Syntax-based Statistical Machine Translation

Synthesis Lectures on Human Language Technologies

Synthesis Lectures on Human Language Technologies is edited by Graeme Hirst of the University of Toronto. The series consists of 50- to 150-page monographs on topics relating to natural language processing, computational linguistics, information retrieval, and spoken language understanding. Emphasis is on important new techniques, on new applications, and on topics that combine two or more HLT subfields.

Natural Language Processing for Social Media
Atefeh Farzindar and Diana Inkpen
2015

Semantic Similarity from Natural Language and Ontology Analysis
Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain
2015

Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition
Hang Li
2014

Ontology-Based Interpretation of Natural Language
Philipp Cimiano, Christina Unger, and John McCrae
2014

Automated Grammatical Error Detection for Language Learners, Second Edition
Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault
2014

Web Corpus Construction
Roland Schäfer and Felix Bildhauer
2013

Recognizing Textual Entailment: Models and Applications
Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto
2013

Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax
Emily M. Bender
2013

Semi-Supervised Learning and Domain Adaptation in Natural Language Processing
Anders Søgaard
2013

Semantic Relations Between Nominals
Vivi Nastase, Preslav Nakov, Diarmuid Ó Séaghdha, and Stan Szpakowicz
2013

Computational Modeling of Narrative
Inderjeet Mani
2012

Natural Language Processing for Historical Texts
Michael Piotrowski
2012

Sentiment Analysis and Opinion Mining
Bing Liu
2012

Discourse Processing
Manfred Stede
2011

Bitext Alignment
Jörg Tiedemann
2011

Linguistic Structure Prediction
Noah A. Smith
2011

Learning to Rank for Information Retrieval and Natural Language Processing
Hang Li
2011

Computational Modeling of Human Language Acquisition
Afra Alishahi
2010

Introduction to Arabic Natural Language Processing
Nizar Y. Habash
2010

Cross-Language Information Retrieval
Jian-Yun Nie
2010

Automated Grammatical Error Detection for Language Learners
Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault
2010

Data-Intensive Text Processing with MapReduce
Jimmy Lin and Chris Dyer
2010

Semantic Role Labeling
Martha Palmer, Daniel Gildea, and Nianwen Xue
2010

Spoken Dialogue Systems
Kristiina Jokinen and Michael McTear
2009

Introduction to Chinese Natural Language Processing
Kam-Fai Wong, Wenjie Li, Ruifeng Xu, and Zheng-sheng Zhang
2009

Introduction to Linguistic Annotation and Text Analytics
Graham Wilcock
2009

Dependency Parsing
Sandra Kübler, Ryan McDonald, and Joakim Nivre
2009

Statistical Language Models for Information Retrieval
ChengXiang Zhai
2008

Syntax-based Statistical Machine Translation

Philip Williams, Rico Sennrich, Matt Post, and Philipp Koehn

www.morganclaypool.com

ISBN: 9781627059008 paperback
ISBN: 9781627055024 ebook

DOI 10.2200/S00716ED1V04Y201604HLT033

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON HUMAN LANGUAGE TECHNOLOGIES

Lecture #33
Series Editor: Graeme Hirst, *University of Toronto*
Series ISSN
Print 1947-4040 Electronic 1947-4059

Syntax-based Statistical Machine Translation

Philip Williams
University of Edinburgh

Rico Sennrich
University of Edinburgh

Matt Post
Johns Hopkins University

Philipp Koehn
Johns Hopkins University

SYNTHESIS LECTURES ON HUMAN LANGUAGE TECHNOLOGIES #33

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

This unique book provides a comprehensive introduction to the most popular syntax-based statistical machine translation models, filling a gap in the current literature for researchers and developers in human language technologies. While phrase-based models have previously dominated the field, syntax-based approaches have proved a popular alternative, as they elegantly solve many of the shortcomings of phrase-based models. The heart of this book is a detailed introduction to decoding for syntax-based models.

The book begins with an overview of synchronous-context free grammar (SCFG) and synchronous tree-substitution grammar (STSG) along with their associated statistical models. It also describes how three popular instantiations (Hiero, SAMT, and GHKM) are learned from parallel corpora. It introduces and details hypergraphs and associated general algorithms, as well as algorithms for decoding with both tree and string input. Special attention is given to efficiency, including search approximations such as beam search and cube pruning, data structures, and parsing algorithms. The book consistently highlights the strengths (and limitations) of syntax-based approaches, including their ability to generalize phrase-based translation units, their modeling of specific linguistic phenomena, and their function of structuring the search space.

KEYWORDS

statistical machine translation, syntax, synchronous grammar formalisms, natural language processing, computational linguistics, machine learning, statistical modeling

Contents

Preface

Statistical machine translation takes a complex human skill and frames it as a clean, crisp mathematical equation. Expressed in words, this equation asks the following question.

> Given a text in the source language, what is the most probable translation in the target language?

For the most part, researchers and practitioners do not worry about the linguistic and philosophical question of what it means to talk about language and cognition as being probabilistic processes. Instead they view statistical models as tools with which to attack a complex and poorly understood problem. The interpretation is pragmatic: how should one construct a statistical model that assigns high probabilities to "good" translations and low probabilities to "bad" translations? Of course, no such model will be perfect. Unavoidably, it will have blind spots and shortcomings, and it will sometimes assign high probabilities to sentences that no human would conceive of uttering.

Restated in more formal terms, here is how statistical machine translation frames the translation problem:

$$t^* = \arg\max_t p(t|s).$$

The variables s and t are strings of tokens in the source and target language, respectively. Typically, the tokens represent words and the strings represent sentences. The statistical model that we want to construct is of the conditional distribution $p(t|s)$.

Since the inception of the field at the end of the 1980s, the most popular models for statistical machine translation—the models of $p(t|s)$—have been sequence-based. In these models, the basic units of translation are words or sequences of words (called phrases). Sequence-based n-gram language models are a core feature. These kinds of models are simple and effective, and they work well for many language pairs, but they exhibit weaknesses inherent to their sequence-based nature. For instance, they are blind to the global structure of the target sentence, and translation decisions are made based on a narrow window of adjacent words. They also do not take into account the cross-lingual syntactic divergences observed by linguists, such as that English is subject-verb-object (SVO) and Japanese subject-object-verb (SOV). Furthermore, heuristics used to rein in the complexity of the search problem in phrase-based translation often preclude the correct translation.

Despite the empirical success of sequence-based models, these deficiencies make them unsatisfying from a scientific perspective. This is especially true when we consider how much work has been independently undertaken in modeling the syntactic structure of language, both natural

and formal. It is therefore unsurprising that the last 20 years have also seen fruitful research on translation models that are in some way based on the syntactic structure of the source and/or target sentences. Incorporating syntactic structure into the translation process allows reordering between languages to be informed by that structure. It also permits long-distance dependencies to be used in selecting translations of ambiguous source-language words, and can help promote fluent translations beyond the local window of phrases and *n*-grams. Since models that produce target-side syntactic structure are naturally extended with syntactic language models, this provides a means of improving global fluency and addressing problems that are morphosyntactic in nature, such as agreement.

Research into syntax-based translation has not just scratched a scientific itch. The effectiveness of syntax-based approaches has been demonstrated in shared translation tasks such as OpenMT12[1] and the yearly WMT evaluations [Bojar et al., 2014, 2015]. These successes have been borne out in automatic and human evaluations. Output from syntax-based systems has long been observed to have a different feel from that of sequence-based models, and, as might be hoped, to improve grammaticality. And while not always empirically the best choice, syntax has been shown to be superior across a range of situations, in high-resource settings and low, between languages disparate and similar.

Sequence-based models, however, remain the default. And while statistical machine translation is a large and important enough area of research to have garnered lengthy surveys and a textbook, syntax-based approaches have to date received little in the way of introductory presentation. We hope this book fills that gap.

OVERVIEW OF THIS BOOK

This book introduces the reader to core models and algorithms that have emanated from research on syntax-based statistical machine translation, and have found their use in different varieties of syntax-based translation models. While we cannot cover every aspect of this subject in a book of this size, we aim to provide coverage that is sufficiently broad and sufficiently in-depth to open up the research literature for further study and to provide a solid foundation for beginning experimental work.

The main focus of this book is models based on *synchronous context-free grammars* (SCFG) and *synchronous tree-substitution grammars* (STSG). Historically, most research in statistical machine translation has focused on translation into English and most syntax-based approaches have been based on constituency. This book reflects that bias. Nonetheless, we hope that readers interested in dependency-based approaches will not feel discouraged: despite their roots in constituency grammar, the methods described here have been successfully adapted and applied to dependency-based translation.

We have tried to keep this book as self-contained as possible; however, we do assume some background knowledge in mathematics and computer science. In particular, we assume that the

[1] http://www.nist.gov/itl/iad/mig/openmt12results.cfm

reader has at least a passing familiarity with basic concepts from probability theory, set theory, graph theory, and algorithms. Any previous exposure to natural language processing, and specifically statistical machine translation, is likely to be helpful but is not essential. For readers looking for a succinct crash course to the field of machine translation, we recommend the survey of Lopez [2008]. For a more in-depth introduction focusing on phrase-based translation, the reader would do well to consult the textbook, *Statistical Machine Translation* [Koehn, 2010].

Here is a sketch of what is to come.

Chapter 1 introduces the syntactic and statistical building blocks that make up a syntax-based model. We introduce SCFG and STSG formally and then discuss the role they play in statistical models. With these formal foundations in place, we begin to introduce the family of model types that exist in the wild. We begin with a high-level classification of approaches, before giving a brief history of syntax-based statistical machine translation.

Chapter 2 describes how the synchronous grammars that serve as translation models can be learned from parallel text. We discuss extraction algorithms for three popular grammars: hierarchical phrase-based grammars, syntax-augmented grammars, and GHKM grammars.

Chapter 3 is the first of three chapters devoted to decoding—the task of finding the most probable translation of a source sentence for a given model. For syntax-based models, decoding is based on a generalization of the graph called a hypergraph. This chapter introduces basic definitions and algorithms on hypergraphs.

Chapter 4 discusses tree decoding, i.e., decoding when the input sentence has already been parsed. It introduces the basic decoding algorithm, and discusses the inclusion of non-local features such as n-gram language models. Non-local features break the independence assumptions that allow for efficient exact decoding with rule-locally weighted grammars. We discuss common approximations made for the sake of efficiency, in particular beam search and cube pruning.

Chapter 5 describes string decoding, i.e., decoding with a string input. While the basic algorithms from the previous two chapters carry over to string decoding, the chapter discusses how to tackle the higher computational complexity of string decoding. The chapter covers topics such as stronger search approximations, grammar binarization and pruning, and efficient parsing algorithms and data structures for non-binary grammars.

Chapter 6 delves into a selection of further topics. We begin by discussing some important extensions of the core models and algorithms. For the most part, these extensions are aimed at relaxing the constraints imposed by constituency. We then move onto alternative grammar formalisms, including the use of dependency structure. Several prominent translation systems are based on dependency syntax, and we discuss dependency-based machine translation, comparing it to the algorithms introduced previously. Finally, we highlight some

linguistic phenomena that have proven challenging for statistical machine translation, especially sequence-based models, but have seen progress through the use of syntax-based approaches.

Chapter 7 wraps up the book with closing remarks, discussing some limitations of syntax-based machine translation, and speculating on its future.

Acknowledgments

Work on the material for this book began with the preparation of a tutorial on syntax-based SMT delivered by two of the authors (Williams and Koehn) at EMNLP 2014. We would like to thank the tutorial attendees for their feedback and questions, and our editor, Graeme Hirst, for proposing that we develop the tutorial into a Synthesis lecture.

The initial draft of this book was improved greatly by the numerous corrections and many thoughtful suggestions of the two anonymous reviewers. We are indebted to them for the time and care they took in thoroughly reviewing the book. We would also like to thank Gillian Foy and Keisuke Sakaguchi, who carefully read through preliminary drafts and provided many helpful suggestions and corrections.

Finally, our thanks to the staff at Morgan and Claypool for their assistance (and patience) throughout the preparation of this book.

Philip Williams, Rico Sennrich, Matt Post, and Philipp Koehn
May 2016

CHAPTER 1

Models

During the last 20 years, researchers have proposed a wide range of syntax-based approaches to statistical machine translation. While a core set of methods has now crystallized and become dominant, much of the diversity of those approaches is still reflected in the models that are in current use. In this chapter, we describe the common components that make up a modern syntax-based model. We will focus initially on the formal syntactic and mathematical building blocks before introducing the family of model types. We will not say too much about specific models—that will come in the next chapter.

In a syntax-based model, the basic units of translation are synchronous grammar rules. We begin by introducing these rules informally in Section 1.1, showing how they compare to phrase pairs, which are the basic translation unit in the dominant phrase-based approach. We proceed in Section 1.2 by formalizing the description, introducing the basic concepts of two synchronous grammar formalisms: synchronous context-free grammar (SCFG) and synchronous tree-substitution grammar (STSG).

In formal grammar, a primary concept is that of derivation. In the current context this roughly means a sequence of translation steps. Once equipped with a formal conception of derivation, we have something over which to define a statistical model. In Section 1.3, we will describe the two main types of statistical modeling framework: the generative noisy-channel framework and the discriminative log-linear framework.

Having covered these foundational topics, we give a classification of syntax-based models in Section 1.4. As with classification in general, the criteria for classifying syntax-based models are somewhat subjective. The specific approach we take is not particularly important. Rather, the main purpose of this section is to introduce and contrast the range of features that can be found in current models.

Finally, in Section 1.5, we give a brief history of syntax-based statistical machine translation, with pointers to many of the seminal papers in the field.

1.1 SYNTACTIC TRANSLATION UNITS

In phrase-based models, the basic unit of translation is the *phrase pair*. A phrase pair is simply a pairing of a sequence of tokens in the source language with a sequence of tokens in the target language. We will write phrase pairs in the form ⟨ *source* | *target* ⟩. For instance, the pair ⟨ `ein groß` | `a big` ⟩ represents a translation unit in which the German fragment `ein groß` is mapped to the English fragment `a big`.

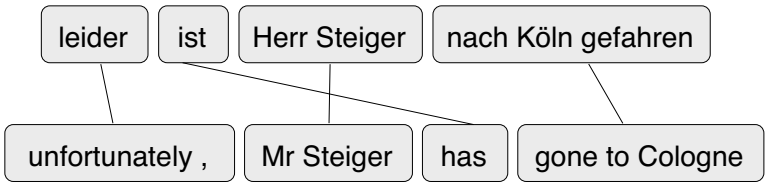


Figure 1.1: A derivation using phrases.

In syntax-based models, the translation unit can be thought of as an extended version of the phrase pair. There is not one type of syntactic translation unit. Rather, there is a family of related types, with different syntax-based models using different types. Within this family, one property can be considered fundamental: the phrases can contain *gaps*. The introduction of gaps allows for the nesting of phrases within phrases and corresponds directly to the notion of recursion in natural language. There are two other main types of extension: the first is *labeling*—the association of phrases with labels; the second is *internal tree structure*—the association of phrases with syntactic tree fragments.

Before we see how these notions are defined formally, let us take a look at some examples of different types of translation units and see how each of them functions as part of a translation. In the examples that follow, we will show a decomposition into translation units of the following German-English translation:[1]

(1) leider ist Herr Steiger nach Köln gefahren
unfortunately has Mr Steiger to Cologne gone

(2) unfortunately, Mr Steiger has gone to Cologne

Our example sentence pair exhibits one of the features of German-English translation that has proven problematic for statistical machine translation: the movement of the main verb from a position just after the subject in English to the end of the clause in German. In the example, the verb does not have far to move, but in longer sentences the distance can be considerable—much further than is allowed by the hard reordering limit imposed by typical phrase-based models.

1.1.1 PHRASES

Figure 1.1 shows a possible derivation of our example sentence pair using phrase pairs. By choosing different segmentations, many alternative derivations are of course possible. When given a choice, longer phrases are usually preferable because they include more context, but in practice a

[1]We have decapitalized the first word of each sentence. Normalizing letter case helps to improve the generality of translation units and is common practice in statistical machine translation (see, for example, Koehn et al. [2008]).

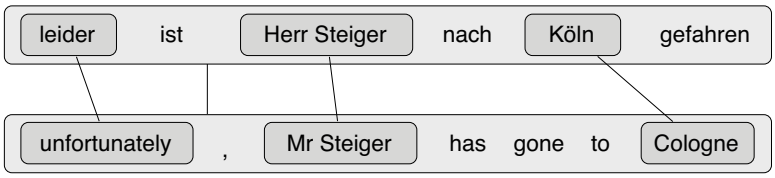


Figure 1.2: A derivation using phrases with gaps.

model must draw from the limited pool of phrase pairs that were observed in training. This means that the context encoded in an individual phrase pair rarely extends beyond three or four words.

For the issue of verb movement in English-German translation, moving the main verb within a phrase pair may be possible for very short distances, as in the example sentence, which uses the phrase pair ⟨ `nach Köln gefahren` | `gone to Cologne` ⟩. However, for movement over longer distances, it becomes increasingly unlikely that a suitable phrase pair will exist, meaning that the model must instead reorder phrases.

1.1.2 PHRASES WITH GAPS

Our second derivation, shown in Figure 1.2, uses phrases with gaps. On the German side, the phrases `leider`, `Herr Steiger`, and `Köln` are nested inside the larger sentence-level phrase `◇ ist ◇ nach ◇ gefahren` (where the ◇ symbol represents a gap). Unlike the previous example, all reordering is now internal to the translation units. The decision about how to move the verbs is encoded within the outermost translation unit. Importantly, the verbal translation is now captured within a single operation:

⟨ `◇ ist ◇ nach ◇ gefahren` | `◇ , ◇ has gone to ◇` ⟩

Notice also that the insertion of a comma is now contextualized within a unit that covers the entire sentence.

While gaps allow the generalization of long phrase pairs, they also carry a danger of over-generalization since there is no restriction on the form of the phrase pair that fills a gap. For instance, suppose that we wish to translate the sentence:

(3) leider ist Herr Steiger **nicht** nach Köln gefahren
*unfortunately has Mr Steiger **not** to Cologne gone*

The outermost translation unit can be overlaid onto this sentence just as well as for the original sentence. Now suppose that during training the model had learned the phrase pair:

⟨ `Herr Steiger nicht` | `Mr Steiger does not` ⟩

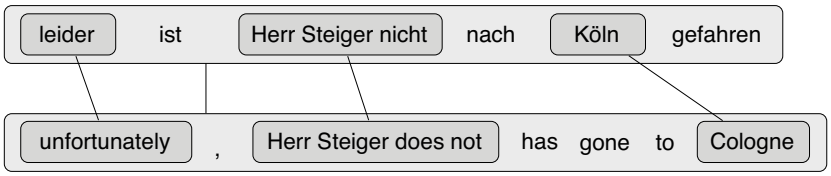


Figure 1.3: An alternative derivation using phrases with gaps. In this instance the English sentence is ungrammatical.

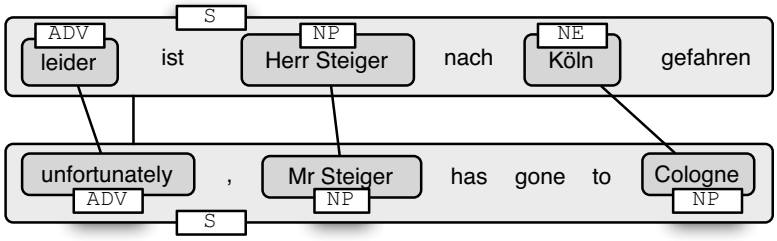


Figure 1.4: A derivation using phrases with labels.

This is a perfectly reasonable translation unit. However, the translation units do not produce a good translation in combination: the resulting sentence, shown in Figure 1.3, is ungrammatical and difficult to make sense of.

1.1.3 PHRASES WITH LABELS

Our third derivation, shown in Figure 1.4, adds labels to the phrases. The labels in this example happen to be treebank constituent labels, but in principle, they could be anything: CCG supertags, dependency relations, semantic roles, and so on. Adding labels can go some way to addressing the overgeneralization problem. For instance, the outermost translation unit requires the second gap to be filled by a phrase pair that is labeled NP (for "noun phrase") on both the source and target side. This rules out the problematic phrase pair from the previous example, since neither phrase is a noun phrase.

Notice that there can be differing labels for the source and target phrases (in the example, `Köln` is labeled NE and `Cologne` is labeled NP). As we will see later, this is a modeling decision: some models require matching source and target labels.

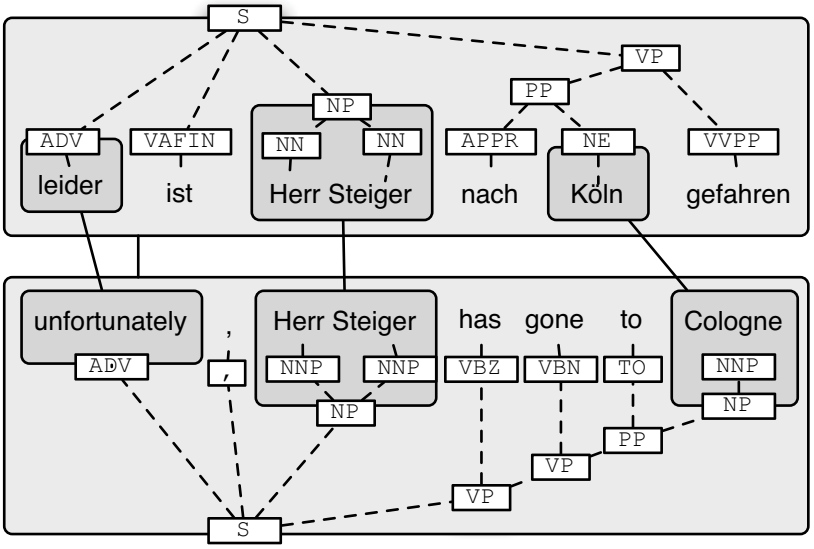


Figure 1.5: A derivation using phrases with internal tree structure.

Now that we are introducing syntactic annotation into the rules, it is natural to take this a step further and fill in the missing tree structure.

1.1.4 PHRASES WITH INTERNAL TREE STRUCTURE

Our final derivation adds internal tree structure (Figure 1.5). We have already showed how gaps in phrase pairs allow us to capture a generalization about verb phrase translation in a single operation. By adding labeled gaps and tree structure, we are in theory able to extend this ability to capture phenomena like paired predicate-argument structure. In the example, it is possible to capture paired tree fragments that explain how the past participle construction of verbs changes from German to English. It can also help disambiguate between syntactic relations. For instance, consider the following sequence of labeled gaps: V NP PP. Does the PP attach to the NP or to the V? Correctly translating a sentence may depend on knowing the correct attachment—a detail that can be encoded in the internal tree structure.

Syntax in rules has other practical advantages. On the source side, syntax-directed rule selection can be used to reduce the search space to enable faster decoding (Section 1.4). On the target side, the syntax can be used to inform scoring features. For instance, a syntactic language model can be used to encourage the generation of syntactically well-formed sentences.

1.2 GRAMMAR FORMALISMS

Most syntax-based models are based on one of three formalisms: synchronous context-free grammar (SCFG), synchronous tree-substitution grammar (STSG), or tree transducers. The first, SCFG, allows the definition of translation units containing gaps, which can optionally be labeled; the last two, STSG and tree transducers, add internal tree structure.

In their most general form, tree transducers are more powerful than STSG, adding operations such as node copying or deletion. However, these facilities have so far not been employed in statistical machine translation and the restricted form of tree transducer rules that are used in practice are equivalent to STSG rules with only minor technical differences. To simplify the presentation, we will use STSG throughout this book.[2]

We begin by defining context-free grammar (CFG). Once the basics of CFG have been understood, it is a short step to understanding the synchronous version and then to understanding STSG. In addition to describing how the translation units can be represented as SCFG or STSG rules, we spend some time introducing the concepts of derivation and weighted grammars, and introducing related terminology and notation. These are prerequisites for the description of statistical models in the following section and decoding algorithms in Chapters 3–5. Rather than scattering the material through the book, we have tried to provide a self-contained introduction here. If these concepts are new to you and some details pass you by on first reading, then you can always return to this section later.

1.2.1 CONTEXT-FREE GRAMMAR

We suspect that most readers will have at least some familiarity with context-free grammar. For those readers, let us bring CFG back into sharp focus with a formal definition. Even if you have not seen CFG before, the basic concepts should quickly become clear.

Definition 1.1 (Context-Free Grammar). A *context-free grammar (CFG)* is a 4-tuple $G = \langle T, N, S^\dagger, R \rangle$, where T is a finite set of terminal symbols; N is a finite set of non-terminal symbols that is disjoint from T; $S^\dagger \in N$ is a start symbol; and R is a set of production rules defined over $N \cup T$. Each rule has the form $A \to \alpha$, where A is a non-terminal and α is a string of terminals and non-terminals.

In the context of natural language processing, the terminals of a CFG typically represent the words of a language and the non-terminals represent grammatical categories. The categories might be those delineated by a linguistic theory or might be derived by other means (for example, using unsupervised machine learning methods). The production rules collectively specify the ways in which the words of the language can be combined. For instance, a CFG for English might contain a production rule S → NP VP specifying that a noun phrase (NP) can be combined with a verb

[2] For readers wishing to learn more about tree transducer formalisms, see Knight and Graehl [2005] for an excellent overview.

phrase (VP) to produce a sentence (S). In turn, further production rules specify precisely how NPs and VPs can be constructed and exactly what combinations of words they can contain.

The flip side to describing how words combine is that CFGs also describe how strings of words can be broken apart. We will focus in this section on the combinatorial aspect: how a CFG combines words into strings through the process of *derivation*. We will return later to the process of *parsing*, which is concerned with breaking down a string into its constituent parts.

Derivation is a sequential process that operates on a symbol string (i.e., a string of terminals and non-terminals). At each step, the symbol string is transformed by rewriting a single non-terminal, A, as a string of symbols, α. Rewriting is constrained by the grammar's set of production rules: a rewrite of non-terminal A with symbol string α is only allowed if the grammar includes the production rule $A \rightarrow \alpha$. For many grammars, the derivation process can continue indefinitely (for instance by continually rewriting a non-terminal X as Y then as X again), but for the most part we are only concerned with finite derivations. We are particularly interested in derivations that begin with a single non-terminal and end with a string of terminals. When a derivation yields a string of terminals, the process *must* end since there is no way of further rewriting the string. When a derivation begins with the grammar's start symbol and ends with a string of terminals, the resulting string is called a *sentence* ("sentence" is a formal term here: a sentence in this sense need not be a sentence in the natural language sense).

Example 1.2 If a CFG G has terminal symbols $T = \{\text{a, beastie, bold, timorous}\}$ and non-terminal symbols $N = \{\text{ADJ, DET, NOUN, NP}\}$, of which $S^{\dagger} = \text{NP}$ is designated the start symbol, and if G contains the production rules:

$$\begin{aligned} \text{NP} &\rightarrow \text{DET ADJ NOUN} \\ \text{DET} &\rightarrow \text{a} \\ \text{ADJ} &\rightarrow \text{bold} \\ \text{ADJ} &\rightarrow \text{timorous} \\ \text{NOUN} &\rightarrow \text{beastie} \end{aligned}$$

then the two sentences "a timorous beastie" and "a bold beastie" (and only those two sentences) are derivable from G. Here is one way of deriving the first sentence.

1. Starting from the string "NP," rewrite it as "DET ADJ NOUN" by applying the first production rule.
2. Next, rewrite the intermediate string to "DET timorous NOUN" (using the fourth rule).
3. Then rewrite that to "DET timorous beastie" (using the fifth rule).
4. Finally, rewrite the string to "a timorous beastie" (using the second rule).

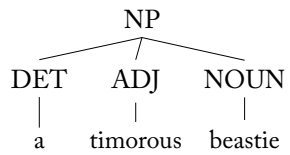
By applying the production rules in different orders, there are six (trivially) distinct ways to derive the sentence.

In Example 1.2, the start symbol NP, the intermediate strings, and the sentence are all examples of *sentential forms*. For any grammar G, a sentential form is either G's start symbol $S^\dagger$ or a sequence of terminals and non-terminals that can be derived from $S^\dagger$ via a sequence of one or more rule applications.

If a symbol string, α, can be rewritten as a second, β, using a single production rule then we say that β is *immediately derivable* from α and denote this relation $\alpha \Rightarrow \beta$. We use the symbol $\overset{*}{\Rightarrow}$ to indicate that a symbol string can be rewritten as another via a sequence of zero or more rule applications (naturally, an application of zero rules leaves the string unchanged).

The derivation process has a hierarchical nature: when a non-terminal symbol is rewritten, we can think of that non-terminal being the parent of the symbols that replace it; and in turn, any child non-terminal symbols being the parents of the symbols that replace them. The resulting hierarchical object is referred to as a *derivation tree*.

Example 1.3 The six derivations from Example 1.2 all share the following derivation tree:



If two derivations result in the same derivation tree then the differences in the order of rewrites are usually of little interest. By adopting a convention of, for instance, always rewriting the leftmost non-terminal in a symbol string, we can uniquely associate a single *canonical derivation* with each distinct derivation tree. In Example 1.2, the two sentences ("the timorous beastie" and "the bold beastie") have a single canonical derivation each.

As well as referring to a process, the word "derivation" is also used to refer to an entity, which can be thought of as the trace of the process:

Definition 1.4 (CFG Derivation). A (finite) *CFG derivation* is a sequence of sentential forms $\sigma = \gamma_1, \gamma_2, \ldots, \gamma_k$ defined over some CFG such that $\gamma_i \Rightarrow \gamma_{i+1}$ for all $1 \leq i < k$. If a canonical derivation form is assumed then σ can equivalently be expressed as a sequence of production rules $r_1, r_2, \ldots, r_{k-1}$, where r_i is the (unique) rule that rewrites γ_i to γ_{i+1}.

The use of a canonical form of derivation is so common that it frequently goes unstated. Often the word "derivation" is used without qualification to mean a canonical derivation or to refer to a derivation tree. Throughout the remainder of this book, unless specified otherwise, we will use the term in this second, order-independent sense.

Having defined a CFG derivation, we can define the *language* of a grammar.

Definition 1.5 (Language of a CFG). Let G be a context-free grammar with start symbol $S^\dagger$. The *language* of G, denoted $\mathcal{L}(G)$, is the set of sentences that can be derived from $S^\dagger$.

At this point, our examples have exhibited an implicit constraint on their form. For a rule $A \to \alpha$, the right-hand side α has been either (a) a single terminal symbol or (b) a sequence of one or more non-terminals. Definition 1.1 does not constrain α in this way, but instead states that α is a sequence of one or more symbols drawn from $N \cup T$. We can produce a variation of the grammar from Example 1.2 that combines some of the rules:

$$\begin{aligned} \text{NP} &\to \text{a ADJ beastie} \\ \text{ADJ} &\to \text{bold} \\ \text{ADJ} &\to \text{timorous} \end{aligned}$$

The first rule here is reminiscent of the labeled gaps discussed above. The language of this grammar is exactly the same as that of Example 1.2: the sentences *a bold beastie* and *a timorous beastie*. This is true despite the fact that the derivations that produce these sentences will differ between the grammars; the first grammar takes four productions per string, while the second takes only two. Because these grammars have identical languages, they are considered to be *weakly equivalent*; if they were isomorphic (up to a relabeling), they would also be *strongly equivalent*.

Definition 1.6 (Equivalence). Two grammars G_1 and G_2 are *weakly equivalent* if $\mathcal{L}(G_1) = \mathcal{L}(G_2)$. They are *strongly equivalent* if every derivation in G_1 is isomorphic to a derivation in G_2, and vice versa.

As we will see later in this book, the form of a grammar's rules can have implications for computational complexity. One important property is the number of non-terminals on the rule's right-hand side. This is referred to as the rule's *rank*.

Definition 1.7 (CFG Rank or Arity). The *rank* (equivalently, *arity*) of a CFG rule is the number of non-terminals on its right-hand side. The rank of a grammar is the maximum rank of any rule in the grammar.

1.2.2 SYNCHRONOUS CONTEXT-FREE GRAMMAR

Having defined CFG, we can now define the synchronous version.

Definition 1.8 (Synchronous Context-Free Grammar). A *synchronous context-free grammar* (SCFG) is a 4-tuple $G = \langle T, N, S^\dagger, R \rangle$ where T, N, and $S^\dagger$ are defined as for CFG (Definition 1.1) and R is a set of synchronous rules. Each rule $r \in R$ is a pair of CFG production rules $A \rightarrow \alpha$ and $B \rightarrow \beta$, A and B being non-terminals, α and β strings of terminals and non-terminals, with the property that α and β contain the same number of non-terminals, and each non-terminal in α is paired exclusively with one non-terminal in β.

In the context of statistical machine translation, the first production rule, $A \rightarrow \alpha$ is a CFG rule in the source language and the second, $B \rightarrow \beta$, is a CFG rule in the target language. The terminal set T comprises the combined vocabularies of the source and target languages (as observed in the data from which the rules are learned). The contents of the non-terminal set N depends on the type of model: in models that use unlabeled gaps, N will typically contain a single generic non-terminal (commonly X) and a separate start symbol (commonly S). In models that use labeled translation units, N will contain symbols derived from linguistic categories for either the source or target language, or (less commonly) both.

We will write a general SCFG rule in the following format:

$$A \rightarrow \alpha \ , \ B \rightarrow \beta$$

using subscripted indices to indicate the non-terminal links.

Example 1.9 The translation units from Example 1.4 can be written as the following SCFG rules:

ADV	→	leider	,	ADV	→	unfortunately
NP	→	Herr Steiger	,	NP	→	Mr Steiger
NE	→	Köln	,	NP	→	Cologne
S	→	ADV_1 ist NP_2 nach NE_3 gefahren	,	S	→	ADV_1 , NP_2 has gone to NP_3

It is often necessary to refer individually to the source or target components of a SCFG rule or grammar. These components are referred to as the source and target projections and are denoted using a function called proj.

Definition 1.10 (Source and Target Projections). If $G = \langle T, N, S^\dagger, R \rangle$ is a SCFG and $r = \langle A \rightarrow \alpha, B \rightarrow \beta \rangle \in R$ then the *source projection* and *target projection* of r, $\text{proj}_s(r)$ and $\text{proj}_t(r)$, are

$A \rightarrow \alpha$ and $B \rightarrow \beta$, respectively. The source and target projections of G, $\text{proj}_s(G)$ and $\text{proj}_t(G)$, are the CFGs with the rule sets $\{\text{proj}_s(r) \mid r \in R\}$ and $\{\text{proj}_t(r) \mid r \in R\}$.

Given a CFG rule q, the *inverse* source (or target) projection returns the set of SCFG rules for which q is the source (or target) side component. Of course, the inverse projection must be applied with respect to a particular SCFG.

Definition 1.11 (Inverse Source and Target Projections). If $G = \langle T, N, S^\dagger, R\rangle$ is a SCFG and $q = \langle C \rightarrow \gamma\rangle$ is a CFG rule then the *inverse source projection* and *inverse target projection* of q with respect to G, $\text{proj}'_s(q)$ and $\text{proj}'_t(q)$, are the rule sets $\{r \mid q = \text{proj}_s(r), r \in R\}$ and $\{r \mid q = \text{proj}_t(r), r \in R\}$, respectively.

Derivation with a SCFG G involves the rewriting of a pair of symbol strings $\langle \gamma_s, \gamma_t\rangle$, where γ_s contains only symbols that occur on the source-side of G's production rules and γ_t contains only symbols that occur on the target-side. As with the production rules, γ_s and γ_t are required to contain the same number of non-terminals, and each non-terminal in γ_s is paired exclusively with one non-terminal in γ_t. A rewrite using a rule $\langle A \rightarrow \alpha, B \rightarrow \beta\rangle$ is allowed if γ_s contains the non-terminal A, γ_t contains the non-terminal B, and A and B are paired with each other.

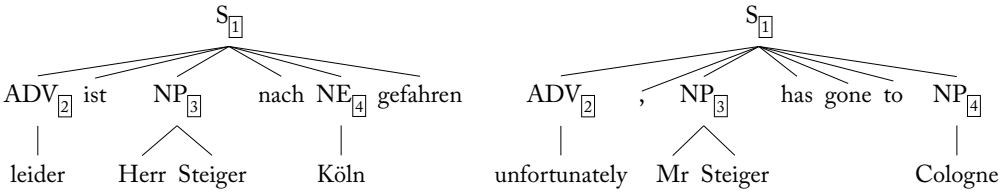
Example 1.12 Using the SCFG rules from Example 1.9, and assuming S to be the start symbol, we can derive a pair of source and target sentences as follows:

$$
\begin{aligned}
& \text{S}_1, \text{S}_1 \\
\Rightarrow\ & \text{ADV}_2 \text{ ist NP}_3 \text{ nach NE}_4 \text{ gefahren ,} \\
& \quad \text{ADV}_2 \text{ , NP}_3 \text{ has gone to NP}_4 \\
\Rightarrow\ & \text{leider ist NP}_3 \text{ nach NE}_4 \text{ gefahren ,} \\
& \quad \text{unfortunately , NP}_3 \text{ has gone to NP}_4 \\
\Rightarrow\ & \text{leider ist Herr Steiger nach NE}_4 \text{ gefahren ,} \\
& \quad \text{unfortunately , Mr Steiger has gone to NP}_4 \\
\Rightarrow\ & \text{leider ist Herr Steiger nach Köln gefahren ,} \\
& \quad \text{unfortunately , Mr Steiger has gone to Cologne}
\end{aligned}
$$

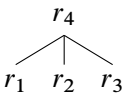
Just as a CFG derivation has an associated tree, a SCFG derivation has an associated pair of trees, one for the source-side and one for the target-side. Each non-terminal node of the source tree is linked with a single non-terminal node of the target tree. Equivalently, a SCFG derivation

can be expressed as a single tree where the nodes are labeled with rule identifiers and each node has one child for each non-terminal pair in the corresponding rule. The ordering of a node's children can follow the source or target non-terminal order, as long as the ordering scheme is consistent across the tree. As with CFG derivations, if a canonical derivation form is assumed then a SCFG derivation can be expressed as a sequence of SCFG rules.

Example 1.13 The derivation from Example 1.12 corresponds to the following pair of trees. Non-terminal links are indicated using co-index boxes:



Equivalently, if we use the identifiers $r_1 - r_4$ to refer to the SCFG rules from Example 1.9 (in the order they are listed) then the derivation can be written:



The concepts of language, equivalence, and rank generalize naturally from CFG to SCFG.

Definition 1.14 (Language of a SCFG). Let G be a SCFG with start symbol $S^\dagger$. The *language* of G, denoted $\mathcal{L}(G)$, is the set of sentence pairs that can be derived from $S^\dagger$.

Definition 1.15 (SCFG Rank). The *rank* of a SCFG rule is the number of non-terminal pairs on its right-hand side. The rank of a SCFG is the maximum rank of any rule in the grammar.

The definition of equivalence is exactly as for CFG and so we do not repeat it here (see Definition 1.6).

There is a widely used variant of SCFG that we will refer to as *shared-category* SCFG. Shared-category SCFG is defined identically to the *distinct-category* version of Definition 1.8, except that the paired CFG rules have the same left-hand side and every non-terminal in α is paired with an identical non-terminal in β. When writing shared-category SCFG rules, we will drop the target left-hand side:

$$A \rightarrow \alpha \ , \beta$$

Distinct-category SCFGs are a natural choice for models in which the translation units are labeled on both the source and target sides, since the two languages are likely to use differing grammatical categories. When only one side is labeled—which is by far the more common case in practice—shared-category SCFGs are usually used.

While the expressivity of distinct-category SCFG is attractive for certain kinds of model, it is sometimes more convenient to work with a shared-category SCFG. We can convert any distinct-category SCFG to shared-category form by conjoining each pair of non-terminals to create a hybrid "source and target" non-terminal.[3]

Example 1.16 The SCFG from Example 1.9 can be expressed in shared-category form as follows:

(ADV, ADV)	→	leider	,	unfortunately
(NP, NP)	→	Herr Steiger	,	Mr Steiger
(NE, NP)	→	Köln	,	Cologne
(S, S)	→	(ADV, ADV)$_1$ ist (NP,NP)$_2$ nach (NE,NP)$_3$ gefahren	,	(ADV, ADV)$_1$, (NP,NP)$_2$ has gone to (NE,NP)$_3$

Note that the notation (A, B) is for our convenience only. Formally, the name is arbitrary: we could equally have chosen a symbol like X_{482}.

1.2.3 SYNCHRONOUS TREE-SUBSTITUTION GRAMMAR

Using SCFG, we can define translation units with labeled gaps. Synchronous tree-substitution grammar (STSG) allows us to add internal tree structure. This is possible since STSG rules are pairs of tree fragments.

Definition 1.17 (Synchronous Tree-Substitution Grammar). A *synchronous tree-substitution grammar* (STSG) is a 4-tuple $G = \langle T, N, S^\dagger, R \rangle$ where T, N, and $S^\dagger$ are defined as for CFG (Definition 1.1) and R is a set of synchronous rules. Each rule $r \in R$ is a pair of trees α and β in which the root and interior nodes are non-terminals drawn from N and each leaf node is either a terminal drawn from T or a non-terminal drawn from N. Each non-terminal leaf in α is paired exclusively with one non-terminal leaf in β.

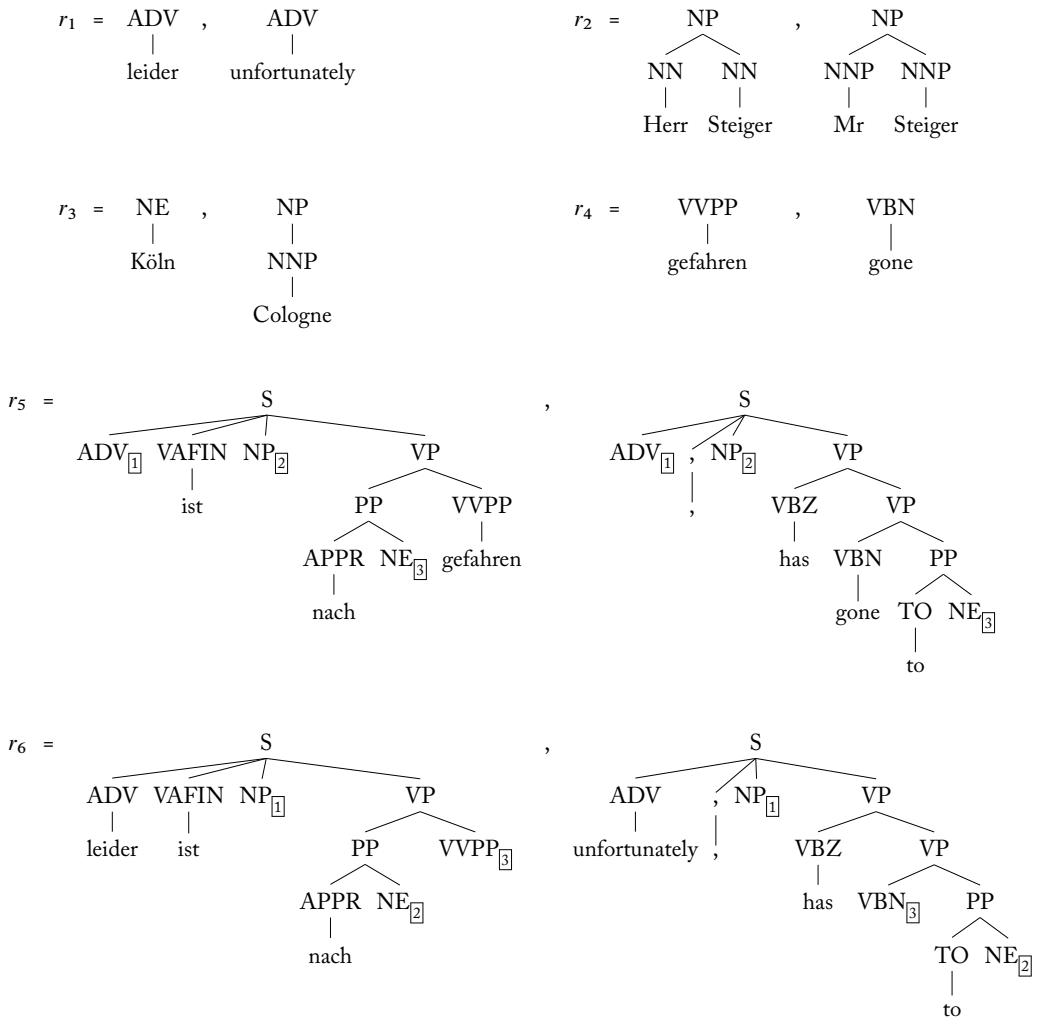
In the context of machine translation, α is a tree in which the terminals represent tokens in the source language and non-terminals represent grammatical categories. Similarly, the termi-

[3] As this conversion makes clear, the distinct-category version does not provide any additional formal expressiveness over its shared-category counterpart.

nals and non-terminals of β represent tokens and grammatical categories in the target language. Typically, the trees are fragments of parse trees over source and target sentences in the training data.

We will write a general STSG rule as a pair of comma-separated trees, using index boxes to indicate the non-terminal links.

Example 1.18 An example of an STSG. The translation units from Figure 1.5 correspond to the rules r_1, r_2, r_3, and r_5.



As for SCFG, it is convenient to have a notation for referring individually to the source and target components of a STSG rule. We will define the analogous projection functions here. Just as the source and target components of a SCFG rule are CFG rules, the components of a STSG rule are tree-substitution grammar (TSG) rules. The relationship between the TSG and STSG formalisms is essentially the same as the relationship between CFG and SCFG. We will therefore leave the definition of TSG implicit, except to note that the projection of a STSG rule is more usually referred to as an *initial tree* or an *elementary tree*, rather than a rule. For uniformity, we will use the term *rule* since many of the methods that we will present later in this book are (more or less) interchangeable between CFG and TSG formalisms (or SCFG and STSG).

Definition 1.19 (Source and Target Projections (STSG)). If $G = \langle T, N, S^\dagger, R \rangle$ is a STSG and $r = \langle \alpha, \beta \rangle \in R$ then the *source projection* and *target projection* of r, $\text{proj}_s(r)$ and $\text{proj}_t(r)$, are α and β, respectively. The source and target projections of G, $\text{proj}_s(G)$ and $\text{proj}_t(G)$, are the TSGs with the rule sets $\{\text{proj}_s(r) \mid r \in R\}$ and $\{\text{proj}_t(r) \mid r \in R\}$.

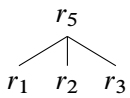
Given a TSG rule q, the inverse source (or target) projection returns the set of STSG rules for which q is the source (or target) side component. The inverse projection must be applied with respect to a particular STSG.

Definition 1.20 (Inverse Source and Target Projections (STSG)). If $G = \langle T, N, S^\dagger, R \rangle$ is a STSG and $q = \gamma$ is a TSG rule then the *inverse source projection* and *inverse target projection* of q with respect to G, $\text{proj}'_s(q)$ and $\text{proj}'_t(q)$, are the rule sets $\{r \mid q = \text{proj}_s(r), r \in R\}$ and $\{r \mid q = \text{proj}_t(r), r \in R\}$, respectively.

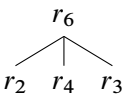
Derivation with a STSG G involves the rewriting of a pair of trees $\langle \gamma_s, \gamma_t \rangle$. Whereas an SCFG rewrite replaces a pair of non-terminals with a pair of symbol strings, an STSG rewrite replaces a pair of non-terminals with a pair of trees. A rewrite using a rule $\langle \alpha, \beta \rangle$ is allowed if γ_s contains the non-terminal A, where $A = \text{root}(\alpha)$, γ_t contains the non-terminal B, where $B = \text{root}(\beta)$, and the sentential form's A and B are paired with each other. The rewrite operation is called synchronous tree substitution. As with SCFG, γ_s and γ_t are required to contain the same number of non-terminals and each non-terminal in γ_s is paired exclusively with one non-terminal in γ_t.

Like CFG and SCFG, a STSG derivation has an associated derivation tree. The derivation tree is distinct from the derived trees and care must be taken to avoid confusing the two types of object.

Example 1.21 Two sentential derivations are possible using the grammar rules from Example 1.18. Their derivation trees are



and

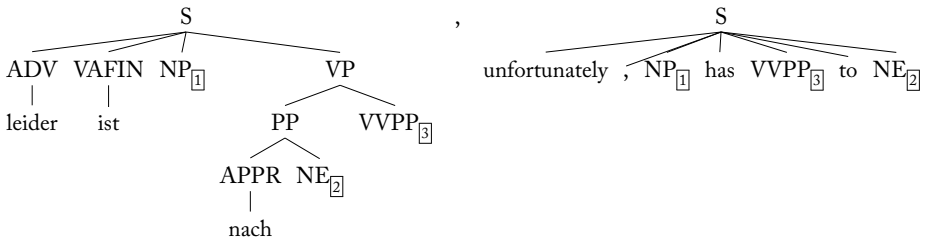


Notice that the two derivations have the same source and target derived trees. The left derivation corresponds to the derivation shown in Figure 1.5.

Language, equivalence, and rank are defined for STSG exactly as for SCFG (see Definitions 1.14, 1.6, and 1.15, respectively).

Like with SCFG, we will distinguish between shared-category and distinct-category forms of STSG. The examples we have given so far have used the distinct-category form. In the shared-category form, it is typical that the tree structure on either the source or target side of a rule is derived from a parse tree; the other side of the rule has a flat structure with non-terminal labels that double those of their counterparts.

Example 1.22 A shared-category version of rule r_6 from Example 1.18 that uses source-side structure only would be:



1.2.4 PROBABILISTIC AND WEIGHTED GRAMMARS

Probabilistic and weighted variants have been developed for CFG, SCFG, and STSG. Among these, the most frequently encountered are probabilistic CFGs (PCFGs), which are widely used in natural language processing, most notably for phrase-structure parsing.

In this book, we will follow recent work in parsing by defining rule weights abstractly as elements of a weight set $\mathbb{K}$. For a particular application, the weight set might contain probabilities, real numbers, Booleans, vectors, or some more exotic type. At this level of abstraction, the

definitions of rule weights and weighted grammars are essentially the same, regardless of the underlying formalism. In the following definition we will use the word "grammar" to mean any of CFG, SCFG, or STSG and the word "rule" to mean a rule of one of those grammars.

Definition 1.23 (Rule Weight Function and Weighted Grammar). Given a rule set R, a *rule weight function* for R is a function $W_R : R \to \mathbb{K}$. A *weighted* grammar is a pair $\langle G, W_R \rangle$, where G is a grammar with rule set R and W_R is a rule weight function for R.

A probabilistic grammar is a weighted grammar $\langle G, P \rangle$ with a probabilistic rule weight function P. Specifically, P is a probability mass function that gives the probability of a rule conditioned on its left-hand-side. To put that another way, suppose that midway through a CFG derivation, we have arrived at a symbol string α, and we want to rewrite the leftmost non-terminal. If G's production rules allow us to derive β in the next step ($\alpha \Rightarrow \beta$) then P gives us the probability of this event (as opposed to any alternative the grammar might allow). The same interpretation applies for synchronous grammars except that then we are dealing with pairs of sentential forms and pairs of non-terminals.

Definition 1.24 (Probabilistic Context-Free Grammar (PCFG)). A *probabilistic context-free grammar (PCFG)* is a pair $\langle G, P \rangle$, where $G = \langle T, N, S^\dagger, R \rangle$ is a CFG (Definition 1.1) and $P : R \to [0, 1]$ is a rule weight function such that for any $A \in N$,

$$\sum_{\alpha} P(A \to \alpha) = 1.$$

Probabilistic SCFGs and STSGs are defined in a similar way to PCFGs. Instead of conditioning on a single left-hand side non-terminal, the SCFG rule probability function conditions on a pair of left-hand side non-terminals. The STSG rule probability function conditions on a pair of root non-terminals.

If we assume that rule applications are independent events, then the probability of a derivation is simply the product of the probabilities of its component rules. While this independence assumption is clearly unrealistic in practice—for example, rewriting a NOUN as "tiger" is going to be more probable if a previous derivation step rewrote the preceeding word as "fierce" rather than "cloudy"—, assuming independence between rule applications greatly simplifies language modeling and this assumption is widely used. The problem of overgeneralization can be ameliorated by using more specific categories. For instance, the category-refinement approach of the Berkeley parser [Petrov et al., 2006] dramatically improves over a basic probabilistic PCFG, bringing the model close to state of the art performance.

While it is often useful to assume independence, there are situations where this assumption is too strong and needs to be relaxed. In this case we can use an alternative derivation weighting

function (and possibly an alternative rule weight function). In general, the weight of a derivation is not necessarily the product of the component rule's weights.

Definition 1.25 (Derivation Weight Function). Given a derivation set D, a *derivation weight function* on D is a function $W_D : D \to \mathbb{K}$ that assigns a weight to every derivation $d \in D$.

1.3 STATISTICAL MODELS

We now return to the fundamental problem of modeling the distribution $p(t|s)$. Using the concepts from the previous section, we are now in a position to formulate a model that expresses the probability of a translation in terms of rules, derivations, and weights.

As in statistical machine translation generally, the first statistical syntax-based models proposed in the literature were generative models—that is, they modeled the joint probability distribution $p(s,t)$. A generative model of translation describes a process that generates a pair (s,t) via a sequence of steps—it provides a *generative story*. We have already seen one example: probabilistic synchronous grammar.

In contrast, discriminative models directly model the distribution of interest: $p(t|s)$. They dispense with the notion of a generative story and instead calculate a conditional probability for a translation in terms of weighted features. It is up to the model's designer to designate a set of features that they deem relevant to the problem. During training, the model learns to weight the features in order that it can discriminate good translations from bad ones.

Following the pattern in phrase-based modeling, generative models have now been supplanted by discriminative ones, with the conditional log-linear approach of Och and Ney [2002] being the dominant approach. Since generative models are no longer widely used, we will not describe them in detail. Nonetheless, we feel it is instructive to examine an example in order to see some of the decisions made in the modeling process. Here we give a representative example, which is due to Galley et al. [2006]. We then move on to discriminative models, focusing on the log-linear framework.

1.3.1 GENERATIVE MODELS

Applied to statistical machine translation, a generative model is a model of the joint probability distribution $p(s,t)$. The distribution of primary interest to us is $p(t|s)$, and the two distributions are related by the chain rule:

$$p(t|s) = \frac{p(s,t)}{p(s)}. \tag{1.1}$$

Since the denominator, $p(s)$, is fixed for a given input sentence, s, we can substitute a generative model directly into our objective function:

$$t^* = \arg\max_t p(t|s) \tag{1.2}$$

$$= \arg\max_t p(s,t). \tag{1.3}$$

We have already seen a simple means of modeling $p(s,t)$: probabilistic synchronous grammar (recall that these grammars define a probability distribution over derivations; this implicitly defines a distribution over string pairs since for any string pair we can obtain its probability by summing up the probabilities of derivations yielding that pair).

Given a probabilistic SCFG or STSG (or data from which to estimate one) we could stop at this point. However, probabilistic synchronous grammars have been found to make poor translation models on their own. Instead, generative statistical machine translation models generally follow the noisy channel approach[4] from information theory. Returning to the chain rule (Equation 1.1), if we switch the s and t variables and rearrange terms, then we get

$$p(s,t) = p(s|t)p(t). \tag{1.4}$$

Substituting into Equation 1.1, we arrive at

$$p(t|s) = \frac{p(s|t)p(t)}{p(s)} \tag{1.5}$$

and our objective function becomes

$$t^* = \arg\max_t p(s|t)p(t). \tag{1.6}$$

Equation 1.5 is known as Bayes' rule and lies at the heart of the seminal IBM word-based models [Brown et al., 1993] and early phrase-based models [Koehn et al., 2003, Marcu and Wong, 2002]. It may not be immediately obvious how applying Bayes' rule can help. After all, for sentences s and t in two arbitrarily chosen natural languages, there is no reason to think that modeling $p(s|t)$ is likely to be any more straightforward than modeling $p(t|s)$, and now there is an additional term, $p(t)$. The motivation for this approach lay in its successful use in the field of automatic speech recognition, where a distribution analogous to $p(t)$ had been modeled using a n-gram language model. Intuitively, it might seem that modeling two distributions, $p(s|t)$ and $p(t)$, instead of one, $p(t|s)$, will lead to a weaker model due to the increasing scope for the introduction of errors. Instead, automatic speech recognition had shown that modeling two components independently led to a more robust model. In the absence of a reliable model of the translation distribution (either $p(t|s)$ or $p(s|t)$), the addition of a language model component proves essential for finding well-formed translations. The language model can be seen as a filter that weeds out ill-formed candidate strings.

It is at this point that the formulation of word-based, phrase-based, and syntax-based models diverges. Let us continue with the syntax-based model by Galley et al. [2006], which as-

[4]Taken literally, the noisy channel approach interprets translation as the task of recovering a corrupted message: the string s is taken to be a version of t that has been garbled during its transmission across a noisy channel (i.e., its expression in the source language). Given a source string, s, the task is to recover the most probable target string, t, given what we know probabilistically about the action of the noisy communication channel and what we know about the probability distribution of strings in the target language.

sumes that we are using a STSG-based translation grammar[5] (for presentations of word-based and phrase-based models, see Brown et al. [1993] and Koehn et al. [2003]).

Applying Bayes' rule splits our model into two components, a translation model, $p(s|t)$, and a language model, $p(t)$. Galley et al. introduced a third component, $p(\pi|t)$, which we will call the tree model. They do this by splitting the translation model through the application of the sum rule:

$$p(s|t) = \sum_{\pi} p(s|\pi)p(\pi|t), \tag{1.7}$$

where π is any target-side derived tree occurring in the set of all derivations in G. (Recall that in STSG a derived tree is distinct from a derivation tree.) Notice that the term $p(\pi|t)$ is exactly the distribution that is modeled by a probabilistic parsing model, such as a PCFG. The motivation for adding this term is the same as for the introduction of the language model: to increase the robustness of the overall model by injecting a source of external modeling knowledge. Loosely speaking, the role of the tree model is to ensure that, all else being equal, translations for which there are plausible parse trees get higher probabilities than translations for which the parse trees are ill-formed.

Now, what about the term $p(s|\pi)$? In general, a single target-side derived tree π can result from multiple STSG derivations, even if the source sentence s is fixed. If we write $D(s, \pi)$ to denote the set of all derivations in G for which the source yield is s and the target derived tree is π then

$$p(s|\pi) = \sum_{d} p(d|\pi), \tag{1.8}$$

where $d \in D(s, \pi)$. To model the term $p(d|\pi)$, Galley et al. [2006] make the assumption that a derivation's rules are independent of one another. This is the standard independence assumption that we encountered in Section 1.2.4.

$$p(s|\pi) = \frac{1}{Z} \sum_{d} \prod_{r} p(\alpha|\beta), \tag{1.9}$$

where $r \in d$ and α and β are the source and target components of rule r. Notice the introduction of the term $1/Z$. This is a normalization factor, which is required to ensure that $\sum_{s'} p(s'|\pi) = 1$ (where s' is any source sentence in G).

Galley et al.'s presentation goes on to propose two methods for estimating $p(\alpha|\beta)$ from data, one based on relative frequency estimation and the other based on expectation maximization. In a typical training scenario, estimation is non-trivial since only sentence pairs are provided, not their derivations. For the remaining details of the model, we refer the reader to the original paper.

The example here demonstrates the distinguishing features of generative syntax-based models.

[5] Technically, Galley et al. [2006] assume a tree transducer, but the distinction is unimportant here.