}(\mathsf{yield}^{-1}(\mathsf{LM})) = \mathcal{E}_2$ as a regular tree language, and thus integrate the language model seamlessly.

However, due to the size of typical language models (trained on millions, or even billions of sentences), the product construction $\mathcal{D}(\mathbf{f}) \cdot \mathcal{F} \cdot \mathcal{T} \cdot \mathcal{E}_1^{\lambda_3} \cdot \mathcal{E}_2^{\lambda_3}$ is not feasible because of combinatorial explosion. In the next section, we present an approximation method where we can sometimes establish a guarantee that the solution is exact, and can iterate until an exact solution has been found.

5.3.4 Exact rescoring

The computational cost of the product construction can be avoided by switching the order of the k-best and the LM function applications. However, it is easy to see that this sacrifices exactness if the best candidate is worse than the k-th candidate using translation model scores only. Let $F = \mathcal{D}(\mathbf{f}) \cdot \mathcal{F} \cdot \mathcal{T} \cdot \mathcal{E}_1^{\lambda_3}$ be a cycle-free forest of translation candidates. Note that F induces an ordering on T_Σ, and we can define an injective mapping $\mathcal{F} : \mathbb{N}^+ \to T_\Sigma$ such that $n \leq n'$ implies $F(\mathcal{F}(n)) \geq F(\mathcal{F}(n'))$. Then for every $k \in \mathbb{N}^+$, the set $\mathcal{F}([k])$ is a k-best list (which can be obtained using the algorithms discussed in Section 5.2.2). Assuming that

$$\mathcal{E}_2(t) = \mathsf{LM}(\mathsf{yield}(\mathsf{out}(t)))$$

for every $t \in \mathcal{F}([k])$, the best candidate in the reordered k-best list is

$$\begin{aligned}
\hat{t} &= \operatorname*{arg\,max}_{t \in \mathcal{F}([k])} F(t) \cdot \mathsf{LM}(\mathsf{yield}(\mathsf{out}(t)))^{\lambda_3} \\
&= \operatorname*{arg\,max}_{t \in \mathcal{F}([k])} (\mathcal{F} \cdot \mathcal{T} \cdot \mathcal{E}_1^{\lambda_3})(t) \cdot \mathcal{E}_2^{\lambda_3}(t) \\
&= \operatorname*{arg\,max}_{t \in \mathcal{F}([k])} L(t)
\end{aligned}$$

which is what we expect from decoding (recall the Viterbi approximation in Equation (42)). However, $\mathcal{F}([k]) \subseteq \mathcal{D}(\mathbf{f})$, and therefore

$$\hat{t} = \operatorname*{arg\,max}_{t \in \mathcal{F}([k])} L(t) \text{ does not imply } \hat{t} = \operatorname*{arg\,max}_{t \in \mathcal{D}(\mathbf{f})} L(t)$$

i.e., we do not have any guarantee that $\hat{t}$ is optimal.

Lemma 5
There is $n' \in \mathbb{N}$ such that for every $n \geq n'$,

$$\operatorname*{arg\,max}_{t \in \mathcal{F}([n])} L(t) = \operatorname*{arg\,max}_{t \in \mathcal{D}(\mathbf{f})} L(t)\,.$$

Proof. Let $\hat{t} = \arg\max_{t \in \mathcal{D}(\mathbf{f})} L(t)$. Since both $\mathsf{ran}(\mathsf{LM}) \subseteq [0,1]$ and $\mathsf{ran}(F) \subseteq [0,1]$, we know that

$$L(t) = F(t) \cdot \mathcal{E}_2^{\lambda_3}(t) \leq F(t)$$

for every tree $t \in T_\Delta$ (because $ab \leq a$ for every $a, b \in [0,1]$). Therefore, $F(t) < L(\hat{t})$ implies $L(t) < L(\hat{t})$, and because F is cycle-free, there exists n such that $F(t') < L(\hat{t})$ for all $t' \notin \mathcal{F}([n])$.

□

Intuitively, this means that we can increase the size of the k-best list until the worst item on the k-best list *before* factoring in the language model is already worse than the best item *after* factoring in the language model. From then on, no candidate can ever become better, because scores always get worse by factoring in the language model.

5.4 Tuning, evaluation, model optimization

Recall that the parameters of a log-linear model (Section 3.3.2), and therefore also the parameters of the models obtained by the methods described in the previous section of this chapter, greatly influence the decoding outcome. We can optimize translation results by optimizing the parameter vector. Let us assume we have a source language text $S = (s_1, \ldots, s_\ell) \in \mathbf{F}^*$ and a reference translation $T = (t_1, \ldots, t_\ell) \in \mathbf{E}^*$. We choose[20]

$$\hat{\lambda} = \arg\min_{\lambda} \sum_{i=1}^{\ell} \Lambda(D_\lambda(s_i), t_i) \tag{53}$$

where $D_\lambda(s_i)$ is the best-scoring translation of s_i with parameter vector λ, to optimize the parameter vector according to the *loss function* $\Lambda : \mathbf{E}^* \times \mathbf{E}^* \to \mathbb{R}$. For instance, we can use a loss function based on the BLEU metric, explained in the next section.

5.4.1 Evaluation metrics

Evaluation of machine translation is the task of assessing the quality of the translations proposed by a given MT system in terms of aspects like *adequacy* and *fluency*. A translation is *adequate* if it conveys the same meaning as the source text. It is not adequate if information was added, lost, or distorted in the translation process. A translation is *fluent* if it is grammatical and idiomatic in the target language. Another way to measure the quality of machine translation is in terms of the post-editing effort needed when used in CAT (computer-aided translation). The evaluation scores can be used to compare the quality of different MT systems, or to track and optimize the quality of an MT system during development.

[20] We define 'arg min' analogously to 'arg max'.

An *automatic evaluation metric* is a function that assigns a score to a candidate translation measuring its quality. Depending on the metric, a lower score can mean higher quality, or vice versa. There are several reasons to use automatic evaluation of machine translation:

- *Efficiency:* Automatic evaluation can be performed in very short time.
- *Cost:* Automatic evaluation is cheap.
- *Easy integration:* Evaluation can easily be integrated in workflows, e.g., parameter tuning (see Section 5.4.2). Depending on the properties of the metric, a local maximum may or may not be reached.

However, there are also drawbacks. In particular, there is no guarantee that an automatic evaluation metric will correlate with human judgment. Optimizing for automatic evaluation scores might or might not improve translation quality.

The most widely used metric is BLEU (**Bil**ingual **E**valuation **U**nderstudy), proposed by Papineni et al. (2002). We will limit ourselves to BLEU as it is simple to present, easy to implement and gives good correlation with human evaluation. However, we will briefly discuss shortcomings of BLEU and suggested improvements at the end of this section. BLEU measures n-gram precision, i.e., how many of the n-grams of a given candidate translation appear in the *reference translation*. BLEU was originally designed to be used with multiple reference translations, but can also be used with a single reference translation. As this is the most common usage, we describe only the single-reference variant.

We now formalize BLEU. To this end, let A be an alphabet. The set of n-grams of $a \in A^*$ is defined by:

$$\mathsf{grams}_n(a) = \{v \in A^n \mid \exists u, w \in A^* \text{ such that } uvw = a\} \tag{54}$$

Let $T = (T_1, \ldots, T_m)$ be a text of length m called input text. Let $C = (C_1, \ldots, C_m)$ be a text of candidate translations and $R = (R_1, \ldots, R_m)$ a text of reference translations. We define the *clipped n-gram precision* of a candidate translation as follows:

$$\mathsf{prec}_n(C, R) = \sum_{i=1}^{m} \frac{\sum_{w \in \mathsf{grams}_n(C_i) \cap \mathsf{grams}_n(R_i)} \min\{\#_{C_i}(w), \#_{R_i}(w)\}}{\sum_{w \in \mathsf{grams}_n(C_i)} \#_{C_i}(w)} \,. \tag{55}$$

The precision measure is "clipped" in the sense that a candidate translation cannot be awarded for the same n-gram more often than it appears in the reference translation. We define $\mathsf{bp}(C, R)$, the *brevity penalty* which punishes translations that are shorter than the reference[21], and a $\mathbb{N}$-indexed family of functions $(\mathsf{bleu}_N(C, R) \mid N \in \mathbb{N})$, as follows:

$$\begin{aligned} \mathsf{bp}(C, R) &= \min\left\{1, \exp\left(1 - \frac{|R|}{|C|}\right)\right\} \\ \mathsf{bleu}_N(C, R) &= \mathsf{bp}(C, R) \cdot \prod_{n \in [N]} \mathsf{prec}_n(C, R)^{1/N} \end{aligned} \tag{56}$$

[21] The brevity penalty is introduced to avoid extremely short translations that contain very frequent n-grams to obtain a high precision score.

Essentially, bleu_N computes the geometric mean of the n-gram scores for $n = \{1, \ldots, N\}$. A common choice for N is 4, when the resulting metric is also called BLEU-4. More sophisticated metrics can be obtained by replacing the uniform n-gram parameter $1/N$ by different parameters to assign different weight to different types of n-grams, but this is the most widely used formulation.

Example 14 (*Calculating* BLEU-4.)
Let us assume the following reference translation and two candidate translations:

Reference translation	`the cat sat on the mat .`
Candidate translation A	`this` `cat sat` `upon a` `mat .`
Candidate translation B	`the cat` `on the mat .`

The maximal n-grams matching the reference translation are highlighted. Since every bigram match implies two unigram matches, every trigram match implies two bigram matches and three unigram matches, and every 4-gram match implies two trigram matches, three bigram matches and four unigram matches, we arrive at the following numbers:

	1-grams	2-grams	3-grams	4-grams	BP	BLEU-4
A	4 out of 7	2 out of 6	0 out of 5	0 out of 4	$e^{1-7/7} = 1$	0
B	6 out of 6	4 out of 5	2 out of 4	1 out of 3	$e^{1-7/6} \approx .85$	$\approx .512$

Candidate translation A has a BLEU-4 score of 0 because it does not contain any 4-gram match. Candidate translation B scores a geometric mean of n-gram matches of

$$(6/6 \cdot 4/5 \cdot 2/4 \cdot 1/3)^{1/4} \approx .604.$$

Taking the brevity penalty of $e^{1-7/6}$ into account, its BLEU-4 is approximately .512, or 51.2 %.

BLEU has been criticized on the grounds that "recall plays a more important role than precision in obtaining high-levels of correlation with human judgments" (Banerjee and Lavie, 2005), but a detailed discussion is out of the scope of this dissertation. Other automated metrics have been proposed that take into account aspects like lexical variation, paraphrases and inflected word forms. For a comparison of automated metrics and their correlation with the assessment of translation quality using *human judgment*, see Stanojević et al. (2015).

5.4.2 Minimum Error Rate Training

A common optimization technique is *Minimum Error Rate Training* (MERT), introduced by Och (2003). The idea of MERT is simple: An initial parameter vector is chosen. A predefined set of source language sentences is translated. For each sentence, a k-best list of translation candidates with their feature function scores is stored.

When the parameter vector is changed, the k-best lists will be reordered. Thus, the parameter vector can be optimized such that the resulting BLEU score achieved by the translations on top of the k-best lists is the highest. This process is then repeated with the newly obtained parameter vector until convergence.

Kumar et al. (2009) describe a method to perform parameter tuning on *hypergraphs*. In their setting, hypergraphs encode the hypotheses of an SCFG SMT system. A hypergraph is a pair $H = (V, E)$ consisting of a vertex set V and a set of hyperedges $E \subseteq V^* \times V$. Thus, each hyperedge connects a head vertex with a sequence of tail vertices. Furthermore, hyperedges are labeled with rules from the SCFG ruleset. In this way, the rules used in the derivation of a hyperpath can be recovered, and the hyperpath can be scored. On this representation of the hypothesis space instead of just k-best lists, algorithms like MERT and MBR (Minimum Bayes-Risk, Kumar and Byrne (2004)) can be used.

It is easy to see that hypergraphs are just regular tree grammars, and each hyperpath in the hypergraph is a derivation tree in our terminology. Therefore, it should be trivial to adapt hypergraph tuning algorithms to our framework.

5.4.3 EM training

So far, we have assigned weights to rules purely based on their shape by using relative frequency counts. Now, we will assign weights to rules based on their relative usefulness in explaining a test corpus. (Graehl et al., 2008) describe how to train an extended top-down tree transducer using the *Expectation-Maximization* (EM) algorithm. In principle, this can be done for any formalism whose derivation languages are weighted regular tree languages, in particular every bimorphism. EM training (Dempster et al., 1977) maximizes the corpus likelihood by repeatedly estimating all possible ways of generating the training corpus given the current parameters and assigning values based on counts to the parameters, and then renormalizing. Each iteration increases the corpus likelihood until a local maximum is reached.

Let $\mathcal{G} = (Q, I, P)$ be a weighted tree automaton over Δ. Recall the inside weight of a state $q \in Q$, repeated from Equation (47):

$$\beta_{\mathcal{G}}(q) = \sum_{(q,\delta(q_1,\ldots,q_k))\in \mathsf{dom}(P)} \left(P(q, \delta(q_1, \ldots, q_k)) \cdot \prod_{i=1}^{k} \beta_{\mathcal{G}}(q_i) \right) .$$

The *outside weight* $\alpha_{\mathcal{G}}(q)$ of a state $q \in Q$ is defined by:

$$\alpha_{\mathcal{G}}(q) = I(q) + \sum_{\substack{(p,\delta(p_1,\ldots,p_k))\in \mathsf{dom}(P) \\ w=P(p,\delta(p_1,\ldots,p_k))}} w \cdot \alpha_{\mathcal{G}}(p) \cdot \sum_{\substack{j\in[k] \\ q=p_j}} \prod_{\substack{i\in[k] \\ i\neq j}} \beta_{\mathcal{G}}(p_i) . \qquad (57)$$

Since $\mathbb{R}$ has multiplicative inverses, division may be used instead of excluding $i \neq j$. Intuitively speaking, $\alpha_{\mathcal{G}}(q)$ is the weight of trees that embed q, without the weight of that particular q-generated subtree.

If $\mathcal{G}$ is acyclic, computation of α and β is straightforward using the recursive definition and memoization to prevent re-computation, or by establishing a topological order on Q. If $\mathcal{G}$ is acyclic, and $I(q_0) = 1$ and $I(Q - \{q_0\}) \subseteq \{0\}$, then Equation (57) can be rewritten to $\alpha_{\mathcal{G}}(q_0) = 1$ and

$$\alpha_{\mathcal{G}}(q) = \sum_{(p,\delta(p_1,\ldots,p_k))\in \mathsf{dom}(P)} P(p,\delta(p_1,\ldots,p_k)) \cdot \alpha_{\mathcal{G}}(p) \cdot \sum_{\substack{j\in[k]\\ q=p_j}} \prod_{\substack{i\in[k]\\ i\neq j}} \beta_{\mathcal{G}}(p_i)\,. \quad (58)$$

Finally, we can define the sum of weights of trees using a particular production $\rho = (p, \delta(p_1, \ldots, p_k)) \in \mathsf{dom}(P)$ where $w = P(\rho)$ as follows:

$$\gamma_{\mathcal{G}}(\rho) = \alpha_{\mathcal{G}}(p) \cdot w \cdot \prod_{i=1}^{k} \beta_{\mathcal{G}}(p_i)\,. \quad (59)$$

Recall our translation bimorphism $\mathcal{D} = (\mathsf{in}, L, \mathsf{out})$. Let us now assume that $\mathcal{D} = (D_i \mid i \in [m])$ is a family of derivation forests over Δ for a parallel corpus $\mathcal{C} = ((\mathbf{f}_i, \mathbf{e}_i) \mid i \in [m])$, i.e., every D_i can be represented as a weighted tree automaton (Q_i, I_i, P_i) over Δ that explains the sentence pair $(\mathbf{f}_i, \mathbf{e}_i)$, i.e., $t \in \mathsf{supp}(D_i)$ implies $\mathsf{yield}(\mathsf{in}(t)) = \mathbf{f}_i$ and $\mathsf{yield}(\mathsf{out}(t)) = \mathbf{e}_i$. (That means that $D_i = \mathcal{D}_{(\mathbf{f}_i,\mathbf{e}_i)}$).

Recall the equivalence relations $\sim$ defined on Δ in Section 5.1.2. In particular, consider the equivalence relation defined by $\delta \sim \delta'$ if and only if $q = q'$ for every $\delta = (q, (q_1, \ldots, q_k), f, e, a)$ and $\delta' = (q', (q'_1, \ldots, q'_{k'}), f', e', a')$. However, the technique explained below is applicable to any equivalence relation on Δ.

Furthermore, note that the production weights in every D_i only depend on the symbol δ generated because there is at most one non-zero production for every $\delta \in \Delta$ in L. Therefore, $P_i(q, \delta(q_1, \ldots, q_k)) = P_L(\delta)$. We will use the EM algorithm to optimize P_L in order to maximize the *corpus likelihood*

$$\sum_{i=1}^{m} \sum_{t\in T_\Delta} L_{D_i}(t)\,. \quad (60)$$

We will now compute the weighted count of every symbol in Δ by summing fractional counts. Let us therefore define

$$\mathsf{ex}(\mathcal{D})(\delta) = \sum_{i=1}^{m} \frac{\sum_{\rho=(p,\delta(p_1,\ldots,p_k))} \gamma_{D_i}(\rho)}{\sum_{q\in Q} I(q) \cdot \beta_{D_i}(q)} \quad (61)$$

for every $\delta \in \Delta$. This is called the *expectation step*. The *maximization step* is defined as a straightforward normalization, for any $P : \Delta \to \mathbb{R}$, by $\mathsf{norm}(P) : \Delta \to \mathbb{R}$ such that

$$\mathsf{norm}(P)(\delta) = \frac{P(\delta)}{\sum_{\delta'\sim\delta} P(\delta')} \quad (62)$$

for every $\delta \in T_\Delta$. Putting both steps after each other, $\mathsf{ex}\,;\mathsf{norm}$ is an expectation-maximization step that maps a sequence of weighted forests to an optimized weight function.

For a family of derivation forests $\mathcal{D} = (D_i \mid i \in [m])$ with every D_i represented as a weighted tree automaton (Q_i, I_i, P_i), let $\mathcal{D}(P) = (D'_i \mid i \in [m])$ be the family of derivation forests where the weight function has been replaced by P, i.e., every D_i can be represented as a weighted tree automaton (Q_i, I_i, P'_i) with $P'_i(q, \delta(q_1, \ldots, q_k)) = P(\delta)$ for every $(q, \delta(q_1, \ldots, q_k)) \in \mathsf{dom}\, P_i$. Obviously, $\mathcal{D} = \mathcal{D}(P_L)$.

Let us now define a family of sequences of derivation forests $(\mathcal{D}^{(0)} \mid i \in \mathbb{N})$ as well as a family of weight functions $(P^{(i)} : \Delta \to \mathbb{R} \mid i \in \mathbb{N})$ such that

$$\begin{aligned} P^{(0)} &= P_L \\ \mathcal{D}^{(n)} &= \mathcal{D}(P^{(n)}) \\ P^{(n+1)} &= (\mathsf{ex}\,;\mathsf{norm})\left(\mathcal{D}^{(n)}\right) \end{aligned} \tag{63}$$

for every $n \in \mathbb{N}$. Then the corpus likelihood increases in every step until a local maximum is reached:

$$j \leq j' \implies \sum_{i=1}^{m} \sum_{t \in T_\Delta} L_{\mathcal{D}_i^{(j)}}(t) \leq \sum_{i=1}^{m} \sum_{t \in T_\Delta} L_{\mathcal{D}_i^{(j')}}(t)\,.$$

Chapter 6

Experiments

Contents

After laying out the theoretical foundations, and explaining the algorithms needed to implement the theory, we will now put the framework to a test. To this end, we will present three experiments in this chapter, serving various objectives.

1. Experiment A (Section 6.2) explores two theoretical questions of modeling. When we approximated the probability distribution $P(\mathbf{e}|\mathbf{f})$, there was a certain lack of theoretical motivation when the log-linear models were introduced. The purpose of this experiment is to show whether or not additional model parameters are reasonable from a practical point of view even if they are not theoretically justifiable. Furthermore, we will examine the difference between an STSG translation model and an In-XMBOT translation model.

2. Experiment B (Section 6.3) will serve as an example on how to use the toolkit for diagnostic purposes. We will replicate the workings of another machine translation toolkit to reason about the exactness of its results.

3. Experiment C (Section 6.4) will prove that the toolkit described in Section 5.2 can handle large-scale data. While not achieving state-of-the art results, we participated twice in the annual WMT shared task.

6.1 Common infrastructure

The decoding framework has been implemented in the programming language Python. Looking back, this might not have been the most fortunate choice, as many of the algorithms are rather computation-heavy, but the project initially started as a proof-of-concept diagnostic tool, not to replace full machine translation pipeline. While the toolkit has been used more or less successfully in large-scale machine translation campaigns (see Section 6.4), the focus remains on education and diagnostics. On the other hand, there may still be big potential for improvement. In the remainder of our section, we will call the framework presented in the previous chapters, and its implementation, the ExEx (for *"experimental exact"*) decoder.

Weighted regular tree grammars have been implemented in a format that is compatible with the popular Tiburon[22] toolkit (May and Knight, 2006b). For convenience, we implemented the intersection of structurally similar weighted regular tree grammars as needed for the translation model by allowing vectors of weights. After fixing a parameter vector λ, these can be converted to a single weight, if needed.

Rules are represented as triples of rule identifier (making up the ranked alphabet Δ used for derivation trees), input structure (a tree with variables by default) and output structure (a tree sequence with variables by default). Interpretation functions for tree homomorphisms and tree k-morphisms have been implemented, but interpretation functions for other types of input and output structure can easily be added. In particular, for efficiency, both tree homomorphisms and tree k-morphisms use string representations of trees and can therefore without any change be used for string homomorphisms and string k-morphisms.

In the remainder of this chapter, particular resources that were used or obtained in the pipeline (such as corpora, translation models, language models etc.) will be written in boldface.

6.1.1 Source data and preprocessing

For all our experiments, we picked the language pair English–German because German exhibits a high degree of discontiguities. The use of discontiguous rules has proven beneficial (Seemann et al. (2015b); Seemann et al. (2015a)). As our

[22] Courtesy of Jonathan May, available from https://github.com/isi-nlp/tiburon.

source data, we chose the freely available data used for the WMT shared task (see Section 6.4). Parallel data is made available from three sources:

- EUROPARL, the proceedings of the European parliament;
- NEWS, a collection of newswire text;
- COMMONCRAWL, a collection of parallel text obtained by crawling the Internet.

In all experiments, EUROPARL and NEWS were used. The English–German part of these will be referred to as the resource **euronews**. COMMONCRAWL was only used in Section 6.4. The concatenation of **euronews** and the English–German part of COMMONCRAWL will be referred to as the resource **euronewscrawl**. Furthermore, WMT makes available large quantities of monolingual text for language model training. We used this data to a varying degree. For testing, i.e., evaluation, we used official test sets from WMT. For development and parameter tuning, older test sets from WMT were used, as is common practice. All BLEU scores reported in this chapter are lowercase BLEU-4 scores, and given as percentages.

Effort has been made not to duplicate existing infrastructure. Therefore, we make extensive use of the data preprocessing scripts that are supplied by the popular Moses[23] toolkit (Koehn et al., 2007).

As for the actual preprocessing, it was made sure that all the input data was encoded in UTF-8. Portions of the data that could not be interpreted, as well as data that did not match the source or target language, were removed. This was achieved by using the n-gram based language guesser TEXTCAT[24], which implements the algorithm by Cavnar and Trenkle (1994). We also removed data from the COMMONCRAWL sub-corpus that was obviously not parallel by manually inspecting the documents it is composed of. Systems were built on lowercased data, and if necessary, capitalization was recovered in a post-processing step (this will be indicated).

6.1.2 Tokenization and parsing

Tokenization of the parallel and monolingual data was carried out using the Stanford Tokenizer[25] with the `preserve_lines` option, and the BITPAR tokenizer which has been adapted to preserve linebreaks. Keeping line breaks is crucial to ensure sentence alignment between both sides of the parallel corpus.

Both sides of the parallel corpus were then parsed. For rule extraction, generally only the best parse tree was kept for each sentence. For the English data, EGRET was used, for the German data, BITPAR was used.

[23] Available from http://statmt.org/moses/ under the LGPL open source licence.
[24] Available from http://odur.let.rug.nl/~vannoord/TextCat/.
[25] Available from http://nlp.stanford.edu/software/ under the GPL 3.0 open source licence.

RB(n't)	RB(not)
VBP('m)	VBP(am)
VBP('re)	VBP(are)
VPZ('s)	VBZ(is)
VBP('ve)	VBP(have)
VBD('d)	VBD(had)
MD('ll)	MD(will)
MD('d)	MD(would)
MD(wo)	MD(will)
MD(ca)	MD(can)

Table 6. Contractions and their long forms

Subsequently, parse failures were filtered out. We normalized contractions (*"is not"* vs. *"isn't"*) by carrying out the following replacements in the English data to recover the full form, using part-of-speech tags to disambiguate between different uses of *"'s"* (the possessive marker cannot be replaced by *"is"*). This was done to prevent data sparsity. Table 6 lists all the contractions that were recovered. Thus, we obtained the resources **euronews.parsed** and **euronewscrawl.parsed** as bilingual, bi-parsed parallel corpora.

6.1.3 Word alignment and rule extraction

Word alignment in these experiments was carried out on lowercased versions of the training data with two different tools:

- GIZA++[26] which is an implementation of the IBM models 1 through 5, as well as HMM alignment (Och and Ney, 2003);
- `fast_align`[27], which is an improved variant of IBM model 2 which is faster than GIZA++ and yields comparable results (Dyer et al., 2013).

The resulting word alignments were symmetrized and used together with the parse trees to extract rules in the ln-XMBOT and STSG formalisms.

In addition to the resulting word alignments, the lexical translation probabilities that are obtained along with them were kept for lexical scoring of the translation rules (see Section 5.1.2).

6.1.4 Language models

We generally used KENLM (Heafield, 2011) to build n-gram language models (unless otherwise indicated, $n = 5$). In addition to the German side of the parallel data, we occasionally used additional, monolingual data. KENLM offers

[26] Available from https://github.com/moses-smt/giza-pp under the GPL open source licence.

[27] Available from https://github.com/clab/fast_align under the Apache 2.0 open source licence.

a command-line interface that can be conveniently queried; it is not necessary to embed KenLM or any libraries.

6.1.5 Tuning

We use Z-Mert (Zaidan, 2009) as a "black box". Z-Mert takes as its input k-best lists with scores for the individual features and optimizes for Bleu. It should be noted as an advantage of our framework that decoding has to happen only once. In each iteration, only the k-best algorithm has to be re-run because the weight vector will have changed.

6.1.6 Coverage

One drawback of our strict adherence to theory is the fact that $\mathcal{D}_{\mathbf{f}}$ may be empty. In this case, no translation can be generated for $\mathbf{f}$. We call the set $C = \{c \in \mathbf{F} \mid \mathcal{D}_c \neq \emptyset\}$ the *coverage* of $\mathcal{D}$. Any sentence $c' \in \mathbf{F} - C$ is *out of coverage*. Taking into account the very specific nature of our ln-XMBOT or STSG translation rules, coverage is limited. Syntax-based translation systems usually increase coverage with the help of *glue rules* (Chiang, 2007) and/or identity translation rules.

6.2 Experiment A: Reasoning about models

We now describe an experiment that is concerned with the effect of different model restrictions on translation performance. First, we will explore the difference between m-morphisms and 1-morphisms. Then, we will try to answer the question whether the generic log-linear model can be replaced by a model with fewer degrees of freedom, closer to theory.

The data chosen for this experiment was the resource **euronews**. It was parsed using Egret and BitPar to obtain the resource **euronews.parsed** (only the best parse tree for each sentence was kept). Parse failures were filtered out to obtain the resource **euronews.filtered**. The lowercase version resource was used to generate lexical translation probabilities **euronews.lex** as well as word alignments **euronews.gdfa** by running `fast_align` in both directions and using the `grow-diag-final` heuristic for combining the word alignments of both directions.

We then built two distinct translation models: an STSG translation model **euronews.stsg** and an ln-XMBOT translation model **euronews.mbot** by using the appropriate rule extraction algorithms on **euronews.parsed** and **euronews.gdfa**. Recall that the difference between these is essentially the output morphism. This way, we can study the difference between m-morphisms and 1-morphisms on the output side. Given that ln-XMBOT rule extraction enables us to extract more, and also more generic rules, and that German exhibits a high degree of discontiguity in its syntactic structure, we expect ln-XMBOT to perform better than STSG.

For the first part of the experiment, the STSG translation model was compared to the ln-XMBOT translation model. The rules of both models were equipped with the following weights (see Section 5.1.2 for details on the rule scoring):

- forward translation probability, modeled by an equivalence relation defined by the root label of the source-side tree;
- backward translation probability, modeled by an equivalence relation defined by the root label(s) of the target-side tree(s);
- symmetric translation probability, modeled by an equivalence relation defined by the root labels of the source-side tree and the target-side tree(s);
- forward and backward lexical weights, obtained from lexical translation probabilities during word alignment.

In addition, the input parse forests were weighted. To this end, we created the resources **newstest2014.forests** and **newstest2015.forests** by parsing the WMT test sets of the 2014 and 2015 shared tasks using Egret. After decoding, two additional weights for each candidate on the k-best list were introduced:

- n-gram based language model sentence score;
- syntactic language model tree score.

The syntactic language model tree score was obtained by using the Berkeley Parser in "tree scoring" mode after recasing the tree (assigning uppercase to the first word in each sentence as well as all nouns, as is required in German). Whenever a tree was assigned a "$-\infty$" score by the parser (which corresponds to 0 because the parser returns log probabilities), we instead assigned it the worst score of any tree for the respective input sentence.

Each of these 8 scores was assigned a parameter λ_i, and the resulting log-linear models were tuned using Z-Mert for Minimum Error Rate Training, obtaining the parameter vectors **stsg.mert** and **mbot.mert**. The parallel corpora used for tuning were the subsets of **newstest2014.forests**, restricted to the coverage of the STSG or ln-XMBOT translation model, respectively. These resources, **newstest2014.stsg** and **newstest2014.mbot** contain 1 071 and 1 190 sentences, respectively, out of the original 2 736 sentences.

The systems were then evaluated by translating the subsets of the test set **newstest2015.forests** that were in coverage. These subsets, named **newstest2015.stsg** and **newstest2015.mbot**, contain 927 and 1 042 sentences, respectively, out of the original 2 169 sentences. The intersection of these subsets is exactly the set **newstest2015.stsg** because the STSG system is subsumed by the ln-XMBOT system. Table 7 summarizes the results. In particular, the ln-XMBOT based system outperforms the STSG based system on the intersection

	newstest2014 (tuning)			**newstest2015** (test)		
	.stsg	**.mbot**	all	**.stsg**	**.mbot**	all
TRAVATAR	13.68	13.45	13.48	15.95	15.89	14.65
STSG	12.60	n/a	n/a	14.07	n/a	n/a
ln-XMBOT restricted	12.63	12.12	n/a	17.15	16.34	n/a
ln-XMBOT	13.52	12.89	n/a	18.17*	17.29*	n/a

Table 7. Findings of Experiment A. The columns **.stsg** and **.mbot** indicate coverage of the STSG and ln-XMBOT based systems, respectively. The row TRAVATAR indicates performance of the system trained in Section 6.3 (with default settings) for comparison. Starred values are significantly better according to bootstrap resampling than values in rows above them ($p < 0.03$).

coverage by a wide margin. This was confirmed using the MOSES implementation of *bootstrap resampling* (Koehn, 2004; Riezler and Maxwell, 2005) with $p < 0.03$.

For the second part of the experiment, recall our bimorphism-based translation system, repeated from Equation (20):

$$\begin{aligned}\Pr(t) &= \Pr(t)^{\lambda_1} \cdot \Pr(t)^{\lambda_2} \cdot \Pr(t)^{\lambda_3} \\ &= \Big(\underbrace{\Pr(f,\mathbf{f})}_{\text{parser}} \cdot \underbrace{\Pr(t|f)}_{\text{forward}}\Big)^{\lambda_1} \cdot \underbrace{\Pr(t)}_{\text{symm.}}{}^{\lambda_2} \cdot \left(\underbrace{\Pr(t|e)}_{\text{backward}} \cdot \underbrace{\frac{\Pr(e,\mathbf{e})}{\Pr(\mathbf{e})}}_{\text{synLM}} \cdot \underbrace{\Pr(\mathbf{e})}_{\text{LM}}\right)^{\lambda_3}. \end{aligned} \tag{64}$$

Although deviating from the strict probabilistic model, it is generally accepted that this is approximated by choosing a different weight vector $\mu = (\mu_1, \ldots, \mu_6)$, in the style of a maximum entropy (Och and Ney, 2002) or log-linear model (cf. Equation (16)):

$$\begin{aligned}\Pr(t) &\approx \Pr(f,\mathbf{f})^{\mu_1} \cdot \Pr(t|f)^{\mu_2} \cdot \Pr(t)^{\mu_3} \cdot \Pr(t|e)^{\mu_4} \cdot \left(\frac{\Pr(e,\mathbf{e})}{\Pr(\mathbf{e})}\right)^{\mu_5} \cdot \Pr(\mathbf{e})^{\mu_6} \\ &\approx \prod_{i=1}^{6} \phi(t)^{\mu_i}\,, \end{aligned} \tag{65}$$

with $(\phi_i \mid 1 \leq i \leq 6)$ modeling aspects of u accordingly. In this experiment, we investigate whether these additional degrees of freedom in Equation (65) lead to better translation quality as compared to Equation (64) after Minimum Error Rate Training. Accordingly, we performed another round of Minimum Error Rate Training for the ln-XMBOT translation model **euronews.mbot**, this time only using three parameters μ_1, μ_2, μ_3, thereby obtaining a parameter vector **mbot-restricted.mert**. Table 7 summarizes the results obtained from both tuning rounds.

We conclude that the maximum entropy model, using arbitrary feature functions, does indeed perform better than the model with fewer degrees of freedom

that is closest to the idealized theoretical formulation. Again, this could be confirmed using bootstrap resampling ($p < 0.03$).

6.3 Experiment B: Exactness and search errors

In this experiment, we will examine whether translation quality can be improved by staying closer to the theoretical foundations. We will measure the number of search errors committed by a system that employs pruning, and we will examine if pruning is justified or not.

6.3.1 Search errors and model errors

As hinted in the previous sections, translation models as well as the grammars and automata that represent the models and the input can easily be of a size that is close to unmanageable. Most current MT systems use at least some form of pruning, i.e., they will sacrifice exactness for efficiency. Usually, this involves cutting off branches of the computation that are deemed unsuccessful or unlikely to succeed, e.g., by restricting the number of hypotheses that are explored. This number is called the *beam size* or *pop limit*. In addition to fixed limits, variable beams (depending on the score of the best partial hypothesis) have been used. While the pruning decisions are justified most of the time, and usually strongly guided by the language model, it is not guaranteed that the result will be the same with and without pruning. Exact decoding on the other hand does not make any decisions that alter the result of the computation. The result is guaranteed to be the result predicted by theory.

We usually evaluate the output of a machine translation system in terms of translation quality. Section 5.4.1 explains human and automatic evaluation in detail. However, while these evaluations target the quality of the actual output of the MT system in question, there is another question to be asked. Given that the implementation of a translation model might only yield an approximate solution, does this approximation hurt translation quality? Or, in other words, would the system benefit from a better approximation or even an exact, faithful application of the translation model?

When an MT system returns a translation that is not optimal according to the translation model, this is called a search error. However, when the optimal translation according to the translation model is not a good translation, this is called a model error. It is far from self-evident that avoiding search errors improves translation quality because evaluation should target model errors instead.

Lagrangian relaxation and Integer Linear Programming have been used for exact decoding. Chang and Collins (2011) compare exact decoding of a phrase-based MT system with a reference implementation of Moses. They report between 4.12% and 18.32% search errors, depending on the beam size systems (between 100 and 10 000). However, Moses commonly places a constraint on the shape of derivations, and when the same restriction is applied to the exact

decoder, no search errors can be observed for beam sizes of 200 and larger. Moreover, when comparing the BLEU scores, exact decoding does not produce a significant increase in translation quality. Rush and Collins (2011) compare exact decoding of an SCFG MT system with HIERO (Chiang, 2007). They report 13.7% search errors even at a large beam size of 1 000. However, they do not present BLEU scores or any kind of qualitative analysis.

6.3.2 Experimental setup

For comparison, we picked the TRAVATAR MT toolkit[28] (Neubig, 2013, 2014; Neubig and Duh, 2014). TRAVATAR is a syntax-based forest-to-string toolkit. The decoder takes as input a parsed sentence or parse forest (Mi et al., 2008), and outputs a string (or k-best list of strings). Rule extraction from aligned parallel corpora where the source is parsed and the output is not parsed is included in the toolkit, implementing the algorithm by Galley et al. (2006). Notably, TRAVATAR can extract composed rules. An extension of the rule extraction algorithm to forests (Mi and Huang, 2008) has also been implemented. The underlying formalism is the extended tree-to-string transducer (Galley et al., 2004).

Rules are scored using relative frequency (see Section 5.1.2). For tuning, TRAVATAR comes with an implementation of MERT (Och, 2003) and its extension to hypergraphs (Kumar et al., 2009), explained in Section 5.4.2.

Like in the experiments above, we used EGRET to parse the English part of **euronews**. The German data was only tokenized. Extended top-down tree-to-string transducer rules were then extracted from the lowercased data using the TRAVATAR toolkit, and transformed into ln-XTOP rules by using a string concatenation homomorphism (similar to Example 7). The equivalence relation on output objects is trivially $a \equiv b$ for all $a, b \in \mathbf{G}$, and therefore stateful behavior is restricted to the input side. In this way, a bijection was achieved between derivations in TRAVATAR and EXEX, which was also empirically validated by inspection of the translation scores.

Figures 28 and 29 show some rules extracted by TRAVATAR and their bimorphism representation. A rule representation in the TRAVATAR rule format[29] consists of five components, divided by `|||`:

- `source`: The source side tree, containing `variable:state` pairs;
- `target`: The target side string, containing variables with no state information;
- `features`: A list of `name=value` pairs;
- `counts`: The occurrence counts of source-target, source, and target, respectively;

[28] Courtesy of Graham Neubig, available under the LGPL 3.0 open source licence from https://github.com/neubig/travatar.

[29] http://www.phontron.com/travatar/model-format.html

```
np ( x0:dt x1:nn ) |||
x0 x1 |||
p=1 egfl=0.0 egfp=-0.042974 fgel=0.0 fgep=-2.648340 |||
608232 634940 8594661 |||
```

$$\frac{\text{DT} \quad \text{NN}}{\text{NP}} \qquad \mathsf{in}(f_1, f_2) = \begin{array}{c} \text{NP} \\ \diagup \quad \diagdown \\ f_1 \quad f_2 \end{array} \qquad \mathsf{out}(e_1, e_2) = e_1 e_2$$

Figure 28. TRAVATAR translation rule without lexical material

```
adjp ( adjp ( jjr ( "lower" ) ) pp ( in ( "than" ) x0:pp ) ) |||
"niedriger" "als" x0 |||
p=1 egfl=-4.740122 w=2 egfp=0.0 fgel=-4.166465 fgep=-2.639057 |||
1 1 14 |||
0-0 1-0 1-1
```

$$\frac{\text{PP}}{\text{ADJP}} \qquad \mathsf{in}(f_1) = \begin{array}{c} \text{ADJP} \\ \diagup \quad \diagdown \\ \text{ADJP} \quad\quad \text{PP} \\ | \quad\quad \diagup \quad \diagdown \\ \text{JJR} \quad\quad \text{IN} \quad f_1 \\ | \quad\quad | \\ \text{lower} \quad \text{than} \end{array} \qquad \mathsf{out}(e_1) = \text{niedriger als } e_1$$

Figure 29. TRAVATAR translation rule with lexical material

- `alignment`: The lexical alignment (for lexical scoring purposes, see Section 5.1.2).

Standard feature functions used by TRAVATAR are:

- `egfp`, `fgep`: conditional probabilities $\log \Pr(e|f)$ and $\log \Pr(f|e)$;
- `egfl`, `fgel`: lexical probabilities (see Section 5.1.2);
- `w`: the number of words on the target side;
- `p`: phrase count feature, always 1.

We used KENLM (Heafield, 2011) to train a 5-gram language model on the German side of the bilingual resource **euronews**. KENLM is integrated into TRAVATAR, but was used in EXEX too, by calling it from the command line.

We tuned the TRAVATAR translation model using MERT on the test set of the WMT-14 translation task (2 736 sentences). However, the parameter

pop limit	search errors	percentage	BLEU	runtime (sec)
10	1 957	90.2%	12.74	1 216.5
20	1 866	86.0%	13.36	1 201.9
50	1 735	80.0%	13.72	1 160.4
100	1 634	75.3%	13.86	1 159.8
200	1 512	69.7%	14.13	957.6
500	1 329	61.3%	14.49	1 170.8
1 000	1 187	54.7%	14.56	1 221.2
2 000	1 058	48.8%	14.65	1 275.8
5 000	861	39.7%	**14.69**	1 422.6
10 000	738	34.0%	14.67	1 671.5
20 000	625	28.8%	14.66	2 260.1
50 000	458	21.1%	14.65	5 327.6
100 000	366	16.9%	14.64	10 829.8
200 000	295	13.6%	14.58	24 317.8
500 000	168	7.7%	14.54	33 076.6
1 000 000	96	4.4%	14.49	35 903.6
2 000 000	3	0.14%	14.49	38 762.6
5 000 000	n/a	n/a	n/a	dnf

Table 8. Findings of Experiment B. Translation performance of TRAVATAR and runtime are given depending on pop limit (default: 2 000).

for glue rules and default handling of unknown rules (which are not part of the above-mentioned translation model) were set to an arbitrary high value of 10 000 because EXEX is not able to use them, and we do not want to use them in TRAVATAR when there are viable alternatives. We then used the optimized parameter vector for decoding using both TRAVATAR and EXEX.

6.3.3 Results and discussion

We evaluated the number of search errors, as well as translation quality using BLEU-4 on the test set from the WMT-15 translation task (2,169 sentences). We decoded this test set using different pop limits in TRAVATAR, directly influencing the number of hypotheses explored ("popped") in the k-best search (see Section 5.2.2). The method used by TRAVATAR is *cube pruning* (Chiang, 2007). Even with a pop limit of 2 000 000, we still identified search errors. We can confirm a search error if a better scoring translation can be found using EXEX or TRAVATAR with a higher pop limit. The TRAVATAR experiment with a pop limit of 5 000 000 did not finish because of lack of memory. Using EXEX, we were able to still identify 3 search errors for a pop limit of 2 000 000. However, the number of search errors reported is only a lower bound because even EXEX was not always able to guarantee optimality due to runtime constraints (see Section 5.3.4 for details on the exact rescoring algorithm).

As can be seen in Table 8, with the default pop limit of 2 000 we commit

at least 48.8% search errors. However, this does not impact translation quality. Even with a pop limit of 5 000 where we report the highest BLEU score, the percentage of search errors is still 39.7%. In fact, the BLEU score drops slightly when decoding is even more accurate, and the higher runtime does not seem to justify the faithfulness to the model. The BLEU score of TRAVATAR at pop limit 2 000 000 combined with EXEX, fixing 3 search errors, is 14.50, still well below the highest BLEU score at pop limit 5 000. We conclude that the default pop limit is indeed a good tradeoff between runtime and accuracy, and while not entirely convincing from a theoretical point of view, it can be justified by the high BLEU score.

6.4 Experiment C: Large scale decoding

WMT is a yearly conference (previously workshop) on statistical machine translation[30]. In addition to peer-reviewed technical papers, participants can build and compare MT systems on a predefined set of data. The systems are then scored automatically and by human annotators (cf. Section 5.4.1).

Both our submissions have been described in system papers in the proceedings of WMT. Our submission to the 9th Workshop on Statistical Machine Translation (WMT-14, Bojar et al. (2014)) is described in Quernheim and Cap (2014). Our submission to the 10th Workshop on Statistical Machine Translation (WMT-15, Bojar et al. (2015)) is described in Quernheim (2015).

As explained in the previous section, we deviated from the theoretical formulation and designed an ad-hoc maximum entropy model using the following feature functions:

1. Translation weight normalized by source root symbol
2. Translation weight normalized by all root symbols
3. Translation weight normalized by leaves on the source side (*)
4. Lexical translation weight target given source
5. Lexical translation weight source given target
6. Target side language model
7. Number of words in output string (*)
8. Number of rules used in the derivation (*)
9. Number of gaps in the target side sequences (*)
10. Penalty for rules that have more lexical material on the source side than on the target side or vice versa (absolute value) (*)
11. Probability of the rule in the input parse forest

Feature functions marked with (*) were only used in the WMT-14 submissions. Furthermore, the WMT-14 and WMT-15 submissions differed in two critical aspects:

1. For WMT-14, morphological preprocessing was carried out to combat

[30] The relevant websites for this experiment are http://statmt.org/wmt14/ and http://statmt.org/wmt15/.

feature	WMT-14 system	WMT-15 system
fallback	phrase-based, full sentence	word-based, subtree
morphology	compound splitting	none
additional features	yes	no
EN parser	EGRET	EGRET
DE parser	BITPAR	BITPAR
word alignment	GIZA^{++}	`fast_align`
MERT	Z-MERT	Z-MERT
language model	5-gram KENLM	5-gram KENLM
BLEU-4	17.0	15.3
in-coverage	unknown	16.7
description	Quernheim and Cap (2014)	Quernheim (2015)

Table 9. Comparison of WMT-14 and WMT-15 submissions

data sparsity. German noun compounds were split into their components to allow for better word alignment and more useful compositional rules. For WMT-15, no morphological preprocessing, in particular no compound splitting was carried out.

2. For WMT-14, sentences that could not be translated using the lnXMBOT-based system were translated by a phrase-based fallback system.

   For WMT-15, instead of an external phrase-based fallback system, a simple word-based system was used to translate subtrees in the input parse forest that could not be handled, instead of handling the whole sentence externally. This was achieved by adding dummy identity translation rules on the fly with a very low score. Thus, only in the absence of a derivation in the original model, a derivation involving a dummy rule would be chosen. Furthermore, scores were modelled at runtime such that the overall lexical material translated by dummy rules would be minimized.

For both submissions, a 5-gram language model was trained using KENLM on all the German monolingual data provided. Word alignment was performed using GIZA^{++} (WMT-14 submission) and `fast_align` (WMT-15 submission). For MERT tuning, we used Z-MERT on the WMT-13 test set. Both systems are compared in Table 9.

Coverage was limited because of data sparsity, especially since fine-grained morphological annotation was used in the German data. This causes highly specific rules that do not generalize well. Furthermore, only minimal rules were extracted, and only one parse tree per sentence was used for rule extraction.

Runtime is another problem. The full training pipeline takes about a week including parsing, word alignment, rule extraction and tuning. However, decoding takes approximately 5 minutes per sentence. Decoding can be trivially parallelized, though, and the full test set of 10 000 sentences (WMT-14) and 3 000 sentences (WMT-15) could be successfully translated within a week. However,

it might be worth to implement an algorithm like A*, where the completion cost of a partial hypothesis is estimated to explore promising hypotheses first, while keeping optimality. A useful heuristic to estimate the completion cost might be an n-gram language model. A* has been described for k-best extraction by Pauls and Klein (2009).

References

André Arnold and Max Dauchet. Morphismes et bimorphismes d'arbres. *Theoretical Computer Science*, 20 (1): 33–93, 1982. doi:10.1016/0304-3975(82)90098-6.

Mark Aronoff and Kirsten Fudeman. *What is Morphology?* Blackwell Publishing, 2005.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, 2005. URL http://aclweb.org/anthology/W05-0909.

Jean Berstel and Christophe Reutenauer. Recognizable formal power series on trees. *Theor. Comput. Sci.*, 18: 115–148, 1982. doi:10.1016/0304-3975(82)90019-6.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 Workshop on Statistical Machine Translation. In *Proceedings of the 9th Workshop on Statistical Machine Translation*, pages 12–58, 2014. URL http://aclweb.org/anthology/W14-3302.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 1–46, 2015. URL http://aclweb.org/anthology/W15-3001.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2 (4): 597–620, 2004. doi:10.1007/s11168-004-7431-3.

Fabienne Braune, Nina Seemann, Daniel Quernheim, and Andreas Maletti. Shallow local multi-bottom-up tree transducers in statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 811–821, 2013. URL http://aclweb.org/anthology/P13-1080.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16 (2): 79–85, 1990. URL http://aclweb.org/anthology/J90-2002.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19 (2): 263–311, 1993. URL http://aclweb.org/anthology/J93-2003.

Matthias Büchse, Mark-Jan Nederhof, and Heiko Vogler. Tree parsing with synchronous tree-adjoining grammars. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 14–25, 2011. URL http://aclweb.org/anthology/W11-2903.

Matthias Büchse. *Algebraic decoder specification: coupling formal-language theory and statistical machine translation*. PhD thesis, TU Dresden, 2015. URL http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-159266.

William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

Yin-Wen Chang and Michael Collins. Exact decoding of phrase-based translation models through Lagrangian relaxation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 26–37, 2011. URL http://aclweb.org/anthology/D11-1003.

David Chiang. An introduction to synchronous grammars, 2006. URL http://www3.nd.edu/~dchiang/papers/synchtut.pdf.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33 (2): 201–228, 2007. URL http://aclweb.org/anthology/J07-2003.

David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. The hiero machine translation system: Extensions, evaluation, and analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005. URL http://aclweb.org/anthology/H05-1098.

Noam Chomsky. *Syntactic Structures*. Mouton de Gruyter, 1957.

Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications, 2007. URL http://tata.gforge.inria.fr/.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39 (1): 1–38, 1977.

Steve DeNeefe. *Tree-Adjoining Machine Translation*. PhD thesis, University of Southern California, 2011.

Bonnie J. Dorr. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20 (4), 1994. URL http://aclweb.org/anthology/J94-4004.

Manfred Droste and Werner Kuich. Semirings and formal power series. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, EATCS Monographs on Theoretical Computer Science, chapter 1, pages 3–28. Springer, 2009.

Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. EATCS Monographs on Theoretical Computer Science. Springer, 2009.

Chris Dyer, Victor Chahuneau, and A. Noah Smith. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 644–648, 2013. URL http://aclweb.org/anthology/N13-1073.

Jason Eisner. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 205–208, 2003. URL http://aclweb.org/anthology/P03-2041.

Joost Engelfriet. Bottom-up and top-down tree transformations - A comparison. *Mathematical Systems Theory*, 9 (3): 198–231, 1975. doi:10.1007/BF01704020.

Joost Engelfriet. Tree automata and tree grammars. *CoRR*, abs/1510.02036, 2015. URL http://arxiv.org/abs/1510.02036. Slightly revised version of lecture notes from 1975.

Joost Engelfriet, Eric Lilin, and Andreas Maletti. Extended multi bottom-up tree transducers: Composition and decomposition. *Acta Informatica*, 46 (8): 561–590, 2009. doi:10.1007/s00236-009-0105-8.

Heidi Fox. Phrasal cohesion and statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 304–311, 2002. URL http://www.aclweb.org/anthology/W02-1039.

Zoltán Fülöp, Andreas Maletti, and Heiko Vogler. Preservation of recognizability for synchronous tree substitution grammars. In *Proceedings of the 2010 Workshop on Applications of Tree Automata in Natural Language Processing*, pages 1–9, 2010. URL http://aclweb.org/anthology/W10-2501.

Zoltán Fülöp, Andreas Maletti, and Heiko Vogler. Weighted extended tree transducers. *Fundamenta Informaticae*, 111 (2): 163–202, 2011.

Zoltán Fülöp and Heiko Vogler. Weighted tree automata and tree transducers. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, EATCS Monographs on Theoretical Computer Science, chapter 9, pages 313–403. Springer, 2009.

William A. Gale and Kenneth W. Church. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19 (1): 75–102, 1993. URL http://aclweb.org/anthology/J93-1004.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 273–280, 2004. URL http://aclweb.org/anthology/N04-1035.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, 2006. URL http://aclweb.org/anthology/P06-1121.

Ferenc Gécseg and Magnus Steinby. Tree automata. *CoRR*, abs/1509.06233, 2015. URL http://arxiv.org/abs/1509.06233. This is a reissue of the book Tree Automata by F. Gécseg and M. Steinby originally published in 1984 by Akadémiai Kiadó, Budapest.

Daniel Gildea. On the string translations produced by multi bottom-up tree transducers. *Computational Linguistics*, 38, 2012. URL http://aclweb.org/anthology/J12-3008.

Jonathan Graehl, Kevin Knight, and Jonathan May. Training tree transducers. *Computational Linguistics*, 34 (3), 2008. URL http://aclweb.org/anthology/J08-3004.

Thomas Hanneforth and Kay-Michael Würzner. Statistical language models within the algebra of weighted rational languages. *Acta Cybernetica*, 19 (2): 313–356, 2009.

Kenneth Heafield. KenLM: Faster and smaller language model queries. In *Proceedings of the 6th Workshop on Statistical Machine Translation*, pages 187–197, 2011. URL http://aclweb.org/anthology/W11-2123.

John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

Liang Huang and David Chiang. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 53–64, 2005. URL http://aclweb.org/anthology/W05-1506. Revised version available at http://www.cis.upenn.edu/~lhuang3/huang-iwpt-correct.pdf.

Liang Huang, Kevin Knight, and Aravind Joshi. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas*, 2006.

Miriam Kaeshammer. Synchronous linear context-free rewriting systems for machine translation. In *Proceedings of the 7th Workshop on Syntax and Structure in Statistical Translation*, pages 68–77, 2013. URL http://aclweb.org/anthology/W13-0808.

Miriam Kaeshammer. Hierarchical machine translation with discontinuous phrases. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 228–238, 2015. URL http://aclweb.org/anthology/W15-3028.

Kevin Knight. Capturing practical natural language transformations. *Machine Translation*, 21 (2): 121–133, 2008. doi:10.1007/s10590-008-9039-0.

Kevin Knight and Jonathan May. Applications of weighted automata in natural language processing. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, EATCS Monographs on Theoretical Computer Science, chapter 14, pages 571–596. Springer, 2009.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004. URL http://aclweb.org/anthology/W04-3250.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86, 2005.

Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133, 2003. URL http://aclweb.org/anthology/N03-1017.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 177–180, 2007. URL http://aclweb.org/anthology/P07-2045.

Alexander Koller and Marco Kuhlmann. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 2–13, 2011. URL http://aclweb.org/anthology/W11-2902.

Shankar Kumar and William Byrne. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 63–70, 2003. URL http://aclweb.org/anthology/N03-1019.

Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 169–176, 2004. URL http://aclweb.org/anthology/N04-1022.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, pages 163–171, 2009. URL http://aclweb.org/anthology/P09-1019.

Yang Liu, Yajuan Lü, and Qun Liu. Improving tree-to-tree translation with packed forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 558–566, 2009. URL http://aclweb.org/anthology/P09-1063.

Andreas Maletti. A tree transducer model for synchronous tree-adjoining grammars. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1067–1076, 2010a. URL http://www.aclweb.org/anthology/P10-1109.

Andreas Maletti. Survey: Tree transducers in machine translation. In *Proceedings of the 2nd International Workshop on Non-Classical Models of Automata and Applications*, volume 263 of *books@ocg.at*, pages 11–32. Österreichische Computer Gesellschaft, 2010b.

Andreas Maletti. How to train your multi bottom-up tree transducer. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 825–834, 2011a. URL http://aclweb.org/anthology/P11-1083.

Andreas Maletti. An alternative to synchronous tree substitution grammars. *Natural Language Engineering*, 17: 221–242, 2011b. doi:10.1017/S1351324911000027.

Andreas Maletti. Every sensible extended top-down tree transducer is a multi bottom-up tree transducer. In *Proceedings of the 12th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 263–273, 2012. URL http://aclweb.org/anthology/N12-1027.

Andreas Maletti. Synchronous forest substitution grammars. In *Proceedings of the 5th International Conference on Algebraic Informatics*, pages 235–246. Springer, 2013. doi:10.1007/978-3-642-40663-8_22.

Andreas Maletti. The power of weighted regularity-preserving multi bottom-up tree transducers. *Int. J. Found. Comput. Sci.*, 26 (7): 987–1005, 2015.

Andreas Maletti and Giorgio Satta. Parsing algorithms based on tree automata. In *Proceedings of the 11th International Workshop on Parsing Technology*, pages 1–12, 2009. URL http://aclweb.org/anthology/W09-3801.

Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. The power of extended top-down tree transducers. *SIAM Journal on Computing*, 39 (2): 410–430, 2009.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19 (2): 313–330, 1993.

Jonathan May and Kevin Knight. A better n-best list: Practical determinization of weighted finite tree automata. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 351–358, 2006a. URL http://aclweb.org/anthology/N06-1045.

Jonathan May and Kevin Knight. Tiburon: A weighted tree automata toolkit. In *Proceedings of the 11th International Conference on Implementation and Application of Automata*, pages 102–113. Springer, 2006b. doi:10.1007/11812128_11.

Jonathan May, Kevin Knight, and Heiko Vogler. Efficient inference through cascades of weighted tree transducers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1058–1066, 2010. URL http://www.aclweb.org/anthology/P10-1108.

Dan I. Melamed, Giorgio Satta, and Benjamin Wellington. Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 661–668, 2004. URL http://aclweb.org/anthology/P04-1084.

Haitao Mi and Liang Huang. Forest-based translation rule extraction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 206–214, 2008. URL http://aclweb.org/anthology/D08-1022.

Haitao Mi, Liang Huang, and Qun Liu. Forest-based translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 192–199, 2008. URL http://aclweb.org/anthology/P08-1023.

Mark-Jan Nederhof and Heiko Vogler. Synchronous context-free tree grammars. In *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 55–63, 2012.

Graham Neubig. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 91–96, 2013. URL http://aclweb.org/anthology/P13-4016.

Graham Neubig. Forest-to-string SMT for Asian language translation: NAIST at WAT 2014. In *Proceedings of the 1st Workshop on Asian Translation*, pages 20–25, 2014. URL http://aclweb.org/anthology/W14-7002.

Graham Neubig and Kevin Duh. On the elements of an accurate tree-to-string machine translation system. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 143–149, 2014. URL http://aclweb.org/anthology/P14-2024.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, 2003. URL http://aclweb.org/anthology/P03-1021.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, 2002. URL http://www.aclweb.org/anthology/P02-1038.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29 (1): 19–51, 2003.

Franz Josef Och and Hermann Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30 (4): 417–449, 2004. URL http://www.aclweb.org/anthology/J04-4002.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002. URL http://aclweb.org/anthology/P02-1040.

Adam Pauls and Dan Klein. K-best A* parsing. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, pages 958–966, 2009. URL http://aclweb.org/anthology/P09-1108.

Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 404–411, 2007. URL http://aclweb.org/anthology/N07-1051.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, 2006. URL http://aclweb.org/anthology/P06-1055.

Daniel Quernheim. Exact decoding with multi bottom-up tree transducers. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 164–171, 2015. URL http://aclweb.org/anthology/W15-3019.

Daniel Quernheim and Fabienne Cap. Large-scale exact decoding: The IMS-TTT submission to WMT14. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 163–170, 2014. URL http://aclweb.org/anthology/W14-3318.

Frank G. Radmacher. An automata theoretic approach to rational tree relations. In *Proceedings of the 34th Conference on Current Trends in Theory and Practice of Computer Science*, pages 424–435. Springer, 2008. doi:10.1007/978-3-540-77566-9_37.

Jean-Claude Raoult. Rational tree relations. *Bulletin of the Belgian Mathematical Society – Simon Stevin*, 4 (1): 149–176, 1997. URL http://eudml.org/doc/119808.

Stefan Riezler and T. John Maxwell. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, 2005. URL http://aclweb.org/anthology/W05-0908.

William C. Rounds. Mappings and grammars on trees. *Mathematical Systems Theory*, 4 (3): 257–287, 1970. doi:10.1007/BF01695769.

Alexander M. Rush and Michael Collins. Exact decoding of syntactic translation models through Lagrangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 72–82, 2011. URL http://aclweb.org/anthology/P11-1008.

Giorgio Satta and Enoch Peserico. Some computational complexity results for synchronous context-free grammars. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 803–810, 2005. URL http://www.aclweb.org/anthology/H05-1101.

Helmut Schmid. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 162–168, 2004. URL http://aclweb.org/anthology/C04-1024.

Helmut Schmid. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics*, pages 177–184, 2006. URL http://aclweb.org/anthology/P06-1023.

Nina Seemann, Fabienne Braune, and Andreas Maletti. A systematic evaluation of MBOT in statistical machine translation. In *Proceedings of the 15th MT-Summit*, pages 200–214. Association for Machine Translation in the Americas, 2015a. URL http://www.amtaweb.org/wp-content/uploads/2015/10/MTSummitXV_ResearchTrack.pdf.

Nina Seemann, Fabienne Braune, and Andreas Maletti. String-to-tree multi bottom-up tree transducers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 815–824, 2015b. URL http://aclweb.org/anthology/P15-1079.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27: 379–423, 1948.

Stuart M. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8 (3): 333–343, 1985.

Stuart M. Shieber. Synchronous grammars as tree transducers. In *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 88–95, 2004.

Stuart M. Shieber. Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006. URL http://aclweb.org/anthology/E06-1048.

Stuart M. Shieber. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proceedings of the Workshop on Syntax and Structure in Statistical Translation*, pages 88–95, 2007. URL http://www.aclweb.org/anthology/W07-0412.

Stuart M. Shieber. Bimorphisms and synchronous grammars. *Journal of Language Modelling*, 2 (1): 51–104, 2014. doi:10.15398/jlm.v2i1.84.

Stuart M. Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 253–258, 1990.

Anders Søgaard. Range concatenation grammars for translation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 103–106, 2008. URL http://aclweb.org/anthology/C08-2026.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. Results of the WMT15 metrics shared task. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 256–273, 2015. URL http://aclweb.org/anthology/W15-3031.

Magnus Steinby and Cătălin Ionuţ Tîrnăucă. Defining syntax-directed translations by tree bimorphisms. *Theoretical Computer Science*, 410 (37): 3495–3503, 2009. doi:10.1016/j.tcs.2009.03.009.

Jun Sun, Min Zhang, and Lim Chew Tan. A non-contiguous tree sequence alignment-based model for statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 914–922, 2009. URL http://aclweb.org/anthology/P09-1103.

James W. Thatcher. Generalized^2 sequential machine maps. *Journal of Computer and System Sciences*, 4 (4): 339 – 367, 1970. doi:10.1016/S0022-0000(70)80017-4.

Cătălin Ionuţ Tîrnăucă. *Syntax-directed translations, tree transformations and bimorphisms*. PhD thesis, Universitat Rovira i Virgili, 2016. URL http://hdl.handle.net/10803/381246.

Bernard Vauquois. A survey of formal grammars and algorithms for recognition and transformation in machine translation. In *Proceedings of the IFIP Congress*, pages 1114–1122, 1968.

Dekai Wu. An algorithm for simultaneously bracketing parallel texts by aligning words. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 244–251, 1995. URL http://aclweb.org/anthology/P95-1033.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23 (3): 377–403, 1997. URL http://aclweb.org/anthology/J97-3002.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530, 2001. URL http://aclweb.org/anthology/P01-1067.

Kenji Yamada and Kevin Knight. A decoder for syntax-based statistical MT. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 303–310, 2002. URL http://aclweb.org/anthology/P02-1039.

Omar F. Zaidan. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91: 79–88, 2009.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. Synchronous binarization for machine translation. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 256–263, 2006. URL http://aclweb.org/anthology/N06-1033.

Hui Zhang, Min Zhang, Haizhou Li, and Lim Chew Tan. Fast translation rule matching for syntax-based statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1037–1045, 2009. URL http://aclweb.org/anthology/D09-1108.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Lim Chew Tan, and Sheng Li. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 559–567, 2008. URL http://aclweb.org/anthology/P08-1064.

Index

Wissenschaftlicher Werdegang

- Februar 2011 bis Juli 2016: Akademischer Mitarbeiter am Institut für Maschinelle Sprachverarbeitung der Universität Stuttgart
    - Mitarbeit im Projekt "Tree Transducers in Machine Translation" unter der Leitung von Dr. Andreas Maletti
    - Arbeit an der vorliegenden Dissertation unter der Betreuung von Dr. Andreas Maletti
- Oktober 2005 bis November 2010: Studium der Computerlinguistik an der Universität Potsdam
    - Abschluss als Diplom-Sprachwissenschaftler *mit Auszeichnung* (Gesamtnote 1,1)
    - Titel der Abschlussarbeit: "Hyper-minimisation of weighted finite automata"
- Juni 2004: Abitur am Archigymnasium Soest (Gesamtnote 1,1)

Bibliographische Daten

Autor: Daniel Quernheim
Titel: Bimorphism Machine Translation
135 Seiten, 29 Abbildungen, 9 Tabellen

Zusammenfassung

Maschinelle Übersetzung (MÜ) hat bedeutenden Fortschritt dank der Anwendung statistischer Methoden gemacht, welche es ermöglichen, MÜ-Systeme automatisch aus einer zweisprachigen Textsammlung zu gewinnen. Manche Ansätze benötigen nicht einmal linguistische Annotationen, um Übersetzungsregeln aus Rohdaten abzuleiten. Viele heutige MÜ-Systeme beachten linguistische Struktur kaum, oder nutzen Ad-hoc-Formalismen und Algorithmen. Dies führt zu vermeidbarem mehrfachem Aufwand, und nicht zuletzt zu einem unnötigen Zwiespalt zwischen Theoretikern und Praktikern.

Um diesem Mangel an Motivation und Stringenz zu begegnen, trägt diese Dissertation folgendes bei:

1. Nach einer Darstellung des historischen Hintergrundes und Kontextes sowie der mathematischen und linguistischen Grundlagen, wird ein rigoros algebraisches Modell der MÜ vorgestellt. Reguläre Baumgrammatiken und Bimorphismen bilden das Rückgrat dieses Ansatzes, der eine modulare Architektur mit vielen verschiedenen möglichen Eingabe- und Ausgabeformalismen darstellt.

2. Die Herausforderungen der Implementierung dieses auf Bimorphismen basierenden Modelles in einem MÜ-Toolkit werden beschrieben, und die nötigen Algorithmen für die Kernkomponenten werden im Detail erklärt.

3. Schließlich werden Experimente vorgestellt, in denen das Toolkit auf praxisgerechte Daten angewandt und für diagnostische Zwecke verwandt wird. Wir diskutieren, wie exakte Übersetzungsmodelle es ermöglichen, Suchfehler und Modellfehler in einem beliebten anderen MÜ-Toolkit aufzuspüren, und wir vergleichen Ausgabeformalismen verschiedener Mächtigkeit.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialen oder erbrachten Dienstleistungen als solche gekennzeichnet.

(Ort, Datum)

(Unterschrift)