Abstract

Title of dissertation: A FORMAL MODEL OF AMBIGUITY AND ITS APPLICATIONS IN MACHINE TRANSLATION

Christopher Dyer, Doctor of Philosophy, 2010

Dissertation directed by: Professor Philip Resnik
Department of Linguistics and
Institute for Advanced Computer Studies

Systems that process natural language must cope with and resolve ambiguity. In this dissertation, a model of language processing is advocated in which multiple inputs and multiple analyses of inputs are considered concurrently and a single analysis is only a last resort. Compared to conventional models, this approach can be understood as replacing single-element inputs and outputs with *weighted sets* of inputs and outputs. Although processing components must deal with sets (rather than individual elements), constraints are imposed on the elements of these sets, and the representations from existing models may be reused. However, to deal efficiently with large (or infinite) sets, compact representations of sets that share structure between elements, such as weighted finite-state transducers and synchronous context-free grammars, are necessary. These representations and algorithms for manipulating them are discussed in depth in depth.

To establish the effectiveness and tractability of the proposed processing model, it is applied to several problems in machine translation. Starting with spoken language translation, it is shown that translating a *set* of transcription hypotheses yields better translations compared to a baseline in which a single (1-best) transcription hypothesis is selected and then translated, independent of the translation model formalism used. More subtle forms of ambiguity that arise even in text-only translation (such as decisions conventionally made during system development about how to preprocess text) are then discussed, and it is shown that the ambiguity-preserving paradigm can be employed in these cases as well, again leading to improved translation quality. A model for supervised learning that learns from training data where sets (rather than single elements) of correct labels are provided for each training instance and use it to learn a model of compound word segmentation is also introduced, which is used as a preprocessing step in machine translation.

A Formal Model of Ambiguity and its Applications in Machine Translation

by

Christopher James Dyer

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2010

For my friends and colleagues.

Acknowledgements

Anything I write here will be an inadequate expression of the thanks owed to many people who helped me complete this dissertation. My advisor, Philip, has been a constant source of ideas (brainstorming with him is like drinking from a fire-hose); but, his most important contribution has been his enthusiasm for my work, and his ability to convince me of the possibilities that it holds. He is especially thanked for patiently suffering through too many very rough drafts of this document, as well as waiting too long for the final draft.

The project leader from the CLSP Summer Workshop at Johns Hopkins in 2006, Philipp, is probably more directly responsible than anyone else for me ending up with this thesis topic. He supported me as an unofficial advisor, as well as occasional benefactor, funding a couple trips to conferences, workshops, and a three month stint in his lab at the University of Edinburgh. While in Edinburgh, I worked with many terrific collaborators and friends (Abhishek, Abby, Barry, Hieu, Josh, Lexi, Sharon, Trevor, and Miles). In particular among my Edinburgh colleagues, Phil Blunsom deserves mention for his influence on this work. Not only did he convince me to bring more rigor and precision to what I do, but he taught me most of what I know about CRFs and Bayesian inference.

From the 2006 workshop, I must also thank Nicola Bertoldi, Marcello Federico, and Richard Zens. Although I had nothing to do with their project, I was so intrigued by their work adding confusion network decoding to Moses (and the modeling possibilities that it afforded) that I decided to replicate their work with Hiero, leading more or less directly to the core idea of ‘translating ambiguity’ that is explored in this dissertation.

The Maryland linguistics department has been a great place to work these last 5 years. Although my research interests ended up being outside of the mainstream of linguistics, they have been consistently supportive of my work. My colleagues in the CLIP lab (David, Asad, Eric, Michael, Smara, Vlad, Hendra, Yuval, and Jordan) have been good friends and helped shape many ideas in this work. Adam Lopez deserves special mention–the lab has not been the same since he left. Finally, the tireless efforts of the UMIACS support staff (particularly Mark, Fritz, and Janet) have meant that the technical aspects of this work were never hampered by computing issues. I am already dreading future labs without them.

During graduate school, I’ve had a ‘second home’ at the CLSP at JHU, where the faculty (especially Fred, Jason, and Sanjeev) have been extremely supportive of me, despite having no formal affiliation with their institution. Fred Jelinek let me take his course on speech recognition, an experience for which I’m particularly grateful in light of his recent passing, and Jason Eisner has helped me with technical aspects of this work on several occasions. The students there (Markus, Zhifei, Jason, Delip, Juri, Jonny, Anni, Byung-Gyu, and Omar) have been my friends and collaborators. Finally, since joining the faculty two years ago, Chris Callison-Burch has been a good friend and confidant.

I also wish to thank Noah Smith, who has not only given me a job but patiently allowed me to delay my start date and turn in the final draft of my thesis after starting. This was done with far less grumbling about it than was deserved. His comments on earlier portions of this thesis have also been invaluable.

Most especially, I would thank my friends, Herb, George, Todd, and Matt, for helping me to hold on to a few shreds of sanity during this whole adventure. When I'm able to have a life again, I look forward to seeing them.

This research was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-001. Any opinions, findings, conclusions or recommendations expressed in this work are those of the author and do not necessarily reflect the view of DARPA. Further support was provided by the EuroMatrix project funded by the European Commission (7th Framework Programme), the Army Research Laboratory, and the National Science Foundation under contract IIS-0838801.

Contents

List of Tables

List of Figures

1 Introduction

Theorists are apt to vex themselves with vain efforts to remove uncertainty just where it has a high aesthetic value.

– Donald Francis Tovey (1935)

Neurosis is the inability to tolerate ambiguity.

–Sigmund Freud (1856–1939)

The quest for certainty blocks the search for meaning.

–Erich Fromm (1900–1980)

Most problems in natural language processing can be viewed as the transformation of one input into one output. A parser transforms a sentence into a parse tree; a speech recognizer transform an acoustic signal into text; and machine translation transforms text in one language into another. This dissertation argues for going beyond this functional relationship (i.e., choosing a single output for every input), because the ideal output of a processing component is often inherently underdetermined by its input: for a single input, there may be many possible correct analyses. In other words, it is (and often should be!)

ambiguous what the correct output is.[1]

With the rise of empirical methods, it has become commonplace to deal with the problem of underdetermined outputs using statistics from large corpora to determine the most likely analysis, given the information at hand. While this approach delivers reasonable (and even very good) results on average, it will necessarily fail in particular cases. Rather than attempting to improve the average performance with ever richer models constructed from larger corpora, this dissertation advocates an alternative paradigm: considering multiple analyses concurrently using standard processing models, but only committing to a single one as a last resort, typically after processing by a cascade of independent modules. Thus, rather than accepting single inputs and producing single outputs, processing components are formalized as applying transformations to *sets* of inputs and yielding *sets* of outputs.

To provide a rigorous, but general framework, this ambiguity-preserving processing model is defined abstractly in terms of weighted sets and relations. However, it can be instantiated using familiar computational constructs: finite-state automata and context-free grammars (together with their transducer equivalents). The remainder of the dissertation is organized around a series of experiments (mostly focusing on problems in machine translation) that are designed to show that this ambiguity-preserving paradigm is both useful and tractable. In addition to the empirical verification of this model, a number

[1]The term ambiguity will be used more broadly than it is traditionally used in computational linguistics. There, it most often refers to specific phenomena such as structural ambiguity, where more than one structural description can characterize a sentence, or lexical semantic ambiguity, where a word in isolation may have many meanings that are resolved with the incorporation of more contextual information (Allen, 1987). Here, I refer to ambiguity as the existence of multiple possible analyses that are compatible with an input, as well as alternative possible *inputs* (typically derived from some ambiguous upstream processing module). The 'classical' cases of ambiguity, as well as their resolution through the incorporation of more knowledge, are naturally modeled in the framework utilized in this dissertation.

of novel algorithms are introduced, as well as contributions to several topics in machine learning.

An outline of the structure of the dissertation is now given.

1.1 Outline of the dissertation

In Chapter 2 the ambiguity-preserving processing model is defined rigorously in terms of weighted sets, binary relations, and operations over them. Under this model, decisions about committing to an analysis are separated from the processing of values, and multiple (i.e., ambiguous) values are handled naturally. While the definition is quite abstract, I show that two classes of familiar formal objects serve as concrete instantiations of it: weighted finite-state automata (WFSAs) and transducers (WFSTs), and weighted context-free grammars (WCFGs) and synchronous context-free grammars (WSCFGs). The chapter also includes a description of a new algorithm for computing the composition of a WFST and a WSCFG, and describes how the problem of synchronous parsing can be viewed as two successive WFST-WSCFG composition operations, leading to a novel synchronous parsing algorithm. These algorithms are used in the remainder of the dissertation to provide efficient and practical implementations of the model's fundamental operations.

Chapters 3, 4, and 5 are organized around experiments designed to show the effectiveness and tractability of the model introduced in Chapter 2. In each, I start with an existing processing model that is 'overly hasty' in resolving ambiguities, which is then recast in terms of the ambiguity-preserving model, and provide an experimental compari-

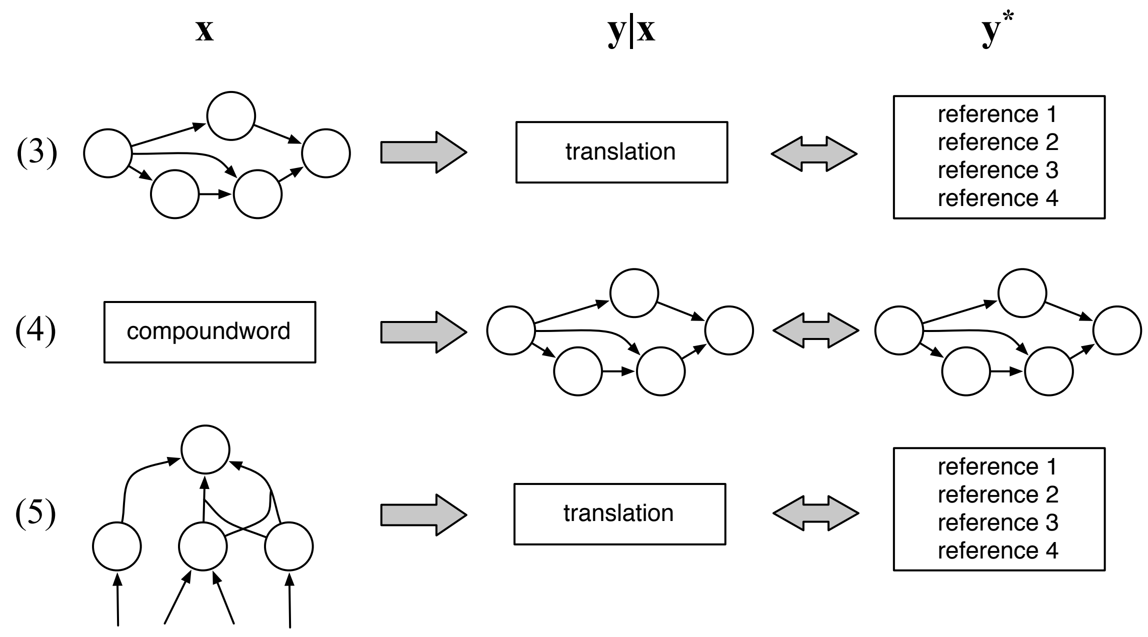


Figure 1.1: Comparison of the forms of the inputs ($\mathbf{x}$), outputs ($\mathbf{y}|\mathbf{x}$), and of the references in the training data ($\mathbf{y}^*$) in Chapters (3), (4), and (5).

son of the performance of the two models. In Chapter 3, I show translation of ambiguous inputs that have a finite-state structure using both WFST-based and WSCFG-based translation models. In Chapter 4, I revisit the supervised learning problem and consider how it can be altered to deal with multiple correct labels for each training instance, which are encoded compactly using finite-state representations. Chapter 5 then returns to the problem of translating input ambiguity; however, this time I explore the possibilities available when the input has a *context-free* structure. In all cases, the ambiguity-preserving model outperforms baselines. Figure 1.1 emphasizes the common elements of the three 'empirical' chapters, comparing the form of the inputs, outputs, and training references in the ambiguity-preserving variant of the processing model considered in that chapter.

Chapter 6 discusses extensions of the work presented in the dissertation as a whole and draws general conclusions.

1.2 Research contributions

This dissertation makes a number of novel contributions in three broad areas: formal foundations of NLP, machine learning, and applications (synchronous parsing, machine translation, and word segmentation).

1.2.1 Formal foundations

- I define an algebra of weighted sets, based on general semirings, and suitable for characterizing the kinds of ambiguity objects and relations that are encountered in language, and use it to define a general model of language processing based on the composition of binary relations. This formalization is independent of any particular grammatical or processing formalism, allowing the behavior of processing pipelines to be characterized abstractly.
- I show that weighted finite-state automata (WFSAs) and transducers (WFSTs) and weighted context free grammars (WCFGs) and synchronous context free grammars (WSCFGs) are concrete instantiations of sets and relations in this system.
- I give novel algorithms for computing the weighted composition of one binary relation represented as a WFST and another as a WSCFG.

1.2.2 Machine learning

- I introduce the upper envelope semiring and show that the line search used in minimum error rate training (Och, 2003) can be reformulated in terms of this semiring, which highlights its relationship to many other inference algorithms.

- I show how that the training of conditional random fields (CRFs) can be carried out efficiently when there multiple reference labels (for each training instance) are compactly encoded in a word lattice.

1.2.3 Applications

- I show that synchronous parsing (recognizing a string *pair* in two languages using a WSCFG) can be formulated as two successive (weighted) composition operations, rather than as a more specialized algorithms (that can be understood as computing a 3-way composition). I give experimental results showing that this algorithm is more efficient than the specialized algorithms on two important classes of WSCFG.

- I show that using a WFSA (specifically, a restricted WFSA called a word lattice) representing a set of ambiguous input possibilities as the translation system produces better translation quality compared to making a forced choice to select a single input sentence to be used. This result holds whether the translation model uses a finite-state or context-free translation model.

- I show that text-only translation systems have input ambiguity that can be encoded in a WFSA—decisions about stemming, segmentation, and other kinds of preprocessing can be treated as part of the model, leading to improved translation quality.

- I describe how to use a semi-CRF model to build segmentation lattices which decompose compound words into the smaller units (morphemes or smaller compounds) that are useful for translation.

- I show that by using dense, linguistically motivated features in the segmentation model, a model trained on German training data works effectively on Turkish and Hungarian.

- Exploiting the fact that permutations can be compactly encoded in a context-free structure, I introduce a novel translation model that reorders the source language into a target-like order in the first phase and performs lexical transduction in the second. This model achieves state-of-the-art performance.

2 A formal model of ambiguity in language and language processing

John saw the man with the telescope.

–Syntax 101

Mathematics takes us still further from what is human, into the region of absolute necessity, to which not only the actual world, but every possible world, must conform.

–Bertrand Russell

Ambiguity is pervasive in language. A single sentence may be compatible with multiple structural descriptions corresponding to different logical forms, individual words have multiple meanings, phonemes are realized as different phones in different contexts, and the acoustic features of phones vary again by context as well as from speaker to speaker. Thus, effective language processing (whether attempting to determine what words were spoken, what the intended meaning of a particular utterance was, or anything in between) must be concerned to a large extent with resolving ambiguity.

In the last decades, the use of statistical models in natural language processing has become commonplace. One reason for the success of such models is that they are a natural fit for processes where ambiguity is found. Alternative (i.e., ambiguous) outcomes can be characterized with a probability distribution reflecting the likelihood that any particular analysis is the correct one, conditioning provides a mathematically rigorous means for incorporating knowledge, and statistical decision theory (Berger, 1985) provides a robust theoretical framework for selecting an analysis or interpreting the results.

Statistical modeling has led to considerable advances in the power and robustness of virtually every kind of language processing component, from part-of-speech taggers (DeRose, 1988) to parsing (Collins, 1996) to speech recognition systems (Jelinek, 1998) to machine translation (Koehn, 2009). However, while these systems use probabilistic inference internally to compute a result, it is still commonplace to define the inputs and outputs as single values. In this conception of computation, the relationship between inputs and outputs is one-to-one; there is an appropriate output for each input. Formally, such relationships have the form of a *function* f of an input x from domain $\mathcal{X}$ to an output y in its codomain $\mathcal{Y}$, written $f : \mathcal{X} \rightarrow \mathcal{Y}$. While this is certainly useful and often necessary behavior in many cases, it is problematic in others.

1. In pipelines or networks of probabilistic processing components, if each component only propagates its 'best guess', upstream errors may flow downstream, leading to a compounding error rates.

2. For some tasks, the idea of an unambiguous input is inherently problematic. For example, the objective of translation may be fairly characterized expressing the *meaning* that underlies text in a source language in some target language. However, the observed sentence is only one possible expression of that underlying meaning.

More mundanely, ambiguity with respect to *preprocessing* (such as how to divide a character sequence in a language like Chinese whose orthography does not indicate word boundaries) can be understood as a kind of ambiguity. Both of these considerations suggest that a more proper input to a translation system is a distribution over sentences.

3. Supervised learning techniques for statistical models require pairs of inputs and labels. However, for many tasks, it may be problematic to identify a single correct label. To continue with the conception of translation as a task that transforms the meaning expressed in a source language into an expression in the target language that corresponds to the same meaning, it is obvious that the inputs and outputs are better characterized by distributions, not single labels.

Recent work has begun to address the issues associated with propagating uncertainty in pipelines of processing components (Bunescu, 2008; Finkel et al., 2006; Ramshaw et al., 2001). Furthermore, one can understand Bayesian inference, which has been widely used for the modeling of language in the last ten years, as a technique for reasoning about uncertainty. In the Bayesian framework, a probability reflects the degree of belief about the truth or falsity of a hypothesis, and in inference, evidence is incorporated (via Bayes' Law) to update these beliefs (Jaynes and Bretthorst, 2003). Crucially, full distributions rather than single hypotheses are maintained throughout.

However, the latter two points in the above list have been mostly neglected, and the common assumption that is made is that processing components naively assume that their inputs or outputs must be reduced to a single, unambiguous value. This work addresses all three problematic results of this assumption by permitting *weighted sets* of inputs and outputs.

This first part of this chapter (§2.1) rigorously defines weighted sets, binary relations, and operations over them. By defining transduction operations and inference processes in terms of set theoretic primitives (rather than specific classes of sets derived from particular kinds of automata, as is commonly done), modeling issues can be explored independently of their realization in particular algorithms and data structures. The intention is that results obtained will be meaningful for other realizations of the theoretical primitives that are not considered explicitly.

Since the focus of this thesis is primarily on the problems associated with translation between languages, many sets and relations of interest are infinite in size (since they represent entire languages or relationships between languages). I therefore focus on two widely used representations of infinite sets and relations: weighted finite-state transducers (WFSTs) and weighted synchronous context-free grammars (WSCFGs). After providing definitions of these objects, I describe their closure properties and then introduce several algorithms that manipulate these objects.

2.1 Formal preliminaries

In the course of this work, several different representations of ambiguous inputs and their use with transduction components that produce ambiguous outputs are explored. Since it is convenient to discuss *models* of ambiguity processing without reference to specific representations or algorithms, I develop a set of mathematical primitives that enable models and inference to be described generally. While probability theory offers one possible framework, I instead opt to describe models and inference using weighted sets that are manipulated using set theoretic operations. Not only can probabilistic models be instantiated in this framework, but non-probabilistic models (which may have advantages in some cases) can also be rigorously defined. Since language processing often deals with large (and even infinite) sets and relations, it is useful to be able to appeal directly to formal language and automata theory, which provide tools for representing and processing sets of this magnitude. The mathematical framework used here makes this connection explicit.

Weighted sets (of sentences, analyses, translations, etc.) are used to represent ambiguous alternatives in inputs and outputs in my model of ambiguity processing. I first begin by defining the characteristics of the weights used by requiring that they form a semiring (§2.1.1). Although many of the models considered will be probabilistic, the weights associated with elements of sets and relations are defined as generally as possible, which enables a number of computations and model types to be expressed using the same basic structures and algorithms together with different semirings. Processing modules are defined as weighted *binary relations* (§2.1.2), which permit a single input element to be associated with many (ambiguous) output elements (Partee et al., 1990). The inference process, that is, constructing a weighted result set given a processing model and some input, is defined in terms of set theoretic operations (§2.1.3). The weighted set algebra I develop can be understood in terms of matrices and operations over matrices (§2.1.4). Although I could develop this ambiguity processing model in terms of matrices, this perspective is more difficult to unify with the automata and language theory work that will be relied heavily upon, so this parallelism is only briefly discussed. This section concludes with the statement of the formal model of ambiguity processing (§2.1.5).

Starting with a definition of weighted sets and relations is somewhat unconventional: weighted sets and relations are more typically constructed in terms of specific generating processes, using concepts from automata theory and formalized as rational power series (Droste and Kuich, 2009; Kuich and Salomaa, 1985). While I will also ultimately utilize such tools to represent weighted sets and relations in later chapters, I wish to frame the problem of processing ambiguous inputs as generally as possible, without making commitments to any specific representation or algorithms. To assist the reader in understanding the formal primitives as well as the processing model I define, a detailed example is given in §2.1.5.2.

2.1.1 Semirings

A semiring is an algebraic structure that I will use to characterize the behavior of weights in weighted sets and relations as various operations (§2.1.3) are carried out over them (Droste and Kuich, 2009).

Definition 1. *A* semiring *K is a quintuple $\langle \mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1} \rangle$ consisting of a set $\mathbb{K}$ (e.g., the reals, natural numbers, $\{0,1\}$, etc.), an addition operator $\oplus$ that is associative and commutative, a multiplication operator $\otimes$ that is associative, and the values $\overline{0}$ and $\overline{1}$ in $\mathbb{K}$, which are the additive and multiplicative identities, respectively. $\otimes$ must distribute over $\oplus$ from the left or right (or both), that is that $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ or $(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$. Additionally, $\overline{0} \otimes u = \overline{0}$ must hold for any $u \in \mathbb{K}$. If a*

Table 2.1: Elements of common semirings.

semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$	notes
Boolean	$\{0,1\}$	$\vee$	$\wedge$	0	1	idempotent
count	$\mathbb{N}_0 \cup \{\infty\}$	$+$	$\times$	0	1	
probability	$\mathbb{R}_+ \cup \{\infty\}$	$+$	$\times$	0	1	
tropical	$\mathbb{R} \cup \{-\infty, \infty\}$	max	$+$	$-\infty$	0	idempotent
log	$\mathbb{R} \cup \{-\infty, \infty\}$	$\oplus_{\log}$	$+$	$-\infty$	0	

semiring K has a commutative $\otimes$ operator, the semiring is said to be commutative. *If K has an idempotent $\oplus$ operator (i.e., $a \oplus a = a$ for all $a \in \mathbb{K}$), then K is said to be* idempotent.

Table 2.1 lists several common semirings (Mohri, 2009).[1] All of these are commutative semirings.

Intuitively, addition operations are associated with set union operations, and multiplication is associated with intersection operations. By altering the semiring, the same set theoretic operations will result in different derived weight functions corresponding to different quantities of interest (for an example, see §2.1.5.2).

In this thesis, all semirings will be commutative. Non-commutative semirings require careful handling when used with the context-free structures I will be working with (Goodman, 1999; Li, 2010).

[1]The operator $\oplus_{\log}$ is defined as:

$$a \oplus_{\log} b = \log\left(e^a + e^b\right)$$

2.1.2 Weighted sets and relations

Definition 2. *A* weighted set $W = \langle A, w \rangle$ *over a semiring* K *is a pair of a set* A *and a weight function* $w : A \to \mathbb{K}$. *By abuse of notation, let* $w(S)$ *be defined where* $S \subseteq A$ *as follows:*

$$w(S) = \begin{cases} \bigoplus_{x \in S} w(x) & |S| > 0 \\ \overline{0} & \textit{otherwise} \end{cases} \tag{2.1}$$

For most of the applications considered below, A will frequently be sets of sentences from a finite vocabulary Σ (i.e., subsets of the free monoid Σ^*). Depending on the model and application, weights will be defined in differently, but often they will represent a probability distribution such that $\mathbb{K} = \mathbb{R}_+ \cup \{\infty\}$ and $\sum_{x \in A} w(x) = w(A) = 1$.

Definition 3. *A* weighted binary relation $\langle R, u \rangle$ *is a weighted set over semiring* K *where* $R \subseteq \mathcal{X} \times \mathcal{Y}$ *specifies how elements from domain* $\mathcal{X}$ *map into codomain* $\mathcal{Y}$ *as with a weight function* $u : R \to \mathbb{K}$.

I will also refer to this as a *weighted transducer*.[2] The weight set $\mathbb{K}$ is defined generally and may be used for a variety of purposes. Often I will use $\mathbb{K} = \mathbb{R}_+$ where values represent probability densities of various kinds (either joint probabilities of the events $p(x,y)$ or conditional probabilities $p(y|x)$ or $p(x|y)$).

Although these concepts are defined for general sets, I focus in particular on subsets of Σ^* and relations that are subsets of $\Sigma^* \times \Delta^*$, where Σ and Δ are finite vocabularies from different languages.

[2]This is not to be confused with a weighted finite-state transducer (§2.2.1), which is a particular representation of a weighted binary relation.

It will often be useful to treat a set A with weight function $u : A \to \mathbb{K}$ as a relation R_A with weight function $w : R_A \to \mathbb{K}$ where:

$$\begin{aligned} w(x,x) &= u(x) \\ R_A \subseteq A \times A &= \{(x,y) : x = y\} \end{aligned}$$

I will refer to this construction as an identity-relation or identity-transducer.

2.1.3 Operations over weighted sets and relations

With this formal representation for ambiguous inputs and relations, several operations that manipulate sets and relations may be defined: union (§2.1.3.1), intersection (§2.1.3.2), projection (§2.1.3.3), composition (§2.1.3.4), and inversion (§2.1.3.5). The processing of ambiguous inputs and pipelines of ambiguous transduction components (§2.1.5) will be defined using these operations and the representations from above.

2.1.3.1 Weighted union

Definition 4. *Given weighted sets $\langle A,u\rangle$ and $\langle B,v\rangle$ over semiring K, let the* weighted union $\langle A,u\rangle \cup \langle B,v\rangle = \langle A \cup B, w : A \cup B \to \mathbb{K}\rangle$ *be defined as follows:*[3]

[3]In general, the domain of a weight function is clear. In some cases it is useful to be able to assign a weight of $\overline{0}$ to values outside of this set.

$$
\begin{aligned}
A \cup B &= \{x : x \in A \vee x \in B\} \\
w(x) &= \begin{cases} u(x) \oplus v(x) & x \in A \wedge x \in B \\ u(x) & x \in A \wedge x \notin B \\ v(x) & x \notin A \wedge x \in B \end{cases}
\end{aligned}
$$

Note that in standard set theory, union is an idempotent operation, that is $A \cup A = A$. While this continues to be true with respect to set membership, a union operation according to this definition induces a new weight function which may differ from that of the operands, unless K is idempotent (§2.1.1).

2.1.3.2 Weighted intersection

Definition 5. *Given weighted sets $\langle A, u\rangle$ and $\langle B, v\rangle$ over semiring K, let the* weighted intersection $\langle A, u\rangle \cap \langle B, v\rangle = \langle C, w : C \to \mathbb{K}\rangle$ *be defined as follows:*

$$
\begin{aligned}
A \cap B &= \{x : x \in A \wedge x \in B\} \\
w(x) &= \begin{cases} u(x) \otimes v(x) & x \in A \wedge x \in B \\ \overline{0} & \textit{otherwise} \end{cases}
\end{aligned}
$$

One particularly useful application of intersection is the application of a language model (§3.1.1) in a noisy channel speech recognition system (Jelinek, 1998). For some utterance **u** (which is a vector of acoustic observations), let A be a set of sentences that a recognizer is capable of recognizing with a weight function u that assigns the likelihood that each element, a sentence **w**, generated the utterance: $p(\mathbf{u}|\mathbf{w})$. If B is a language model, that is, the set of all sentences in the recognition language with a weight function v that represents the (prior) probability of each sentence occurring in the language, then $\langle A,u\rangle \cap \langle B,v\rangle$ computes the set of sentences corresponding to the utterance **u**, weighted by their posterior probabilities.

2.1.3.3 Weighted projection

Definition 6. *Given a weighted binary relation $\langle R,u\rangle$ over semiring K where $R \subseteq \mathcal{X} \times \mathcal{Y}$, one may* project *$R$ onto its domain or codomain, resulting in a weighted set. Projection onto its codomain (or* output projection*), $\langle R,u\rangle{\downarrow} = \langle A,w\rangle$ is defined as follows.*

$$\begin{aligned} A &= \{y \mid \exists x \in \mathcal{X} : (x,y) \in R\} \qquad (\subseteq \mathcal{Y}) \\ w(y) &= \bigoplus_{x \in \mathcal{X} : (x,y) \in R} u(x,y) \end{aligned}$$

Projection onto the domain (or *input projection*), ${\downarrow}\langle R,u\rangle$, is defined similarly. Also note that in the probability semiring, given a joint probabilistic weighting $u(x,y) = p(x,y)$, projection is equivalent to marginalizing out a variable.

2.1.3.4 Weighted composition

Definition 7. *Given two weighted binary relations $\langle R \subseteq \mathcal{X} \times \mathcal{Z}, u : R \to \mathbb{K} \rangle$ and $\langle S \subseteq \mathcal{Z} \times \mathcal{Y}, v : S \to \mathbb{K} \rangle$ over semiring K, their* composition $\langle R, u \rangle \circ \langle S, v \rangle = \langle R \circ S, w : R \circ S \to \mathbb{K} \rangle$ *is defined as follows.*

$$\begin{aligned} R \circ S &= \{(x,y) \in \mathcal{X} \times \mathcal{Y} \mid \exists z \in \mathcal{Z} : (x,z) \in R \wedge (z,y) \in S\} \subseteq \mathcal{X} \times \mathcal{Y} \\ w(x,y) &= \bigoplus_{z \in \mathcal{Z} : (x,z) \in R \wedge (z,y) \in S} u(x,z) \otimes v(z,y) \end{aligned}$$

From this definition, it is not difficult to show that composition is associative, i.e., $(S \circ R) \circ T = S \circ (R \circ T)$.

Furthermore, Assuming that X and Y are conditionally independent given Z, composition is equivalent to marginalizing out a latent variable. If $p(z|y) = w(y,z)$ and $p(x,y) = w(x,y)$ then $p(x,y,z) = w(x,y) \cdot w(y,z)$ and $p(x,z) = w(x,z)$. Likewise if $p(z|y) = w(y,z)$ and $p(y|x) = w(y,x)$ then $p(z,y|x) = w(x,y) \cdot w(y,z)$ and $p(z|x) = w(x,z)$.

Intuitively, weighted composition can be thought of as finding a mapping from $\mathcal{X}$ to $\mathcal{Z}$, summing over all paths taken through some intermediate step $\mathcal{Y}$. As an illustration $\mathcal{X}$ may represent sentences in a source language, $\mathcal{Y}$ may be source sentences divided into phrases, and $\mathcal{Z}$ target language sentences.

Further note that composition is a generalization of the intersection operation described above.

2.1.3.5 Weighted inversion

Definition 8. *Given a weighted binary relation $\langle R \subseteq \mathcal{X} \times \mathcal{Y}, u : R \to \mathbb{K} \rangle$ over semiring K its inversion $\langle R, u \rangle^{-1} = \langle S, v \rangle$ is defined as follows:*

$$
\begin{aligned}
S &= \{(y,x) : (x,y) \in R\} \subseteq \mathcal{Y} \times \mathcal{X} \\
v(y,x) &= u(x,y)
\end{aligned}
$$

The following theorem is useful since it says that using inversion can be used to switch the order of operands in a composition operation.

Theorem 1. *(Inversion theorem). Given weighted binary relations $\langle R, u \rangle$ and $\langle S, v \rangle$, then*

$$\langle R, u \rangle \circ \langle S, v \rangle = (\langle S, v \rangle^{-1} \circ \langle R, u \rangle^{-1})^{-1} .$$

Proof. The proof follows directly from the definitions of weighted inversion and weighted composition. $\square$

2.1.3.6 Total weight of a set

Definition 9. *The* total weight *of a weighted set $\langle A, u \rangle$ over semiring K is a value $w \in \mathbb{K}$, defined as follows:*

$$w = \bigoplus_{a \in A} u(a)$$

The total weight of a set corresponds to a variety of useful quantities. In the log semiring, it corresponds to the value of the partition function, which can be used to renormalize the weight function into a probability distribution. In the tropical semiring, it is the maximum weight in the set. For sets of infinite cardinality, this quantity may be infinite (which also requires that $\mathbb{K}$ contain infinities) or finite.

2.1.4 Weighted sets as matrices

The weighted sets and operations over them that were just defined can be understood, respectively, as matrices and matrix operations (Bhatia, 1996), where the matrices are indexed by elements from an arbitrary set (or, in the case of binary relations, pairs of elements from two sets) and where the value is the weight function applied to that element or pair. A weighted set is therefore a vector (possibly of infinite dimensionality, or empty), and binary relations are two-dimensional matrices. The identity-relation (§2.1.2) is equivalent to an identity matrix.

The weighted operations defined in the previous section also have matrix theoretic equivalents. Composition is equivalent to matrix multiplication; union is matrix addition; and intersection is the Hadamard product. What is called weighted inversion (of a binary relation) here is equivalent to matrix transposition.[4] I will not rely particularly heavily on

[4]This operation should not be confused with the concept of the *matrix inverse*.

this interpretation, except as a means to more thoroughly explain weighted set calculus. However, it is worth keeping in mind since matrix calculus may provide a useful source of operations, and the automata theory literature remarks on the parallelism (e.g., Mohri (2009)).[5]

I do note that since matrices are usually defined over *fields* and not semirings (fields are semirings with the addition of subtraction and division and the requirement that addition and multiplication commute), many common matrix operations (inverse, determinant, permanent, etc.) have no correspondence in the weighted set algebra defined above.

2.1.5 A general model for ambiguity processing

Using the definitions from the preceding section, it is now possible to formulate a general statement for a processing component that accepts ambiguous inputs and produces ambiguous outputs. It may be helpful to refer to the example in §2.1.5.2 while reading this section. Given a weighted transducer $\langle T, v\rangle$ where $T = \mathcal{X} \times \mathcal{Y}$ and $v : T \to \mathbb{K}$, some ambiguous input $\langle I, u\rangle$ where $I \subseteq \mathcal{X}$ is a set of ambiguous inputs and $u : I \to \mathbb{K}$ is its weighting, let $\langle O, w\rangle$, the output set $O \subseteq \mathcal{Y}$ with weighting $w : O \to \mathbb{K}$ be defined as follows:

$$\langle O, w\rangle = (\langle I, u\rangle \circ \langle T, v\rangle)\downarrow$$

[5]Although weighted synchronous context-free grammars can be used as weighted binary relations (§2.2.2), and the composition operation with finite automata is comparable to a generalized parsing algorithm (§2.3.1), the correspondences between Boolean matrix multiplication and parsing explored by Lee (2002) are not directly related to this view of weighted sets as matrices and composition as matrix multiplication. In Lee's formulation, Boolean matrix multiplication is used to formalize the parsing process (specifically, the search for all possible sub-spans that can derive a larger span), whereas here it is used as a means of transforming an entire language (not just a single sentence) via a relation.

The specific values computed will, of course, depend on the semiring used for the composition and projection operations.

The 'greedy' pipeline approach that is often taken in naïve architectures, where a single 'best' analysis (§2.1.5.1) is selected from among a set of ambiguous inputs and used as if it were certain, can be understood as a special case where $|I| = 1$.

Note that this computation has a probabilistic interpretation if the semiring used for the composition and projection operations is the probability semiring. Let I be weighted according to a distribution $u(x) = p(x|\mathbf{o})$ where x is the output of, for example, a preprocessing step with input $\mathbf{o}$, and T is a transducer is weighted according to a conditional distribution $p(y|x)$; then the composition and projection computation produces an output set O that is weighted according to the posterior distribution $p(y|\mathbf{o}) = \sum_x p(y|x) \cdot p(x|\mathbf{o})$. When used in a pipeline where the output of one module becomes the input to the next, this corresponds to the pipeline model advocated by Finkel et al. (2006).

2.1.5.1 Decision rules

It is often necessary to select a single 'best' value from a weighted set $\langle A, w \rangle$. The strategy used is referred to as a *decision rule*. A very commonly used one is the maximum weight rule, which says to select the element with the maximum weight:

$$\hat{x} = \arg\max_{x \in A} w(x)$$

Note that this decision rule requires that the weights assigned by w have a *partial ordering*. While this is not a requirement for the semirings used with weighted sets, many of the value sets (the $\mathbb{K}$'s) used do fulfill this requirement.

Aside from maximum weight, other decision rules are possible. One alternative criterion that is often used in a variety of applications is to select an element from the set so as to minimize risk (expectation of loss) with respect to a specified loss function (Kumar and Byrne, 2004). This requires that the weighting of the set have a probabilistic interpretation as well as a supplemental loss function $\ell : A \times A \rightarrow \mathbb{R}_+$ that indicates how 'bad' a hypothesis is compared to a reference. The Bayesian minimum risk decision function is defined as follows:

$$\begin{aligned} \hat{x}_{\text{risk}} &= \arg\min_x \sum_{x' \in A} w(x')\ell(x,x') \\ &= \arg\min_x \mathbb{E}_{p(x')}\ell(x,x') \end{aligned}$$

Minimum risk decision rules are useful in cases when the task is to maximize performance with respect to a particular loss function; however, they are generally more expensive to use than maximum weight rules. Minimum risk is equivalent to maximum weight when $\ell(x,x') = 0$ when $x = x'$ and 1 otherwise.

The maximum weight decision rule will be used in all experiments in this dissertation.

2.1.5.2 Example

Consider now a more detailed example of modeling an ambiguous processing pipeline and carrying out inference using the ambiguity processing model defined in §2.1.5. In this section, I define an example distribution over inputs, an example transducer, and compute an output using three techniques: compose and project using the probability semiring, compose and project using the tropical semiring, and a ‘greedy’ processing approach (Figure 2.1).

The example task a system that translates elements from an input language $\mathcal{X}$ (in this case, letters) into an output language $\mathcal{Y}$ (in this case, card suits). However, I assume that direct observations of the sentences in $\mathcal{X}$ are unavailable, instead only a distribution over possibilities is available. Let the distribution over inputs be represented as $I \subseteq \mathcal{X}$ together with a weight function $p : I \to \mathbb{R}_+$ that maps an element $x \in I$ onto a positive real number representing the probability $p(x)$, such that $\sum_{x \in I} p(x) = 1$. To further complicate matters, the translation process is itself ambiguous: even if there was complete certainty about the input, there would be multiple possible outputs. The translation process is therefore represented as a binary relation $R \subseteq \mathcal{X} \times \mathcal{Y}$ with a weight function $w : R \to \mathbb{R}_+$ that maps pairs of inputs and outputs $(x, y) \in R$ onto a positive real number representing the conditional probability $p(y|x)$, that is, the probability that y is the the desired translation given input x. Let $\mathcal{X}$, $\mathcal{Y}$, I, and R be defined as shown below. The complement relation of

R (designated S) is given for clarity.

$$
\begin{aligned}
\mathcal{X} &= \{\texttt{a}, \texttt{b}, \texttt{c}\} \\
\mathcal{Y} &= \{\spadesuit, \clubsuit, \blacklozenge\} \\
I &= \mathcal{X} \\
R &= \{\langle \texttt{a}, \spadesuit\rangle, \langle \texttt{a}, \clubsuit\rangle, \langle \texttt{a}, \blacklozenge\rangle, \langle \texttt{b}, \clubsuit\rangle, \langle \texttt{b}, \blacklozenge\rangle, \langle \texttt{c}, \spadesuit\rangle, \langle \texttt{c}, \blacklozenge\rangle\} \subseteq \mathcal{X} \times \mathcal{Y} \\
S = (\mathcal{X} \times \mathcal{Y}) \setminus R &= \{\langle \texttt{b}, \spadesuit\rangle, \langle \texttt{c}, \clubsuit\rangle\}
\end{aligned}
$$

Let the weight functions p and w be defined as indicated in Figure 2.1. Given I and R and their weight functions, I compare three different possibilities to compute an output set $O \subseteq \mathcal{Y}$ and its weight function $q : O \to \mathbb{R}_+$. For each, the 'the best' translation according using the maximum weight decision rule will be selected.[6]

I distinguish three approaches for computing O: (1) greedy decision making, (2) weighted composition using the probability semiring, and (3) weighted composition using the tropical semiring. For reference, the elements of the probability and tropical semirings are shown again here in Table 2.2. Note that they differ only in the definition of the $\oplus$ operator.

Table 2.2: Elements of the probability and tropical semirings.

semiring	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
probability	$\mathbb{R}_+$	$+$	$\times$	0	1
tropical	$\mathbb{R}_+$	max	$\times$	0	1

[6]This decision rule can only be used when the range of the weight function is a *partially ordered set*. This is not a requirement of the value set in a semiring.

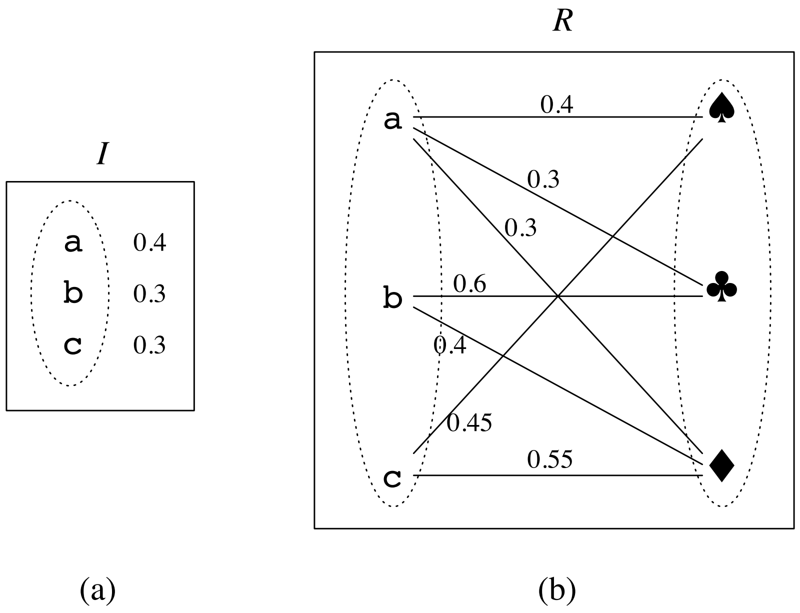


Figure 2.1: A weighted set representing a distribution over inputs (a) and a weighted relation representing a transducer (b).

Greedy processing. As noted above, greedy processing refers to a strategy that uses a decision rule to select a single input from among ambiguous ones and further processing is carried out as if this were an unambiguous input. I use the maximum weight decision rule, which in the case of the example selects a. If a is treated as the unambiguous input to the transducer in Figure 2.1(b), then the output set is the set consisting of the right-hand component of all elements in R whose left-hand component is a. Thus, $O = \{♠, ♣, ♦\}$ and the weight function can be defined to be the weighting from the corresponding elements in $\mathcal{Y}$, that is $q(y) = w(\mathtt{a}, y)$. The first line in Table 2.4 gives the posterior weighting on output symbols computed using this strategy. Using the maximum weight decision rule on $\langle O, q \rangle$ yields the output $\hat{y} = ♠$.

Compose and project using the probability semiring. The compose and project strategy is more intuitively satisfying because it does not throw away information the way the greedy processing approach does. I begin with an example where K is the probability semiring. Using the sets and weights specified in Figure 2.1, the result is the same output set as with the greedy processing rule $O = \{\spadesuit, \clubsuit, \blacklozenge\}$, but a different weight function q_{prob}, shown in the following table.

y		$q_{\text{prob}}(y)$
$\spadesuit$	$0.4 \times 0.4 + 0.3 \times 0.45$	$= 0.295$
$\clubsuit$	$0.4 \times 0.3 + 0.3 \times 0.6$	$= 0.4$
$\blacklozenge$	$0.4 \times 0.3 + 0.3 \times 0.4 + 0.3 \times 0.55$	$= 0.405$

Using the maximum weight decision rule on this set yields the output $\blacklozenge$. Note that the results of this computation are a proper probability distribution (i.e., they sum to 1 and are non-negative).

Compose and project using the tropical semiring. While compose and project using the probability semiring fulfills the requirement of incorporating a distribution over inputs (in contrast to the greedy approach), it is often desirable for efficiency reasons to make use of the tropical semiring, which, rather than summing over all intermediate stages in the transduction, selects the *maximum*.[7] As a result, while the output set is the same as in the previous two cases, its weight function is different yet again, as shown in Table 2.3.

[7]For readers familiar with hidden Markov Models (HMMs; Jelinek (1998)), using the tropical semiring is equivalent to doing decoding by selecting the Viterbi path (the best single path) from an HMM, whereas using the log semiring and maximum weight decision rule is more similar to MAP decoding, which selects the maximum probability state at each time, given the observation sequence.

Table 2.3: The q_{Vit} weight function.

y		$q_{\text{Vit}}(y)$
♠	$\max\{0.4 \times 0.4, 0.3 \times 0.45\}$	$= 0.16$
♣	$\max\{0.4 \times 0.3, 0.3 \times 0.6\}$	$= 0.18$
♦	$\max\{0.4 \times 0.3, 0.3 \times 0.4, 0.3 \times 0.55\}$	$= 0.165$

Summary of processing strategies. The three different processing strategies all produce different results, as the summary in Table 2.4 makes clear. But which one should be utilized? In general, this is a matter of taste or an empirical question: which one works best. However, a priori, the 'greedy' approach seems quite poor, since it uses a single element from the input set to represent the entire set. The probability semiring is arguably more appealing than the tropical semiring since it combines evidence from all input-output pairs, whereas the Viterbi approach just selects a maximum from among all pairs. In a naïve implementation, where sets are represented with an exhaustive list of their members (and therefore necessarily limited to be of certainly finite and probably rather small sizes), there is no difference between the complexity of using the tropical semiring in conjunction with the maximum weight decision rule. But, this will not be the case when automata of various kinds are used to compactly represent sets. In this case, the requirement of only keeping track of a single maximum makes the search for the maximum of the whole set very efficient to compute.

Table 2.4: Output weight computed using different processing strategies. Bold indicates the highest weighted output for a particular strategy.

	♠	♣	♦	$\hat{y}$
greedy	**0.4**	0.3	0.3	♠
probability	0.295	0.4	**0.405**	♦
Viterbi	0.16	**0.18**	0.165	♣

2.2 Tractable representations of weighted sets and relations

In the preceding section, weighted sets, binary relations, and several operations (union, intersection, composition, etc.) were defined, and I showed how ambiguous transduction operations can be formalized in terms of these primitives. I now explore possibilities for compactly representing weighted sets and transducers. In the example from the previous section, the sets and relations used were small enough that they could be represented simply by enumerating each element and its weight explicitly in a list. The operations could be implemented by iterating over the elements in the relevant sets and performing the specified operations. Unfortunately, most of the sets that are encountered in language applications will be far too large (if not infinite) for this naïve representation, so more sophisticated methods must be used.

Fortunately, formal language theory provides a means to define very large weighted sets of strings and relations over strings, as well as the necessary operations. The focus will be on two weighted representations of sets of strings, weighted finite-state automata (WFSAs; §2.2.1) and weighted context-free grammars (WCFGs; §2.2.2). Both of these can be generalized to transducers, enabling them to generate weighted relations between two languages. Finally, using these representations of sets and relations, efficient algorithms exist for implementing all the required operations. While both these representations have been described extensively in previous work (Goodman, 1999; Mohri, 2009), I define them here as specific instantiations of the more fundamental weighted set algebra introduced above. That is, this common set of properties emphasizes that these representations are fungible and, subject to a few restrictions discussed below (§2.2.3), either of them can be used when weighted sets and relations are required.

2.2.1 Weighted finite-state automata and transducers

Definition 10. $A = \langle \Sigma, Q, \langle I, \lambda \rangle, \langle F, \rho \rangle, \langle E, w \rangle \rangle$ *is a* weighted finite-state automaton *over semiring K if* Σ *is a finite input alphabet;* Q *is a finite set of states;* $I \subseteq Q$ *is the weighted set of initial states with weight function* $\lambda : I \to \mathbb{K}$*;* $F \subseteq Q$ *is the weighted set of final or accepting states with weight function* $\rho : F \to \mathbb{K}$*; and* $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ *with weight function* $w : E \to \mathbb{K}$ *is a weighted (finite) set of transitions or edges.*[8]

For an edge (transition) $e \in E$, $i[e]$ denotes its label (in Σ), $p[e]$ its previous state (in Q), $n[e]$ its next state (in Q), and $w[e]$ its weight (in $\mathbb{K}$). For any state $q \in Q$, $E[q]$ denotes the set of transitions leaving q, that is the set of transitions where $\{e \in E : p[e] = q\}$ and which may be $\emptyset$. $\pi = \langle e_1, e_2, \ldots, e_\ell \rangle \in E^*$ is a *path* iff $n[e_{i-1}] = p[e_i]$ for $i \in [2, \ell]$. The functions p, n, i, and w can be extended from edges to paths as follows: $p[\pi] = p[e_1]$, $n[\pi] = n[e_\ell]$, $i[\pi]$ is the concatenation of labels $i[e_1]i[e_2] \cdots i[e_\ell]$, and $w[\pi] = \bigotimes_{i=1}^{\ell} w[e_i]$. $P(q,r)$ denotes the set of paths from $q \in Q$ to $r \in Q$. $P(q,x,r) = \{\pi \in P(q,r) : i[\pi] = x\}$ where $x \in \Sigma^*$. This definition is generalized to sets $Q' \subseteq Q$ and $R \subseteq Q$ as follows: $P(Q',x,R) = \bigcup_{q \in Q' \wedge r \in R} P(q,x,r)$.

[8]This definition of a WFSA is a slightly modified version of the definition given by Mohri (2009) since it makes use of weighted sets as defined in §2.1.2.

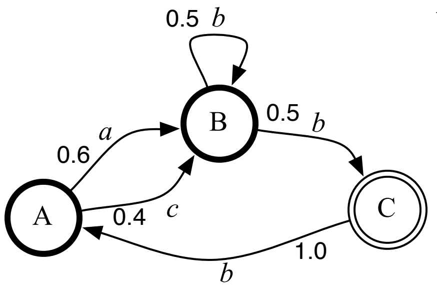


Figure 2.2: A graphical representation of a weighted finite-state automaton.

A weighted automaton A assigns a weight (in $\mathbb{K}$) to every string $x \in \Sigma^*$ as follows:

$$u(x) = \begin{cases} \overline{0} & P(I,x,F) = \emptyset \\ \bigoplus_{\pi \in P(I,x,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi]) & \text{otherwise} \end{cases} \tag{2.2}$$

Let $L(A) \subseteq \Sigma^*$ denote the set of strings accepted by A, that is $L(A) = \{x \in \Sigma^* : P(I,x,F) \neq \emptyset\}$. A weighted finite-state automaton A therefore represents a weighted set $\langle L(A), u \rangle$ over semiring K with u defined as in Equation 2.2.

WFSAs are often represented and illustrated as *directed graphs*, with nodes and edges representing states and transitions, respectively. Figure 2.2 shows an example. In this case, a bold circle indicates an initial state (with the node label indicating $\lambda(q)$) and a double circle indicates a final state (with the node label indicating $\rho(q)$).

Definition 11. *A* weighted finite-state transducer *is a 6-tuple* $\langle \Sigma, \Delta, Q, \langle I, \lambda \rangle, \langle F, \rho \rangle, \langle E, w \rangle \rangle$. Σ *is the finite input alphabet;* Δ *is the finite output alphabet;* Q *is a finite set of states;* $I \subseteq Q$ *is the weighted set of initial states with weight function* $\lambda : I \to \mathbb{K}$*;* $F \subseteq Q$ *is the weighted set (as defined in §2.1.2) of final or accepting states with weight function* $\rho : F \to \mathbb{K}$*; and* $E \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times (\Delta \cup \{\varepsilon\}) \times Q$ *with weight function* $w : E \to \mathbb{K}$ *is a* *weighted (finite) set of transitions or edges.*

A WFST is thus a generalization of a WFSA so that each edge has a label in each of two vocabularies, Σ and Δ. The notion of paths and the functions p, n, i, and w are defined as above, and augmented by the function o which maps edge e or path π to its output symbol ($\in \Delta$) or sequence of output symbols ($\in \Delta^*$). Additionally define $P(q,x,y,r) = \{\pi \in P(q,r) : i[\pi] = x \wedge o[\pi] = y\}$ where $x \in \Sigma^*$ and $y \in \Delta^*$, and let $P(Q',x,y,R)$ be the generalization to sets of states, as above. A weighted finite-state transducer T assigns a weight (in $\mathbb{K}$) to every string pair $(x,y) \in \Sigma^* \times \Delta^*$ as follows:

$$v(x,y) = \begin{cases} \overline{0} & P(I,x,y,F) = \emptyset \\ \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \otimes w[\pi] \otimes \rho(n[\pi]) & \text{otherwise} \end{cases} \tag{2.3}$$

Letting $Rel(T) \subseteq \Sigma^* \times \Delta^*$ denote the set of string pairs accepted by T, that is $Rel(T) = \{(x,y) \in \Sigma^* \times \Delta^* : P(I,x,y,F) \neq \emptyset\}$. A weighted finite-state transducer T therefore represents a weighted binary relation $\langle Rel(T), v\rangle$ over semiring K with v defined as in Equation 2.2.

2.2.2 Weighted context-free grammars and weighted synchronous CFGs

Definition 12. *A* weighted context-free grammar *G is a 4-tuple $\langle \Sigma, V, \langle S, \sigma\rangle, \langle R, \upsilon\rangle\rangle$ over semiring K where Σ is a finite vocabulary of terminal symbols; V is a finite vocabulary of non-terminal variables and $\Sigma \cap V = \emptyset$; $\langle S, \sigma\rangle$ is a weighted set of start symbols where $S \subseteq V$ with weight function $\sigma : S \to \mathbb{K}$; and $\langle R, \upsilon\rangle$ is the weighted set of rewrite rules which* *is a weighted binary relation where $R \subseteq V \times (V \cup \Sigma)^*$ maps from non-terminal variables to sequences of terminal and non-terminal symbols with weight function $\upsilon : R \to \mathbb{K}$.*

For a rewrite rule $r \in R$, let $\textsc{lhs}(r)$ refer to the left-hand side (a symbol in V) of the rewrite rule, and $\textsc{rhs}(r)$ refer to its right-hand side or *yield*, which is a symbol in $(V \cup \Sigma)^*$ and will write $\textsc{lhs}(r) \to \textsc{rhs}_i(r)$. $a[r]$ denotes its arity, which is the number of symbols in V that are in $\textsc{rhs}_i(r)$, and $\upsilon[r]$ is its weight (in $\mathbb{K}$). For strings $u, v \in (V \cup \Sigma)^*$, u yields v, iff $\exists \alpha \to \beta \in R$ and $u_1 \in \Sigma^*$, $u_2 \in (V \cup \Sigma)^*$ such that $u = u_1 \alpha u_2$ and $v = u_1 \beta u_2$; this is written $u \overset{\alpha\to\beta}{\Rightarrow} v$.[9] $\delta = \langle r_1, r_2, \ldots, r_\ell \rangle \in R^*$ is a *derivation* iff $\textsc{lhs}(r_1) \overset{r_1}{\Rightarrow} (V \cup \Sigma)^* \overset{r_2}{\Rightarrow} \cdots \overset{r_\ell}{\Rightarrow} \Sigma^*$, which may be written $\textsc{lhs}(r_1) \overset{\delta}{\Rightarrow} \Sigma^*$. The functions $\textsc{lhs}$, $\textsc{rhs}_i$, and υ can be extended to a derivation δ, where $\textsc{lhs}(\delta) = \textsc{lhs}(r_1)$, $\textsc{rhs}_i(\delta) = \left(\textsc{lhs}(r_1) \overset{\delta}{\Rightarrow} \Sigma^*\right)$, and $\upsilon[\delta] = \bigotimes_{i=1}^{\ell} \upsilon[r_i]$, with $D(q)$ where $q \in V$ denotes the set of derivations rooted at non-terminal q, that is $\{\delta \in R^* : \textsc{lhs}(\delta) = q\}$. Let $D(q, x)$ where $q \in V$ and $x \in \Sigma^*$ represent $\{\delta \in D(q) : \textsc{rhs}_i(\delta) = x\}$. Finally, D is generalized to support sets of non-terminals $Q \subseteq V$ as follows: $D(Q, x) = \bigcup_{q \in Q} D(q, x)$. A weighted context-free grammar G thus assigns a weight (in $\mathbb{K}$) to every string $x \in \Sigma^*$ as follows:

$$w(x) = \begin{cases} \overline{0} & D(S,x) = \emptyset \\ \bigoplus_{\delta \in D(S,x)} \sigma(\textsc{lhs}[\delta]) \otimes \upsilon[\delta] & \text{otherwise} \end{cases} \tag{2.4}$$

Let $L(G) \subseteq \Sigma^*$ denote the set of strings generated by G, that is $L(G) = \{x \in \Sigma^* : D(S, x) \neq \emptyset\}$. A weighted context-free grammar G therefore represents a weighted set $\langle L(G), w \rangle$ over semiring K with w defined as in Equation 2.4.

[9]Rule application is defined always to apply to the *left-most* non-terminal in a sequence. This ensures that unique derivations correspond to unique trees.

The structure of context-free grammars can be understood as a generalization of weighted finite-state automata, and understanding the particulars of this relationship can be illuminating. The non-terminal vocabulary V plays a similar role as the state set Q in a WFSA. Rewrite rules R play a similar role to an the transition set E, in fact, $\textsc{lhs}[r]$ and $n[e]$ can be seen as equivalent. The set of *final* states F in an FSA is equivalent to the *starting* non-terminals S in a CFG.[10] Derivations and paths represent similar concepts. The set of derivations $D(q,x)$ starting a non-terminal q and yielding $x \in \Sigma^*$ is equivalent to the set of paths $P(q,x,F)$ starting in state q, yielding x, and ending in any final state. This similarity helps show that any WFST can be trivially encoded as a WCFG, corresponding to the well-known result that context-free languages are a proper superset of the regular languages. Briefly, states in the WFSA become non-terminals in the WCFG, a transition $e = \langle q,x,r \rangle$ becomes a rewrite rule $r \to qx$, final states in the WFSA become start states in the WCFG, and initial states become epsilon rewrite rules.[11] Figure 2.3 shows the WFSA from Figure 2.2 encoded as a WCFG. Further note that the number of symbols $|G|$ in the equivalent WCFG, G, is in $O(|A|)$ where $|A|$ is the number of symbols in the finite automaton A.

Definition 13. *A* weighted synchronous context-free grammar[12] *is defined to be a 6-tuple*

[10]Most definitions of context-free grammars require that S is a single non-terminal symbol in V, not a subset of non-terminals. I have used a more general definition to emphasize the similarities to FSAs (where there may be many final states); however, this change is merely a notational convenience, it does not alter the generative capacity of the CFG.

[11]This is only one of many possible encodings of a WFSA in a WCFG. For an unweighted encoding, see, for example, Lewis and Papadimitriou (1981).

[12]Non-weighted synchronous context-free grammars were introduced by Lewis and Stearns (1968). In addition to introducing weights, the definition given here makes use of *indexed gaps*, which were not part of the original definition of SCFGs, but which simplify the statement of several algorithms and are useful in system implementation. Using indexed gaps (instead of the original bijective correspondence between input and output non-terminals) does not alter the expressivity of the formalism.

$$\Sigma = \{a,b,c\} \quad V = \{\mathrm{A},\mathrm{B},\mathrm{C}\} \quad S = \{\mathrm{C}\} \quad \sigma(\mathrm{C}) = 1.0$$

$$\begin{aligned}
R = \{ \quad & \mathrm{A} \overset{1.0}{\to} \mathrm{C}\ a, \\
& \mathrm{B} \overset{0.6}{\to} \mathrm{A}\ a, \\
& \mathrm{B} \overset{0.4}{\to} \mathrm{A}\ c, \\
& \mathrm{B} \overset{0.5}{\to} \mathrm{B}\ b, \\
& \mathrm{C} \overset{0.5}{\to} \mathrm{B}\ b, \\
& \mathrm{A} \overset{0.2}{\to} \varepsilon, \\
& \mathrm{B} \overset{0.8}{\to} \varepsilon \quad \}
\end{aligned}$$

Figure 2.3: Encoding the WFSA in Figure 2.2 as a weighted context-free grammar. Note that the start symbol is C.

$\langle \Sigma, \Delta, V, \langle S, \sigma \rangle, \langle R, \upsilon \rangle \rangle$ *over semiring* K *where* Σ *is a finite vocabulary of terminal symbols in the input language;* Δ *is a finite vocabulary of terminal symbols in the output language,* V *is a finite vocabulary of non-terminal variables and* $\Sigma \cap V = \Delta \cap V = \emptyset$*;* $\langle S, \sigma \rangle$ *is a weighted set of start symbols where* $S \subseteq V$ *with weight function* $\sigma : S \to \mathbb{K}$*; and* $\langle R, \upsilon \rangle$ *is the weighted set of input and output rewrite rules which is a weighted relation where* $R \subseteq V \times (V \cup \Sigma)^* \times (\{\boxed{1}, \boxed{2}, \ldots\} \cup \Delta)^*$ *relates non-terminal variables to a pairs* $\langle \beta, \gamma \rangle$*, where* β *is a sequence of non-terminals and terminals from the* input *alphabet;* γ *is a sequence of indexed gaps and terminals from the* output *alphabet; and the pair* $\langle \beta, \gamma \rangle$ *is subject to the* gap correspondence constraint.

Definition 14. *The* gap correspondence constraint *states that an input-output string pair* $\langle \beta, \gamma \rangle \in (V \cup \Sigma)^* \times (\{\boxed{1}, \boxed{2}, \ldots\} \cup \Delta)^*$ *is well-formed iff (1) the number of non-terminal symbols in* β *is equal to the number of indexed gaps in* γ*, and (2) while any non-terminal may occur any number of times* β*, each index used must occur only once in* γ *and the*

$$\Sigma = \{a, b, c\} \quad \Delta = \{x, y, z\} \quad V = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\} \quad S = \{\mathrm{C}\} \quad \sigma(\mathrm{C}) = 1.0$$

$$\begin{array}{rlll} R = \{ & \mathrm{A} & \overset{0.5}{\rightarrow} & \langle \mathrm{A}\ a\ \mathrm{B},\ \boxed{2}\,\boxed{1}\,x\rangle, \\ & \mathrm{A} & \overset{0.5}{\rightarrow} & \langle b\ \mathrm{C},\ y\,\boxed{1}\rangle, \\ & \mathrm{B} & \overset{0.6}{\rightarrow} & \langle a\ b\ c,\ z\ z\rangle, \\ & \mathrm{B} & \overset{0.4}{\rightarrow} & \langle a\ b\ c,\ x\ y\ z\rangle, \\ & \mathrm{C} & \overset{0.8}{\rightarrow} & \langle \mathrm{A}\ \mathrm{A},\ \boxed{1}\,\boxed{2}\rangle, \\ & \mathrm{C} & \overset{0.2}{\rightarrow} & \langle c, z\rangle \qquad\qquad \} \end{array}$$

Figure 2.4: Example of a weighted synchronous context-free grammar (WSCFG). Note that C is the start symbol.

indices used must be contiguous from 1 *to* $a[\gamma]$*, although they may occur in any order.*

Like weighted finite-state transducers, which generalize weighted finite-state automata to generate (or accept) strings in two languages, weighted synchronous context-free grammars (WSCFGs) generalize WCFGs to generate strings in two languages. Figure 2.4 shows an example WSCFG. The functions defined above for WCFGs continue to apply to WSCFGs, and I add an additional function over rules $\textsc{rhs}_o[r]$ that returns the output language yield in $(\{\boxed{1}, \boxed{2}, \ldots\} \cup \Delta)^*$.

Let the function $Y_k : (\{\boxed{1}, \boxed{2}, \ldots\} \cup \Delta)^* \times (\{\boxed{1}, \boxed{2}, \ldots\} \cup \Delta)^*$ be defined so as to add to the index of every gap the integer k, for example $Y_{-1}(a\ b\ \boxed{2}) = a\ b\ \boxed{1}$. For string pairs $\langle u, x\rangle, \langle v, y\rangle \in (V \cup \Sigma)^* \times (\{\boxed{1}, \boxed{2}, \ldots\} \cup \Delta)^*$ with both pairs fulfilling the gap-correspondence constraint, $\langle u, x\rangle$ yields $\langle v, y\rangle$ iff $\exists \alpha \rightarrow \langle \beta, \gamma\rangle \in R$ and $u_1 \in \Sigma^*,\ u_2 \in (V \cup \Sigma)^*$ and $x_1, x_2 \in (\{\boxed{2}, \boxed{3} \ldots\} \cup \Delta)^*$ such that $\langle u, x\rangle = \langle u_1 \alpha u_2, x_1 \boxed{1} x_2\rangle$ and $\langle v, y\rangle = \langle u_1 \beta u_2, Y_{a[\beta]-1}(x_1) \gamma Y_{a[\beta]-1}(x_2)\rangle$. $\delta = \langle r_1, r_2, \ldots, r_\ell\rangle \in R^*$ is a *derivation* iff $\langle \textsc{lhs}(r_1), \boxed{1}\rangle \overset{r_1}{\Rightarrow} \langle (V \cup \Sigma)^*, (\{\boxed{1}, \boxed{2}, \ldots\} \cup \Delta)^*\rangle \overset{r_2}{\Rightarrow} \cdots \overset{r_\ell}{\Rightarrow} \langle \Sigma^*, \Delta^*\rangle$, which may be written $\langle \textsc{lhs}(r_1), \boxed{1}\rangle \overset{\delta}{\Rightarrow}$

Yield	$\alpha \to \langle \beta, \gamma \rangle$	Weight
$\langle \text{C}, \boxed{1} \rangle \Rightarrow$	$\text{C} \to \langle \text{A A}, \boxed{1}\boxed{2} \rangle$	1.0×0.8
$\langle \text{A A}, \boxed{1}\boxed{2} \rangle \Rightarrow$	$\text{A} \to \langle \text{A}\ a\ \text{B}, \boxed{2}\boxed{1}x \rangle$	$\times 0.5$
$\langle \text{A}\ a\ \text{B A}, \boxed{2}\boxed{1}x\boxed{3} \rangle \Rightarrow$	$\text{A} \to \langle b\ \text{C}, y\boxed{1} \rangle$	$\times 0.5$
$\langle b\ \text{C}\ a\ \text{B A}, \boxed{2}y\boxed{1}x\boxed{3} \rangle \Rightarrow$	$\text{C} \to \langle c, z \rangle$	$\times 0.2$
$\langle b\ c\ a\ \text{B A}, \boxed{1}y\,z\,x\boxed{2} \rangle \Rightarrow$	$\text{B} \to \langle a\ b\ c, x\ y\ z \rangle$	$\times 0.4$
$\langle b\ c\ a\ a\ b\ c\ \text{A}, x\,y\,z\,y\,z\,x\boxed{1} \rangle \Rightarrow$	$\text{A} \to \langle b\ \text{C}, y\boxed{1} \rangle$	$\times 0.5$
$\langle b\ c\ a\ a\ b\ c\ b\ \text{C}, x\,y\,z\,y\,z\,x\,y\boxed{1} \rangle \Rightarrow$	$\text{C} \to \langle c, z \rangle$	$\times 0.2$
$\langle b\ c\ a\ a\ b\ c\ b\ c, x\,y\,z\,y\,z\,x\,y\,z \rangle$		$= 0.0016$

Figure 2.5: Example synchronous derivation using the WSCFG shown in Figure 2.4.

$\langle \Sigma^*, \Delta^* \rangle$. Figure 2.5 shows an example WSCFG derivation.

As was done in the monolingual case, the functions $\textsc{lhs}$, $\textsc{rhs}_i$, $\textsc{rhs}_o$ and υ can be extended to a derivation δ. $D(q)$ where $q \in V$ denotes the set of derivations rooted at non-terminal q, that is $\{\delta \in R^* : \textsc{lhs}(\delta) = q\}$. Let $D(q,x,y)$, where $q \in V$ and $\langle x,y \rangle \in \Sigma^* \times \Delta^*$, represent $\{\delta \in D(q) : \textsc{rhs}_i(\delta) = x \wedge \textsc{rhs}_o(\delta) = y\}$. Finally, D is generalized to support sets of non-terminals $Q \subseteq V$ as follows: $D(Q,x,y) = \bigcup_{q \in Q} D(q,x,y)$. A weighted synchronous context-free grammar G assigns a weight (in $\mathbb{K}$) to every string pair $(x,y) \in \Sigma^* \times \Delta^*$ as follows:

$$\upsilon(x,y) = \begin{cases} \bar{0} & D(S,x,y) = \emptyset \\ \bigoplus_{\delta \in D(S,x,y)} \sigma(\textsc{lhs}[\delta]) \otimes \upsilon[\delta] & \text{otherwise} \end{cases} \tag{2.5}$$

Let $Rel(G) \subseteq \Sigma^* \times \Delta^*$ denote the set of string pairs accepted by G, that is $Rel(G) = \{(x,y) \in \Sigma^* \times \Delta^* : D(S,x,y) \neq \emptyset\}$. A weighted synchronous context-free grammar G therefore represents a weighted binary relation $\langle Rel(G), \upsilon \rangle$ over semiring K with υ defined as in Equation 2.5.

2.2.2.1 Hypergraphs as grammars

An *ordered, directed hypergraph* (Gallo et al., 1993; Huang, 2008; Huang and Chiang, 2005) can be used to represent an arbitrary WSCFG in a manner similar to the graphical representation of an FSA described above. An ordered hypergraph generalizes the concept of a directed graph to include edges with a tail represented as a vector comprising zero or more nodes. While some definitions of hypergraphs define the tail nodes to be an unordered set, to ensure a complete isomorphism between hypergraphs and WSCFGs, the same node must be able to appear represented multiple times in the tail of the edge. Furthermore, the order of the nodes in the tail matter; therefore, I define the tail nodes in terms of a vector. Figure 2.6 shows the representation of the WSCFG from Figure 2.4 encoded as a hypergraph (for simplicity, the rule weights and start symbol weights are not shown). Briefly, non-terminal symbols become nodes, rewrite rules become edges (with non-terminal variables corresponding to tails), and the start symbols become goal nodes (indicated with a double circle).

While prior work has tended to focus on hypergraphs as encodings for (typically non-recursive) parse forests—themselves a kind a CFG (§2.3.1)—it should be emphasized that hypergraphs are a means of representing *any* arbitrary WCFG or WSCFG. Furthermore, thinking of a WSCFG as a hypergraph emphasizes the similarity to WFSTs (which can be represented as convention graphs). When I discuss inference algorithms (§2.4), I will rely on a common hypergraph representation of both WSCFGs and WFSTs to express several algorithms that can apply to both classes of objects without any special handling.

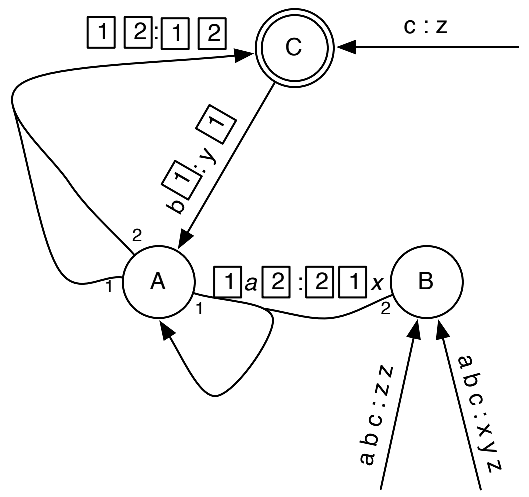


Figure 2.6: The SCFG from Figure 2.4 represented as a hypergraph (note: weights are not shown).

2.2.2.2 Pushdown automata and transducers

The definitions of WCFGs and WSCFGs derive from constructions from *formal language theory*, whereas those of WFSAs and WFSTs are based on *automata theory*.[13] To a certain extent, this discrepancy is a historical accident: algorithms for manipulating regular languages and transducers have been developed using the mathematics of automata theory, whereas the manipulation of context-free languages has tended to favor representation in terms of grammars. But, since I am attempting to define representations of sets and transducers so as to emphasize commonalities whenever possible, it is reasonable to ask whether there are automata that would be a more appropriate formal object with which to describe (weighted) context-free languages and transducers.

Of course, there *are* automata theoretic objects that represent context-free languages: pushdown automata (PDAs), which have been generalized to pushdown transducers (PDTs). Why not use weighted PDAs and PDTs instead of CFGs and SCFGs?

[13]I thank Michael Riley for bringing these issues to my attention.

Although the argument that CFGs are more familiar is compelling, there is a more basic reason: *there is no equivalent PDT for general SCFGs* (Aho and Ullman, 1972). PDTs can only represent an SCFG when the gap indices in the output labels occur in strictly increasing order (Ibid.; see Lemma 3.2). In other words, they cannot reorder non-terminals during translation. Since the ability to explore a large number of reorderings in polynomial time is precisely what makes SCFGs so compelling in applications, I will not make further use of PDTs.[14]

2.2.3 Selecting a representation: finite-state or context-free?

In the previous section, I gave examples of four structures for representing weights sets and relations of possibly infinite size using a finite number of symbols: WFSAs and WFSTs, which are finite-state, and WCFGs and WSCFGs, which are context-free. Since context-free languages and relations are a strict superset of the regular languages and relations, and the former can express the same objects with grammars of approximately equal complexity (in terms of the number of symbols), context-free representations would seem preferable, if for no other reason than because they provide more powerful representations.

However, it must first be established to what extent these automata support the operations introduced in §2.1.3, since it is not sufficient merely to represent sets and relations. Of particular interest is the *composition* operation, since it is with this that inputs are transduced to outputs. Since composition implicitly computes the intersection of the output

[14]Aho and Ullman (1973) do sketch the possibility an automaton, which they call a *pushdown processor*, which is capable of representing any SCFG. However, it is not rigorously defined, nor is it clear if composition operations would be natural to implement against it.

Table 2.5: Language closure properties under intersection.

$\downarrow \cap \rightarrow$	FSA	CFG
FSA	FSA	CFG
CFG	CFG	*undecidable*

of the left transducer and the input of the right transducer, I start with a discussion of the closure properties of *intersection*. Table 2.5 summarizes the closure properties for intersection of FSAs and CFGs (Aho and Ullman, 1972). In summary: there are intersection algorithms for any pair of finite-state and context-free transducers, with the exception of two context-free transducers, for which intersection is in general undecidable (Hopcroft and Ullman, 1979). However, if one of the CFGs is *non-recursive* (which is the case in many applications), the intersection will result in a new CFG; however, the algorithm is PSPACE-complete (Nederhof and Satta, 2004). Although Post and Gildea (2008) describe how this algorithm can be utilized to incorporate a context-free language model into a context-free translation model, and work on approximate inference algorithms suggests alternative solutions (Rush et al., 2010), I will leave models requiring the composition of two context-free structures for future work.

While these language theoretic results are well-known, the implications for organizing processing models have only been partially explored. That WFSTs are closed under composition has been exploited to factor problems like translation, speech-recognition, or joint translation and speech recognition into a cascade of transducers (Kumar et al., 2006). Recognizing that SCFGs can be composed with any number of FSTs has been theoretically appreciated (Satta, submitted), but few applications that factor problems into cascades of transducers and include a context-free component have been developed.

2.3 Algorithms for composition of a WFST and a WSCFG

Composition is the fundamental operation in my ambiguity-preserving processing model (§2.1.5). While composition algorithms for general WFSTs using arbitrary semirings have been developed and explored in considerable detail (Mohri, 2009), general algorithms that compose a WFST and a WSCFG to produce a new WSCFG have received less attention.[15] In this section, I give a practical composition algorithm that makes very few assumptions about the structure of its inputs or the semiring used. While this algorithm has a similar structure to Earley's algorithm for parsing (Earley, 1970) and incorporates elements of the FSA-CFG intersection construction first described in Bar-Hillel et al. (1961), my presentation as a generalized algorithm is novel.

I start by giving Bar-Hillel's simple but extremely inefficient algorithm for computing the intersection of a CFG and a FSA. I describe how its efficiency can be improved using a top-down, left-to-right search while also incorporating weights from an arbitrary semiring (§2.3.1). I then discuss how the weighted intersection algorithm can be altered so as to compute the *composition* of a WFST and a WSCFG (§2.3.2). I conclude by showing that synchronous parsing (parsing a pair of strings $\langle \mathbf{e}, \mathbf{f} \rangle$ with a WSCFG) can be factored into two successive WFST-WSCFG composition operations, and I provide experimental evidence showing that this strategy is far more efficient than specialized synchronous parsing algorithms that have been proposed previously (§2.3.4).

[15] The closely related problems of intersection of unweighted FSAs and CFGs (van Noord, 1995), as well as intersection of probabilistic FSAs and CFGs (Nederhof and Satta, 2003), have, however, been explored in some detail.

2.3.1 Intersection of a WFSA and a WCFG

Bar-Hillel et al. (1961) proposed an intersection algorithm that, given an (unweighted) FSA, A, and a CFG, $\mathcal{G}$, constructs a new CFG, $\mathcal{G}_{\cap A}$, that generates the intersection of the languages $L(A)$ and $L(\mathcal{G})$, that is, $\mathcal{G}_{\cap A}$ generates only strings that are also generated by both A and $\mathcal{G}$. Although this method was originally specified only for unweighted FSAs and CFGs, Nederhof and Satta (2003) show that there is a straightforward generalization to probabilistic FSAs and probabilistic CFGs such that the probabilities assigned by the two inputs are multiplied in the output PCFG.[16] The construction proceeds as follows. The members of the FSA, Q (states), E (edges), F (final states), and I (initial states) are defined as in §2.2.1, except with weighted sets replaced with conventional unweighted sets. The rules in $\mathcal{G}$ have the form $\mathrm{X} \rightarrow \alpha_1 \ldots \alpha_k$, where $\alpha_i \in \Sigma \cup V$. For each rule, all symbols in the grammar (both terminals and non-terminals, left and right sides) become annotated with *pairs of states* from the input FSA as follows: $\mathrm{X}_{q_0,q_k} \rightarrow \alpha_{1\,q_0,q_1} \alpha_{2\,q_1,q_2} \ldots \alpha_{k\,q_{k-1},q_k}$, for all sequences of states $q_0 \ldots q_k$. Each symbol $\alpha_{i\,q,r}$ is a non-terminal symbol in the output grammar. Next, terminal rules $x_{q,r} \rightarrow x$ (where x is a symbol in Σ) are added if $\langle q,x,r \rangle \in E$. Finally, the start states $\mathcal{G}_{\cap A}$ are the input CFG's start states annotated with the pairs $\langle q,r \rangle \in I \times F$, which ensures that $\mathcal{G}_{\cap A}$ will only derive strings in $L(A)$.

This is the original Bar-Hillel construction. However, for simplicity, I will make one minor change. Observe that if a symbol $x_{q,r}$ exists in a rule (after the transformation just described has been applied), this symbol can be replaced by x if $\langle q,x,r \rangle \in E$ and otherwise the rule can be *deleted*, since any derivation containing this rule will never be able to rewrite into a string of terminals. Therefore, to summarize the output of the

[16]The generalization to arbitrary semirings is likewise trivial.

modified algorithm: from each input rule from $\mathcal{G}$, zero, one, or many rules are constructed for the output grammar. These rules have the same length and structure as the rule, in that terminal symbols are untransformed and non-terminals retain the same 'type' in the output grammar, but are annotated with state pairs from A. I note that, as a result, if a string $x \in \Sigma^*$ is derivable by both A and $\mathcal{G}$, then the derivation of x under $\mathcal{G}_{\cap A}$ (which must exist) will have the same derivation tree structure as it did under $\mathcal{G}$, only with different node labels.

Although theoretically useful, the Bar-Hillel construction is impractical to use, for two reasons. Most seriously, there are $|Q|^n$ unique sequences of states of length n, so a single rule with k symbols in its right-hand side in the input grammar is expanded into $|Q|^{k+1}$ variants in the output grammar.[17] Second, even when $\mathcal{G}_{\cap A}$ derives only the empty set, the algorithm simply generates a massive grammar that ultimately produces no derivations of strings of terminal symbols! However, despite these limitations the Bar-Hillel construction does provide a crucial insight into the nature of the intersection of CFGs and FSAs: *intersection occurs by mapping rules from the input grammar into one or more rules in the output grammar with the same length, structure, and terminals, but with different non-terminals*.

To summarize, the Bar-Hillel intersection grammar is not generally as efficient as it could be since it may contain many rules that are unreachable from the start symbols or that can never be rewritten into terminal symbols. One would therefore would like to find an efficient means of gathering only the rules that are 'useful' in the intersection grammar, without exploring the exponential number of transformation possibilities suggested by

[17]While this combinatorial explosion is actually the correct intersection in rare cases (specifically, when an FSA is fully connected and every pair of states rewrites with every symbol in Σ), most intersections for 'real' automata and grammars are massively more constrained.

the Bar-Hillel construction (unless, of course, the intersection grammar requires all these rules). To do so a strategy of *top-down filtering* will be used to produce an algorithm that efficiently computes the intersection of FSA and a CFG. Although the worst case complexity of the filtered algorithm is the same as of the Bar-Hillel construction, for many useful restricted classes of CFGs and FSAs, the performance is substantially better. Since weighted intersection (§2.1.3.2) is the focus here, the algorithm introduced handles semiring-weighted variants of these operands, that is, WFSAs and WCFGs. This adds only minimal complexity as the unweighted algorithm can be seen as a special case using the Boolean semiring.

2.3.1.1 Weighted deductive logic

I will present my efficient top-down WFSA-WCFG intersection algorithm (and the WFST-WSCFG composition algorithm later) as a *weighted deductive proof system* (Chiang, 2007; Goodman, 1999; Shieber et al., 1995). I define a space of *items* paired with weights from semiring K as follows. *Axioms* are items that are true automatically (with some weight), and *inference rules* have the following form and describe how to create new items from already existing items (that is, axioms and items created by the application of inference rules):

$$\frac{I_1 : w_1 \quad I_2 : w_2 \quad \cdots \quad I_k : w_k}{I : w} \quad \phi$$

This states that if items I_i (the *antecedents*) are true with weights w_i, then the I (the *consequent*) is true with weight w, if side condition ϕ is true. When an item can be derived in multiple ways, its weight is defined to be the sum using the semiring's $\oplus$

operator. Finally, *goals* are specified, which are a set of items that are to be proved.

I use weighted logic to construct new grammars, rather than the more conventional uses discussed in the literature (e.g., to select a best parse or to compute a quantity over the resulting proof structure, such as an inside score). The programs are assumed to run exhaustively, terminating when no new items can be derived by applying inference rules. As a result, termination is *not* a condition of being able to derive a goal node; however, if no goal is derived when all inference rules have been expanded, then the algorithm results in failure. It is therefore possible to give programs that will never terminate; so, for each algorithm given, I explicitly state the preconditions for successful termination.

Other authors have discussed the realization of logic programs in software in considerable depth. Shieber et al. (1995) provides a detailed discussion for the unweighted case as it relates to parsing algorithms, and Goodman (1999) and Klein and Manning (2001) explore this topic particularly with regard to weighted items. Additionally, the Dyna programming language can directly compile such a formalism into C++ code (Eisner et al., 2005).

2.3.1.2 A top-down, left-to-right intersection algorithm

I now give a more efficient algorithm for computing the intersection of WCFG, $\mathcal{G}$, and WFSA, A.[18] Let x be a free variable that refers to a symbol in Σ; q, r, and s are free variables ranging over states in Q; X and Y are free variables taking on values in V; α and β are strings of terminals and non-terminals (possibly of length zero), that is, they are elements of $(\Sigma \cup V)^*$; and u and v are weights in $\mathbb{K}$. Figure 2.7 provides the axioms,

[18] Grune and Jacobs (2008) sketch a similar algorithm for intersection of unweighted FSAs with CFGs.

goals, and inference rules for a top-down intersection algorithm for computing $\mathcal{G} \cap A$. Items have the form $[\mathrm{X} \rightarrow \alpha \bullet \beta, q, r]$, which corresponds to the assertion that there is a path π from q to r, such that $\alpha \overset{*}{\Rightarrow} i[\pi]$. The inference rules are quite similar to those found in Earley's algorithm.

For clarity, I remark on the differences between this algorithm and Earley's, as well as the differences to other well-known presentations of weighted logic programming (Chiang, 2007; Eisner et al., 2005; Goodman, 1999). Most importantly from the perspective of logic programming, the weights of the items in this logic program are defined so as to compute the *local weight* of the resulting intersection rules in the intersection grammar. This is an important difference compared to other presentations of weighted parsing, for example the standard presentations of the INSIDE algorithm using weighted deductive logic, which define the weight of an item so as to include the weight of all its antecedents so that when the goal item is derived, it represents the total weight of the set defined by the intersection grammar. Thus, COMPLETE and ACCEPT do not incorporate the weights of the completed item that triggered their traversal (and annotation) of a non-terminal symbol, just as the weight associated with an item generated by PREDICT is just the weight from the grammar and does not include the score of the trigger item. The reason for this is that most presentations of weighted parsing conflate two computations that I prefer to treat as distinct: computation of the weighted intersection grammar and inference over the weighted set defined by this grammar. Inference is discussed below (§2.4).

Since this algorithm is a 'parser' for WFSA input (rather than for unambiguous string input, which is the input assumed by conventional parsers) and since this WFSA may contain ε-transitions, the Earley SCAN operation must be split into two different

Start / end conditions:

Axioms:

$$\frac{}{[S' \to \bullet \mathrm{X}, q, q] : \lambda(q) \otimes \sigma(\mathrm{X})} \qquad (q \in I) \wedge (\mathrm{X} \in S)$$

Goals:

$$[S' \to \mathrm{X}\bullet, q, r] \qquad (q \in I) \wedge (r \in F) \wedge (\mathrm{X} \in S)$$

Inference rules:

PREDICT

$$\frac{[\mathrm{X} \to \alpha \bullet \mathrm{Y}\beta, q, r] : u}{[\mathrm{Y} \to \bullet\gamma, r, r] : v} \qquad \mathrm{Y} \overset{v}{\to} \gamma \in R$$

SCAN-Σ

$$\frac{[\mathrm{X} \to \alpha \bullet x\beta, q, s] : u}{[\mathrm{X} \to \alpha x \bullet \beta, q, r] : u \otimes w(s, x, r)} \qquad \langle s, x, r\rangle \in E$$

SCAN-ε

$$\frac{[\mathrm{X} \to \alpha \bullet \beta, q, s] : u}{[\mathrm{X} \to \alpha \bullet \beta, q, r] : u \otimes w(s, \varepsilon, r)} \qquad \langle s, \varepsilon, r\rangle \in E$$

ε-RULE

$$\frac{[\mathrm{X} \to \bullet\varepsilon, q, r] : u}{[\mathrm{X} \to \varepsilon\bullet, q, r] : u}$$

COMPLETE

$$\frac{[\mathrm{X} \to \alpha \bullet \mathrm{Y}\beta, q, s] : u \quad [\mathrm{Y} \to \gamma\bullet, s, r] : v}{[\mathrm{X} \to \alpha\mathrm{Y}_{s,r} \bullet \beta, q, r] : u} \qquad \mathrm{X} \neq S'$$

ACCEPT

$$\frac{[S' \to \bullet\mathrm{X}, q, q] : u \quad [\mathrm{X} \to \gamma\bullet, q, r] : v}{[S' \to \mathrm{X}_{q,r}\bullet, q, r] : u \otimes \rho(r)} \qquad r \in F$$

Figure 2.7: Weighted logic program for computing the intersection of a WCFG, $\mathcal{G} = \langle \Sigma, V, \langle S, \sigma\rangle, \langle R, \upsilon\rangle\rangle$, and a WFST, $A = \langle \Sigma, Q, \langle I, \lambda\rangle, \langle F, \rho\rangle, \langle E, w\rangle\rangle$.

cases: one that handles transitions with labels from Σ and one that handles ε-transitions. Second, the ACCEPT rule is necessary to incorporate accept weights ρ. Third, the COMPLETE and ACCEPT rules *annotate* the non-terminal symbols they traverse with the starting and ending states of the FSA used to traverse the completed item in the antecedent (for example, in COMPLETE, the symbol Y in the right-hand side of the X rule in the antecedent becomes $\mathrm{Y}_{s,r}$ in the consequent).[19] Once the space of items has been generated by repeated application of the inference rules, the intersection of $\mathcal{G}$ and A is encoded in the item chart, which can then trivially be turned back into WCFG rules of the proper form.

To illustrate the algorithm, Figure 2.9 shows an example deduction using the weighted logic program in Figure 2.7 to compute the intersection of the WCFG and a WFSA given in Figure 2.8. I now turn to an algorithm for converting items back into WCFG rules, thereby completing the construction of $\mathcal{G}_{\cap A}$ with the top-down algorithm.

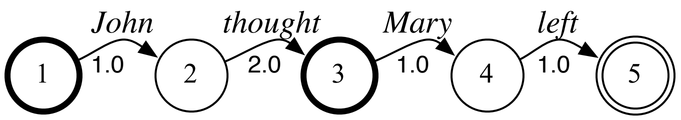
2.3.1.3 Converting the item chart into $\mathcal{G}_{\cap A}$

The items generated by running the logic program in Figure 2.7 on $\mathcal{G}$ and A can now be re-encoded into a well-formed WCFG $\mathcal{G}_{\cap A} = \langle \Sigma, V_{\cap A}, \langle S_{\cap A}, \sigma \rangle, \langle R_{\cap A}, \upsilon \rangle \rangle$. Items where the $\bullet$ has reached the *right edge* of the rule (sometimes called *passive edges* or *passive items*) are transformed into rules in $R_{\cap A}$; all other items are discarded. The existence of any item of the form $[\mathrm{X} \rightarrow \alpha \bullet, q, r] : u$ where $\mathrm{X} \in V$ indicates that $R_{\cap A}$ should contain the rule $X_{q,r} \rightarrow \alpha$ with weight u. ACCEPT items with weight v indicate that the single

[19]Non-terminal annotation by COMPLETE does not change the result computed by the algorithm; however, in certain ambiguous cases, it will result in more items than the non-annotated grammar would have made use of.

$$\Sigma = \{\textit{John}, \textit{Mary}, \textit{left}, \textit{thought}, \textit{saw}\} \quad V = \{\mathrm{S}, \mathrm{NP}, \mathrm{VP}\} \quad S = \{\mathrm{S}\} \quad \sigma(\mathrm{S}) = 1.0$$

$$\begin{aligned} R = \{ \quad \mathrm{S} &\overset{1.0}{\rightarrow} \mathrm{NP}\ \mathrm{VP}, \\ \mathrm{NP} &\overset{0.5}{\rightarrow} \textit{John}, \\ \mathrm{NP} &\overset{0.5}{\rightarrow} \textit{Mary}, \\ \mathrm{VP} &\overset{0.5}{\rightarrow} \textit{saw}\ \mathrm{NP}, \\ \mathrm{VP} &\overset{0.4}{\rightarrow} \textit{left}, \\ \mathrm{VP} &\overset{0.1}{\rightarrow} \textit{thought}\ \mathrm{S} \quad \} \end{aligned}$$



Weight Functions

$\lambda(1) = 0.4$
$\lambda(3) = 0.2$
$\rho(5) = 1.0$

Figure 2.8: A WCFG representing a weighted set of infinite size (above) and a WFSA representing the finite weighted set $\{\textit{John thought Mary left} : 0.8, \textit{Mary left} : 0.2\}$ over the probability semiring (below).

(annotated) non-terminal is a start symbol $S_{\cap A}$, with weight v. To illustrate, the item chart in Figure 2.9 converts into the WCFG in Figure 2.10.

Note several things about the example. First, $\mathcal{G}_{\cap A}$ describes exactly the weighted set $\{\textit{John thought Mary left} : 0.008, \textit{Mary left} : 0.04\}$. That these weights are correct can easily be verified by looking at the weights of the derivations associated with them in the WFSA $(0.8, 0.2)$ and the WCFG $(0.01, 0.2)$ from which the weighted intersection grammar was derived, and multiplying them. Second, the intersection grammar has the structure of a context-free grammar, but it defines a *finite* language. As such, the weighted

1	$[S' \to \bullet \text{S}, 1, 1] : 0.4$	AXIOM
2	$[S' \to \bullet \text{S}, 3, 3] : 0.2$	AXIOM
3	$[\text{S} \to \bullet \text{NP VP}, 1, 1] : 1.0$	PREDICT from 1
4	$[\text{S} \to \bullet \text{NP VP}, 3, 3] : 1.0$	PREDICT from 2
5	$[\text{NP} \to \bullet \textit{John}, 1, 1] : 0.5$	PREDICT from 3
6	$[\text{NP} \to \bullet \textit{Mary}, 1, 1] : 0.5$	PREDICT from 3
7	$[\text{NP} \to \bullet \textit{John}, 3, 3] : 0.5$	PREDICT from 4
8	$[\text{NP} \to \bullet \textit{Mary}, 3, 3] : 0.5$	PREDICT from 4
9	$[\text{NP} \to \textit{John} \bullet, 1, 2] : 0.5 \times 1.0$	SCAN-Σ from 5
10	$[\text{NP} \to \textit{Mary} \bullet, 3, 4] : 0.5 \times 1.0$	SCAN-Σ from 8
11	$[\text{S} \to \text{NP}_{1,2} \bullet \text{VP}, 1, 2] : 1.0$	COMPLETE from 3 and 9
12	$[\text{S} \to \text{NP}_{3,4} \bullet \text{VP}, 3, 4] : 1.0$	COMPLETE from 4 and 10
13	$[\text{VP} \to \bullet \textit{saw}\ \text{NP}, 2, 2] : 0.5$	PREDICT from 11
14	$[\text{VP} \to \bullet \textit{left}, 2, 2] : 0.4$	PREDICT from 11
15	$[\text{VP} \to \bullet \textit{thought}\ \text{S}, 2, 2] : 0.1$	PREDICT from 11
16	$[\text{VP} \to \bullet \textit{saw}\ \text{NP}, 4, 4] : 0.5$	PREDICT from 12
17	$[\text{VP} \to \bullet \textit{left}, 4, 4] : 0.4$	PREDICT from 12
18	$[\text{VP} \to \bullet \textit{thought}\ \text{S}, 4, 4] : 0.1$	PREDICT from 12
19	$[\text{VP} \to \textit{thought} \bullet \text{S}, 2, 3] : 0.1 \times 2.0$	SCAN-Σ from 15
20	$[\text{VP} \to \textit{left} \bullet, 4, 5] : 0.4 \times 1.0$	SCAN-Σ from 17
21	$[\text{S} \to \bullet \text{NP VP}, 3, 3] : 1.0$	PREDICT from 19 (**duplicate of 4**)
22	$[\text{S} \to \text{NP}_{3,4}\ \text{VP}_{4,5} \bullet, 3, 5] : 1.0$	COMPLETE from 12 and 20
23	$[\text{VP} \to \textit{thought}\ \text{S}_{3,5} \bullet, 2, 5] : 0.2$	COMPLETE from 19 and 22
24	$[S' \to \text{S}_{3,5} \bullet, 3, 5] : 0.2 \times 1.0$	ACCEPT from 2 and 22 (**goal**)
25	$[\text{S} \to \text{NP}_{1,2}\ \text{VP}_{2,5} \bullet, 1, 5] : 1.0$	COMPLETE from 11 and 23
26	$[S' \to \text{S}_{1,5} \bullet, 1, 5] : 0.4 \times 1.0$	ACCEPT from 1 and 25 (**goal**)

Figure 2.9: Item chart produced when computing the intersection of the WCFG and the WFSA from Figure 2.8 using the top-down filtering program shown in Figure 2.7.

set can be exactly expressed by a WFSA.

2.3.1.4 Alternative forms of $\mathcal{G}_{\cap A}$

So far, two ways of deriving $\mathcal{G}_{\cap A}$ have been considered: the modified Bar-Hillel construction and the top-down intersection algorithm. While the Bar-Hillel construction may include many unreachable rules, it otherwise produces the exact same set of 'good' rules as the top-down algorithm. However, this is not the only possible intersection grammar

$$\Sigma_{\cap A} = \Sigma \quad V_{\cap A} = \{\mathrm{S}_{1,5}, \mathrm{S}_{2,5}, \mathrm{NP}_{1,2}, \mathrm{NP}_{3,4}, \mathrm{VP}_{2,5}, \mathrm{VP}_{4,5}\} \quad S_{\cap A} = \{\mathrm{S}_{1,5}, \mathrm{S}_{3,5}\}$$
$$\sigma(\mathrm{S}_{1,5}) = 0.4 \quad \sigma(\mathrm{S}_{3,5}) = 0.2$$

$$R = \{ \quad \begin{array}{rcl} \mathrm{NP}_{1,2} & \overset{0.5}{\rightarrow} & \textit{John}, \\ \mathrm{NP}_{3,4} & \overset{0.5}{\rightarrow} & \textit{Mary}, \\ \mathrm{VP}_{4,5} & \overset{0.4}{\rightarrow} & \textit{left}, \\ \mathrm{S}_{3,5} & \overset{1.0}{\rightarrow} & \mathrm{NP}_{3,4}\ \mathrm{VP}_{4,5}, \\ \mathrm{VP}_{2,5} & \overset{0.2}{\rightarrow} & \textit{thought}\ \mathrm{S}_{3,5}, \\ \mathrm{S}_{1,5} & \overset{1.0}{\rightarrow} & \mathrm{NP}_{1,2}\ \mathrm{VP}_{2,5} \quad \} \end{array}$$

Figure 2.10: Converting the item chart from Figure 2.9 into the intersection grammar WCFG $\mathcal{G}_{\cap A}$.

that generates $L(\mathcal{G}) \cap L(A)$. To see one possible alternative, observe that, like Earley's algorithm, the top-down algorithm can be understood to perform an on-the-fly binarization of $\mathcal{G}$. The COMPLETE rule corresponds to the binary combination of two non-terminals, and the SCAN rule combines a terminal symbol with a non-terminal; both productions result in a non-terminal symbol (corresponding to the consequent item). Thus, another strategy for inferring the intersection grammar is to convert the *structure* of the item chart *directly* into a grammar.[20] Unlike the Bar-Hillel construction and the top-down algorithm where the chart items are converted using specialized rules, the derivations in this alternative intersection grammar will generally have a different structure than derivations of the same string in $\mathcal{G}$.

While implicit binarization is a useful strategy for parsing, the (binary) item chart

[20]In fact, Nederhof (2003) points out that *any* derivation encoded as a weighted deduction can be represented as a weakly equivalent CFG—even those corresponding to grammar formalisms more powerful than CFGs, such as tree adjoining grammars (Vijay-Shanker and Weir, 1993), combinatorial categorial grammars (Steedman, 2000), and range concatenation grammars (Boullier, 2000). However, the size of these weakly equivalent grammars may be substantially larger than the more powerful source grammar.

structure will not always be useful for computing $\mathcal{G}_{\cap A}$ since it will not always be meaningful in the context of SCFGs, where there may be no binary equivalent of a higher order SCFG (Wu, 1997). Fortunately, by using the top-down intersection algorithm, where annotated non-terminals are used to reconstruct intersection grammar rules, binarization can be used during intersection with one language at a time without sacrificing structure of the original grammar, enabling this algorithm to be used with SCFGs.

2.3.1.5 Remarks on the relationship between parsing and intersection

As was already mentioned, the top-down WFSA/WCFG intersection algorithm of §2.3.1.2 and Earley's parsing algorithm are closely related.[21] Although the Bar-Hillel construction has been known since the early 1960's and context-free parsing algorithms for at least as long, an appreciation of the relationship between parsing and intersection has been slower to develop, having become appreciated mostly in the last 20 years (Grune and Jacobs, 2008). I therefore briefly remark on the relationship between *parsing* and *intersection*. For this discussion, note that an input sentence to a parser can be understood to be a simple linear-chain FSA, with states corresponding to the positions between words, and with one state (the initial state) before the first word and one after the final word (the final state).

Intuitively, parsing is the problem of returning a representation of all parse structures compatible with an input sentence and input grammar. Rather than committing to any particular output representation when discussing parsers in general, Lee (2002) pro-

[21]Earley's algorithm can be understood as a non-probabilistic variant of the algorithm that applies only to restricted kinds of FSAs and operates with a specific control strategy.

poses that a parser must only produce an oracle that can be queried (in constant time) to ask if some *span* $[i,j]$ in the input sentence is derivable, in the context of the full sentence, by some non-terminal in the input grammar.[22] Under this definition, the intersection grammar $\mathcal{G}_{\cap A}$ generated by the top-down algorithm is itself such an oracle.[23] Testing for the existence of the non-terminal $\mathrm{X}_{i,j}$ in $V_{\cap A}$ is equivalent to querying the oracle to see whether X derives $w_{i \cdots j}$. Additionally, the right-hand sides of the rules that this non-terminal rewrites as function like backpointers in many traditional formulations of parsers.

2.3.2 From intersection to composition

The previous section described how to compute the weighted intersection of an arbitrary WCFG and WFSA, resulting in a new WCFG. With minor variations the same algorithm can be used to compute the *composition* of a WSCFG and a WFST, resulting in a new WSCFG. Recall that weighted composition (§2.1.3.4) is a binary operation over weighted relations $\langle R, u\rangle$ and $\langle S, v\rangle$ with $R \subseteq \Sigma^* \times \Delta^*$, and I assume $S \subseteq \Delta^* \times \Omega^*$. I only consider the case where the left relation is represented as a WFST and the right relation is a WSCFG. If the reverse is required (i.e., left relation is a WSCFG and right relation is a WFST), the inversion theorem (§2.1.3.5) can be utilized to compute the composition.

[22]The requirements for the information provided by the oracle are actually a bit more complicated. A more detailed discussion is omitted since they are not material to the point being made. For more details, consult Lee (2002).

[23]To be a proper oracle, it must be possible to query the existence of any non-terminal in $V_{\cap A}$ in constant time.

2.3.2.1 A top-down, left-to-right composition algorithm

Let T be a WFST $\langle \Sigma, \Delta, Q, \langle I, \lambda \rangle, \langle F, \rho \rangle, \langle E, w \rangle \rangle$ that represents $\langle Rel(T), u \rangle$ and let $\mathcal{G}$ be a WSCFG $\langle \Delta, \Omega, V, \langle S, \sigma \rangle, \langle R, \upsilon \rangle \rangle$ that represents $\langle Rel(\mathcal{G}), v \rangle$ over semiring K. The composition WSCFG $\mathcal{G}_{\circ T} = \langle \Sigma, \Omega, V_{\circ T}, \langle S_{\circ T}, \sigma_{\circ T} \rangle, \langle R_{\circ T}, \upsilon_{\circ T} \rangle \rangle$ must be computed. Intuitively, composition can be carried out by treating the *output* side of T as a WFSA and running the intersection algorithm on the WCFG that is defined by the *input* side of $\mathcal{G}$. However, the resulting *composition* grammar $\mathcal{G}_{\circ T}$ must consist of rules that map from strings in Σ^* to strings in Ω^*. Therefore, in addition to non-terminal annotation introduced by the COMPLETE and ACCEPT rules in the intersection algorithm, the composition algorithm must transform *terminal* symbols during the SCAN operations. Figure 2.11 shows the altered algorithm. Most of the symbols retain the same meanings they had in the intersection (e.g., x is a free variable over symbols in Σ); however, there are a number of changes. Briefly, y is a free variable over symbols in the output alphabet, Δ; ζ and ξ are strings in $(\{\boxed{1}, \boxed{2}, \boxed{3}, \ldots\} \cup \Omega)^*$. Because symbols in Δ are transformed (by the FST) into symbols in Σ, α is a string in $(\Sigma \cup V_{\circ T})^*$ but β is a string in $(\Delta \cup V)^*$.

This composition algorithm terminates in the vast majority of cases. However, the presence of ε-rules on the *output* side of T can generate an unbounded chain of inferences, causing it to never halt. This occurs precisely in the case where $P(q, \Sigma\Sigma^*, \varepsilon, q) \neq \mathbb{0}$ for any state $q \in Q$ (refer to §2.2.1 for the definition of P). Fortunately, the grammars and transducers that are considered for the remainder of the dissertation will not contain any such ε-cycles. This problem is therefore avoided by stipulation, and its proper resolution relegated to future work.[24] Furthermore, by relying on this stipulation, the structural

[24] Eisner (2000) proposes an algorithm for detecting such cycles that would be one possibility for dealing

Start / end conditions:

Axioms:

$$\frac{}{[S' \to \langle \bullet \mathrm{X}, \boxed{1} \rangle, q, q] : \lambda(q) \otimes \sigma(\mathrm{X})} \qquad (q \in I) \wedge (\mathrm{X} \in S)$$

Goals:

$$[S' \to \langle \mathrm{X} \bullet, \boxed{1} \rangle, q, r] \qquad (q \in I) \wedge (r \in F) \wedge (\mathrm{X} \in S)$$

Inference rules:

PREDICT

$$\frac{[\mathrm{X} \to \langle \alpha \bullet \mathrm{Y}\beta, \zeta \rangle, q, r] : u}{[\mathrm{Y} \to \langle \bullet \gamma, \xi \rangle, r, r] : v} \qquad \mathrm{Y} \xrightarrow{v} \langle \gamma, \xi \rangle \in R$$

SCAN-$\Sigma : \Delta$

$$\frac{[\mathrm{X} \to \langle \alpha \bullet y\beta, \zeta \rangle, q, s] : u}{[\mathrm{X} \to \langle \alpha x \bullet \beta, \zeta \rangle, q, r] : u \otimes w(s, x, y, r)} \qquad \langle s, x, y, r \rangle \in E$$

SCAN-$\varepsilon : \Delta$

$$\frac{[\mathrm{X} \to \langle \alpha \bullet y\beta, \zeta \rangle, q, s] : u}{[\mathrm{X} \to \langle \alpha \bullet \beta, \zeta \rangle, q, r] : u \otimes w(s, \varepsilon, y, r)} \qquad \langle s, \varepsilon, y, r \rangle \in E$$

SCAN-$\Sigma : \varepsilon$ *(naive)*

$$\frac{[\mathrm{X} \to \langle \alpha \bullet \beta, \zeta \rangle, q, s] : u}{[\mathrm{X} \to \langle \alpha x \bullet \beta, \zeta \rangle, q, r] : u \otimes w(s, x, \varepsilon, r)} \qquad \langle s, x, \varepsilon, r \rangle \in E$$

ε-RULE

$$\frac{[\mathrm{X} \to \langle \bullet \varepsilon, \Omega^* \rangle, q, r] : u}{[\mathrm{X} \to \langle \varepsilon \bullet, \Omega^* \rangle, q, r] : u}$$

COMPLETE

$$\frac{[\mathrm{X} \to \langle \alpha \bullet \mathrm{Y}\beta, \zeta \rangle, q, s] : u \quad [\mathrm{Y} \to \langle \gamma \bullet, \xi \rangle, s, r] : v}{[\mathrm{X} \to \langle \alpha \mathrm{Y}_{s,r} \bullet \beta, \zeta \rangle, q, r] : u} \qquad \mathrm{X} \neq S'$$

ACCEPT

$$\frac{[S' \to \langle \bullet \mathrm{X}, \boxed{1} \rangle, q, q] : u \quad [\mathrm{X} \to \langle \gamma \bullet, \zeta \rangle, q, r] : v}{[S' \to \langle \mathrm{X}_{q,r} \bullet, \boxed{1} \rangle, q, r] : u \otimes \rho(r)} \qquad r \in F$$

Figure 2.11: Weighted logic program for computing the composition of a WFST $T = \langle \Sigma, \Delta, Q, \langle I, \lambda \rangle, \langle F, \rho \rangle, \langle E, w \rangle \rangle$, and WSCFG $\mathcal{G} = \langle \Delta, \Omega, V, \langle S, \sigma \rangle, \langle R, \upsilon \rangle \rangle$.

isomorphism that is found between the input and output grammars in intersection holds for the outputs of composition as well (that is, $\mathcal{G}$ and $\mathcal{G}_{\circ T}$ have the same basic structure, in terms of number and orientation of non-terminal symbols). However, the proper solution to the ε-cycle problem requires inserting auxiliary non-terminal symbols in $\mathcal{G}_{\circ T}$ that have no correspondence in the input rules, thus breaking this isomorphic structural relationship.[25] All other ε's are handled properly by the algorithm.

2.3.2.2 A bottom-up composition algorithm

The algorithm presented in the previous section makes few assumptions about the input, making it quite general and potentially useful for many applications. However, it can be challenging to implement, since efficiency demands creating several indices (for example, to efficiently retrieve the set of items that must be completed after the • reaches the right edge of the rule), managing a queue of items to process, and handing multiple derivations of the same items properly. However, when the grammar $\mathcal{G}$ is ε-free on the input side of its rules, and where T is acyclic (and therefore defines a finite language), a simpler bottom-up algorithm can be utilized. These conditions are met in many situations in machine translation; for example, where $\mathcal{G}$ is a hierarchical phrase-based translation grammar (§3.1.2.2) and T is a sentence or non-recursive WFST. Figure 2.12 gives the weighted logic program. Note that this is similar to the CKY+ algorithm used in SCFG-based translation (Chiang, 2007), but it has been adapted to annotate non-terminals and transduce terminals to create a composition forest. The bottom-up algorithm is quite sim-

with this issue.

[25]It is not difficult to prove that the excluded composition cases involving ε-cycles require the addition of new non-terminals to the composition grammar.

ilar to the top-down one, only it lacks a PREDICT rule and the axioms are different. A primary advantage of the bottom-up algorithm is its simplicity: it can be implemented by looping over all states q, r, and s in topological order and building items that span progressively longer distances in the state space.[26] Keeping track of an agenda of items is not required (as it is with the top-down algorithm).

The process of converting the resulting items from the bottom-up chart is similar to the top-down approach.[27] The altered conversion procedure is as follows: for each passive item of the form $[\mathrm{X} \rightarrow \langle \alpha \bullet, \zeta \rangle, q, r] : w$, add X_q, r to $V_{\circ T}$. If the item is a goal item, add $\mathrm{X}_{q,r}$ to $S_{\circ T}$ with weight $\sigma(\mathrm{X}) \otimes \lambda(q) \otimes \rho(r)$. Always, add the rule $\mathrm{X}_{q,r} \rightarrow \langle \alpha, \zeta \rangle$ to $R_{\circ T}$ with weight w.

2.3.3 A note on terminology: sets vs. relations

In the remainder of the thesis, I make use of the convention to represent weighted sets with the appropriate identity relation. That is, WFSAs will be represented using identity-transducer WFSTs, and WCFGs with identity-transducer WSCFGs. This convention has been widely used among authors who work with WFSTs (see e.g., Allauzen et al. (2007)), if for no other reason than implementations need only have a single representation. It also means that which is logically intersection is implemented with composition. However, it may at first seem unusual to readers unfamiliar with this convention when sets of sentences are described as being encoded by a WFST or WSCFG. However, this is quite

[26]Because T is acyclic by stipulation, a topological ordering of the states is guaranteed to exist.

[27]An alternative conversion strategy is necessary since the bottom-up logic program incorporates neither the σ and λ start weights, nor the ρ accept weights, and there are not separate ACCEPT items. Fortunately, by adapting the item-to-rule conversion to account for the missing ACCEPT items, the missing weights can easily be incorporated.

Start / end conditions:

Axioms:

$$\frac{}{[\mathrm{X} \to \langle \bullet\beta, \zeta\rangle, q, q] : u} \qquad (q \in Q) \wedge (\mathrm{X} \xrightarrow{u} \langle \beta, \zeta\rangle \in R)$$

Goals:

$$[\mathrm{X} \to \langle \alpha\bullet, \zeta\rangle, q, r] \qquad (q \in I) \wedge (r \in F) \wedge (\mathrm{X} \in S)$$

Inference rules:

SCAN-$\Sigma : \Delta$

$$\frac{[\mathrm{X} \to \langle \alpha \bullet y\beta, \zeta\rangle, q, s] : u}{[\mathrm{X} \to \langle \alpha x \bullet \beta, \zeta\rangle, q, r] : u \otimes w(s,x,y,r)} \qquad \langle s,x,y,r\rangle \in E$$

SCAN-$\varepsilon : \Delta$

$$\frac{[\mathrm{X} \to \langle \alpha \bullet y\beta, \zeta\rangle, q, s] : u}{[\mathrm{X} \to \langle \alpha \bullet \beta, \zeta\rangle, q, r] : u \otimes w(s,\varepsilon,y,r)} \qquad \langle s,\varepsilon,y,r\rangle \in E$$

SCAN-$\Sigma : \varepsilon$

$$\frac{[\mathrm{X} \to \langle \alpha \bullet \beta, \zeta\rangle, q, s] : u}{[\mathrm{X} \to \langle \alpha x \bullet \beta, \zeta\rangle, q, r] : u \otimes w(s,x,\varepsilon,r)} \qquad \langle s,x,\varepsilon,r\rangle \in E$$

COMPLETE

$$\frac{[\mathrm{X} \to \langle \alpha \bullet \mathrm{Y}\beta, \zeta\rangle, q, s] : u \quad [\mathrm{Y} \to \langle \gamma\bullet, \xi\rangle, s, r] : v}{[\mathrm{X} \to \langle \alpha \mathrm{Y}_{s,r} \bullet \beta, \zeta\rangle, q, r] : u} \qquad \mathrm{X} \neq S'$$

Figure 2.12: Weighted logic program for computing the intersection of a WSCFG, $\mathcal{G} = \langle \Sigma, \Delta, V, \langle S, \sigma\rangle, \langle R, \upsilon\rangle\rangle$, and an *acyclic* WFST, $A = \langle \Delta, \Omega, Q, \langle I, \lambda\rangle, \langle F, \rho\rangle, \langle E, w\rangle\rangle$ using bottom-up inference.

intentional.

2.3.4 Application: Synchronous parsing via weighted composition

In the previous two sections, I described an algorithm for intersection of a WFSA and WCFG (§2.3.1) which was then extended to compute the composition of a WFST and a WSCFG (§2.3.2). I now show how the composition algorithm can be used to perform synchronous parsing, the problem of finding the best derivation (or forest of derivations) of a source and target sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$, under a WSCFG, $\mathcal{G}$.[28]

Solving the synchronous parsing problem is necessary for several applications; for example, optimizing how well an SCFG translation model fits parallel training data. Wu (1997) describes a specialized, bottom-up $O(n^6)$ synchronous parsing algorithm for ITGs, a binary SCFG with a restricted form. For general grammars, the situation is even worse: the problem has been shown to be NP-hard (Melamed, 2004; Satta and Peserico, 2005). Unfortunately, even if the class of grammars considered is restricted to binary ITGs, the $O(n^6)$ run-time makes large-scale learning applications infeasible. The usual solution is to use a heuristic search that avoids exploring edges that are likely (but not guaranteed) to be have a low weight (Haghighi et al., 2009; Zhang et al., 2008).

Here, I derive an alternative synchronous parsing algorithm starting from a conception of parsing with SCFGs as a composition of binary relations. This enables the synchronous parsing problem to be factored into two successive *monolingual* parses. My algorithm runs more efficiently than $O(n^6)$ with many grammars (including those that required using heuristic search with other parsers), making it possible to take advantage

[28]This section contains work originally published in Dyer (2010).

of synchronous parsing without developing search heuristics; and the SCFGs are not required to be in a normal form, making it possible to easily parse with more complex SCFG types. I call this algorithm the *two-parse algorithm*.

Before presenting this algorithm, I review the $O(n^6)$ synchronous parser for binary ITGs.[29] Wu (1997) describes a bottom-up synchronous parsing algorithm that can be understood as a generalization of the CKY monolingual parsing algorithm. CKY defines a table consisting of n^2 cells, with each cell corresponding to a span $[i, j]$ in the input sentence; and the synchronous variant defines a table in 4 dimensions, with cells corresponding to a source span $[s,t]$ *and* a target span $[u, v]$. The bottom of the chart is initialized first, and pairs of items are combined from bottom to top. Since combining items from the n^4 cells involves considering two split points (one source, one target), it is not hard to see that this algorithm runs in time $O(n^6)$.

2.3.4.1 The two-parse algorithm

Let WFST, F, be the self-transducer which encodes the source sentence $\mathbf{f}$ with weight $\bar{1}$. Figure 2.13 shows an example of self-transducers encoded as WFSTs. Since F contains no ε's at all, it is certain that F can be composed with the WSCFG, $\mathcal{G}$, using the composition algorithms from the previous section. The result is the composition grammar, $\mathcal{G}_{\circ F}$, which I will now write as $F \circ \mathcal{G}$. While $\mathcal{G}$ can generate a potentially infinite set of strings in the source and target languages, $\mathcal{G}$ generates *only* $\mathbf{f}$ in the input language (albeit with possibly infinitely many derivations), but any number of different strings in

[29]Generalizing the algorithm to higher rank grammars is possible (Wu, 1997), as is converting a grammar to a weakly equivalent binary form in some cases (Huang et al., 2009).

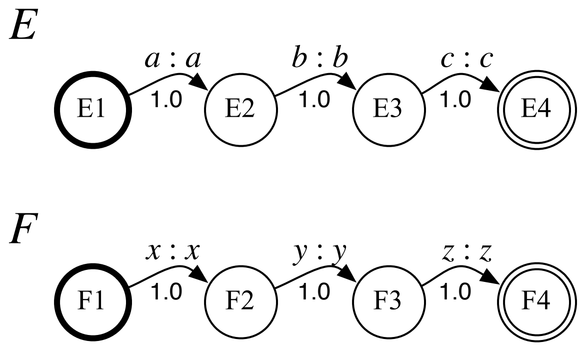


Figure 2.13: The two WFSTs, E and F, required to compute the synchronous parse of the pair $\langle a\ b\ c, x\ y\ z\rangle$.

the output language. This structure is commonly referred to as a $-$LM translation forest (Chiang, 2007), since it encodes all translation alternatives with weights based only on the translation model, not a language model.

From here, it is not hard to see that a second composition operation of a self-transducer WFST, E, that encodes the target string $\mathbf{e}$ with the output side of $F \circ \mathcal{G}$ will produce a grammar that derives exactly the sentence pair $\langle \mathbf{e}, \mathbf{f}\rangle$. Therefore $F \circ \mathcal{G} \circ E$ encodes the synchronous parse forest for $\langle \mathbf{e}, \mathbf{f}\rangle$. Note that in $F \circ \mathcal{G} \circ E$, the non-terminals consist of a non-terminal symbol from $\mathcal{G}$ annotated both with pairs of states from E and pairs of states from F.

Synchronous parsing of the pair $\langle \mathbf{e}, \mathbf{f}\rangle$ with a grammar $\mathcal{G}$ is therefore equivalent to computing $F \circ \mathcal{G} \circ E$, where E and F are self-transducers representing $\mathbf{e}$ and $\mathbf{f}$, respectively. Furthermore, composition is associative, so I can compute this quantity either as $(F \circ \mathcal{G}) \circ E$ or $F \circ (\mathcal{G} \circ E)$.

The *two-parse algorithm* refers to performing a synchronous parse by computing either $(F \circ \mathcal{G}) \circ E$ or $F \circ (\mathcal{G} \circ E)$. Each composition operation is carried out using a standard composition algorithm, rather than the more commonly used approach of doing a 3-way composition directly. In the experiments below, since I use ε-free grammars and acyclic WFSTs, the bottom-up composition algorithm (§2.3.2.2) may be utilized. Once the first composition is computed, the resulting WSCFG must be inverted (§2.1.3.5). Since the composition algorithm used operates more efficiently with a determinized grammar, the grammar is left-factored during the inversion process as well (Klein and Manning, 2001).

Analysis. The bottom-up composition algorithm runs in $O(|\mathcal{G}| \cdot n^3)$ time, where n is the length of the input being parsed and $|\mathcal{G}|$ is a measure of the size of the grammar (Graham et al., 1980). Since the grammar term is constant for most typical parsing applications, it is generally not considered carefully; however, in the two-parse algorithm, the size of the grammar term for the second parse is not $|\mathcal{G}|$ but $|F \circ \mathcal{G}|$, which clearly depends on the size of the input F; and so understanding the impact of this term is key to understanding the algorithm's run-time.

If $\mathcal{G}$ is assumed to be an ε-free WSCFG with non-terminals V and maximally two non-terminals in a rule's right hand side, and n is the number of states in F (corresponding to the number of words in the $\mathbf{f}$ in a sentence pair $\langle \mathbf{e}, \mathbf{f} \rangle$), then the number of nodes in the parse forest $F \circ \mathcal{G}$ will be $O(|V| \cdot n^2)$. This is easy to see since it is known from above that $V_{\circ F}$ consists of symbols from V annotated with *pairs* of states, and there are $n+1$ states in F. The total number of rules will be $O(|V| \cdot n^3)$, which occurs when every new non-terminal can be derived from all possible binary splits of the span it dominates. This bound on the number of rules implies that $|F \circ \mathcal{G}| \in O(n^3)$. However, how tight these bounds are depends on the ambiguity in the grammar with respect to the input: to generate

n^3 edges, every item in every cell must be derivable by every combination of its subspans. Most grammars are substantially less ambiguous. Therefore, the worst case run-time of the two-parse algorithm is $O(|V| \cdot n^3 \cdot n^3 + |\mathcal{G}| \cdot n^3) = O(|V| \cdot n^6)$, the same as the bound on the binary ITG algorithm. Note that while the ITG algorithm requires that the SCFGs be rank-2 and in a normal form, this analysis of the two-parse algorithm analysis holds as long as the grammars are rank-2 and ε-free. Since many widely used SCFGs meet these criteria, including hierarchical phrase-based translation grammars (Chiang, 2007), SAMT grammars (Zollmann and Venugopal, 2006), and phrasal ITGs (Zhang et al., 2008), an analysis of ε-containing and higher rank grammars is left to future work.

2.3.4.2 Two-parse algorithm experiments

An empirical characterization of the performance of the two-parse algorithm is now provided by comparing the running time of the two-parse algorithm with that of alternative synchronous parsing algorithms, on synchronous parsing tasks for two different classes of SCFGs (each described below). In both experiments, a synchronous context-free grammar, $\mathcal{G}$ is constructed from a parallel corpus C using established grammar induction techniques. Then, the sentence pairs $\langle \mathbf{f}_i, \mathbf{e}_i \rangle$ in the parallel corpus C are iterated over, and two identity WFSTs, F from $\mathbf{f}_i$, and E from $\mathbf{e}_i$, are constructed. Both E and F are defined os as to have a single path with weight $\overline{1}$. The time required to compute the compute the synchronous parse forest $(F \circ \mathcal{G}) \circ E$ using the two-parse algorithm is measured and compared to that of another standard synchronous parsing algorithm.

Phrasal ITGs. In the first experiment, I compare performance of the two-parse algorithm and the $O(n^6)$ ITG parsing algorithm on an Arabic-English phrasal ITG alignment task. The corpus consisted of 120k sentence pairs (3.3M Arabic tokens; 3.6M English tokens) drawn from the NIST MT evaluation newswire training data. Sentences were filtered to a length of maximally 64 tokens on either side. For $\mathcal{G}$, I used a variant of the phrasal ITG described by Zhang et al. (2008). The restriction that phrases contain exactly a single alignment point was relaxed, instead the grammar was restricted to contain all phrases consistent with the word-based alignment up to a maximal phrase size of 5. This resulted in a synchronous grammar with 2.9M rules. Figure 2.14 plots the average run-time of the two algorithms as a function of the Arabic sentence length, and Table 2.6 shows the overall average run-times. Both presentations make clear that the two-parse approach is dramatically more efficient. In total, aligning the 120k sentence pairs in the corpus completed in less than 4 hours with the two-parse algorithm but required more than 1 week with the baseline algorithm.[30]

Table 2.6: Comparison of synchronous parsing algorithms on Arabic-English.

Algorithm	avg. run-time (sec)
ITG alignment	6.59
Two-parse algorithm	0.24

“Hiero” grammars. In the second experiment, I evaluate an alternative approach to computing a synchronous parse forest that is based on *cube pruning* (Huang and Chiang,

[30]A note on implementation: the ITG aligner was implemented quite efficiently; it only computed the probability of the sentence pair using the inside algorithm and did not build a representation of the parse chart. With the two-parse aligner, the complete item chart was stored during both the first and second parses. Therefore the implementation was biased in favor of the baseline ITG parsing algorithm.

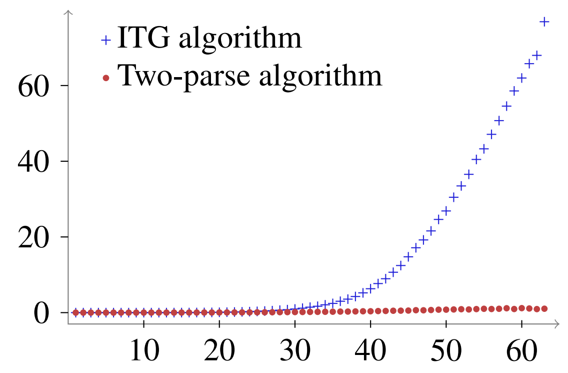


Figure 2.14: Average synchronous parser run-time (in seconds per sentence) as a function of Arabic sentence length (in words).

2007). While more commonly used to integrate a target m-gram LM during decoding, Blunsom et al. (2008a), who required synchronous parses to discriminatively train an SCFG translation model, repurposed this algorithm to discard partial derivations during translation of $\mathbf{f}$ if the derivation yielded a target m-gram not found in $\mathbf{e}$ (p.c.). I replicated their BTEC Chinese-English system and compared the speed of their 'cube-parsing' technique and the two-parse algorithm. The BTEC Chinese-English corpus consists of 44k parallel sentences (330k Chinese tokens; 360k English tokens). The SCFG was extracted from a word-aligned corpus, as described in Chiang (2007), and contained 3.1M rules with RHS's consisting of a mixture of terminal and non-terminal symbols, up to rank 2. To the extent possible, the two experiments were carried out using the exact same code base, which was a C++ implementation of an SCFG-based decoder. Figure 2.15 plots the average run time as a function of Chinese sentence length, and Table 2.7 compares the average per sentence synchronous parse time. The BTEC timing plot is less smooth than the Arabic-English plot for two reasons. First, there are fewer longer sentences in the BTEC corpus in comparison to the Arabic-English newswire corpus, so the average time

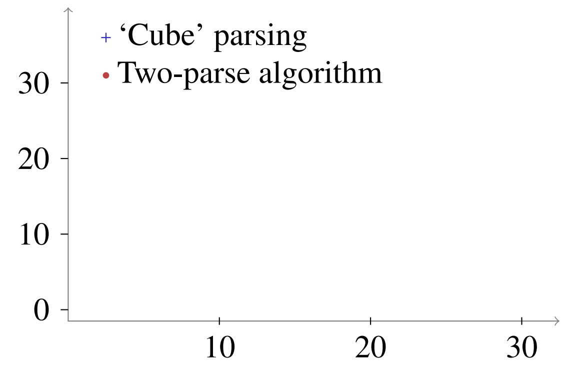


Figure 2.15: Average synchronous parser run-time (in seconds per sentence) as a function of Chinese sentence length (in words).

is estimated from a smaller population so there will be some additional variance due to the smaller sample sizes. But, more significantly, the running time of the cube parsing algorithm depends much more strongly on the contents of the grammar than the ITG parsing algorithm does (whose run-time is almost wholly determined by the sentence lengths). Again, the timing differences are striking.

Table 2.7: Comparison of synchronous parsing algorithms on Chinese-English.

Algorithm	avg. run-time (sec)
'Cube' parsing (Blunsom et al., 2008a)	7.31
Two-parse algorithm	0.20

Discussion of two-parse results. As the results of the two experiments just reported show, the two-parse strategy clearly outperforms both the ITG parsing algorithm (Wu, 1997), as well as the 'cube-parsing' technique for synchronous parsing. The latter result points to a connection with recent work showing that determinization of edges before LM integration leads to fewer search errors during decoding (Iglesias et al., 2009). Taken together, this suggests it may be worth rethinking the dominant language model integration strategy (i.e., cube pruning) that is used in syntactic translation models.

These results are somewhat surprising in light of work showing that 3-way composition algorithms for FSTs operate far more efficiently than performing successive pairwise compositions (Allauzen and Mohri, 2009). This is certainly because the 3-way algorithm used here (the ITG algorithm) does an exhaustive search over all n^4 span pairs without awareness of any top-down constraints. This suggests that faster composition algorithms that incorporate top-down filtering may still be discovered.

2.4 Inference

So far in this chapter, I have introduced weighted sets, two tractable relations for them (WFSTs and WSCFGs), and intersection and composition algorithms. I now turn to two inference problems that will come up repeatedly in the remainder of this dissertation: computing the total weight of all paths in a WFST (or equivalently, derivations in a WSCFG) and computing marginal transition weights in a WFST (or, equivalently, marginal rule weights in a WSCFG).[31] Before describing algorithms for computing the total weight and marginal weights, I discuss a common representation for both WFSTs and WSCFGs which enables a statement of generic algorithms that applies to both classes of inputs.

Note that the algorithms in this section are specific to weighted sets represented by

[31]Finding the *k*-best derivations of a WSCFG or WFST when path weights define a partial ordering is also a common problem; however, the models considered in later chapters do not require it so I do not discuss this. For details, the topic has been explored in considerable depth by Huang (2008). Like the algorithms for computing the total weight and edge marginals, the *k*-best algorithm operates on a unified hypergraph representation.

WFSTs or WSCFGs, not general weighted sets.

2.4.1 Correspondences between WFSTs and WSCFGs

Table 2.8 summarizes the correspondences between elements of WFSTs (§2.2.1) and elements of WSCFGs (§2.2.2). While these two classes of grammars have different generative power, the two formalisms are similar enough that a number of algorithms can be designed so as to apply to objects from both classes without special handling. To do so, I will assume are represented in as a directed hypergraph, where WFSTs are directed acyclic graphs, and I will use names in the first column of the table to refer, in a generic way, to elements of WSCFGs or WFSTs. For example, rather than referring specifically to transitions (in a WFST) or rewrite rules (in a WSCFG), I will use the term *edges* which can refer to *both*.

This common representation for WFSTs and WSCFGs can be understood as treating WFSTs as a degenerate form of WSCFG, one where there is always a single nonterminal on the left edge of every rewrite string. The close relationship between WFSTs and WSCFGs was noted above (§2.2.2). Furthermore, the correspondence is related to the fact that the regular languages (which are generated by FSAs) are a proper subset of the context-free languages (which are generated by CFGs).

2.4.2 Computing the total weight of all derivations

It is often necessary to compute the $\oplus$-total weight of all derivations in a hypergraph (WFST or WSCFG), $\mathcal{G}$. For example, computing the maximum weight derivation with

Table 2.8: Summary of correspondences between WFSTs (§2.2.1) and WSCFGs (§2.2.2).

	symbol	WFST	WSCFG
Graph. rep.	$\mathcal{G}$	directed graph	B-hypergraph (Gallo et al., 1993)
Node	q	state	non-terminal
Edge	e	transition	rewrite rule
Edge weight	$w(e)$	$w(e)$	$\upsilon(r)$
Edge tail nodes	$\mathbf{t}(\mathrm{e})$	$\mathrm{p(e)}$	RHS non-terminals (0 or more)
Edge head node	$\mathrm{n(e)}$	$\mathrm{n(e)}$	$\textsc{lhs}(\mathrm{r})$
Edge arity	$\lvert\mathbf{t}(e)\rvert$	1	# of non-terminals in r
In-edges	$B(q)$	$\{e : n(e) = q\}$	$\{r : \textsc{lhs}(r) = q\}$
Goal	q_{goal}	final state	start symbol
Source		initial state	—
Derivation	$\mathbf{d}$	path (π)	derivation
Output yield	$o(\mathbf{d})$	$o(\pi)$	$\textsc{rhs}_o(\mathbf{d})$
Input yield	$i(\mathbf{d})$	$i(\pi)$	$\textsc{rhs}_i(\mathbf{d})$
Total weight	$w_K(\mathcal{G})$	FORWARD	INSIDE
Edge marginal	$\gamma(e)$	FWDBACKWARD	INSIDEOUTSIDE
Max derivation	$\mathbf{d}_{\max}(\mathcal{G})$	shortest path	best derivation

the tropical semiring can be used to find the maximum derivation in the semiring, which is a frequently used decision rule. A naïve approach to computing this quantity would be to enumerate all derivations in $\mathcal{G}$, compute their weights (which decompose multiplicatively over the edges), and sum. Unfortunately, $\mathcal{G}$ will often contain a number of derivations that is exponential in its size (there may even be an infinite number of derivations), making this an intractable proposition.

The INSIDE algorithm, which takes a hypergraph representation, $\mathcal{G}$, of a WFST or WSCFG and an arbitrary semiring, K, is given in Figure 2.16 and can be used to find the total weight of a WFST or WSCFG in time and space that is linear in the size of $\mathcal{G}$ (measured by the number of edges and nodes), using dynamic programming. The algorithm computes a vector weights of paths terminating at successive nodes in the hypergraph rep-

```
1: function INSIDE($\mathcal{G}, K$)          ▷ $\mathcal{G}$ is an acyclic hypergraph and $K$ is a semiring
2:     for $q$ in topological order in $\mathcal{G}$ do
3:         if $B(q) = \emptyset$ then
4:             $\alpha(q) \leftarrow \overline{1}$          ▷ assume states with no in-edges are axioms
5:         else
6:             $\alpha(q) \leftarrow \overline{0}$
7:             for all $e \in B(q)$ do          ▷ all in-coming edges to node $q$
8:                 $k \leftarrow w(e)$
9:                 for all $r \in \mathbf{t}(e)$ do          ▷ all tail (previous) nodes of edge $e$
10:                    $k \leftarrow k \otimes \alpha(r)$
11:                $\alpha(q) \leftarrow \alpha(q) \oplus k$
12:    return $\alpha$
```

Figure 2.16: The INSIDE algorithm for computing the total weight of a non-recursive WFST or WSCFG.

resentation of the input, and $\alpha(q_{goal})$ is the total weight, the quantity in Equation (2.6).

$$\alpha(q_{goal}) = \bigoplus_{\mathbf{d} \in \mathcal{G}} \bigotimes_{e \in \mathbf{d}} w(e) \tag{2.6}$$

The INSIDE algorithm assumes a non-recursive (acyclic) input.[32] In this statement of the algorithm, I further assume that there is only a single goal node q_{goal}, with accept weight $\overline{1}$. Keep in mind that any hypergraph with multiple goal nodes (or a goal node with a non-1 weight) can be converted to a such a graph by adding an edge with an ε-label whose weight is equal to the goal node's accept weight. The non-recursive assumption that is made means that the nodes in $\mathcal{G}$ can be topologically ordered, meaning that any node q that directly or indirectly depends on another node r is ordered after r. Dependency is defined by asserting that the tail nodes of an edge are dependents of the edge's head node.

[32]For a discussion of the issues surrounding computing total weights of recursive grammars, which have an infinite number of derivations, refer to Goodman (1999).

2.4.3 Computing marginal edge weights

The marginal weight of an edge e, $\gamma(e)$, is the $\oplus$-total weight of all derivations that include e. Like the total derivation weight computed by the INSIDE algorithm, the marginal weight of an edge can be computed more efficiently than by enumerating all applicable derivations using the INSIDEOUTSIDE dynamic programming algorithm that is given in Figure 2.17.[33] In a probabilistic FSA or CFG, under the probability semiring, the edge marginal is the probability that a particular transition will be taken (or rule will be used). When using real-valued weights and the tropical semiring, the marginal weight is the maximum score of all derivations that include e. When using the count semiring, the marginal weight is the number of derivations that include e.

2.5 Summary

This chapter has set up a general model of language processing in terms of weighted sets and weighted binary relations. Inputs that are unambiguous can be accounted for in the model, as well as ones where inputs (and outputs) are inherently ambiguous, which will be my focus in subsequent chapters. I then described how weighted sets and relations can be encoded as WFSTs and WSCFGs. Although the operations for manipulating and combining multiple WFSTs have been well-studied, there is much less relevant research for the problems encountered when combining WFSTs and WSCFGs. I therefore presented several algorithms for intersection and composition of WFSAs/WFSTs and

[33] A slightly different formulation of the INSIDEOUTSIDE algorithm is sometimes given, for example by Li and Eisner (2009). Their formulation is useful when the algorithm is used to compute the product of a marginal and another function, for example when computing expectations.

```
1: function OUTSIDE($\mathcal{G}, K, \alpha$)                    ▷ $\alpha$ is the result of INSIDE($\mathcal{G}, K$)
2:     for all $q \in \mathcal{G}$ do
3:         $\beta(q) \leftarrow \overline{0}$
4:     $\beta(q_{goal}) = \overline{1}$
5:     for $q$ in reverse topological order in $\mathcal{G}$ do
6:         for all $e \in B(q)$ do                         ▷ all in-coming edges to node $q$
7:             for all $r \in \mathbf{t}(e)$ do            ▷ all tail (previous) nodes of edge $e$
8:                 $k \leftarrow w(e) \otimes \beta(q)$
9:                 for all $s \in \mathbf{t}(e)$ do        ▷ all tail (previous) nodes of edge $e$, again
10:                    if $r \neq s$ then
11:                        $k \leftarrow k \otimes \alpha(s)$        ▷ incorporate inside score
12:                $\beta(r) \leftarrow \beta(r) \oplus k$
13:    return $\beta$
```

```
1: function INSIDEOUTSIDE($\mathcal{G}, K$)                              ▷ compute edge marginals
2:     $\alpha \leftarrow$ INSIDE($\mathcal{G}, K$)
3:     $\beta \leftarrow$ OUTSIDE($\mathcal{G}, K, \alpha$)
4:     for edge $e$ in $\mathcal{G}$ do
5:         $\gamma(e) \leftarrow w(e) \otimes \beta(n(e))$      ▷ edge weight and outside score of edge's head node
6:         for all $q \in \mathbf{t}(e)$ do
7:             $\gamma(e) \leftarrow \gamma(e) \otimes \alpha(q)$                  ▷ inside score of tail nodes
8:     return $\gamma$                                   ▷ $\gamma(e)$ is the edge marginal of $e$
```

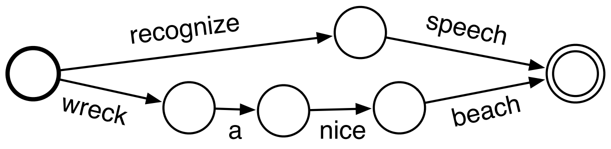
Figure 2.17: The OUTSIDE and INSIDEOUTSIDE algorithms for computing edge marginals of a non-recursive WFST or WSCFG.

WCFGs/WSCFGs. Although these are closely related to previous work in parsing and formal language theory, this is the first time the weighted intersection and composition processes have been formalized for general inputs. Finally, I showed that the composition algorithm leads to an alternative solution to the synchronous parsing problem. For two common SCFG types, this algorithm turns out to be far more efficient than existing specialized synchronous parsing algorithms. Algorithms for performing inference over non-recursive WFSTs and WSCFGs using a common representation of both classes of structures based on hypergraphs were then reviewed.

3 Finite-state representations of ambiguity

One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. 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 (1947)



–Speech Recognition 101

In the following chapters, I focus on the task of *statistical machine translation* (SMT; Brown et al. (1993)) to provide various situations at which it is useful to represent ambiguity of elements rather than single elements.[1] Since SMT furnishes most of the tasks upon which the approach described in the previous chapter is to be evaluated, I begin this chapter with a brief introduction to machine translation (§3.1). For a more comprehensive overview, refer to the recent survey by Lopez (2008a) or the textbook by Koehn (2009).

[1]This chapter contains material originally published in Dyer (2007) and Dyer et al. (2008).

Although SMT was originally formulated in terms of concepts from probability and information theory, I will recast the problem in terms of the weighted set operations described in the previous chapter, which subsume and extend the older models.

While much research attempts to improve models of the translation process and improve the resolution of the ambiguities that are inherent in these models, this chapter considers the translation problem when the *input* to the translation component is itself ambiguous—for example, when the task is to translate the output of a statistical automatic speech recognition (ASR) system. I then show that even in the context of text-to-text translation, where the identity of the input would seem to be *un*ambiguous, the techniques of translating a weighted set of inputs can be utilized so as to model decisions stochastically that would otherwise be made arbitrarily (such as what the optimal segmentation of the input is), leading to major improvements over strong baselines in a variety of machine translation tasks.

In this chapter, I focus on using WFSTs to represent input alternatives, in particular I use a restricted class called a *word lattice* to do so. I then turn to a discussion of the issues that arise when translating input structured as a WFST (§3.2). A description of experiments with a number of different kinds of ambiguous inputs follows (§3.3).

Chapter contributions. In the introductory section describing machine translation, I define a novel semiring, which I name the *upper envelope semiring*. I show that the hypergraph MERT algorithm described by Kumar et al. (2009) is equivalent to running the INSIDE algorithm over a hypergraph with this semiring. Although this algorithm is no more powerful or efficient than the Kumar et al. algorithm, it is expressed in terms of the familiar concepts of semirings and the INSIDE algorithm, which emphasizes its relationship to other algorithms used in natural language processing that the more specialized algorithm obscures. In the experimental section, I show that hierarchical phrase-based translation models can efficiently translate input encoded as a word lattice. I provide experimental evidence that translation quality (as measured by BLEU) can be improved by encoding ambiguous alternatives of the input in a word lattice, rather than making a decision about the analysis before translation. In particular, I show that decisions that are typically made during system development (such as the kind of source language segmentation or amount of morphological simplification) can be treated as a source of ambiguity and encoded in a lattice, leading to improved translation. These improvements hold for both finite-state and context-free translation models.

3.1 An introduction to statistical machine translation

Statistical machine translation (SMT) is typically formulated as a statistical inference problem; however, since more generally weighted (i.e., not probabilistic) models have become the dominant paradigm in SMT (Och, 2003; Zollmann et al., 2008), I will formulate the problem in terms of the weighted set operations defined in the previous chapter. A general SMT system consists of a weighted transducer $\mathcal{G}$ that relates sentences in some *source language* (conventionally designated $\mathbf{f}$) to sentences in some *target language* (conventionally designated $\mathbf{e}$).[2] $\mathcal{G}$ is called a *translation model* and assigns weights. I will assume it is structured either as a WFST (§2.2.1) or WSCFG (§2.2.2). It typically has

[2]The use of **f** and **e** as variable names is the result of the first statistical machine translation experiments being carried on out the **F**rench-**E**nglish Canadian parliamentary proceedings.

three responsibilities: to change the order of the words in the source sentence into the appropriate target order (if it differs), to translate the words from the source language into words in the target language, and to assign a weight to each translation hypothesis in the output set according to how faithful the translation is to the original meaning. After translation, a target language model $\mathcal{E}$ is typically used, which assigns a weight (usually a log-probability weighted by some factor) to every string in the target language indicating how fluent and grammatical a translation hypothesis is.

In the standard model, the transducer is applied to an unweighted input set F consisting of a single sentence $\mathbf{f}$ in the source language using composition. Therefore, $F \circ \mathcal{G}$ produces a weighted set over outputs in the target language. Whether $F \circ \mathcal{G}$ is a WFST or a WSCFG, I will assume that it defines a finite set of strings, or, equivalently stated, that its graphical representation is *acyclic*.[3] $F \circ \mathcal{G}$ is then rescored by applying the target language model $\mathcal{E}$, again with a composition operation, that is $F \circ \mathcal{G} \circ \mathcal{E}$. An output from the system is chosen by applying a decision rule to the target projection, that is $(F \circ \mathcal{G} \circ \mathcal{E})\downarrow$. I will select $\mathbf{d}_{\max}$, that is, make use of the maximum weight decision rule (§2.1.5.1), to find the maximum weighted derivation.[4]

To summarize the introduction to statistical machine translation, I will discuss language models (§3.1.1) and two common forms of translation models (§3.1.2): phrase-based translation models (which make finite-state assumptions) and hierarchical phrase-based translation models (which are context-free). Then I will discuss model parameteri-

[3]This assumption holds for the translation models considered here. Some others, for example the IBM lexical translation models, define output strings of unbounded length by permitting insertions (Brown et al., 1993).

[4]The term derivation is used to refer to both paths in a WFST and derivations in a WSCFG. Refer to §2.4.1 for more information about terminology.

zation using linear models (§3.1.3), the optimization of such models using minimum error training (§3.1.4), and conclude with a discussion of the evaluation of machine translation (§3.1.5).

3.1.1 Language models

A language model $\mathcal{E}$ assigns a score to every string in a language that reflects the string's grammaticality and well-formedness. As such, it is just an instance of a weighted set (of infinite cardinality), and any of the weighted set representations discussed in the previous chapter may be used to represent it. In this dissertation, I will use n-gram language models, which assign (log) probabilities to strings of any length using a order-n Markov assumption (Manning and Schütze, 1999). The Markov assumption states that the probability of a word in some context depends only on the recent context (where recency is measured in the number of words). For example, using $n = 2$, this is:

$$\begin{aligned}
p(\mathbf{e}) &= p(e_1 e_2 \dots e_n) \\
&= p(e_1)p(e_2|e_1)p(e_3|e_1e_2)p(e_4|e_1e_2e_3)\cdots p(e_n|e_1\dots e_{n-2}e_{n-1}) \\
&\approx p(e_1)p(e_2|e_1)p(e_3|\overline{e_1}e_2)p(e_4|\overline{e_1e_2}e_3)\cdots p(e_n|\overline{e_1\dots e_{n-2}}e_{n-1}) \ , \\
&= p(e_1)p(e_2|e_1)p(e_3|e_2)p(e_4|e_3)\cdots p(e_n|e_{n-1}) \ ,
\end{aligned}$$

and for general m,

$$= \prod_{i=1}^{|\mathbf{e}|} p(e_i|e_{i-m+1}\dots e_{i-1})$$

The conditional probability distributions that n-gram models are composed of are estimated from large corpora of monolingual text.

Because of the linear Markov assumption, language models are equivalent to WFSAs (Allauzen et al., 2003). Any n-gram language model can be explicitly encoded as a WFSA (or equivalently, an identity relation WFST). While a naïve encoding will have $|\Sigma|^{n-1}$ states (each state will correspond to a context of $n-1$ words and the $|\Sigma|$ transitions leaving it will be weighted with the conditional probability of that symbol, given the context implied by the state), it is often possible to have much more compact encodings of models, even those that make use of back-off smoothing.

Although it is possible to encode a language model as a WFST and use standard composition algorithms to incorporate the language model, many specialized language model composition algorithms exist that do not require explicitly instantiating an n-gram language model as a WFST (Huang and Chiang, 2007; Koehn et al., 2003). In the experiments used in this dissertation, the approximate composition algorithm known as *cube pruning* (Huang and Chiang, 2007) is utilized to incorporate an n-gram language model into translation models with a context-free component, and the beam-search approach described by Koehn et al. (2003) is used with phrase-based models.

Unless otherwise noted, experiments use a 3-gram language model trained on the target side of the parallel training data. In the case of systems evaluated on test sets from NIST MT Evaluations, the training data was been supplemented with the AFP and Xinhua portions of the English Gigaword corpus, version 3. Evaluations that make use of Workshop on Statistical Machine Translation (WMT) data use the provided English monolingual training data. Modified Kneser-Ney smoothing is used for all language models (Chen and Goodman, 1996).

3.1.2 Translation models

Many translation models can be represented either as weighted finite-state transducers or weighted synchronous context-free grammars (Lopez, 2008a). I review a specific instance of each, phrase-based translation models (Koehn et al., 2003) and hierarchical phrase-based translation models (Chiang, 2007).

3.1.2.1 Phrase-based translation

Phrase-based translation (Koehn et al., 2003) models the translational equivalence between two languages using pairs of *phrases* (strings of words) in the source and target language that have the same meaning. Phrases, which are learned automatically from parallel corpora, are not required to correspond to phrases in any particular syntactic theory: they must only be contiguous sequences of words. Phrases are learned automatically from a word-aligned parallel corpus, which is a corpus of text in two languages, where the word-to-word (or phrase-to-phrase) correspondences are marked with alignment links. A word alignment may be visualized with a two dimensional grid with the words of each sentence in the x- and y-axes. Figure 3.1 shows an example word alignment. Word alignments are induced from a parallel corpus, typically using expectation maximization; for more information see Och and Ney (2003).

From every sentence pair, phrases are extracted so as to be *consistent* with the word alignment. Consistency means simply that if you draw a box around the proposed phrase pair in the alignment grid, then no alignment links will fall directly above, below, to the left or right of it (diagonal points may remain). Figure 3.1 shows two consistent pairs

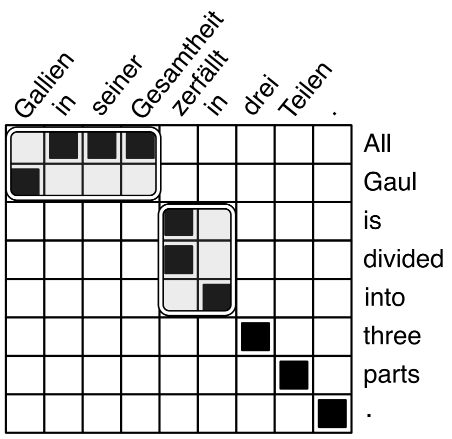


Figure 3.1: A German-English sentence pair with word alignment and two consistent phrases marked.

with rounded rectangles superimposed over the word alignment grid. Table 3.1 shows all the phrases (up to length 4 on the German side) that may be extracted from the example aligned sentence pair. When inducing a phrase based translation model, phrase pairs are using this heuristic and gathered into a collection which is referred to as a *phrase table*.

While phrase pairs capture lexical differences across languages, word order differences between the two languages are handled in one of two ways: 1) phrase pairs memorize local word order differences or 2) phrases from the source language may be translated 'out of order'. For an example of the latter, when translating from Arabic (a VSO language) into English (an SVO language), the subject in the Arabic sentence can be translated first, followed by the verb, leading to the correct English order.

Model features. The typical model features used with phrase based translation are: the log relative frequency of the phrase pairs used in a translation (in both directions), a lexical phrase translation log probability of each phrase pair, a count of the number of words and phrases used, a distortion score (typically representing how much reordering

Table 3.1: Phrases up to size 4 (source side) extracted from the aligned sentence pair in Figure 3.1.

German	English
Gallien	*Gaul*
zerfällt	*is divided*
in	*into*
drei	*three*
Teilen	*parts*
.	.
zerfällt in	*is divided into*
in drei	*into three*
drei Teilen	*three parts*
Teilen .	*parts .*
in seiner Gesamtheit	*All*
zerfällt in drei	*is divided into three*
in drei Teilen	*into three parts*
drei Teilen .	*three parts .*
Gallien in seiner Gesamtheit	*All Gaul*
zerfällt in drei Teilen	*is divided into three parts*
in drei Teilen .	*into three parts .*

was used), and the target language model log probability. These are combined in a linear model (§3.1.3).

Translation algorithms. Phrase based translation can be understood as composing an identity-transducer representing $\mathbf{f}$ with a WFST representing the phrase table,[5] a WFST representing the reordering model, and a WFST representing the target language model (Kumar et al., 2006). Standard WFST composition algorithms can be used. However, the number of states in $\mathbf{f} \circ \mathcal{G} \circ \mathcal{E}$ is in $O(2^d \cdot d^2 \cdot m)$, where d is a limit on how far a word may be reordered during translation (Lopez, 2009). Because of the intractably large number of states that result, it is more common to employ a heuristic beam-search composition

[5]An example WFST encoding of a phrase transducer is shown in Figure 5.2 (in Chapter 5)

algorithm to compute (with high probability) the maximum-weight path in $(F \circ \mathcal{G} \circ \mathcal{E})\downarrow$ or a highly weighted subset of it (Koehn et al., 2003). I review this algorithm now.

Phrase-based translation translates an input sentence **f** by selecting sub-spans of the input of that have not yet been translated and matching an entry in the phrase table, choosing a translation from among the possible translations listed and writing it down. Then another untranslated sub-span is selected, translated and placed to the right of the previous one. Once all words in **f** have been translated, creating a hypothesis **e** from left to right, the process completes. Both to reduce the size of the translation space and to prevent having to score implausible candidates with the relatively weak reordering models that are used, when translating source phrases in a non left-to-right order, the requirement is usually imposed that the distance from the first untranslated position in the sentence may not exceed d source words.

The phrase-based translation algorithm defines a search space in terms of states uniquely identified by the following elements:

1. The source *coverage vectors*, which is a bit string indicating what words in the source sentence have been translated.

2. The target language model context, which is the final $n-1$ words of the hypothesis in an n-gram Markov language model (but see Li and Khudanpur (2008)).

3. The position of the last source word translated, which is used to compute the distortion, or reordering, score.

The partial hypotheses, represented by what state they terminate in, are organized into stacks (priority queues), based on the number of words that are translated in them, and

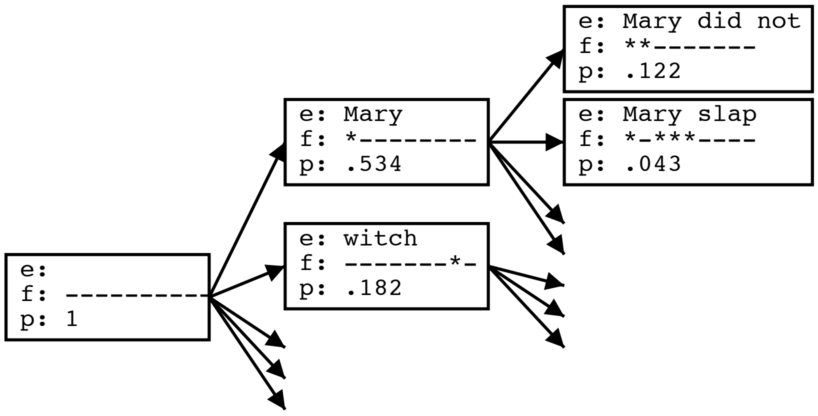


Figure 3.2: A fragment of the search space for a phrase-based decoder, reprinted from Koehn (2004).

translation proceeds by extending states from the stacks associated with ever increasing number of translated words. Partial hypotheses whose scores are too far away from the best-scoring hypothesis (using the linear model on the completed part of the hypothesis and a heuristic 'future cost' estimate that attempts to model how expensive it will be to translate the remainder of the sentence) are discarded. The others are extended by selecting an uncovered portion of the source sentence and generating a state. This state will either be new, in which case it is added to a stack, or otherwise it will be combined with an existing item, which is referred to as *recombination*. By only keeping track of a certain number of hypothesis in each stack (and discarding the rest), the running time of the translation process using this algorithm is can be bound by trading off running time for search errors. Figure 3.2 illustrates a phrase-based decoder search graph, translating the Spanish sentence *Maria no daba una bofetada a la bruja verde* (in English, *Mary did not slap the green witch*).

The space of items explored by this algorithm is structured as the WFST, $\mathbf{f} \circ \mathcal{G} \circ$

$\mathcal{E}$, where edges are permitted to have multiple words on them (Ueffing et al., 2002). Specifically, each item processed is a state in the WFST, and each extension by a phrase pair, is the transition label. Recombination of items indicates that a state has multiple incoming transitions.

Strengths and weaknesses. Phrase-based translation models are an efficient,[6] simple model that performs extremely well, especially when large amounts of parallel training data are available (Zollmann et al., 2008). Their two most serious weaknesses are an inability to model or efficiently represent long-range reordering patterns (that is, word order differences more than 10 or so words) and their inability to model discontinuous spans (but see Galley and Manning (2010)). Despite this, they are used in a number of high-quality translation products, such as Google Translate.

3.1.2.2 Hierarchical phrase-based translation

To address the weaknesses of phrase-based translation models, Chiang (2007) introduced hierarchical phrase-based translation models, which permit phrase pairs learned from data to contain *variables* into which phrases can be substituted, in a hierarchical manner. Not only does this mean that discontinuous spans can be modeled, but it also provides a means of modeling longer-range reordering patterns. With such gaps, phrase pairs therefore have the form of synchronous context-free grammar rules with a single non-terminal category,

[6]As noted above, their efficiency is due to the existence of good quality search heuristics.

X; for example:[7]

$$\begin{aligned}
\mathrm{X} &\rightarrow \langle \mathrm{X}\ \textit{in seiner Gesamtheit}, \textit{All}\ \boxed{1} \rangle \\
\mathrm{X} &\rightarrow \langle \textit{zerfällt in}\ \mathrm{X}, \textit{is divided into}\ \boxed{1} \rangle \\
\mathrm{X} &\rightarrow \langle \textit{den}\ \mathrm{X}\ \textit{hat der}\ \mathrm{X}\ \textit{gesehen}, \textit{the}\ \boxed{2}\ \textit{saw the}\ \boxed{1} \rangle
\end{aligned}$$

Like non-gap phrase pairs from above, these gapped rules are learned from word aligned corpora. Intuitively, they can be understood as arising by the 'subtraction' of a consistent phrase pair from a larger phrase pair. For more information refer to Chiang (2007) and Lopez (2008b).

To the automatically extracted rules are added two additional 'glue rules', which mean that sentence pairs can be derived by the grammar by simple left-to-right concatenation.

$$\begin{aligned}
\mathrm{S} &\rightarrow \langle \mathrm{X}, \boxed{1} \rangle \\
\mathrm{S} &\rightarrow \langle \mathrm{S}\ \mathrm{X}, \boxed{1}\ \boxed{2} \rangle
\end{aligned}$$

Because the translation model has the form of a WSCFG, translation can be carried out

[7]The notation used here is slightly different than Chiang's notation, which uses correspondence indices on both the input and outputs the rules. That is, where I write

$$\mathrm{X} \rightarrow \langle \textit{den}\ \mathrm{X}\ \textit{hat der}\ \mathrm{X}\ \textit{gesehen}, \textit{the}\ \boxed{2}\ \textit{saw the}\ \boxed{1} \rangle \ ,$$

Chiang, and a number of other authors following his lead, write:

$$\mathrm{X} \rightarrow \langle \textit{den}\ \mathrm{X}_1\ \textit{hat der}\ \mathrm{X}_2\ \textit{gesehen}, \textit{the}\ \mathrm{X}_2\ \textit{saw the}\ X_1 \rangle \ .$$

The decision to deviate from his convention was only to simplify the definition of well-formed WSCFG rules (see §2.2.2); the capacity of the grammars is not changed.

using any of the WFST-WSCFG composition algorithms introduced in the previous chapter (§2.3.1.5), which are similar to well-known monolingual parsing algorithms.[8] Since hierarchical phrase base translation models are constructed so that they do not contain any ε-rules, a bottom-up composition (parsing) algorithm is typically used for translation. The result of this composition operation is a WSCFG that encodes all translations of the source sentence in the output language. This object is typically encoded as a directed, ordered hypergraph and called a *translation forest*. Figure 3.3 shows an example hierarchical phrase based translation grammar and two representations of the translation forest generated by applying it to an example input sentence. Typically this forest will be composed with $\mathcal{E}$ and then a derivation will be selected, for example, by finding the highest-weighted derivation.

Since an n-gram language model is equivalent to a WFST (§3.1.1), any general WFST-WSCFG composition algorithm (such as the one described in §2.3) can be used to incorporate the language model. Although language model composition requires only polynomial time and space (as a function of the length of the input sentence), an exhaustive composition is nevertheless too expensive to be computed exactly (even for bigram language models), making a heuristic approach necessary. I therefore use an approximate composition algorithm called *cube pruning* (Huang and Chiang, 2007). Like the heuristic beam search used for phrase-based translation, cube pruning only composes the most promising derivations with the language model, making it possible again to trade running time for search errors.[9]

[8]The input sentence **f** is ‘parsed’ by the source side of the synchronous grammar, which induces a translation forest in the target side.

[9]Other approximate composition algorithms with similar characteristics (namely the ability to trade speed for accuracy), have been proposed as well (Venugopal et al., 2007).

f – input sentence: *diànzi shàng de māo*

$\mathcal{G}$ – translation model (weights not shown; start symbol is S):

S	$\rightarrow$	$\langle$X, $\boxed{1}\rangle$	X	$\rightarrow$	$\langle$X *zài* X, $\boxed{1}$ *in* $\boxed{2}\rangle$
X	$\rightarrow$	$\langle$*māo*, *a cat*$\rangle$	X	$\rightarrow$	$\langle$X *de* X, $\boxed{1}\,\boxed{2}\rangle$
X	$\rightarrow$	$\langle$*gǒu*, *a dog*$\rangle$	X	$\rightarrow$	$\langle$X *de* X, $\boxed{1}$ *'s* $\boxed{2}\rangle$
X	$\rightarrow$	$\langle$*shǔ*, *a rat*$\rangle$	X	$\rightarrow$	$\langle$X *de* X, $\boxed{2}$ *of* $\boxed{1}\rangle$
X	$\rightarrow$	$\langle$*diànzi shàng*, *the mat*$\rangle$	X	$\rightarrow$	$\langle$X *de* X, $\boxed{2}$ *on* $\boxed{1}\rangle$

$(\mathbf{f} \circ \mathcal{G})\downarrow$ – output-projected translation forest (as a CFG; start symbol is ${}_0S_4$):

${}_0S_4$	$\rightarrow$	${}_0X_4$	${}_0X_4$	$\rightarrow$	${}_0X_2$ *'s* ${}_3X_4$
${}_3X_4$	$\rightarrow$	*a cat*	${}_0X_4$	$\rightarrow$	${}_3X_4$ *of* ${}_0X_2$
${}_0X_2$	$\rightarrow$	*the mat*	${}_0X_4$	$\rightarrow$	${}_3X_4$ *on* ${}_0X_2$
${}_0X_4$	$\rightarrow$	${}_0X_2$ ${}_3X_4$			

$(\mathbf{f} \circ \mathcal{G})\downarrow$ – output-projected translation forest (as a directed hypergraph):

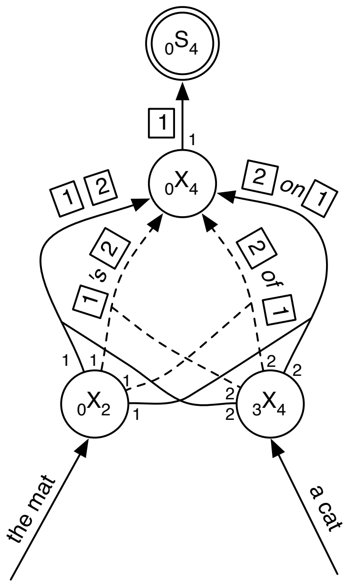


Figure 3.3: An example of a hierarchical phrase based translation. Two equivalent representations of the translation forest are given. Example adapted from Li and Eisner (2009).

Model features. The typical model features used with hierarchical phrase based translation are similar to those used in the original phrase based translation: the log relative frequency of the rule pairs used in a translation (in both directions), a lexical phrase translation log probability of each phrase pair, a count of the number of words and rules used, and the target language model log probability. These are combined in a linear model (§3.1.3).

Strengths and weaknesses. Hierarchical phrase-based translation models have become extremely popular since their introduction. Not only are they attractive from a modeling perspective (they model discontinuous elements, they deal more effectively with mid-range reordering phenomena, and they capture the intuition that language is hierarchical), they can deal with reordering in polynomial time and space, unlike phrase-based models (or any WFST model). On the other hand, hierarchical phrase-based translation models consist of grammars that can be much larger than an original phrase-based translation model, making them much more cumbersome to deal with large amounts of data. One solution is to extract the applicable grammar rules on-the-fly by indexing of the training data in a suffix array (Lopez, 2008b).

3.1.3 Model parameterization: linear models

Model parameterization refers to how weights are assigned to elements in the translation relation and language model. Although in later chapters, probabilistic interpretations of weighted sets will be required, these will still be instances of *linear models* that assign a real-valued weight to a *derivation*.[10] However, in general, this weight may or may not have a particular probabilistic interpretation. The weight of a derivation **d** under a linear model is defined to be the inner product of an m-dimensional weight vector $\Lambda = \langle \lambda_1, \lambda_2, \ldots, \lambda_m \rangle$ and an m-dimensional feature function $\vec{H}(\mathbf{d}) = \langle H_1(\mathbf{d}), H_2(\mathbf{d}), \ldots, H_m(\mathbf{d}) \rangle$ that is a function of the derivation:

$$f(\mathbf{d}, \Lambda) = \Lambda \cdot \vec{H}(\mathbf{d}) = \sum_{i=1}^{m} \lambda_i \cdot H_i(\mathbf{d})$$

It is further stipulated that global feature functions $H_i(\mathbf{d})$ decompose additively over edges in the derivation **d** in terms of *local* feature functions h_i. The decomposition can be written as follows:

$$H_i(\mathbf{d}) = \sum_{e \in \mathbf{d}} h_i(e)$$

As a result of this independence assumption, $\vec{H}(\mathbf{d}) = \sum_{e \in \mathbf{d}} \mathbf{h}(e)$ and if edge weights are assigned by the following:

$$w(e) = \Lambda \cdot \mathbf{h}(e) = \sum_i \lambda_i \cdot h_i(e) \ ,$$

then computing the total weight of the set using the tropical semiring (§2.1.1) computes the weight of the best derivation (or, if back-pointers are used, the best derivation itself).

Intuitively, the individual feature functions $H_i(\mathbf{d})$ represent some piece of evidence about a given *translation derivation*. By changing the weight vector, the relative impor-

[10] I use the general term *derivation* to designate what is commonly called a path in a WFST and a derivation in a WSCFG. I will refer to the units of a derivation as *edges*, corresponding to transitions in a WFST and rules in a WSCFG.

tance of different aspects of the relationship between inputs and outputs is emphasized.

It will often be necessary to compute the maximum derivation $\mathbf{d}_{\max}(A;\Lambda)$ under a linear model with parameters Λ from some set or relation A. This can be done in linear time using the INSIDE algorithm (§2.4) with the tropical semiring, where addition is defined to be the max operator, and edge weights are defined to be the inner product of the weight vector and the local features $\Lambda \cdot \mathbf{h}(e)$.

3.1.4 Minimum error rate training (MERT)

In the context of machine translation (both phrase-based and hierarchical phrase-based), the kinds of linear models just described rank the derivations representing alternative translations of inputs. With different settings of the weight vector Λ, a the model assign different weights resulting in a different ranking of outputs. Och's algorithm (Och, 2003) for minimum error rate training (MERT) is a gradient-free optimization method for setting the weight vector Λ in linear models (§3.1.3) that is widely used in machine translation (Koehn, 2009; Kumar et al., 2009; Lopez, 2008a; Macherey et al., 2008; Och, 2003). Although it is limited in the number of features that it can effectively optimize (maximally on the order of tens of features), it has two characteristics that make it particularly useful. Since it is used to optimize a small number of features, it can generally do so with relatively small amounts of development data (thousands of parallel sentences, typically a mere fraction of the perhaps millions of parallel sentences available for many languages). It is also possible to efficiently optimize non-differentiable loss functions, such as the 1-best BLEU score, as well as loss functions that are defined in terms of *global properties* of the model's hypothesized output (and hence do not decompose over the model structure).[11] Second, MERT optimizes the parameters of the model so as to obtain the best possible performance on a development set in terms of the the maximum weight derivation decision rule. Since finding the maximum weight derivation is typically the most efficient decoding strategy possible, models learned by MERT are particularly effective in situations where decoding speed will be of concern.

I do not give a complete overview of the MERT algorithm or an empirical verification of its effectiveness (for a comprehensive introduction, refer to the above citations), but I will show that the line search inference procedure that it relies on can be expressed as a total weight computation with a particular semiring (e.g., using the INSIDE algorithm; see §2.4.2). Although the computation performed with the INSIDE algorithm using this semiring is equivalent to an algorithm that has been introduced previously (Kumar et al., 2009), by recasting the MERT algorithm in terms of familiar concepts, I hope to make it more accessible to readers from other disciplines who are familiar with semirings and inside algorithms, but who may not necessarily be comfortable with MERT. For example, researchers in parsing and speech recognition might find it useful to be able to refine their models using an optimizer that can target an arbitrary non-differentiable loss function.

3.1.4.1 Line search and error surfaces

Line search is a technique for minimizing an objective function $f : \mathbb{R}^m \to \mathbb{R}$. Briefly, the algorithm works as follows. Starting from an initial point $\Lambda \in \mathbb{R}^m$, a descent direction

[11] Note that while the global feature functions, H_k, are required to decompose with the structure of the model into sums of local feature functions, h_k, there is no such requirement for the error function.

$\mathbf{v} \in \mathbb{R}^m$ is chosen (the criteria for selecting the direction depend on the specific line search algorithm being used) and the algorithm searches in this direction to find the minimal value of the objective function is found (in the case of MERT, the line search will find the global minimum; however, other line search algorithms may only find local minima). Then, the starting point is updated to this point and a new descent vector is chosen. A common variant (and one that I use) is that many different descent direction vectors are searched at each iteration, and the minimal value of the objective is selected across all of them. The algorithm typically terminates when the objective fails to improve on successive iterations.

The MERT algorithm makes use of a line search algorithm for finding the parameters of a linear model (§3.1.3) that minimize the *error score* of the *maximum weight derivation* under the model, using those parameters. Each iteration j starts with a model $\Lambda^{(j)} \in \mathbb{R}^m$, a search direction $\mathbf{v} \in \mathbb{R}^m$, and searches the space of parameters given by $\tilde{\Lambda}(x)$, where $x \in (-\infty, \infty)$.

$$\tilde{\Lambda}(x) = \Lambda^{(j)} + x \cdot \mathbf{v} \tag{3.1}$$

For machine translation models, the development set that is used to optimize parameters consists of n sentences in the source language paired with references—typically one or more human translations—in the target language,[12] $\mathcal{D} = \{\langle \mathbf{f}^1, \mathbf{e}^1_{\text{ref}} \rangle, \ldots, \langle\langle \mathbf{f}^n, \mathbf{e}^n_{\text{ref}} \rangle\}$. An error function $E(\hat{\mathbf{e}}, \mathbf{e}_{\text{ref}})$ computes an error score given a hypothesis $\hat{\mathbf{e}}$ and a reference set $\mathbf{e}_{\text{ref}}$.[13] The error function must not depend on the model score of the maximum weight

[12]Recent work has also demonstrated that supplementing human-generated references with computer-generated references is very effective when using MERT to optimize BLEU (Madnani, 2010).

[13]To keep the notation simple I have assumed that references are sentences; however, in practice they may be sets of sentences or any other information (parse trees, etc.) used to compute an error score from a hypothesis translation.

derivation $\tilde{\Lambda}(x) \cdot \vec{H}(\mathbf{d})$ (or any other derivation). Succinctly, at each iteration j, the MERT algorithm performs the following optimization:[14]

$$\hat{x} = \arg \min_{x \in (-\infty,\infty)} \left\{ \sum_{i=1}^{n} E\big(\underbrace{o(\arg\max_{\mathbf{d}}(\mathbf{f}^i \circ \mathcal{G} \circ \mathcal{E}; \overbrace{\Lambda + x \cdot \mathbf{v}}^{=\tilde{\Lambda}(x)}))}_{\text{max weight translation of } \mathbf{f}^i \text{ given } \tilde{\Lambda}(x)}, \mathbf{e}^i_{\text{ref}}\big) \right\} \tag{3.2}$$

The value $\hat{x}$ is then used in Equation (3.1) to update the model parameters for the next iteration:

$$\Lambda^{(j+1)} = \Lambda^{(j)} + \hat{x} \cdot \mathbf{v}$$

In practice, at each iteration, many random starting points and many random direction vectors are chosen (each pair of which yields a different corpus error surface), and the minimum error score across *all* search directions and start points is computed. This computation can be trivially parallelized.

Because the line search in Equation (3.2) indicates that an uncountably infinite number of different models are searched, even a single starting point and search direction may seem daunting. I now look at how to compute this minimum exactly—and in finite time. Note that the inner product of any feature vector of a derivation $\vec{H}(\mathbf{d})$ and $\tilde{\Lambda}(x)$ defines a line in $\mathbb{R}^2$ where x is how far along $\mathbf{v}$ the starting parameters have been moved and y is the model score of the derivation, *not* its error score—MERT requires that a derivation $\mathbf{d}$ have a constant error score, regardless of its model score:

[14]In Equation (3.2), o indicates that output string yield of the derivation is used by the error function to score it against the reference (§2.4.1).

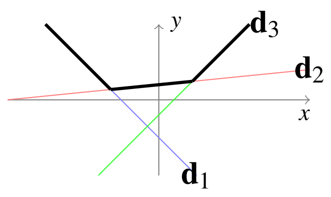


Figure 3.4: A set of three derivations as lines. The height on the y-axis represents the model score of the derivation as a function of x, which determines how far along the descent vector $\mathbf{v}$ the starting parameters Λ are translated.

$$
\begin{aligned}
f_{\mathbf{d}}(x) &= \tilde{\Lambda}(x) \cdot \vec{H}(\mathbf{d}) \\
&= (\Lambda^{(j)} + x \cdot \mathbf{v}) \cdot \vec{H}(\mathbf{d}) \\
&= \underbrace{\mathbf{v} \cdot \vec{H}(\mathbf{d})}_{a=\text{slope}} \cdot x \;\; + \underbrace{\Lambda^{(j)} \cdot \vec{H}(\mathbf{d})}_{b=y\text{-intercept}}
\end{aligned}
$$

Since it has been stipulated that global features of a derivation, $H_i(\mathbf{d})$, must decompose into local feature functions of edges (§3.1.3), this can be further rewriten as follows:

$$
\begin{aligned}
f_{\mathbf{d}}(x) &= \sum_{e \in \mathbf{d}} \Big(\mathbf{v} \cdot \mathbf{h}(e) \cdot x + \Lambda^{(j)} \cdot \mathbf{h}(e) \Big) \\
&= \underbrace{\left(\sum_{e \in \mathbf{d}} \mathbf{v} \cdot \mathbf{h}(e) \right)}_{a=\text{slope}} \cdot x + \underbrace{\left(\sum_{e \in \mathbf{d}} \Lambda^{(j)} \cdot \mathbf{h}(e) \right)}_{b=y\text{-intercept}}
\end{aligned}
$$

Given a *set* of competing derivations $\{\mathbf{d}_i\}$, for example, the set of different translations of an input under a translation model, each with its own feature vector, $\vec{H}(\mathbf{d}_i)$, this induces a set of lines $C = \{y = a_1 x + b_1, y = a_2 x + b_2, \ldots, y = a_k x + b_k\}$. Figure 3.4

shows a plot of the lines corresponding to the scores of 3 derivations, given a $\Lambda^{(j)}$ and $\mathbf{v}$. Furthermore, for every x, it is possible to select the best derivation in the set from such a plot. Moreover, the set of the most highly-weighted derivations at some x form an *upper envelope* (de Berg et al., 2000)– that is, given a model Λ and a direction vector $\mathbf{v}$, the best model score is piecewise linear in the step size x. Since MERT seeks to optimize the error count of the derivation with the highest model score, it need only pay attention to the derivations in the upper envelope, disregarding all others (hence, it is not necessary to explicitly search the range $(-\infty, \infty)$ nor is it necessary to explicitly search every derivation in a set!). In Figure 3.4, the three segments that make up the upper envelope shown in bold.[15]

At this point, it will be more convenient to think of each derivation line $y = f_{\mathbf{d}}(x)$ as a single *point*, taking advantage of the *point-line duality* (de Berg et al., 2000). This dual form represents each line $y = ax + b$ in the primal space with the point $(a, -b)$ in the dual space (conversely, primal points become lines in the dual). Note that the y-intercept is negated in the dual transform. Thus, the line corresponding to the derivation $\mathbf{d}$ is the dual point $\langle \sum_{e \in \mathbf{d}} \mathbf{v} \cdot \mathbf{h}(e), -\sum_{e \in \mathbf{d}} \Lambda \cdot \mathbf{h}(e) \rangle$. Figure 3.5 shows the relationship between primal and dual forms, and also displays the isomorphic geometric concepts of upper envelope (in the primal plane) and lower hull (in the dual plane).

The MERT algorithm finds parameters that minimize the error score of the *maximum weight derivations* of the model using those parameters—therefore, for each training instance, the possible maximum weight derivations correspond to lines that are part of the

[15]The figures in this section were generated using code adapted from Adam Lopez's PhD dissertation (Lopez, 2008b).

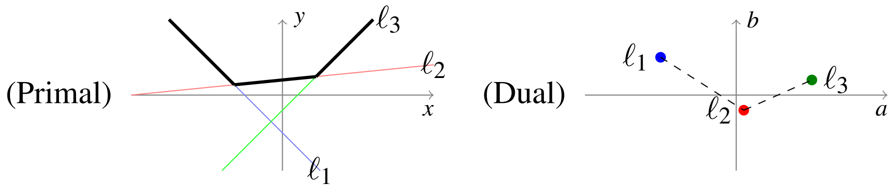


Figure 3.5: Primal and dual forms of a set of lines. The upper envelope is shown with heavy line segments in the primal form. In the dual plane, an upper envelope of lines corresponds to the extreme points of a lower hull (lower hull shown with dashed lines).

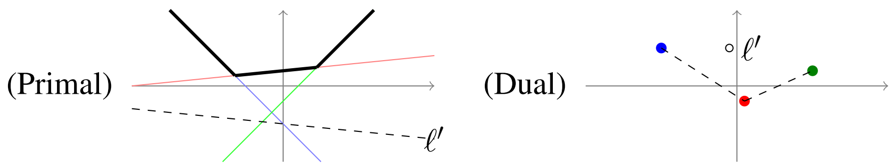


Figure 3.6: A hidden line ℓ' is obscured by the upper envelope in the primal form and (equivalently) is not part of the lower hull in the dual form.

upper envelope.[16] The algorithm can also exploit the fact that some lines (derivations) are not part of the upper envelope. For example in Figure 3.6, the line (ℓ') is hidden beneath the upper envelope. In the dual plane, an upper envelope of lines corresponds to the points that form a lower hull (de Berg et al., 2000). Hidden lines correspond to points that do not lie on the lower hull. Lines obscured beneath the upper envelope correspond to derivations that will never be the maximally weighted derivation, under any value of x, and the algorithm ignores them. However, a different starting Λ and direction vector $\mathbf{v}$ will typically produce an upper envelope containing lines corresponding to different derivations. For this reason, a number of Λ and $\mathbf{v}$ values are tried at each iteration.

For a set of derivations corresponding to alternative translations of an input, a starting weight vector Λ and a descent direction $\mathbf{v}$, the *transition points* (the x-coordinates

[16] If they were not part of the upper envelope, they would not be the maximum derivation—the upper envelope is defined by taking the maximum of a set of lines at each point.

where the lines of the upper envelope intersect) are the points where the best derivation in a set changes as the weights are changed by varying the model parameters according to Equation (3.1). The x-coordinate of the transition points (which is all that is necessary to compute) are the positions where segments in the upper envelope intersect, and (somewhat less obviously) they are equal to the negative slopes of the lines that form the lower hull in the dual plane.[17] Since finding the x-coordinate (actually, it will be a range on the x-axis) that minimizes the error count is the goal, an *error surface* can be created from an upper envelope by evaluating the error function E for each segment (corresponding to a unique derivation) in the upper envelope.[18] Figure 3.7 shows an example of how the error surfaces relate to the transition points in the upper envelope, using an example development set consisting of two sentences. By assumption, the error function is constant for a given derivation, so the error surface for a set of competing derivations is piecewise constant in x.

Since it is assumed that the error function decomposes linearly across sentences in a corpus, the per-sentence error surfaces can be merged into a corpus-level error surface by adding the error surfaces together.[19] Figure 3.8 shows how the error surfaces add, producing the development set error surface. Note that when adding error surfaces, the

[17]I omit the derivation showing the correspondence between the dual slopes and transition points since it requires only basic algebra to show and provides no obvious further insight into this problem.

[18]Note that this means the algorithm need only have evaluate the error function for the (possibly small) number of hypotheses whose derivations correspond to lines that are actually part of the upper envelope. This fact has been exploited to use human evaluators to score hypotheses manually (Zaidan and Callison-Burch, 2009).

[19]Although many useful metrics do not appear to decompose additively across independent training examples, these can often still be optimized with MERT. In these cases, the error function is defined so as to return vectors of sufficient statistics which are added together across sentences and used to compute a corpus level metric at the end. For example, both BLEU and F-measure can be optimized, although neither of these decomposes linearly across instances. When BLEU is optimized, the sufficient statistics are clipped n-gram match counts and hypothesis lengths; when optimizing F-measure, they are the number of matches and the number of misses.

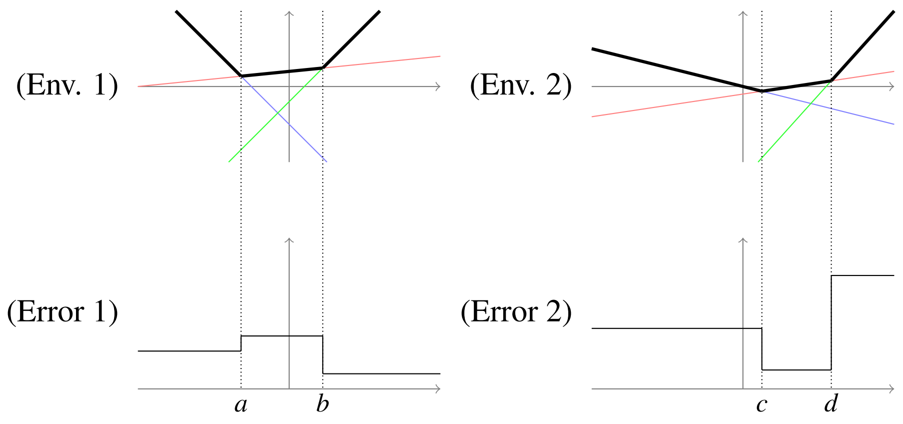


Figure 3.7: Each segment from the upper envelope (above) corresponds to a hypothesis with a particular error score, forming a piecewise constant error surface (below). The points a, b, c, and d are the transition points.

transition points in the sum of the error surfaces are the union of the transition points of the individual error surfaces.

Once the error surface for the entire development set has been computed, the segment with the lowest error is chosen. This is accomplished with a simple search through the segments of the development set error surface. Even though the entire space $x \in (-\infty, \infty)$ is implicitly searched by the line search, this is guaranteed to be manageable in size (see discussion below). In the example development set consisting of 2 sentences, the segment with the best error score is the segment bd, and the value $\hat{x}$ is selected that is the midpoint of this segment. This $\hat{x}$ is then used to update the parameter settings for the next iteration: $\Lambda^{(j+1)} = \mathbf{v} \cdot \hat{x} + \Lambda^{(j)}$.

3.1.4.2 The upper envelope semiring

In the previous section, I showed how to optimize weights of a linear model using an error function E so that the error score of the maximum weight derivations of a model applied to

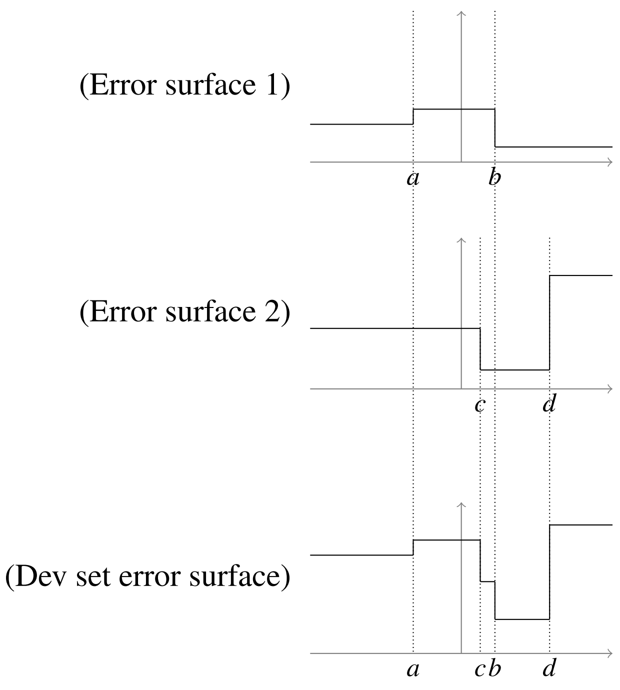


Figure 3.8: Adding two error surfaces (each from a single sentence) to create a corpus error surface corresponding to the error surface of a development set consisting of two sentences.

a development set is minimized. However, the algorithm crucially depends on being able to efficiently compute the upper envelope for a set of derivations, given a starting weight vector Λ and descent direction $\mathbf{v}$. The original presentation of the algorithm advocated approximating the contents of this set using k-best lists (Och, 2003). However, a k-best list is only a minuscule fraction of the derivations encoded in $\mathbf{f}^i \circ \mathcal{G} \circ \mathcal{E}$, which makes this approach unstable. However, Macherey et al. (2008), whose work was extended by Kumar et al. (2009), showed that for a given Λ and $\mathbf{v}$ the upper envelope of derivations could be computed exactly using dynamic programming for WFST and WSCFG representations of $\mathbf{f}^i \circ \mathcal{G} \circ \mathcal{E}$. Furthermore, while $\mathbf{f}^i \circ \mathcal{G} \circ \mathcal{E}$ does in general encode a set of derivations whose size is exponential in the size (in terms of edges and nodes) of $\mathbf{f}^i \circ \mathcal{G} \circ \mathcal{E}$, the upper envelope is guaranteed to contain a number of segments that is only linear in this size (Ku-

Table 3.2: Upper envelope semiring. See text for definitions of LOWERHULL and the run times of the operations.

Element	Definition
$\mathbb{K}$	$\{\ell_1, \ell_2, \ldots, \ell_n\} \in 2^{\mathbb{R}^2}$ where ℓ_i is of the form $\langle a_i, b_i \rangle$, and the points ℓ_i form the extreme points of a lower hull in the (a,b)-plane.
$A \oplus B$	$\text{LOWERHULL}(A \cup B)$
$A \otimes B$	$\{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A \wedge \mathbf{b} \in B\}$ (Minkowski addition)
$\overline{0}$	$\emptyset$
$\overline{1}$	$\{\langle 0, 0 \rangle\}$

mar et al., 2009; Macherey et al., 2008). Unfortunately, Kumar et al. gave a specialized algorithm whose relationship to more familiar inference algorithms was rather unclear. I describe the algorithm now in terms of more familiar concepts, semirings and the INSIDE algorithm.

In this section, I introduce a novel semiring, which I designate the *upper envelope semiring*, which can be used with the INSIDE algorithm to efficiently compute the exact upper envelope of all derivations in $\mathbf{f}^i \circ \mathcal{G} \circ \mathcal{E}$ for a given Λ and $\mathbf{v}$. Although the output and running time of the INSIDE algorithm with the upper envelope semiring are equivalent to the specialized algorithm described by Kumar et al. (2009), formulating the computation in terms of a semiring is intended to make this material accessible to a broader audience. Furthermore, properties of the MERT algorithm can be proved as properties of the elements of the semiring (which are well-known geometric objects). The elements of the upper envelope semiring are given in Table 3.2. Below, I prove that this system fulfills the semiring axioms as well as the correctness of using this with the INSIDE algorithm to compute the upper envelope.

The upper envelope semiring has values that are sets of the lines forming an upper envelope, but they are represented as points in the dual plane. The semiring addition operation depends on the LOWERHULL algorithm, which removes any points from the set that are not the extreme points of the set's lower hull. There are number of $O(|A|\log|A|)$ algorithms for doing so, for example Graham's Scan and the sweep-line algorithm (de Berg et al., 2000; Macherey et al., 2008). Figure 3.6 shows an example of removing a point from the lower hull (in the dual plane), which corresponds to deleting a line that is not part of the upper envelope in the primal plane. The multiplication operation is the Minkowski addition, a pairwise addition of sets of points. While this operation may be naïvely implemented with an $O(|A|\cdot|B|)$ algorithm, because A and B are convex sets (because they form a lower hull), a specialized algorithm applies that runs in time $O(|A|+|B|)$ (de Berg et al., 2000).

Theorem 2. *The system shown in Table 3.2 (upper envelope semiring) fulfills the semiring axioms and is both commutative and idempotent.*

Proof. The values in $\mathbb{K}$ are sets of lines with equations $y = a_i x + b_i$ and represented by sets of points $\langle a_i, -b_i\rangle$ in the dual plane. $\mathbb{K}$ is closed under $\oplus$ since, while the union of two sets of points may generate a set that does not form a lower hull, those extraneous points are removed by LOWERHULL. $w \oplus \overline{0} = w$ since the union of a set A with the empty set is A, and the LOWERHULL operation is guaranteed not to have an effect since, by stipulation, any value in $\mathbb{K}$ is already a lower hull. Addition is commutative, associative, and idempotent. If the LOWERHULL operation were not applied, this would follow trivially from the properties of union. Even with this supplemental filter, the commutativity of addition is holds since LOWERHULL$(A \cup B)$ = LOWERHULL$(B \cup A)$ for all A and B.

Furthermore, since $A \cup A = A$ for all A, addition is idempotent.

What about associativity? The lines between adjacent points in the lower hull (with vertical lines emitting from the left- and right-most points) can be understood as a way of dividing the plane: the part above the lower hull (the inside region) and the part below (the outside region). During filter and union, the extent of the inside region is growing left, right, and downward. If, during union, a point is added that falls in the outside region, it must necessarily become part of the union set's lower hull or lower hull's inside region, which will have a greater extent than (and encompass) the inside region of operand sets. Thus, any order of addition operations must yield the same division of the plan into inside and outside regions, meaning associativity holds.

Now consider $\otimes$, which is defined to be the Minkowski sum of the two operands. The semiring is closed under multiplication since Minkowski addition of two lower envelopes produces another set of points which is itself lower envelope; additionally, Minkowski addition is commutative and associative (de Berg et al., 2000). $\{\langle 0,0\rangle\}$ is the multiplicative identity since adding $\langle 0,0\rangle$ to all points in a set A will not change them. Distributivity holds because multiplication is commutative. $\square$

Although the semiring axioms hold so long as the values used are sets of points forming a lower hull, in MERT, each point a set has a particular semantics: for a point $\langle a,b\rangle$, the value $a - bx$ corresponds to the score (under a linear model) of a derivation as the weight vector is varied (as a function of x) according to Equation (3.1). I now prove that using the upper envelope semiring with the INSIDE algorithm together with a particular edge weighting of a hypergraph $\mathcal{F} = \mathbf{f}^i \circ \mathcal{G} \circ \mathcal{E}$ computes the dual of the desired MERT upper envelope. This upper envelope is suitable for transformation into an error surface as described above.

Theorem 3. *Using the upper envelope semiring with the* INSIDE *algorithm over a hypergraph $\mathcal{F}$ with each edge e having weight $w(e) = \{\langle \mathbf{v} \cdot \mathbf{h}(e), -\Lambda \cdot \mathbf{h}(e) \rangle\}$ computes the upper envelope of the entire set of derivations in $\mathcal{F}$ where derivation $\mathbf{d}$ corresponds to the line $f_{\mathbf{d}}(x) = (\Lambda + x \cdot \mathbf{v}) \cdot \vec{H}(\mathbf{d})$, assuming global feature functions H_k decompose additively in terms of local features functions h_k.*

Proof. Consider the case where $\mathcal{F}$ has a single derivation $\mathbf{d}$. The appropriate upper envelope is trivially a set containing a single line with equation $f_{\mathbf{d}}(x) = (\Lambda + x \cdot \mathbf{v}) \cdot \vec{H}(\mathbf{d})$. By the point-line duality, this is equivalent to a set containing the single point $\langle \mathbf{v} \cdot \vec{H}(\mathbf{d}), -\Lambda \cdot \vec{H}(\mathbf{d}) \rangle$. This set can be rewritten as follows:

$$\begin{aligned}
\left\{ \langle \mathbf{v} \cdot \vec{H}(\mathbf{d}), -\Lambda \cdot \vec{H}(\mathbf{d}) \rangle \right\} &= \left\{ \langle \sum_{e \in \mathbf{d}} \mathbf{v} \cdot \mathbf{h}(e), \sum_{e \in \mathbf{d}} -\Lambda \cdot \mathbf{h}(e) \rangle \right\} \\
&= \left\{ \sum_{e \in \mathbf{d}} \langle \mathbf{v} \cdot \mathbf{h}(e), -\Lambda \cdot \mathbf{h}(e) \rangle \right\} \\
&= \bigotimes_{e \in \mathbf{d}} w(e)
\end{aligned}$$

The last step is in the previous derivation is justified by the definition of $\otimes$ in the upper envelope semiring. Now, what if a hypergraph $\mathcal{F}$ contains multiple derivations $\{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n\}$? The appropriate upper envelope of a set of derivations is the upper envelope of the *union* of all characteristic lines for every derivation in $\mathcal{F}$. In the dual form, this is the set of points $\mathcal{L}$ that form the lower hull of the points corresponding to every derivation in $\mathcal{F}$.

$$\begin{aligned}
\mathcal{L} &= \textsc{LowerEnvelope}\left(\bigcup_{\mathbf{d}\in\mathcal{F}} \left\{\langle \mathbf{v}\cdot\vec{H}(\mathbf{d}), -\Lambda\cdot\vec{H}(\mathbf{d})\rangle\right\}\right) \\
&= \textsc{LowerEnvelope}\left(\bigcup_{\mathbf{d}\in\mathcal{F}} \bigotimes_{e\in\mathbf{d}} w(e)\right) \\
&= \bigoplus_{\mathbf{d}\in\mathcal{F}} \bigotimes_{e\in\mathbf{d}} w(e) \qquad (3.3)
\end{aligned}$$

This last step in this derivation follows from a fact that was remarked on in the previous theorem: namely, the lower hull of a set of points S is the same whether it it is computed as $\textsc{LowerHull}(S)$ or if S is divided into subsets A and B such that $S = A \cup B$, the lower hull is computed as $\textsc{LowerHull}(\textsc{LowerHull}(A) \cup \textsc{LowerHull}(B))$. Equation (3.3) is equal to Equation (2.6), the value computed by the INSIDE algorithm with *any* semiring (§2.4.2), therefore INSIDE may be used to compute this value with the upper envelope semiring. □

Other properties. A remarkable thing about the upper envelope semiring is that the growth in the cardinality of the set under multiplication (Minkowski addition) is more tightly bounded than it might otherwise appear to be. Since both A and B consist of points in the dual plane that form a lower hull (and are thus a convex set), a theorem from computational geometry applies that says $|A \otimes B| < |A| + |B|$ and can be computed in linear time (de Berg et al., 2000).[20]

The number of lines in MERT upper envelope of any acyclic WFST (such as one encoding $\mathbf{f}^i \circ \mathcal{G} \circ \mathcal{E}$) with edges E and states Q is bounded by $|E| - |Q| + 2$, which follows

[20]Thanks are due to Dave Mount who pointed out the relevant theorems from computational geometry.

from a result from Macherey et al. (2008). Kumar et al. (2009) point out that these results also hold for acyclic hypergraphs, and therefore, for non-recursive WSCFGs. These results are extremely important since it means the number of segments in the upper envelope is linear in the size of the encoding of the input (that is, the number of nodes and edges), even when the input represents an exponential number of derivations. Since the INSIDE algorithm runs in $O(|E|+|Q|)$, this also guarantees that the time required to compute the total upper envelope using the INSIDE algorithm will be polynomial in $|E|+|Q|$, *not* in the number of derivations, of which there may be $O(|E|^{|Q|})$.

3.1.4.3 Minimum error training summary

Although the upper envelope semiring only permits the restatement of an algorithm that has already been described in the literature, this restatement has considerable practical value. First, the presentation here makes use of semirings and the INSIDE algorithm which are familiar to a broad audience in natural language processing research. This will hopefully improve understanding of this widely used (but poorly understood) optimization algorithm. Second, there are a number of toolkits that support generic semiring computations over a variety of finite-state and context-free structures, including OpenFST (Allauzen et al., 2007), the Joshua toolkit (Li et al., 2009a), and the `cdec` decoder (Dyer et al., 2010). The upper envelope semiring can easily be implemented in any of them (and has already been implemented in `cdec`). Finally, this generic presentation ensures that novel grammar formalisms that are parameterized using semirings can make use of this algorithm.

All translation experiments in this thesis make use of MERT to tune their parameter weights on a held-out development set using the upper envelope semiring. For the experiments in Chapters 4 and 5, I use a generic semiring weight framework and the INSIDE algorithm to compute the upper envelopes from which error surfaces are generated (Dyer et al., 2010). For the experiments reported in Chapter 4 and Chapter 5, the upper envelope semiring implementation of MERT is used. For the experiments in this chapter, a k-best approximation was used (Och, 2003).

3.1.5 Translation evaluation

Automatic evaluations of the quality of machine translation output is challenging and an area of active and ongoing research. Matters are further complicated by the fact that human annotators give rather unreliable judgments on many evaluation tasks when system differences are slight (Callison-Burch et al., 2009). I will avoid participating in the lively debate on the usefulness or quality of various translation metrics and instead rely on established metrics. I primarily utilize the BLEU-4 metric (Papineni et al., 2002), which is the geometric mean of n-gram precisions (where $n \leq 4$) in a translation output counterbalanced by a *brevity penalty*, which penalizes outputs that attempt to 'game' the precision-oriented nature of the metric by being overly short. The BLEU metric ranges between 0 and 1, and higher scores are better. The absolute range depends on the number of reference translations used in scoring (multiple references permit n-gram matches from any reference). Relative differences are also affected by the number of references. As a rule, with 4 references (common with NIST evaluation sets) an increase of 1 BLEU is considered noteworthy, and with a single reference (common in the Workshop on Statistical Machine Translation evaluations), smaller increases in the range of 0.5 are of interest.

3.2 Translation of WFST-structured input

Now that I have established the basics of statistical machine translation using phrase-based and hierarchical phrase-based models, discussed their training using MERT, and briefly touched on the evaluation of their output, I turn to an exploration of the translation of ambiguous input, when the various input alternatives are encoded as a WFST. Since the translation process was formalized as a WFST composition cascade of weighted transducers, that is, $(F \circ \mathcal{G} \circ \mathcal{E})\downarrow$, the process remains well defined the unambiguous inputs encoded in F are replaced with ambiguous ones. However, little has been said about the *source* of the ambiguity: where does it come from and what alternatives does it entail? What are the properties of a WFST that represents these alternatives? These questions are considered now followed by a discussion of decoding algorithms for phrase-based translation models.

3.2.1 Sources of input finite-state ambiguity

One obvious source of 'input ambiguity' is when the output of a speech recognition system is used as the input to a machine translation system. For example, the task of spoken language translation is to recognize speech (using an ASR system) in the source language and then translate it into some target language. While it is common to use only the single best guess from the output of the recognizer, ASR systems typically define a distribution over possible recognition strings in the source language, which can be encoded in F and translated, producing a weighted set of translation outputs consisting of strings that may derive from many different transcription hypotheses. Furthermore, since ASR systems are typically based on finite-state models, their output distribution is usually already structured as a WFST (Jelinek, 1998). Using WFSTs representing the hypothesis space of a recognizer as input to a translation system has been explored in considerable detail in previous work, especially when the translation model ($\mathcal{G}$) is also structured as a WFST (Bertoldi et al., 2007; Mathias and Byrne, 2006). However, I consider here the performance of translation models with a *context-free* structure on WFST input that encodes recognition ambiguity.

Although spoken language translation is a natural application for the ability to translate weighted sets of source language strings, another source of ambiguity can be seen when one conceives of the development of a translation system as consisting of a series of decisions which could have several different outcomes. Such 'development time' decisions include committing to a particular approach to word segmentation or the amount of morphological simplification (such as stemming) to use as preprocessing. Alternative outcomes for each of these decisions may give rise to a different string of input words which can be encoded easily and compactly in a finite-state object. Input sets that represent preprocessing alternatives will be used as the inputs in experiments below.

3.2.2 Properties of finite-state inputs

When a WFST is used to encode a weighted set of inputs, observe that the inputs to a translation system are only meaningful when they have a finite length. It is therefore necessary (as well as useful) to require that the WFSTs representing the input be acyclic, therefore defining a star-free language.

Word lattices are a restricted subset of WFSTs that have no cycles as well as unique start and end states.[21] It is further useful to distinguish among three kinds of word lattices (examples of each class are shown in Figure 3.9): (a) word lattices that unambiguously generate a single sentence; (b) a confusion networks (CNs), which preserve the linear structure of a sentence, but which may have multiple edges between adjacent nodes, thereby encoding a multitude of paths.[22] CNs are typically constructed from unrestricted word lattices lattices by 'pinching' different branches of the lattice together and merging the weights such that the CN arc represents the posterior probability (under, for example, a speech recognition system) of the word (Mangu et al., 2000). Unless specified otherwise, I will use the term *confusion network* to refer to this structure without implying any particular weight semantics. The last class (c) are arbitrary word lattices, WFSTs restricted only to have a unique initial and final state and no cycles. In contrast to fully general word lattices, which can represent any set of strings, CNs may *over*generate the strings that they are intended to model (this is necessary because CNs must adhere to a particular structure). The number of strings encoded by a word lattice (or CN) is expo-

[21]This restriction makes WFSTs share more properties with a sentence: namely a unique source and end point, making it easier to adapt algorithms translation algorithms that were originally designed to manipulate sentences rather than WFSTs.

[22]Although every path passes through every node in a CN, the strings encoded may be of different lengths on account of ε-transitions.

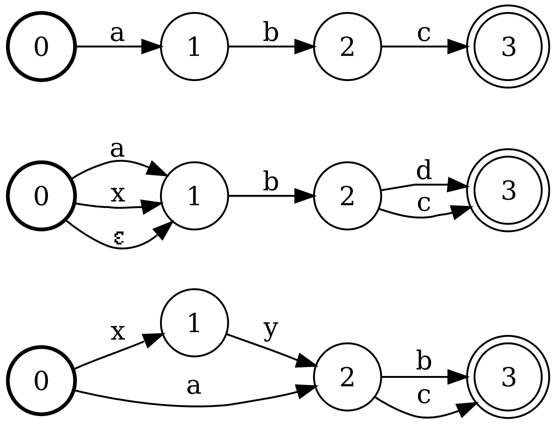


Figure 3.9: Three examples of word lattices: (a) sentence, (b) confusion network, and (c) non-linear word lattice.

nential in the number of nodes in the lattices, with the base of the exponent related to the edge density.

A word lattice is a useful representation of ambiguity because it permits any finite set of strings to be represented, and allows for substrings common to multiple members of the set to be represented with a single piece of structure.[23] Additionally, all paths spanning a pair of nodes form an equivalence class and contain approximately equivalent content. Figure 3.10 illustrates some possible alternations, giving examples of the kinds of alternations that can be expressed in lattices. In the upper lattice, the lattice encodes alternative lexical choices for (approximately) the same underlying meaning, with the span [3,5] forming an equivalence class expressing the meaning *celebrities*. The middle

[23]It should be pointed out that not every common substring found in a subset of the language generated by $\mathcal{G}$ can be encoded using shared structure. For example, although each sentence in the language $\{xab, yab, zab, abw\}$ contains the substring *ab*, an FSA representation must still contain at least two distinct paths that generate *ab*: at least one that leads directly to the final state, and a different one that goes on to generate *w*. If a more sophisticated formalism is used, such as a WCFG, a single edge could represent the substring *ab*; however, this structure would no longer be finite-state; it has the generative capacity of a context-free grammar. I consider such representations of ambiguity in Chapter 5.

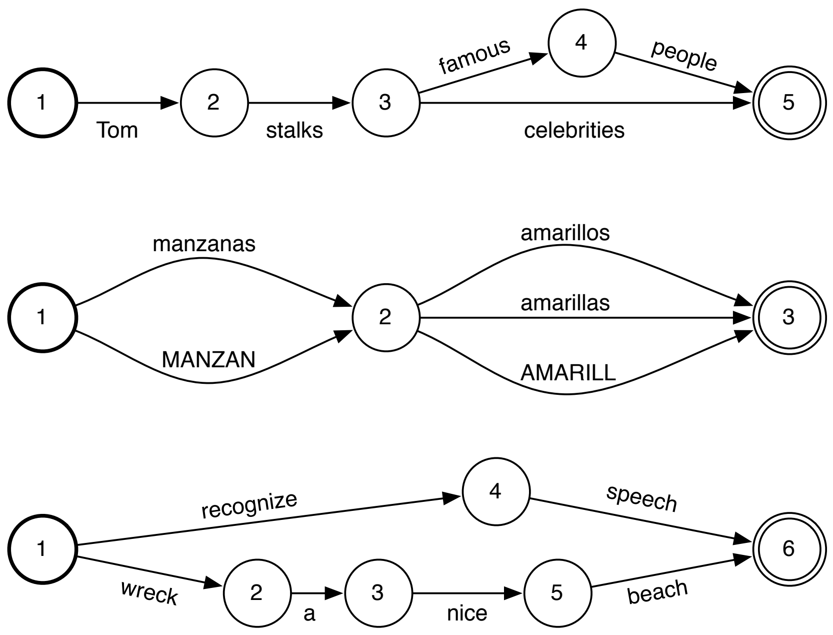


Figure 3.10: Three example lattices encoding different kinds of input variants.

lattice shows morphological variants of Spanish word forms (this lattice has the form of a confusion network), and the lower example represents the kind of confusion that may be found in automatic speech recognition, where spans in the lattice correspond (approximately) to spans of real time. Pairs of nodes not linked by any path do not generally have a discernible relationship.

Encoding a word lattice in a chart. To simplify the adaptation of the phrase-based translation algorithm discussed above (§3.1.2.1), it will be useful to encode a word lattice $\mathcal{G}$ in a chart based on a topological ordering of the nodes, as described by (Cheppalier et al., 1999). The starting node should have index 0 and the ending node will have index $|Q|-1$, where $|Q|$ is the number of states in the lattice. The nodes in the lattices shown in Figure 3.9 are labeled according to an appropriate numbering.[24]

The chart representation of the graph is a triple of 2-dimensional matrices $\langle \mathbf{F}, \mathbf{p}, \mathbf{R} \rangle$,

[24]A lattice may have several possible topological orderings, any of which may be chosen.

which can be constructed from the numbered graph. $\mathbf{F}_{i,j}$ is the word label of the j^{th} transition leaving node i. The corresponding transition cost is $\mathbf{p}_{i,j}$. $\mathbf{R}_{i,j}$ is the node number of the destination of the j^{th} transition leaving node i. Note that because lattices are acyclic, $\mathbf{R}_{i,j} > i$ for all i, j. Table 3.3 shows the word lattices from Figure 3.9 represented in matrix form as $\langle \mathbf{F}, \mathbf{p}, \mathbf{R} \rangle$, weights assume the probability semiring and a uniform distribution over paths.

0			1			2		
$\mathbf{F}_{0j}$	$\mathbf{p}_{0j}$	$\mathbf{R}_{0j}$	$\mathbf{F}_{1j}$	$\mathbf{p}_{1j}$	$\mathbf{R}_{1j}$	$\mathbf{F}_{2j}$	$\mathbf{p}_{2j}$	$\mathbf{R}_{2j}$
a	1	1	b	1	2	c	1	3
a	$\frac{1}{3}$	1	b	1	2	c	$\frac{1}{2}$	3
x	$\frac{1}{3}$	1				d	$\frac{1}{2}$	3
ε	$\frac{1}{3}$	1						
x	$\frac{1}{2}$	1	y	1	2	b	$\frac{1}{2}$	3
a	$\frac{1}{2}$	**2**				c	$\frac{1}{2}$	3

Table 3.3: Topologically ordered chart encoding of the three lattices in Figure 3.9. Each cell ij in this table is a triple $\langle \mathbf{F}_{ij}, \mathbf{p}_{ij}, \mathbf{R}_{ij} \rangle$

3.2.3 Word lattice phrase-based translation

The chart encoding of word lattices introduced in the previous section is closely related to the way sentences are represented in a standard phrase-based decoder. I describe how the decoding algorithm (which can be understood as a specialized finite-state composition algorithm) can be adapted to translate word lattices that are encoded in a chart. Previous work has adapted the phrase-based translation algorithm for confusion network decoding (Bertoldi et al., 2007); however, I adapt the algorithm so as to translate general word lattices.[25]

As described above in the introduction to phrase-based translation (§3.1.2.1), the standard sentence-input decoder builds a translation hypothesis from left to right by selecting a span consisting of untranslated (source language) words and adding translations of this phrase to the end of the hypothesis being extended. Phrase-based translation models translate a foreign sentence $\mathbf{f}$ into the target language $\mathbf{e}$ by breaking up $\mathbf{f}$ into a sequence of phrases $\overline{f}_{1\dots i}$, where each phrase $\overline{f}_i$ can contain one or more contiguous words and is translated into a target phrase $\bar{e}_i$ of one or more contiguous words. Each word in $\mathbf{f}$ must be translated exactly once.

To generalize this algorithm to accept a word lattice F as input, it is necessary to choose both a valid path through the lattice and a partitioning of the sentence this induces into a sequence of phrases $\overline{f}_{1\dots i}$. Although the number of source phrases in a word lattice can be exponential in the number of nodes, enumerating the *possible translations* of every span in a lattice is, in practice, tractable, as described by Bertoldi et al. (2007).

The word lattice decoder keeps track not of the words that have been covered, but of the *nodes*, given a topological ordering of the nodes. For example, assuming the lattice in Figure 3.9(c) is the decoder input, if the edge with word *a* is translated, this will cover *two* untranslated nodes [0,1] in the coverage vector, even though it is only a single word. As with sentence-based decoding, a translation hypothesis is complete when all nodes in the input lattice are covered. Since the decoder supports non-monotonic decoding (where source phrases are translated in an order that is not strictly left to right), the score for

[25]Decoders implemented using generalized WFST composition operations can deal with word lattices without modification (Mathias and Byrne, 2006).

each hypothesis in the stack includes an estimate of the cost of translating the remaining untranslated words. Without this, there would be a bias to translate high-probability words first. For word lattices, the future cost estimate proposed by Koehn et al. (2003) is generalized to be the best possible translation cost through *any* path of the remaining untranslated nodes. When a sentence is encoded in a lattice, my generalization of the future cost is equivalent to the Koehn definition.

Non-monotonicity and unreachable nodes. The changes to the decoding algorithm described thus far are straightforward adaptations of the sentence decoder; however, non-monotonic decoding of word lattices introduces some minor complexity that I discuss now. In a standard sentence decoder, any translation of any span of untranslated words is an allowable extension of a partial translation hypothesis, provided that the coverage vectors of the extension and the partial hypothesis do not overlap. However, when decoding lattice input, a further constraint must ensure that there is always a path from the starting node of the translation extension's source to the node representing the nearest right edge of the already-translated material, as well as a path from the ending node of the translation extension to any future translated spans (if they exist). Figure 3.11 illustrates the constraint. If the edge labeled `a` is translated, the decoder must not consider translating `x` as a possible extension of this hypothesis, since there is no path from node 1 to node 2.

3.2.4 Word lattice translation with WSCFGs

Because phrase-based decoders rely on specialized composition algorithms to apply their translation models, it was necessary to describe the adaptation of these algorithms to deal

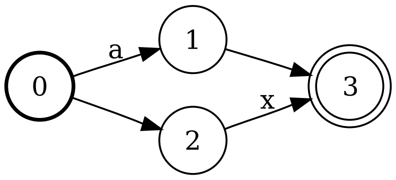


Figure 3.11: The span $[0,3]$ has one inconsistent covering: $[0,1]+[2,3]$.

with word lattice input. However, hierarchical phrase-based translation systems (based on WSCFGs) can make use of the general composition algorithms (either the top-down or bottom-up variants; §2.3) described in the previous chapter for the $F \circ \mathcal{G}$ portion of the translation process. Furthermore, the techniques used to incorporate a target language model (i.e., $\circ\mathcal{E}$) do not need to be adapted when F is changed from a single sentence to a word lattice. It is therefore not necessary to discuss the adaptation of a WSCFG decoder to deal with word lattice input as I did with phrase-based decoders.

3.3 Experiments with finite-state representations of uncertainty

I now describe three experiments looking at applications of source language lattices: spoken language translation, where the lattice encodes the speech recognizer's transcription hypothesis space (§3.3.1), morphological variant lattices (§3.3.2), where morphological variants of the source words are encoded in a lattice, and segmentation lattices (§3.3.3), where morphemes are segmented at different granularities.

3.3.1 Spoken language translation

Although word lattices and confusion networks have been used to improve the performance of statistical systems in spoken language translation tasks, their utility has only been verified using phrase based models. I thus compare a hierarchical (WSCFG) system, Hiero, and an WFST system, Moses, modified to accept word lattices, as described above. As an introduction, I give a motivating example for why WSCFGs are useful in particular when translating the ambiguous output of a speech recognition system.

Because Arabic is primarily VSO, a large class of important collocations (a verb and a characteristic object or preposition) are separated by an intervening subject. This poses a challenge for finite-state models of Arabic-English translation, since any common verb-object or verb-preposition pair, if treated as a unit, must be learned in phrases with many distinct subjects. As an example, verb *sa:fara* (*traveled*) characteristically selects preposition *ʔila* (*to*) to express a destination, e.g.

$$\langle \textit{sa:fara}\ \text{bu:ʃ}\ \textit{ʔila}\ \text{london},\ \text{Bush}\ \textit{traveled to}\ \text{London}\rangle$$

This useful generalization cannot be expressed as a non-hierarchical phrase pair, but it is expressed naturally as a pair of synchronous context-free rules:

$$\text{X} \rightarrow \langle \textit{sa:fara}\ \text{X}\ \textit{ʔila}\ \text{X}, \boxed{1}\ \textit{traveled to}\ \boxed{2}\rangle$$

Rules of this sort have been shown to improve translation quality for text input, and I argue that they can be equally advantageous when coping with input ambiguity

1		2		3		4		5		6		7	
saːfara	**0.8**	**al**	**0.9**	**raʔiːs**	**0.9**	li	0.5	**ʔamiriːkij**	**0.9**	ʕala	0.3	**baɣdaːd**	**1.0**
safiːra	0.2	ε	0.1	ε	0.1	**al**	**0.4**	ʔamiriːkaː	0.1	laː	0.3		
						ε	0.1			**ʔila**	**0.2**		
										fiː	0.1		
										ε	0.1		

Figure 3.12: Example confusion network. Each column has a distribution over possible words that may appear at that position.

from speech. Consider the confusion network shown in Table 3.12. This example shows some ambiguities typical of an ASR system for the Arabic sentence *saːfara al-raʔiːs al-ʔamiriːkij ʔila baɣdaːd* (in English, *the American president traveled to Baghdad*). In this artificial example, the 1-best transcription hypothesis (top row) contains two mistakes. The first mistake is at position 4, where correct word, the definite article *al*, has lower probability than the preposition *li*. A conventional phrase-based system using confusion network input can handle this case: one would expect any system trained on recent news to contain the following correspondence with a high probability:

$$\langle \textit{al-raʔiːs al-ʔamiriːkij},\ \textit{the American president} \rangle$$

Phrase-based CN decoding effectively intersects the Arabic side of this phrase pair with span [2,5] in the confusion network, which yields the phrase *al-raʔiːs al-ʔamiriːkij*. This allows a relatively higher probability in the translation model to counterbalance the higher ASR posterior probability for *al-raʔiːs li-ʔamiriːkij*, making it possible for the translation to favor the ASR system's less likely path.

However, consider the second error in the best ASR hypothesis, at position 6. Here ASR misidentifies the preposition *ʔila* (*to*) as *ʕala* (*on*). A phrase-based system must

	BNAT05	IWSLT-read	IWSLT-spont.
avg. length	31	14	14
avg. depth	1.7	5.4	6.8
max. depth	20	95	83
avg. # derivations	10^{25}	10^{25}	10^{33}

Table 3.4: Confusion network statistics for test sets.

compare of *ʔila baɣdaːd* versus *ʕala baɣdaːd*, but it will find little basis for distinguishing them because their conditional probabilities are roughly equal.[26] In contrast, a *WSCFG* model allows the ambiguity to be resolved using a more robust comparison, ignoring the irrelevant intervening material between verb *saːfara* and the preposition. Since the verb generally co-occurs with *ʔila*, the grammar will have high probability for rule (3.3.1), making it possible to favor a translation containing *ʔila* even though the ASR system has given it lower probability.

Chinese-English travel domain. I now examine some experimental results. Word lattices of the Chinese-English ASR data were provided in the IWSLT 2006 distribution. ASR word lattices (in this section and the next) were converted to confusion networks using the SRI Language Modeling Toolkit (Stolcke, 2002). The word error rate (WER) reported for confusion networks is the *oracle* WER, also computed using the SRI tools. Chinese-English models were trained using a 40K-sentence subset of the BTEC corpus (Takezawa et al., 2002); this corresponds to the training data provided for the IWSLT 2006 Chinese-English translation task. For the Chinese-English experiments, the language model was trained using (only) the English side of this training bitext.[27]

[26] Verified in a corpus of newswire text.
[27] The Chinese side of the corpus came pre-segmented; I used a standard tokenizer on the English side.

Table 3.5: Chinese-English results for IWSLT-2006. Confusion net WER numbers are oracles.

Input	WER	Hiero	Moses
verbatim	0.0	19.63	18.40
read, 1-best (CN)	24.9	16.37	15.69
read, full CN	16.8	16.51	15.59
spontaneous, 1-best (CN)	32.5	14.96	13.57
spontaneous, full CN	23.1	15.61	14.26

The systems were tuned on DEV1 (506 sentences, 16 reference translations), a text-only development set from IWSLT 2006. For the confusion network translation model, the feature weight for the confusion network log posterior probability was set such that $\lambda_{CN} = \lambda_{LM}$, since no ASR output was available for any set but DEV4, making it impossible to tune λ_{CN} automatically.

Table 3.5 shows the results of the hierarchical (Hiero) and non-hierarchical (Moses) systems translating the DEV4 set from the IWSLT 2006 data (489 sentences, 7 reference translations), along with an upper bound (translation of verbatim transcription).[28] Both models show an improvement for decoding the full confusion network when compared to a one-best baseline and confirm previous results that have shown that using confusion networks with the more degraded input associated with recognizing spontaneous speech yields larger gains in translation quality than situations where the WER is low to begin with (Bertoldi et al., 2007). The hierarchical system outperforms the non-hierarchical system in every category, including those where confusion networks are used as input.

[28]The IWSLT data include both read and spontaneous speech. Confusion network complexity statistics for the IWSLT test sets are shown in Table 3.4.

Table 3.6: Arabic-English training data. Sizes are in millions (M) of words.

LDC catalog	Corpus	Size
LDC2004T17	Arabic News Translation Text	4.4M
LDC2005E46	Arabic Treebank English Translation	1.2M
LDC2004T18	Arabic English Parallel News	1.0M
LDC2004E72	eTIRR Arabic English News Text	0.1M

Arabic-English news domain. Arabic-English models were trained on the sources shown in Table 3.16. The Arabic-English lattices were generated by a state-of-the-art 617k word vocabulary Arabic ASR system trained on 136 hours of transcribed speech and 1,800 hours of unlabeled data (Soltau et al., 2007). The language model for Arabic-English experiments was trained on the English side of the training bitext, combined with the LDC English Gigaword v2 AFP and Gigaword v1 Xinhua corpora. Training text and confusion networks were preprocessed to separate clitics and attached particles from stems using tools based on the Buckwalter Morphological Analyzer.

The Arabic-English experiment tested a situation where the baseline WER was far lower and much larger amounts of training data were used.[29] The non-overlapping development (477 sentences) and evaluation (468 sentences) sets consist of automatically recognized broadcast news and conversations in Modern Standard Arabic from several Arabic language satellite channels that were translated into English. Only one reference translation was available.

Because ASR development data were available, the models were tuned according to the input they were to be evaluated on. That is, if non-ambiguous input was being

[29]The lattices used were generated from speech data that was part of the ASR system's training data, so the errors are much lower than is typical of the system. The reported WER was calculated after postprocessing to separate clitics and particles.

Table 3.7: Arabic-English results (BLEU) for BNAT05-test

Input	WER	Hiero	Moses
verbatim	(0.0)	26.46	25.13
1-best	12.2	23.64	22.64
full CN	7.5	24.58	22.61

evaluated, text was used to tune the model (since no λ_{CN} is necessary); if ambiguous input was being evaluated, confusion networks were used for tuning. Input confusion network complexity statistics are shown in Table 3.4 (BNAT05). Table 3.7 summarizes the results of the Arabic-English experiment.

As expected because of the lower WER, the margin for improvement between the 1-best confusion network hypothesis and the verbatim transcription is rather small. However, the hierarchical model still shows considerable improvement when decoding confusion networks. Interestingly, the non-hierarchical model shows no improvement at all when the full confusion network is incorporated. The hierarchical model outperforms the non-hierarchical baseline in all categories, the pattern typically seen when translating unambiguous input (Chiang, 2007; Zollmann et al., 2008).

Efficiency results. Timing experiments for the WSCFG-based decoder were conducted on a AMD Opteron 64-bit server with 1.0GB RAM operating at 1.0GHz. For the IWSLT test set, decoding a text sentence with the hierarchical model took an average of 3.0 seconds. Decoding a confusion network took 12.8 seconds on average, a factor of 4.3 times slower. This compares to a slowdown factor of 3.8 with the non-hierarchical phrase-based model.

Summary. Similar gains are possible when using confusion network input to represent alternative source language transcriptions in both hierarchical phrase-based (WSCFG-based) translation system. Additionally, results of Bertoldi et al. (2007) for phrase-based translation systems were reconfirmed. The impact of using confusion networks rather than 1-best inputs on decoding speed is modest, especially considering the effective size of the input space that is being searched.

3.3.2 Morphological variation

In the previous section, it was showed that by using confusion networks to encode the hypothesis space of an ASR system, it was possible to efficiently search for the best translation of *any* of the transcription hypotheses represented in that space. Translation systems that model translation using WFSTs as well as WSCFGs were both able to take advantage of input word lattices, an efficient finite-state representation of ambiguity, and improve translation performance over a baseline in which ambiguity was not preserved. In the next sections leveraging this technique of propagating uncertainty are extended to the kinds of ambiguity that are found even in text-only translation systems.

The first of these sources of ambiguity concerns morphological analysis and decisions made based on morphological analysis. Conventional statistical translation models are constructed with no consideration of the relationships between lexical items and instead treat different inflected (observed) forms of identical underlying lemmas as completely independent of one another. While the variously inflected forms of one lemma may express differences in meaning that are crucial to correct translation, the strict independence assumptions normally made exacerbate data sparseness and lead to poorly estimated models and suboptimal translations.

A variety of solutions have been proposed: Niessen and Ney (2001) use morphological information to improve word reordering before training and after decoding. Goldwater and McClosky (2005) show improvements in a Czech to English word-based translation system when inflectional endings are simplified or removed entirely. Their method can, however, actually harm performance, since the discarded morphemes carry some information that may have a bearing on the translation. Talbot and Osborne (2006) use a data-driven approach to attempt to cluster source-language morphological variants that are meaningless in the target language, and Yang and Kirchhoff (2006) propose the use of a backoff model that uses morphologically reduced forms only when the translation of the surface form is unavailable.

All of these approaches have in common that the decisions about whether to use morphological information are made in either a pre- or post-processing step. I extend the concept of translating from an ambiguous set of source hypotheses to the problem of determining how much morphological information to include by defining the input to the translation system, F, to be a weighted set sentences derived by applying *morphological transformations* (such as stemming, compound splitting, clitic splitting, etc.) to a source sentence $\mathbf{f}$. Whereas in the context of an ASR transcription hypothesis, each path in F was weighted by the log posterior probability of that transcription hypothesis, I redefine the path weight to be a *backoff penalty* in the morphology model. This can be intuitively thought of as a measure of the "distance" that a given morphological alternative is from the observed input sentence. Just as it did in translating ASR output, my approach allows decisions about what variant form to use to be made during decoding, rather than in advance, as has been done in prior work.

3.3.2.1 Czech morphological simplification

The first experiment where decisions about what morphological transformations to apply to the input into a translation system are deferred from development time to decoding time looks at strategies for improving Czech-English translation by reducing the complexity of Czech inflectional morphology.[30] I describe a series of experiments using different strategies for incorporating morphological information during preprocessing of the News Commentary Czech-English data set provided for the WMT07 Shared Task. Czech was selected because it exhibits a rich inflectional morphology, but its other morphological processes (such as compounding and cliticization) that affect multiple lemmas are relatively limited. The relative morphological complexity of Czech, as well as the potential benefits that can be realized by stemming, can be inferred from the corpus statistics given in Table 3.8, which show that the surface form of Czech has a very large number of types, relative to other European languages. Since word types are the minimal unit of phrase-based translation, this suggests that data sparsity could become a problem.

Data preparation and translation model. The Czech morphological analyzer by Hajič and Hladká (1998) was used to extract the lemmas from the Czech portions of the training, development, and test data (the Czech-English portion of the News Commentary corpus distributed as as part of the WMT07 Shared Task).[31] Data sets consisting of truncated

[30]The Czech-English experiments were originally published in Dyer (2007).

[31]http://www.statmt.org/wmt07/

Table 3.8: Corpus statistics, by language, for the WMT07 training subset of the News Commentary corpus.

Language	Tokens	Types	Singletons
Czech surface	1.2M	88,037	42,341
Czech lemmas	1.2M	34,227	13,129
Czech truncated	1.2M	37,263	13,093
English	1.4M	31,221	10,508
Spanish	1.4M	47,852	20,740
French	1.2M	38,241	15,264
German	1.4M	75,885	39,222

forms were also generated; using a length limit of 6, which Goldwater and McClosky (2005) experimentally determined to be optimal for translation performance. I refer to the three data sets and the models derived from them as SURFACE, LEMMA, and TRUNC. Table 3.9 illustrates the differences in the forms. Czech-English grammars were extracted from the three training sets using the methods described in Chiang (2007). Two additional grammars were created by combining the rules from the SURFACE grammar and the LEMMA or TRUNC grammar and renormalizing the conditional probabilities, yielding the combined models SURFACE+LEMMA and SURFACE+TRUNC.

Table 3.9: Examples of different Czech preprocessing strategies.

Model	Type
SURFACE	amerického
TRUNC	americ
LEMMA	americký

Confusion networks for the development and test sets were constructed by providing a single backoff form at each position in the sentence where the lemmatizer or truncation process yielded a different word form. The backoff form was assigned a cost of 1 and

1	2	3	4	5	6	7	8	9	10	11	12
z	amerického	břehu	atlantiku	se	veskerá	taková	odů.	jeví	jako	naprosto	biz.
	americký	břeh	atlantik	s		takový		jevit			

Figure 3.13: Example confusion network generated by lemmatizing the source sentence to generate alternates at each position in the sentence. The upper element in each column is the surface form and the lower element, when present, is the lemma.

the surface form a cost of 0. Numbers and punctuation were not truncated. A "backoff" set, corresponding approximately to the method of Yang and Kirchhoff (2006) was generated by lemmatizing only unknown words. Figure 3.13 shows a sample surface+lemma CN from the test set.

Experimental results. Table 3.10 summarizes the performance of the six Czech-English models on the WMT07 Shared Task development set. The basic SURFACE model tends to outperform both the LEMMA and TRUNC models, although the difference is only marginally significant. This suggests that the Goldwater and McClosky (2005) results are highly dependent on the kind of translation model and quantity of data. The backoff model, a slightly modified version of the method proposed by Yang and Kirchhoff (2006),[32] does substantially better than the baseline. However, the SURFACE+LEMMA model outperforms both surface and backoff baselines. The SURFACE+TRUNC model is an improvement over the SURFACE model, but it performances significantly worse than the SURFACE+LEMMA model.

[32]The backoff model implemented here has two differences from model described by Yang and Kirchhoff (2006). The first is that the ambiguity-preserving model based on composition effectively creates backoff forms for *every* surface string, whereas their model does this only for forms that are not found in the surface string. This means that in the model considered here, the probabilities of a larger number of surface rules have been altered by backoff discounting than would be the case in the more conservative model. Second, the joint model I used has the benefit of using morphologically simpler forms to improve alignment.

Input	BLEU
SURFACE	22.74
LEMMA	22.50
TRUNC (l=6)	22.07
backoff (SURFACE+LEMMA)	23.94
CN (SURFACE+LEMMA)	**25.01**
CN (SURFACE+TRUNC)	23.57

Input	Sample translation
SURFACE	From the **US side** of the Atlantic all such **odůvodnění** appears to be **a** totally bizarre.
LEMMA	From the **side** of the Atlantic **with** any such **justification** seem completely bizarre.
TRUNC	From the **bank** of the Atlantic, all such **justification** appears to be totally bizarre.
backoff	From the **US bank** of the Atlantic, all such **justification** appears to be totally bizarre.
SUR+LEM	From the **US side** of the Atlantic all such **justification** appears to be **a** totally bizarre.
SUR+TR	From the **US** Atlantic any such **justification** appears to be **a** totally bizarre.

Table 3.10: Czech-English results on WMT07 Shared Task DEVTEST set. The sample translations are translations of the sentence shown in Figure 3.13.

Interpretation of results. By allowing the decoder to select among the surface form of a word or phrase and variants of morphological alternatives on the *source* side, the lattice-input system outperforms baselines where hard decisions about what morphological variant to use are made in advance of decoding, as is typically been in systems that make use of morphological information.

The results shown in Table 3.10 also illustrate a further benefit that lattice-based translation can have. Observe that for the LEMMA system, the word **odůvodnění** has been correctly translated. This is true despite the lemmatizer failing to analyze this form properly (see Figure 3.13). However, the word alignment that was used during the induction of the translation grammar was able to correctly align this word when the surrounding words had been lemmatized.

3.3.2.2 Arabic diacritization

The rich morphology of Czech makes it a challenge for translation. In some ways, Arabic faces the opposite problem. Arabic orthography does not capture all the phonemic distinctions that are made in the spoken language since optional diacritics are used to indicate consonant quality as well as the identity of short vowels. When these diacritics are absent (as is the case in virtually all text genres), a single orthographic form will correspond to several phonemically distinct words. While homography is common in most written languages, and machine translation models successfully deal with it by incorporating contextual information in the source and target languages, the incidence and number of Arabic homographs is far greater than in most languages. Furthermore, some distinctions which are lost in the written language, such as whether a past tense verb is in the active or passive voice, seem like they would be useful to be aware of when modeling translational relationships. This insight was the motivation for a study by Diab et al. (2007), who used SVMs to predict missing diacritics as a preprocessing step for MT. Although a variety of diacriticization schemes were attempted, the study failed to find any set of diacritics that consistently improved translation quality on an Arabic-English task. The authors conclude that the fragmentation of training data resulting from the proliferation of distinctive forms resulted in poorly estimated translation models.

In this experiment, a lattice representing alternative diacritizations of the source sentence should enable the translation system to make use of the more detailed morphological information when the exact form was observed during training but to back off to less specific forms when not. The hypothesis is that quality will increase, since when there is sufficient evidence to meaningfully distinguish the translations of two words that are distinguished by a reconstructed diacritic these sharper distributions will be available, but if not, the decoder will be able to back off to a surface model. Unlike in the previous experiment, where alternatives were generated by removing information from a highly inflected surface form, this experiment generates inflected forms (of varying complexity) from a simplified surface form.

Data preparation. In this experiment, 10M words of Arabic-English parallel newswire data was used to train translation models. In the baseline condition, the Arabic side of the parallel corpus with no diacritics (which is standard for Arabic-English machine translation) was aligned. For the experimental conditions, the Arabic side of the training data was analyzed using a suite of SVM classifiers used to predict the missing diacritics, as described by (Habash et al., 2007). Six versions of the corpus were created which correspond to five hypothetical best diacritizations for MT (one with no diacritics, corresponding to the baseline, one with case markings, another with gemination symbols, one with passive voice markers, one with silence markers, and one with full diacritics) as proposed by Diab et al. (2007). Table 3.11 shows the six diacritization schemes used for the Arabic phrase */saturammamu aljidra:nu/* (pronounced [*saturammam uljidra:nu*]), meaning *the walls will be restored*. The fully specified form (with all diacritics) is written in the Buckwalter Romanization of Arabic as *saturam∼amu AljidorAnu*; but, in text, this will generally appear as *strmm AljdrAn*.

To build the translation grammar, the six diacritized versions of the corpus were concatenated and a hierarchical phrase-based translation grammar was extracted from the

Table 3.11: Six diacritizations of the Arabic phrase *strmm AljdrAn* (adapted from Diab et al. (2007)).

	Romanization
NONE	strmm AljdrAn
PASS	sturmm AljdrAn
CASE	strmmu AljdrAnu
GEM	strm∼m AljdrAn
SUK	strmm AljdorAn
FULL	saturam∼amu AljidorAnu

Table 3.12: Results of Arabic diacritization experiments.

Condition	MT03	MT05	MT06
BASELINE	45.0	42.2	44.2
LATTICE	**45.9**	**43.1**	**45.1**

union. A lattice was constructed by adding arcs between each node that correspond to each of the five diacritization schemes used to train the model. Thus, the decoder can use any of the diacritization schemes to translate any span of the source text. The results reported here are for the SCFG-based decoder.

Experimental results. Table 3.12 shows the results of the Arabic diacritization lattice experiments. For each test set (which were identical to those used by Diab et al. (2007)), improvements in translation quality as measured by BLEU are observed. These results are meaningful since they suggest that diacritics do carry information that is useful for translation, and furthermore that these diacritics can be recovered using the techniques proposed in Habash et al. (2007). However, being able to gracefully fall back to more general forms (with diacritics removed) is crucial if the increased sparsity of the models is not to overwhelm the gains attainable by the more precise models.

3.3.3 Segmentation alternatives with word lattices

In the previous two experiments, a distribution over possible inputs to an MT system was utilized, both in the context of spoken language translation tasks, and in situations where the optimal amount of morphological simplification that should be carried out during preprocessing is unclear. In this section, word lattices are used to encode different segmentations of the input. Like the question of the optimal level of morphological simplification, this question is one that is conventionally answered when the translation system is built; and here, as with the morphological simplification and diacritic restoration examples above, the ambiguity-preserving processing model defers the decision to decoding time, allowing the translation system to select sub-sententially among segmentation alternatives. These lattices will be referred to as *segmentation lattices*.

Segmentation lattices also take full advantage of the word lattice's representational capabilities. Until now, only considered patterns of ambiguity that could be represented using confusion networks were used. In this experiment, general word lattices are used, and the problems associated with the ambiguity of path lengths in unrestricted lattices must be considered. The use of segmentation lattices for translating into English from Chinese and Arabic is now considered.

3.3.3.1 Chinese segmentation

Chinese orthography does not indicate word breaks. As such, a necessary first step in translating Chinese using standard models of translation is segmenting the character stream into a sequence of words, which is a (perhaps surprisingly) challenging task. First,

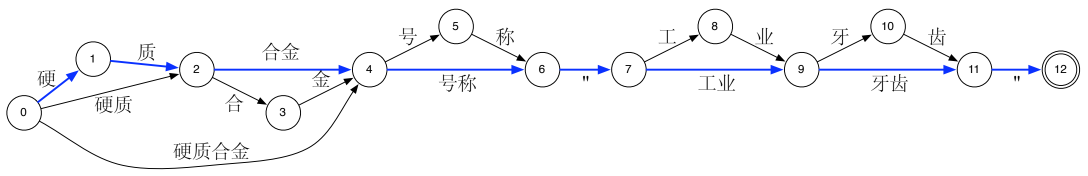


Figure 3.14: Example Chinese segmentation lattice using three segmentation styles.

the segmentation process is ambiguous, even for native speakers (readers) of Chinese. Thus, even if a segmenter performs quite well relative to some gold standard segmentation that has been agreed upon by annotators, it is reasonable that there will still be other alternative segmentations that would have been reasonable. Second, different segmentation granularities may be more or less optimal for translation: some parts of the sentence may benefit from a less compositional translation, making a less granular segmentation more natural. On the other hand, other translations may be relatively direct translations of minimal elements of Chinese words. By encoding segmentation alternatives in the input in a word lattice, the decision as to which granularity to use for a given span can be resolved during decoding rather than when constructing the system. Figure 3.14 illustrates a lattice showing segmentation alternatives. Before looking at translation results, it is necessary to first deal with one additional challenge that occurs in general word lattice translation—the problem of measuring ‘distance’ between nodes in a lattice.

3.3.3.2 The distortion problem in word lattices

The distance between words in the source sentence is used to limit where in the target sequence their translations will be generated in both phrase-based and WSCFG-based translation models. In phrase based translation, distortion is modeled explicitly with a distortion limit d and a distortion penalty. Models that support non-monotonic decoding

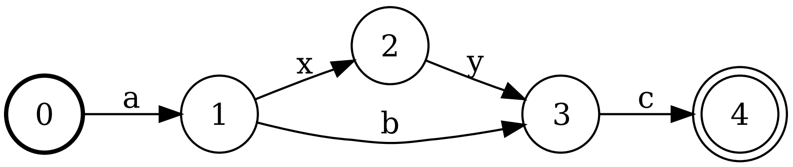


Figure 3.15: Distance-based distortion problem. What is the distance between node 4 and node 0?

generally include a distortion cost, such as $|a_i - b_{i-1} - 1|$ where a_i is the starting position of the foreign phrase $\overline{f}_i$ and b_{i-1} is the ending position of phrase $\overline{f}_{i-1}$ (Koehn et al., 2003). The intuition of this model is that since most translation is monotonic, the cost of skipping ahead or back in the source should be proportional to the number of words that are skipped. In hierarchical phrase-based (WSCFG) models, a distortion limit is usually imposed that prevents the parser from constructing items using anything but the restricted 'glue rule' (§3.1.2.2) when the span size is greater than some size d (Chiang, 2007).

With confusion networks, where all paths pass through the same number of nodes, the distance metric used for the distortion penalty and for distortion limits is well defined; however, in a non-linear word lattice, it poses the problem illustrated in Figure 3.15. Assuming the decoding algorithm described above, if c is generated by the first target word, the distortion penalty associated with 'skipping ahead' should be either 3 or 2, depending on what path is chosen to translate the span [0,3]. In large lattices, where a single arc may span many nodes, the possible distances may vary quite substantially depending on what path is ultimately taken, and handling this properly therefore crucial.

Since a distance metric that will constrain as few of the desired local reorderings as possible on *any* path is wanted, a function $\xi(a,b)$ is used, which returns the length of the shortest path between nodes a and b. Since this function is not dependent on the exact path chosen, it can be computed in advance of decoding using an all-pairs shortest path algorithm (Cormen et al., 1989). Note that in sentences and confusion networks, the shortest path definition does not change the computed distances since every path between two nodes is the same length in those restricted word lattices.

Experimental results. The effect of the proposed distance metrics on translation quality was experimentally determined using Chinese word segmentation lattices (which will be described in detail in the following section), using both a hierarchical and phrase-based system. The shortest-path distance metric was compared with a baseline which uses the difference in node number as the distortion distance. Table 3.13 summarizes the results of the phrase-based systems. On both test sets, the shortest path metric improved the BLEU scores. As expected, the lexicalized reordering model improved translation quality over the baseline; however, the improvement was more substantial in the model that used the shortest-path distance metric (which was already a higher baseline). Table 3.14 summarizes the results of the experiment comparing the performance of two distance metrics to determine whether a rule has exceeded the decoder's span limit. The pattern is the same, showing a clear increase in BLEU for the shortest path metric over the baseline. For the remaining experiments reported, the shortest-path distance metric is used.

3.3.3.3 Chinese segmentation experiments

In the Chinese segmentation experiments two state-of-the-art Chinese word segmenters were used: one developed at Harbin Institute of Technology (Zhao et al., 2001), and one

Distance metric	MT05	MT06
Difference	29.4	27.9
Difference + lex. reordering	29.7	28.9
Shortest path	29.9	28.7
Shortest path + lex. reordering	**30.7**	**29.9**

Table 3.13: Effect of distance metric on phrase-based model performance.

Distance metric	MT05	MT06
Difference	30.6	29.6
Shortest path	**31.8**	**30.4**

Table 3.14: Effect of distance metric on hierarchical model performance.

developed at Stanford University (Tseng et al., 2005). In addition, a simple character-based segmentation (Xu et al., 2004) was used. In the remainder of this section, cs stands for character segmentation, hs for Harbin segmentation and ss for Stanford segmentation. Manual inspection of the segmentations suggested that the Stanford segmenter favored larger groupings of characters than the Harbin segmenter and both were larger than the character-based segmentation. Two types of word lattices were constructed: one that combines the Harbin and Stanford segmenters (hs+ss), and one which uses all three segmentations (hs+ss+cs). An example of a lattice containing three segmentation types is given in Figure 3.14. It was observed that the translation coverage of named entities (NEs) in the baseline systems was rather poor. Since names in Chinese can be composed of relatively long strings of characters that do not translate individually, when generating the segmentation lattices that included cs arcs, segmenting NEs of type PERSON, as identified using a Chinese named-entity tagger (Florian et al., 2004), was avoided.

Data and Settings. The systems used in these experiments were trained on the NIST MT06 Evaluation training corpus, excluding the United Nations data (approximately 950K sentences).[33] The corpus was segmented with the three segmenters. For the systems using word lattices, the training data contained the versions of the corpus appropriate for the segmentation schemes used in the input. That is, for the hs+ss condition, the training data consisted of two copies of the corpus: one segmented with the Harbin segmenter and the other with the Stanford segmenter.[34] MERT was used to optimize the parameters of the translation model using the NIST MT03 test set. The testing was done on the NIST 2005 and 2006 evaluation sets (MT05, MT06).

Experimental results. Using both the FST-based and SCFG-based decoders with word lattices to deserve selection of a segmentation alternative until decoding time yields significant improvements of a single-best baseline where a single segmentation style was committed to in advance of decoding. The results are summarized in Table 3.15. Using word lattices improves BLEU scores both in the phrase-based model and hierarchical model as compared to the single-best segmentation approach. All results using word-lattice decoding for the hierarchical models (hs+ss and hs+ss+cs) are better than the best segmentation (ss). For the phrase-based model, substantial gains were obtained using the word-lattice decoder that included all three segmentations on MT05. The other results, while better than the best segmentation (hs) by at least 0.3 BLEU points, are not as large. In addition to the improvements in BLEU, the number of out-of-vocabulary (OOV)

[33]http://www.nist.gov/speech/tests/mt/

[34]The corpora were word-aligned independently and then concatenated for rule extraction, as used in the Czech (§3.3.2.1) and Arabic experiments (§3.3.2.2) morphology experiments above.

	MT05	MT06
Source	BLEU	BLEU
cs	28.3	26.9
hs	29.1	28.4
ss	28.9	28.0
hs+ss	29.4	28.7
hs+ss+cs	29.9	28.7

(a) Phrase-based model

	MT05	MT06
Source	BLEU	BLEU
cs	29.0	28.2
hs	30.0	29.1
ss	30.7	29.6
hs+ss	31.3	30.1
hs+ss+cs	31.8	30.4

(b) Hierarchical model

Table 3.15: Chinese word segmentation results.

words is improved. For MT06 the number of OOVs in the hs translation is 484. The number of OOVs decreased by 19% for hs+ss and by 75% for hs+ss+cs.

Example. To qualitatively illustrate the benefits of using word-lattices to combine various segmentations, consider the example in Table 3.16 using the hierarchical model. Translations based on Harbin and Stanford segmenters have 3 and 2 untranslated tokens respectively. The example demonstrates that using word-lattices based on all three segmentations yields a net qualitative gain in the translation output. The word-lattice input for the translated phrase *hard alloy known as " industrial teeth ,* is given in Figure 3.14, where the maximal derivation path selected taken by the decoder is in bold. This example makes clear that including the character segmentation in the word-lattice allows the decoder to take the path 0-1-2-4, and thus to correctly output *hard alloy*.

3.3.3.4 Arabic segmentation experiments

Segmentation of the input is also an issue when translating Arabic, in addition to the diacritization issues discussed earlier. Arabic orthography is problematic for lexical and phrase-based MT approaches since a large class of functional elements (prepositions,

cs	hard alloy tooth - called " industrial " , because the hardness and its very much patience of high - pressure for cutting tools , instruments and mining and construction of the railway engineering machinery .
hs	**ying**$_4$ **zhi**$_2$ **he**$_2$ **jin**$_1$ industrial teeth , " because it is known as " a very high fruit and **nai**$_4$ **mo**$_2$ used to **qie**$_1$ **xue**$_1$ tools , and high - pressure equipment and mining and engineering machinery .
ss	**ying**$_4$ **zhi**$_2$ alloy industrial teeth , " because it is known as " a very high hardness and **nai**$_4$ **mo**$_2$ **xing**$_4$ used for building high - pressure - cutting machines tools , instruments and mining and engineering machinery .
hs+ss	**ying**$_4$ **zhi**$_2$ **he**$_2$ **jin**$_1$ alloy known as " industrial teeth , " because it is very high and hardness **nai**$_4$ **mo**$_2$ hardness of for - cutting machines tools , high - pressure equipment building and mining and engineering machinery .
hs+ss+cs	hard alloy known as " industrial teeth , due to the hardness of high and durable grinding nature for cutting tools , high - pressure equipment and the development of mining and the building of roads , engineering machinery .
reference	hard alloy is called " industrial teeth " . due to its high hardness and durability , it is used for cutting tools , high-tension tools as well as for mining and road-building machinery .

Table 3.16: Example translations using the hierarchical model

pronouns, tense markers, conjunctions, definiteness markers) are attached to their host stems. Thus, while the training data may provide good evidence for the translation of a particular stem by itself, the same stem may not be attested when attached to a particular conjunction. For example, the word *sywf* (swords), could appear in a form prefixed by *w-* (and), *l-* (for), and with a suffix *-hm* (their), appearing as *wlsywfhm*, which, although composed of common elements, is infrequent in this particular context.

As with Chinese word segmentation and the morphological simplifications considered above, the usual solution is to commit to the best guess as to the segmentation, in advance of decoding. This is typically done by performing a morphological analysis of the text (where it is often ambiguous whether a piece of a word is part of the stem or merely a neighboring functional element), and then making a subset of the bound functional elements in the language into freestanding tokens. Figure 3.16 illustrates the unsegmented Arabic surface form as well as the segmented variant. Habash and Sadat (2006) showed that a limitation of this approach is that as the amount and variety of training

Form	
surface	wxlAl ftrp AlSyf kAn mEZm AlDjyj AlAElAmy m&ydA llEmAd .
segmented	w- xlAl ftrp Al- Syf kAn mEZm Al- Djyj Al- AElAmy m&ydA l- Al- EmAd .
(English)	During the summer period , most media buzz was supportive of the general .

Figure 3.16: Example of Arabic segmentation driven by morphological analysis.

data increases, the optimal segmentation strategy changes: more aggressive segmentation results in fewer OOV tokens, but evaluation metrics indicate lower translation quality, presumably because the smaller units are being translated less idiomatically.

As was the case with Chinese, segmentation lattices allow the decoder to make decisions about what granularity of segmentation to use *sub-sententially*. Furthermore, since morphological analysis is an inherently ambiguous process, word lattices can effectively capture the resulting ambiguity.

Lattices were constructed from an unsegmented version of the Arabic test data and generated alternative arcs where clitics as well as the definiteness marker and the future tense marker were segmented into tokens. The Buckwalter morphological analyzer was used to perform the necessary morphological analysis, and disambiguation was performed with a unigram model trained on the Penn Arabic Treebank.

Data preparation. For the Arabic segmentation lattice experiments the parallel Arabic-English training data provide for the NIST MT08 evaluation was used. The sentences containing *n*-grams overlapping with the test set were selected using a subsampling method proposed by Kishore Papineni (personal communication), which reduced the amount of training data that the system must process without negatively affecting the quality of the system on the test set of interest. A 5-gram English LM trained on 250M words of En-

	MT05	MT06
Source	BLEU	BLEU
SURFACE	46.8	35.1
MORPH	50.9	38.4
MORPH+SURFACE	**52.3**	**40.1**

(a) Phrase-based model

	MT05	MT06
Source	BLEU	BLEU
SURFACE	52.5	39.9
MORPH	53.8	41.8
MORPH+SURFACE	**54.5**	**42.9**

(b) Hierarchical model

Table 3.17: Arabic morpheme segmentation results

glish training data was used. The NIST MT03 test set was used as development set for optimizing the model weights using MERT. Evaluation was carried out on the NIST 2005 and 2006 evaluation sets (MT05, MT06).

Experimental results. Results are presented in Table 3.17. Using word-lattices to combine the surface forms with morphologically segmented forms substantially improves BLEU scores both in the phrase-based and hierarchical models compared to a baseline where a single segmentation was used.

3.4 Summary

This chapter has demonstrated some practical benefits of using input structured as a WFST to represent alternative source variants in statistical machine translation when compared to systems that must select a single, unambiguous sentence for input. A variety of sources of ambiguity were incorporated into the model to improve translation, some of which have been previously considered in considerable detail (spoken language translation), and others of which are novel (preprocessing decisions). Both lead to improvements. These benefits were available using both a phrase-based (WFST) and hierarchical phrase-based (WSCFG) translation models, suggesting that the generic weighted set framework described in the previous chapter has value as a useful abstraction for describing transduction pipelines.

4 Learning from ambiguous labels

It is wrong always, everywhere, and for everyone, to believe anything upon insufficient evidence.

–W. K. Clifford

When we make inferences based on incomplete information, we should draw them from that probability distribution that has maximum entropy permitted by the information we do have.

–E. T. Jaynes

In the previous chapter, I demonstrated that it is beneficial to utilize a weighted set of strings, rather than single strings, as the input into a statistical machine translation system for a number of quite disparate problems. I turn now to another situation where it is customary to assume a single value, but where a set or distribution can also be more appropriate: in the reference annotations used in supervised learning.[1]

Supervised learning is widely used in natural language processing applications. Rather than burdening the developer with the task of manually writing rules for classification, supervised learning uses training data (consisting of pairs of inputs, from a set

[1]This chapter is a revised and expanded presentation of material published originally in Dyer (2009).

$\mathcal{X}$, with their desired outputs or labels, from a set $\mathcal{Y}$), to induce a model that will (hopefully) generalize and correctly transform novel inputs (novel, but still from $\mathcal{X}$) into the correct output (in $\mathcal{Y}$). Training data usually consists of pairs of a single element from $\mathcal{X}$ (the input) with a single element from $\mathcal{Y}$. In this chapter, I introduce techniques for supervised learning when there is ambiguity about the label for particular inputs. That is, I consider the case where training data consists of pairs of a single element from $\mathcal{X}$ and a subset of elements in $\mathcal{Y}$.

Broadly, two kinds of label ambiguity can be distinguished. The first, which is the primary focus of this chapter, is often called *multi-label classification* and refers to the situation where there may be multiple correct annotations for a single input (Tsoumakas and Katakis, 2007).[2] For example, in document classification, a single document may be assigned to several categories (as an illustration, an article about the Apple iPad could be classified as belong to both the TECHNOLOGY and BUSINESS categories by a document classification system that supports those categories). Machine translation can also be understood as a kind of multi-label classification: there may be many target language sentences that are translations of a source language sentence. The second source of label ambiguity arises when learning from unreliable or noisy annotators. In this case, there is still only a single correct annotation, but the training data are noisy, so a particular training instance may be inaccurately labeled (Dawid and Skene, 1979; Dredze et al., 2009; Jin and Ghahramani, 2002). While the problem of learning from noisy annotations will not be the focus, the techniques developed in this chapter for learning multiple classes are

[2]This is not to be confused with *multi-class* classification, where a single correct class must be selected from among more than 2 classes.

closely related to work that has been done for learning from noisy annotation, and could be utilized for this purpose as well. I return to this topic in the discussion of future work (§4.6).

In the previous chapter, I discussed in some depth one technique for supervised learning, MERT (§3.1.4). While MERT has considerable value for some applications, and can even make use of multiple references (this depends on whether the error function can utilize multiple references when evaluating a candidate prediction), MERT crucially optimizes the performance only of the maximum weight path in the model, otherwise ignoring the scores of the other candidates entirely. As a result, since MERT only cares which candidate has the highest weight, the difference in scores among alternative candidates may be arbitrarily large. Thus, it is problematic to gauge the value of multiple predictions under models trained with MERT: since I would like to predict multiple labels from the model, the model should be informative for all paths.[3] I therefore turn to a different model as the starting point, *conditional random fields* (CRFs; Lafferty et al. (2001)), a probabilistic model whose value is meaningful (as a conditional probability) for every prediction. Since the standard CRF training objective is defined in terms of unambiguous labels, I introduce a technique for generalizing it to multiple labels by positing that a specific label is selected by a latent variable.

As an example of a learning problem where inputs have multiple correct labels, I apply the model to the problem of *compound word segmentation for machine translation*. Using the multi-label CRF training objective, a model is learned that produces sets of

[3]Despite this limitation, multiple outputs from MERT-trained models *are* sometimes used. However, it is generally necessary to learn a secondary scaling factor to make sense of the scores (Tromble et al., 2008).

segmentations (which I encode as a lattice), rather than single segmentations. Note that this is a model that produces, as its *output*, lattices of the kind that served as *input* to the systems described in the previous chapter— see Figure 1.1 in the introduction.

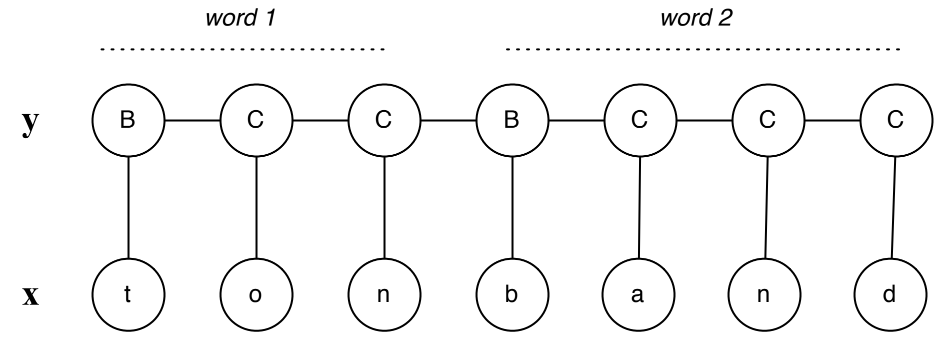
The chapter is organized as follows. First, I review CRFs and their training (§4.1). I then show how they can be extended to use multiple references during training, and trained particularly efficiently when the reference labels are encoded as lattices (§4.2). I then discuss the word segmentation problem, in particular as it applies as a preprocessing step for machine translation (§4.3). I then introduce the concept of a *reference segmentation lattice* (§4.3.2), which is a lattice encoding all correct segmentations of the input.[4] I report the results of an experimental evaluation using reference lattices to train a model for compound segmentation, evaluating the model both in terms of the segmentations learned and on translation tasks (§4.4). I include a brief discussion of related work (§4.5) and then outline future work, focusing in particular on an alternative learning criterion that has more attractive properties than the one used in the rest of the chapter (§4.6).

4.1 Conditional random fields

Since they will form the basis of the multi-label model, I begin with a review of conditional random fields (CRFs). CRFs are discriminatively trained undirected models for structured prediction, in which typically only a single correct label is provided for each training instance (Lafferty et al., 2001). They define a graphical structure relating elements of an input structure to an output structure. In particular, I will focus on two

[4]As with translation, where there may be multiple correct translations, segmentation is a task where there are multiple correct segmentations.

Markov CRF for word segmentation (Tseng et al., 2005):



Semi-Markov CRF for word segmentation (Andrew, 2006; this work):

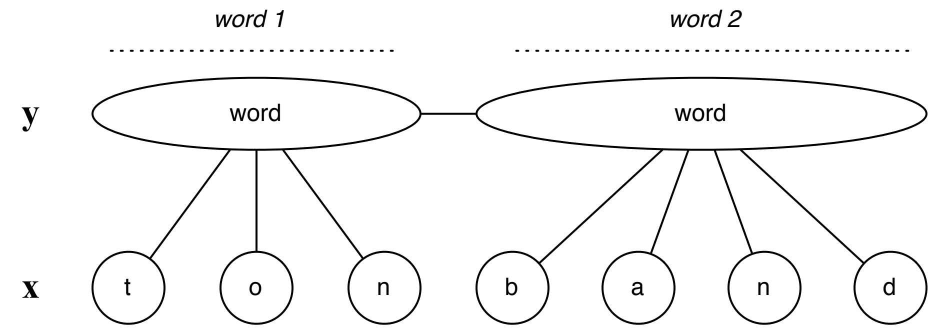


Figure 4.1: Two CRF segmentation models: a fully Markov CRF (above) and a semi-Markov CRF (below).

specific kinds of structures: *sequence models*, where input and label sequences have the same length (fully-Markov linear CRFs; Sha and Pereira (2003)) and models where label sequences may be shorter than the input sequence (semi-Markov or segmental CRFs; Sarawagi and Cohen (2004)). Figure 4.1 shows examples of both, as they are applied to the word segmentation problem.[5]

CRFs are specified by a fixed vector of m global real-valued feature functions $\vec{H}(\mathbf{x},\mathbf{y}) = \langle H_1(\mathbf{x},\mathbf{y}), H_2(\mathbf{x},\mathbf{y}), \ldots, H_m(\mathbf{x},\mathbf{y})\rangle$, where $\mathbf{x}$ and $\mathbf{y}$ are input and output sequences,

[5]Figure 4.1 is not a plate diagram; it simply shows example assignments of values to random variables in a CRF. Each structure would correspond to a particular posterior probability.

respectively, and $H_k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. CRFs are parameterized by a vector of m feature weights $\Lambda = \langle \lambda_1, \lambda_2, \ldots, \lambda_m \rangle \in \mathbb{R}^m$. CRFs define a conditional probability distribution of labels $\mathbf{y}$ given an input $\mathbf{x}$ as follows:

$$p(\mathbf{y}|\mathbf{x};\Lambda) = \frac{\exp \sum_k \lambda_k \cdot H_k(\mathbf{x},\mathbf{y})}{Z(\mathbf{x};\Lambda)} \quad \text{where } Z(\mathbf{x};\Lambda) = \sum_{\mathbf{y}'} \exp \sum_k \lambda_k \cdot H_k(\mathbf{x},\mathbf{y}') \tag{4.1}$$

The function $Z(\mathbf{x};\Lambda)$,[6] called the *partition function*, ensures that the conditional probability distribution is properly normalized. Note that it only depends on the input $\mathbf{x}$, not the prediction $\mathbf{y}$. As a result, to infer the prediction $\hat{\mathbf{y}}$ with the maximum posterior probability, it is not necessary to compute this value; however, it *is* required in training (discussed below).

CRFs further stipulate that every feature function $H_k(\mathbf{x},\mathbf{y})$ must additively decompose into sums of *local* feature functions h_k over the cliques C in the graph $\mathcal{G}$:

$$H_k(\mathbf{x},\mathbf{y}) = \sum_{C \sqsubset \mathcal{G}} h_k(\mathbf{y}|_C, \mathbf{x})$$

Where $\mathbf{y}|_C$ are the components associated with subgraph C. Therefore, in the fully-Markov linear CRF, the cliques are just the nodes and edges between adjacent nodes in a graph. In such models, feature functions can be rewritten as follows:

$$H_k(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{|\mathbf{x}|} h_k(y_i, y_{i-1}, \mathbf{x})$$

[6]The symbol Z is short for the (appropriately) German compound word *Zustandssumme*, meaning *the sum over states*; this notation derives from statistical physics, where it is used to relate various thermodynamic quantities.

In the semi-Markov case, a prediction node y_i has a *start time* s_i and a *duration* d_i, such that $d_i > 0$ for all i and $|\mathbf{x}| = \sum_{i=1}^{|\mathbf{y}|} d_i$ and $s_1 = 1$ and $s_i = s_{i-1} + d_{i-1}$. The global feature functions decompose as follows:

$$H_k(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{|\mathbf{y}|} h_k(y_i, y_{i-1}, s_i, d_i, s_{i-1}, d_{i-1}, \mathbf{x})$$

Fully-Markov CRFs can be interpreted as semi-Markov models where $d_i = 1$ for all i.

Conveniently, for a given $\mathbf{x}$, the posterior distribution of semi-Markov CRFs (and therefore also fully Markov CRFs) can be represented by an acyclic WFST (Lafferty et al., 2001). In the WFST representation, the network is organized such that the predicted labels occur on the edges (whereas in the usual undirected graphical model representation, the predicted values are nodes), and states correspond to particular unique settings of cliques in the graph. Because each state corresponds to the settings of all variables (nodes) in a clique, the size of the WFST is exponential in the size of the clique in nodes. The weight of a path from source to final states produces a label sequence $\mathbf{y}$ and its weight is the numerator of Equation (4.1), and the global features decompose in terms of transitions in the WFST. Figure 4.2 shows a WFST encoding of the fully-Markov CRF from Figure 4.1, which predicts a sequence of B's and C's of length 7, given the input $\mathbf{x} =$ *tonband* (the meanings of the labels are discussed below in §4.1.2).

Fully- or semi-Markov? Since semi-Markov CRFs offer more flexibility than fully-Markov CRFs, it is natural to ask why fully-Markov CRFs would ever be used in sequence modeling. First, in applications where the predicted sequence should always be equal

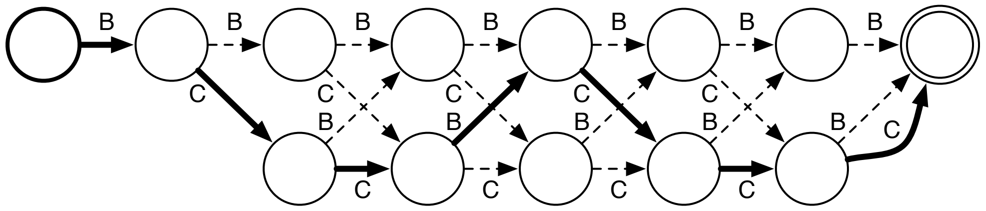


Figure 4.2: A WFST encoding (weights not shown) of the posterior distribution of the CRF from the upper part of Figure 4.1. The highlighted path corresponds to the variable settings shown in the example.

in length to the input sequence, e.g. part-of-speech tagging, semi-Markov CRFs have little value. Second, fully-Markov CRFs can be somewhat more efficient than their semi-Markov counterparts, although in general the overhead is not substantial. Finally, with semi-Markov CRFs, it is slightly easier to define features that will be extremely sparse, making very large amounts of training data necessary. However, in general, semi-CRFs are indeed a much more flexible representation.

4.1.1 Training conditional random fields

Conditional random fields are trained using the maximum conditional likelihood training criterion or maximum *a posteriori* (MAP) criterion. I review these here. Given a set of training data $\mathcal{D} = \{\langle \mathbf{x}^j, \mathbf{y}^j \rangle\}_{j=1}^{\ell}$, the maximum conditional likelihood estimator (MCLE) can be written as follows.

$$\begin{aligned}\Lambda^*_{\text{MCLE}} &= \arg\max_{\Lambda} \prod_{j=1}^{|\mathcal{D}|} p(\mathbf{y}^j|\mathbf{x}^j;\Lambda) \\ &= \arg\min_{\Lambda} -\sum_{j=1}^{|\mathcal{D}|} \log p(\mathbf{y}^j|\mathbf{x}^j;\Lambda) \qquad (4.2)\end{aligned}$$

Supervised learning can be formalized as the minimization of a particular training objective $\mathcal{L}$, which is a function of the model parameters Λ and the training data. From Equation (4.2), the MCLE training objective is

$$\mathcal{L}_{\text{MCLE}}(\Lambda) = -\sum_{j=1}^{|\mathcal{D}|} \log p(\mathbf{y}^j|\mathbf{x}^j;\Lambda)\ .$$

Substituting the CRF definition of the condition probability from Equation (4.1) yields

$$\mathcal{L}_{\text{MCLE}}(\Lambda) = -\sum_{j=1}^{|\mathcal{D}|} \left(\sum_k \lambda_k \cdot H_k(\mathbf{y}^j,\mathbf{x}^j) - \log Z(\mathbf{x}^j;\Lambda) \right)$$

Since $\mathcal{L}_{\text{MCLE}}(\Lambda)$ is globally convex (Lafferty et al., 2001), it can be minimized by solving to find where the gradient $\nabla\mathcal{L}_{\text{MCLE}}(\Lambda) = \mathbf{0}$. The gradient $\nabla\mathcal{L}_{\text{MCLE}}(\Lambda)$ is:

$$\frac{\partial \mathcal{L}_{\text{MCLE}}}{\partial \lambda_k} = \sum_{j=1}^{|\mathcal{D}|} \left(\underbrace{H_k(\mathbf{y}^j,\mathbf{x}^j)}_{\text{empirical feature value}} - \underbrace{\sum_{\mathbf{y}'} p(\mathbf{y}'|\mathbf{x}^j;\Lambda) H_k(\mathbf{y}'|\mathbf{x}^j)}_{\text{expected value of } H_k \text{ in model}} \right) \tag{4.3}$$

The right-hand term in this summation, the expected value of H_k under the model, is the derivative of the log partition function with respect to λ_i. The derivation requires only basic algebra, but since it does not offer any insight, I omit it here. For a detailed derivation, see Smith (2004).

Unfortunately, $\nabla\mathcal{L}_{\text{MCLE}}(\Lambda) = \mathbf{0}$ has no analytic solution; however, numerical methods work quite well (Sha and Pereira, 2003). For the experiments reported below, the quasi-Newtonian method called limited-memory (L-) BFGS is used (Liu and Nocedal, 1989). It should also be noted that the form of the gradient has a natural interpretation: it is the difference between the feature function's value in the training data (the 'empirical' expectation of the feature function) and the expected value of the feature in the model's posterior distribution. This means the gradient is zero when every feature's expectation under the model (labeling the input half of the training data) matches the feature's empirical expectations (that is, the feature's average value in the full training data).

Before concluding the introduction to CRFs, it is worth remarking that maximum likelihood estimators typically overfit the training data. Therefore, it is often advantageous to define a *prior distribution* over the space of models, $p(\Lambda)$, and solve for the maximum *a posteriori* (MAP) estimator, as follows.

$$\begin{aligned}\Lambda^*_{\mathrm{MAP}}(\Lambda) &= \arg\max_{\Lambda}\left(p(\Lambda)\prod_{j=1}^{|\mathcal{D}|}p(\mathbf{y}^j|\mathbf{x}^j;\Lambda)\right)\\ &= \arg\min_{\Lambda}\left(-\log p(\Lambda)-\sum_{j=1}^{|\mathcal{D}|}\log p(\mathbf{y}^j|\mathbf{x}^j;\Lambda)\right)\end{aligned}$$

Using a Gaussian prior with mean μ (usually $= \mathbf{0}$) and a diagonal covariance matrix with elements σ^2, yielding a new objective, which is still convex (Chen and Goodman, 1996):

$$\mathcal{L}_{\mathrm{MAP}}(\Lambda) = -\sum_k \frac{(\lambda_k-\mu_k)^2}{2\sigma^2} - \sum_{j=1}^{|\mathcal{D}|}\left(\sum_k \lambda_k \cdot H_k(\mathbf{y}^j,\mathbf{x}^j) - \log Z(\mathbf{x}^j;\Lambda)\right) \tag{4.4}$$

The corresponding gradient $\nabla\mathcal{L}_{\mathrm{MAP}}(\Lambda)$ is:

$$\frac{\partial\mathcal{L}_{\mathrm{MAP}}}{\partial\lambda_k} = \underbrace{\frac{\lambda_k-\mu_k}{\sigma^2}}_{\text{prior penalty}} - \sum_{j=1}^{|\mathcal{D}|}\left(\underbrace{H_k(\mathbf{y}^j,\mathbf{x}^j)}_{\text{empirical feature value}} - \underbrace{\sum_{\mathbf{y}'}p(\mathbf{y}'|\mathbf{x}^j;\Lambda)h_k(\mathbf{y}'|\mathbf{x}^j)}_{\text{expected feature value in model}}\right) \tag{4.5}$$

With the fully-Markov linear CRFs and the semi-CRFs used here, the forward-backward algorithm can be used to compute, in polynomial time, the log partition function $\log Z(\mathbf{x};\Lambda)$ and the feature expectations under the current model parameters. The empirical expectations can be computed once just by iterating over the training examples and evaluating the feature functions on the pairs.

The MAP estimator with a Gaussian prior with $\mu = \mathbf{0}$ indicates that models should not have too much weight assigned to any feature. This means that no single feature can come to dominate. This improves the quality of the models learned since it prevents features that only accidentally correlate strongly with various labels in the training data from getting too much weight. Intuitively, the variance parameter σ determines how strictly the prior is enforced. Usually, this value is tuned on a held-out development set.

4.1.2 Example: two CRF segmentation models

As an illustration of the applications of CRFs, consider these two closely related models, one fully-Markov model and one semi-Markov model used for word segmentation. Tseng et al. (2005) introduce a fully-Markov CRF segmenter for Chinese text. Their model defines the probability of every segmentation $\mathbf{y}$, given an unsegmented input word $\mathbf{x}$, where x_i is the i^{th} character of the input. Every x_i is classified with a y_i as being the start of a new word (B = BEGIN) or the continuation of one (C = CONTINUE), that is, $y_i \in \{\text{B}, \text{C}\}$. The graphical structure of this model is shown for a German compound segmentation example in the upper part of Figure 4.1. In the example, the string *tonband* is being segmented into two words *ton* (audio) and *band* (tape). Although this model attains very good performance in comparison to many models on a Chinese segmentation task, fully-Markov CRFs are inherently limited for the word segmentation task since they are incapable of using features that are sensitive to the words they are predicting, unless the words are shorter than the Markov window used. This limitation arises because the model is restricted to only compute features that are functions over cliques in the graph—which in this case are just adjacent pairs of characters!

Andrew (2006) suggests improving this model by using a semi-CRF where the states predicted encompass the entire word segment predicted. This enables using features of the words predicted (such as unigram and bigram features), regardless of the length of the segments. The graphical structure of Andrew's model is shown in the lower part of Figure 4.1.

4.2 Training CRFs with multiple references

As seen in the previous section, the standard training objective of CRFs is formulated in terms of a single labels for each training instance. However, when the training data consists of sets of correct labels, i.e., $\mathcal{D} = \{\langle \mathbf{x}^j, Y^j \rangle\}_{j=1}^{\ell}$, where $Y^j \subseteq \mathcal{Y}$, the training criterion must be altered. In particular, I use the EM 'trick' (Dempster et al., 1977) of summing over the values in the reference set to compute the *marginal* conditional likelihood (this treats one of them as the latent 'true' answer):

$$\Pr(Y|\mathbf{x}) = \sum_{\mathbf{y}' \in Y} p(\mathbf{y}'|\mathbf{x};\Lambda)$$

The maximum marginal conditional likelihood estimator (MMCLE) objective can therefore be written as follows:

$$
\begin{aligned}
\mathcal{L}_{\text{MMCLE}}(\Lambda) &= -\log\left(\prod_{j=1}^{\mathcal{D}} \sum_{\mathbf{y}'\in Y^j} \frac{\exp\sum_k \lambda_k \cdot H_k(\mathbf{x},\mathbf{y}')}{Z(\mathbf{x};\Lambda)}\right) \\
&= -\sum_{j=1}^{\mathcal{D}} \log \underbrace{\sum_{\mathbf{y}'\in Y^j} \exp\sum_k \lambda_k \cdot H_k(\mathbf{x}^j,\mathbf{y}')}_{=Z(\mathbf{x}^j,Y^j;\Lambda)} - \log Z(\mathbf{x}^j;\Lambda) \\
&= -\sum_{j=1}^{|\mathcal{D}|} \log Z(\mathbf{x}^j,Y^j;\Lambda) - \log Z(\mathbf{x}^j;\Lambda) \qquad (4.6)
\end{aligned}
$$

I denote the summation over the possible labels in Y using $Z(\mathbf{x},Y;\Lambda)$ in Equation (4.6) since this value has the exact same structure as the regular partition function, $Z(\mathbf{x};\Lambda)$, only it is restricted to sum over the settings of the CRF that match an element in Y. The gradient can has a similar form:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_{\text{MMCLE}}}{\partial \lambda_k} &= \sum_{j=1}^{|\mathcal{D}|}\left(\underbrace{\sum_{\mathbf{y}'\in Y^j} p(\mathbf{y}'|\mathbf{x}^j;\Lambda) H_k(\mathbf{y}'|\mathbf{x}^j)}_{\text{empirical feature expectation}} - \underbrace{\sum_{\mathbf{y}'} p(\mathbf{y}'|\mathbf{x}^j;\Lambda) H_k(\mathbf{y}'|\mathbf{x}^j)}_{\text{expected feature value in model}}\right) \\
&= \sum_{j=1}^{|\mathcal{D}|}\left(\ \mathbb{E}_{p(\mathbf{y}'|\mathbf{x}^j,Y^j;\Lambda)}[H_k(\mathbf{x}^j,\mathbf{y}')] - \mathbb{E}_{p(\mathbf{y}'|\mathbf{x}^j;\Lambda)}[H_k(\mathbf{x}^j,\mathbf{y}')]\ \right) \qquad (4.7)
\end{aligned}
$$

Equation 4.7 is quite similar to the original statement of the gradient in the case of unambiguous inputs, given in Equation 4.3. However, in this case, the empirical feature values have been replaced with an *expectation* of the feature values, weighted by the posterior distribution of over segmentations given the model. Unfortunately, unlike the standard

CRF objective, $\mathcal{L}_{\text{MMCLE}}(\Lambda)$ is not globally convex in the model parameters, and therefore gradient descent will only find a local minima. However, in the case of the experiments discussed below, a significant initialization effect is not observed.

The multi-reference model is closely related to the discriminative latent variable models that have been utilized for many different tasks (Blunsom et al., 2008a,b; Clark and Curran, 2004; Dyer and Resnik, 2010; Koo and Collins, 2005; Petrov and Klein, 2008; Quattoni et al., 2004; Sun et al., 2009). However, in this previous work, the latent variables were used for modeling convenience (that is, they were nuisance variables), rather than as ambiguous alternative prediction possibilities. Dredze et al. (2009), who were exploring strategies for dealing with ambiguous, possibly *incorrect*, training labels, independently derived a model that is identical to this, except each label in the reference set has a prior probability of correctness (in their experiments, the prior correctness probability was estimated by annotator agreement). With a uniform prior, their model is identical to ours.[7] However, while the models are closely related, they explicitly sum over every reference in Y. In the following section, I show how these references can be compactly encoded in an FST and the summation over all paths can be efficiently computed using dynamic programming.

Efficient training with reference lattices. It is possible to compactly encode a very large number of correct annotations if the label set Y is encoded using *reference lattices*, which compactly encode many alternatives.[8] This encoding permits the summation over

[7] The two papers introducing this training strategy, Dyer (2009) and Dredze et al. (2009), were published concurrently and independently.

[8] In the experiments below, the reference lattices will be segmentation lattices (§4.3.2), but in principle the lattices could encode any kind of labels for any task.

the labels that is encoded to be carried out efficiently using a dynamic programming algorithm.

The key to efficient training relies on the fact that the posterior distribution of any fully- or semi-Markov sequential CRF can be encoded as an acyclic WFST (§4.1). As described aboe, during standard CRF training, a summation over multiple variable settings is only necessary when computing the denominator, $Z(\mathbf{x};\Lambda)$, of the probability distribution (and the associated feature expectations under the model, for the gradient). For this, the INSIDE algorithm on the WFST encoding of the CRF can be used. Since the feature functions decompose over the transitions in the state machine, the INSIDEOUTSIDE algorithm wll compute their expected value under the model.

With multiple references, the numerator *also* involves a summation (as does the corresponding term in the gradient). However, if the references, Y, are themselves encoded as a WFST, the numerator can be computed by composing the WFST-encoding of the posterior distribution of the CRF with Y, and then running the INSIDE and INSIDE-OUTSIDE algorithms to compute the required quantities. Figure 4.3 gives the pseudocode for training.

Intuitively, this training procedure can be understood as iteratively moving probability mass from paths in the CRF posterior lattice that are not in the reference lattice to those paths that are found in the reference.

```
function $\textsc{ObjectiveAndGradient}(\mathcal{D}, \Lambda, \mathbf{h}(\cdot))$
    $\mathcal{L}_{\text{MMCLE}} \leftarrow 0$
    $\nabla\mathcal{L}_{\text{MMCLE}} \leftarrow \mathbf{0}$                                    $\triangleright$ $m$ dimensions
    for all $\langle \mathbf{x}, Y \rangle \in \mathcal{D}$ do                $\triangleright$ $Y$ is a FST encoding the reference lattice for $\mathbf{x}$
        $T \leftarrow \textsc{BuildCRFasWFST}(\mathbf{x}, \Lambda, \mathbf{h}(\cdot))$      $\triangleright$ $T$ encodes all segs. (see Figure 4.2)
        $\log Z(\mathbf{x}; \Lambda) \leftarrow \textsc{Inside}(T, \text{log semiring})$
        $\mathbb{E}_{p(\mathbf{y}'|\mathbf{x})}[\mathbf{h}(\mathbf{x}, \mathbf{y}')] \leftarrow \textsc{InsideOutside}(T, \text{log semiring})$
        $T \leftarrow T \circ Y$                                  $\triangleright$ Compose $T$ with the reference $Y$
        $\log Z(\mathbf{x}, Y; \Lambda) \leftarrow \textsc{Inside}(T, \text{log semiring})$
        $\mathbb{E}_{p(\mathbf{y}'|\mathbf{x},Y)}[\mathbf{h}(\mathbf{x}, \mathbf{y}')] \leftarrow \textsc{InsideOutside}(T, \text{log semiring})$
        $\mathcal{L}_{\text{MMCLE}} \leftarrow \mathcal{L}_{\text{MMCLE}} + \log Z(\mathbf{x}, Y; \Lambda) - \log Z(\mathbf{x}; \Lambda)$
        $\nabla\mathcal{L}_{\text{MMCLE}} \leftarrow \nabla\mathcal{L}_{\text{MMCLE}} + \mathbb{E}_{p(\mathbf{y}'|\mathbf{x},Y)}[\mathbf{h}(\mathbf{x}, \mathbf{y}')] - \mathbb{E}_{p(\mathbf{y}'|\mathbf{x})}[\mathbf{h}(\mathbf{x}, \mathbf{y}')]$
    return $\langle \mathcal{L}_{\text{MMCLE}}, \nabla\mathcal{L}_{\text{MMCLE}} \rangle$
```

Figure 4.3: Pseudo-code for training a segmentation CRF with reference lattices.

4.3 Word segmentation and compound word segmentation

As an application of training CRFs with multiple references encoded in a lattice, I turn to the problem of compound word segmentation for machine translation. I begin with an overview of the word segmentation problem in general.

Word segmentation is the problem of determining where in a sequence of symbols from a finite alphabet Σ one morpheme ends and the next begins, assuming morphemes are themselves composed of sequences of one or more symbols from Σ (in text processing, these are letters; in speech segmentation these may be phones or phonemes). More formally, given a language L built of strings of words from lexicon Δ, it is assumed that every word w consists of one or more letters from a finite alphabet Σ of letters, that is, $w \in \Sigma^+$. Word segmentation is the task of breaking a string $\mathbf{x} = \sigma_1\sigma_2\ldots\sigma_\ell \in \Sigma^*$ into a list of k words $w_1, w_2, \ldots, w_k \in \Delta^*$ such that $w_1w_2\ldots w_k = \sigma_1\sigma_2\ldots\sigma_\ell$. For example, the German compound word *tonbandaufnahme*, meaning *audio cassette recording*, consists of at least 3 morphemes: *ton* (audio), *band* (cassette), and *aufnahme* (recording).[9]

The problem of recovering the segmentation into words of a sequence of tokens in Σ is challenging. While Σ (the phonemic inventory or alphabet of a language) is finite and often quite small, Δ (the lexicon) may be quite large and is, in fact, may not be finite (productive word creation processes, like derivational morphology and borrowing from other languages may allow for the creation of an unbounded number of new morphemes). More serious still, determining the content of the lexicon is difficult: does a sequence of tokens correspond to multiple words or just a single (long) one? Even with a 'gold-standard' lexicon, there may be multiple valid ways of decomposing a sequence of words that is consistent with the lexicon. Models that use features of the words, phonology of the language, and even semantics to adequately resolve all word segmentation challenges are therefore necessary.

4.3.1 Compound segmentation for MT

As demonstrated in the previous chapter (§3.3.3), how the input to a machine translation system is segmented can substantially influence the quality of the translations it produces. In the experiments in this chapter, I focus on the translation into English from languages such as German, which exhibit productive compounding, and where the orthography does not indicate breaks between constituent morphemes. These compound words pose particular problems for statistical translation systems because translation systems do not typically consider the internal structure of words; however, the creation of these words is a

[9]We say 'at least 3 morphemes' because *aufnahme* can arguably be decomposed into two morphemes *auf* and *nahme*; however, the semantics do not decompose.

standard part of language use and a translation system that does not deal with them will be inadequate for most applications. This problem is widely known, and the conventional solution (which has been shown to work well for many language pairs) is to segment compounds into their constituent morphemes, as a source preprocessing step, using either morphological analyzers or empirical methods, and then to translate from this segmented variant (Dyer et al., 2008; Koehn et al., 2008; Yang and Kirchhoff, 2006).

But into what units should a compound word be segmented? When viewed as a stand-alone task, the goal of a compound segmenter is a segmentation of the input that matches the linguistic intuitions of native speakers of the language. However, for MT, fidelity to linguistic intuition is less important. Instead, segmentations that produce correct word-by-word translations are considered appropriate (Ma et al., 2007). Unfortunately, determining the optimal segmentation for MT is challenging, often requiring extensive experimentation and frequently disagreeing with linguistic intuitions about how best to segment input (Chang et al., 2008; Habash and Sadat, 2006; Koehn and Knight, 2003).

Now, as seen in Chapter 3, translation quality could be improved by using segmentation lattices which combine the output from *multiple* segmenters, compared to systems where only a single segmenter was used. A further advantage of this approach is that the problem of having to identify the best single segmentation can be sidestepped since *all* segmentations are used. Unfortunately, the way segmentation lattices were constructed the previous chapter required multiple segmenters that behaved *differently* on the same input. However, only a few languages (for example, Arabic and Chinese, the example source languages from the previous chapter) have had such a wealth of resources developed for them, making the technique of limited utility. I therefore would like to model segmentation lattices directly, rather than relying on the good fortune of having multiple systems with different behaviors that ‘accidentally’ produce segmentation lattices.

In this chapter, I solve the problem of generating segmentation lattices using supervised learning: for a training set of example compound words, an annotator provides *reference segmentation lattices*, whose paths are all *possible* segmentations of the word. From the pairs of unsegmented inputs and reference segmentation lattices the model learns how to generate novel lattices like them for words unseen during training. I now describe reference segmentation lattices as a means of representation alternative possible good segmentations (for the purposes of translation) of an input which I will then use to train a CRF segmentation model using the training algorithm described above (§4.2).

4.3.2 Reference segmentation lattices for MT

Figure 4.4 shows segmentation lattices for two typical German compound words, whose paths are compatible with the German lexicon (with English glosses) shown in Table 4.1. While these two words are structurally quite similar, translating them into English is most straightforward when they have been segmented differently. In the upper example, *tonbandaufnahme* can be rendered into English by following 3 different paths in the lattice and translating the words independently, *ton*/audio *band*/tape *aufnahme*/recording, *tonband*/tape *aufnahme*/recording, and *tonbandaufnahme*/tape recording. In contrast, *wiederaufnahme* (English: resumption) is most naturally translated correctly using the unsegmented form, even though in German the meaning of the full form is a composition

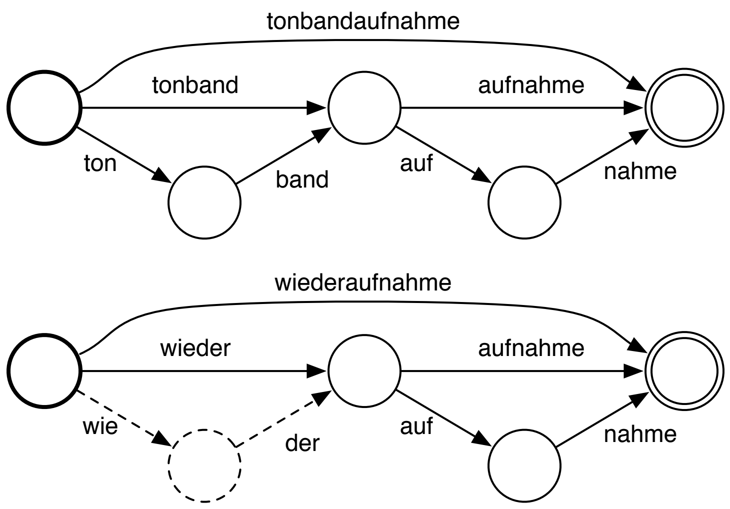


Figure 4.4: Segmentation lattice examples. The dotted structure indicates linguistically implausible segmentation that might be generated using dictionary-driven approaches.

Table 4.1: German lexicon fragment for words present in Figure 4.4.

German	English
auf	*on, up, in, at, ...*
aufnahme	*recording, entry*
band	*reel, tape, band*
der	*the, of the*
nahme (misspelling of *nähme*)	*took* (3P-SG-CONJ)
ton	*sound, audio, clay*
tonband	*tape, audio tape*
tonbandaufnahme	*tape recording*
wie	*how, like, as*
wieder	*again*
wiederaufnahme	*resumption*

of the meaning of the individual morphemes.[10] I define a reference segmentation lattice to be a lattice containing only paths that lead to 'compositional' translations like this.

It should be mentioned that phrase-based models can translate multiple words as a unit, and therefore capture non-compositional meaning. Thus, by default, if the training

[10]The English word *resumption* is likewise composed of two morphemes, the prefix *re-* and a kind of bound morpheme that never appears in other contexts (sometimes called a 'cranberry morpheme'), but the meaning of the whole is idiosyncratic enough that it cannot be called compositional.

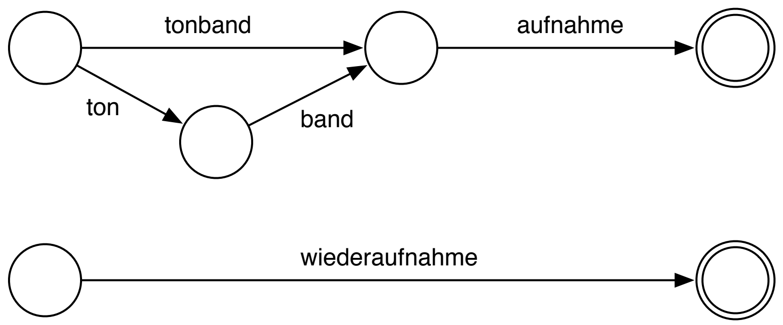


Figure 4.5: Manually created reference lattices for the two words from Figure 4.4. Although only a subset of all linguistically plausible segmentations, each path corresponds to a plausible segmentation for word-for-word German-English translation.

data is processed such that, for example, *aufnahme*, in its sense of *recording*, is segmented into two words, then more paths in the lattices become plausible translations. However, using a strategy of 'over segmentation' and relying on phrase models to learn the non-compositional translations has been shown to degrade translation quality significantly (Habash and Sadat, 2006; Xu et al., 2004). I thus desire lattices containing as little oversegmentation as possible.

In summary, the reference segmentation lattice should be a lattice where every path corresponds to a reasonable word-for-word direct translation into the target language. Figure 4.5 shows an example of the reference lattice for the two words I just discussed.

4.4 Experimental evaluation

In this section, I describe a compound segmentation model that is trained optimizing the multi-label CRF objective (introduced in §4.2), using reference lattices as defined in the previous section. I report the promising results of an intrinsic evaluation of the segmentations, as well as an extrinsic evaluation, where the segmentation model is used to generate inputs to a statistical machine translation system. In the extrinsic evaluation, not only does the model perform well in German, the language it was trained in, but it performs well when used with languages with similar compounding processes, Hungarian and Turkish.

4.4.1 Segmentation model and features

In the introduction to CRFs above, I discussed two possibilities for modeling word segmentation with CRFs: either using a fully-Markov CRF and predicting a word break or continuation at each position in the word, or using a semi-Markov CRF and modeling segments. Since the test language is German, whose compound segments are typically many characters in length and where previous work has tended to use features of the predicted segments to make segmentation decisions, I make use of the semi-Markov alternative.

Feature functions. Feature engineering is of crucial importance for any model. I summarize the requirements for the ones used in model here. First, I will favor 'dense', rather than sparse features, since this will minimize the amount of training data required.[11] Second, I will favor linguistically motivated features, such as phonotactic probability, which tend to be cross-linguistically meaningful. Because the model uses cross linguistic features that should be well defined in most languages with productive compounds, the model (which is trained on German compound words) will be evaluated on two other compounding languages, Hungarian and Turkish. Finally, for computational tractability, output contextual features (that is, features that depend on previous decisions made in

[11] It is quite common to use large numbers of sparse binary features with CRFs, such as features indicating the presence of specific words or *n*-grams.

the model)—the model will make its predictions using only local information about the segment being hypothesized.

Table 4.2 lists the features used in two variants of the compound segmentation model, one trained to deal specifically with some relatively unique processes in German compounds and another that (while likewise trained on German), ignores those processes. An explanation of the features is as follows. y_i is the i^{th} hypothesized segment with length $|y_i|$. It starting at position s_i in $\mathbf{x}$. I include features that depend on the relative frequency of a hypothesized segment in a large monolingual corpus, $f(y_i)$, which builds on prior work of (Koehn and Knight, 2003) that relied heavily on the frequency of the hypothesized constituent morphemes in a monolingual corpus. Binary predicates evaluate to 1 when true and 0 otherwise. $f(y_i)$ is the frequency of the token y_i as an independent word in a large monolingual corpus. $\phi(\#|x_{s_i} \cdots x_{s_i+4})$ is a proxy for phonotactic probability: it is the probability of a word start preceding the letters $x_{s_i} \cdots x_{s_i+4}$, as estimated by a 5-gram character model trained in the reverse direction on a large monolingual corpus.[12]

Since German compound morphology inserts small 'functional' morphemes between the some constituent words inside compounds, the model permits the strings *s*, *n*, and *es* (the so-called *Fugenelemente*) to be deleted between words in the 'German' model. Each deletion fired a count feature (listed as *fugen* in the table). Figure 4.6 shows an example of a noun-noun compound that has an *s* placed between the two constituent nouns.

Analysis of errors in development data indicated that the segmenter would periodi-

[12]In general, this feature helps avoid situations where a word may be segmented into a frequent word and then a non-word string of characters since the non-word typically violated the phonotactics of the language in some way.

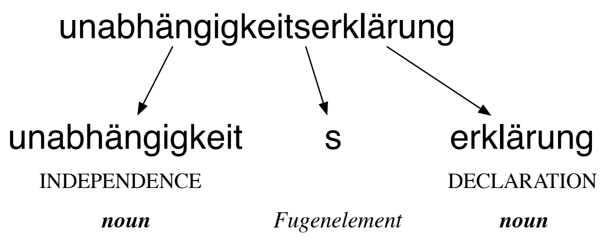


Figure 4.6: An example of a *Fugenelement* in the German word *Unabhängigkeitserklärung* (English *Declaration of Independence*), where *s* is inserted as part of the compounding process but does not occur with the words when they occur in isolation.

Table 4.2: Features and weights learned by maximum marginal conditional likelihood training, using reference segmentation lattices, with a Gaussian prior with $\mu = \mathbf{0}$ and $\sigma^2 = 1$. Features sorted by weight magnitude.

Feature	German	all-language
$f(y_i) > 0.005$	-2.98	-3.31
$^\dagger y_i \in \mathcal{N}$ (= list of 'bad' segments)	-2.32	–
$\lvert y_i \rvert \leq 4$	-1.02	-1.18
†*fugen*	-0.71	–
$\lvert y_i \rvert \leq 10,\ f(y_i) > 2^{-10}$	0.60	-0.82
$\lvert y_i \rvert \geq 12$	-0.58	-0.79
$\sqrt{\lvert y_i \rvert}$	0.58	-0.64
$\log \phi(\# \lvert x_{s_i} x_{s_i+1} x_{s_i+2} x_{s_i+3})$	0.46	2.11
segment penalty	0.33	2.04
$2^{-10} < f(y_i) < 0.005$	-0.28	-0.45
$f(y_i) > 0$	0.23	3.64
$\log f(y_i)$	-0.26	-0.36
$f(y_i) = 0$	0.10	-1.09
$\lvert y_i \rvert^2$	0.038	0.018

cally propose an incorrect segmentation where a single word could be divided into a word and a non-word consisting of common inflectional suffixes. To address this, an additional feature was added that fired when a proposed segment was one of a set $\mathcal{N}$ of 30 non-words that were observed to occur frequently in the development set. The weights shown in Table 4.2 are those learned by MAP training criterion summing over all paths in the reference segmentation lattices. German-specific features are indicated with †.

4.4.2 Training data

There are two separate sets of training and test data, one for the segmentation component, which is the focus of the learning innovation in this chapter, and one for the translation component, which exploits the segmentation lattices generated as input to the translation system (§3.2).

Segmentation training data. Segmentation training data was produced by randomly choosing 11 German newspaper articles (from news stories published in 2008 and available freely on the Internet), identifying all words greater than 6 characters in length, and then manually segmenting each word so that the resulting units could be translated 'directly' into English.[13] This is a rather unspecific task definition, but since reference lattices may contain any number of segmentations, it was not particularly challenging since it was never necessary to select *the* 'correct' segmentation. For a test set, 7 further newspaper articles and one article from the German language version of Wikipedia were selected. This resulted in 621 training sentences corresponding to 850 paths for the dev set, and 279 words (302 paths) for the test set. Each word segments into on average 1.4 different segmentations (range, 1 to 8). The development and test data are publicly available for download.[14]

Translation training data. For translation experiments, a hierarchical phrase-based translation system (§3.1.2.2) capable of translating word lattice input (§3.2) was utilized. For the language model, a 5-gram English language model trained on the AFP and Xin-

[13]This segmentation was carried out by the author.
[14]http://github.com/redpony/cdec/tree/master/compound-split/de/

Table 4.3: Training corpus statistics.

	f-tokens	f-types	Eng. tokens	Eng. types
German (surface)	38M	307k	40M	96k
German (1-best)	40M	136k	”	”
Hungarian (surface)	25M	646k	29M	158k
Hungarian (1-best)	27M	334k	”	”
Turkish (surface)	1.0M	56k	1.3M	23k
Turkish (1-best)	1.1M	41k	”	”

hua portions of the Gigaword v3 corpus (Graff et al., 2007) with modified Kneser-Ney smoothing (Kneser and Ney, 1995) was used. The training, development, and test data for German-English and Hungarian-English systems used were distributed as part of the 2009 EACL Workshop on Machine Translation,[15] and the Turkish-English data corresponds to the training and test sets used in the work of (Oflazer and Durgar El-Kahlout, 2007).

Since the grammar induction procedure for hierarchical phrase-based translation models requires aligned sentence pairs (not lattices), I used two variants of the corpus to learn the grammar: the original, unsegmented corpus and a variant created by extracting the maximum weight segmentation under the model. This is designated the 1-BEST variant. Corpus statistics for all language pairs are summarized in Table 4.3. In all language pairs, the 1-BEST segmentation variant of the training data results in a significant reduction in types.

Word alignment was carried out by running Giza++ implementation of IBM Model 4 initialized with 5 iterations of Model 1, 5 of the HMM aligner, and 3 iterations of Model 4 (Och and Ney, 2003) in both directions and then symmetrizing using a heuristic technique

[15]http://www.statmt.org/wmt09

(Koehn et al., 2003). For each language pair, the corpus was aligned twice, once in its non-segmented variant and once using the single-best segmentation variant.

The MERT algorithm (§3.1.4) was used to tune the feature weights of the translation model on a held-out development set so as to maximize an equally weighted linear combination of BLEU and 1-TER (Papineni et al., 2002; Snover et al., 2006). The weights were independently optimized for each language pair and each experimental condition.

4.4.3 Max-marginal pruning

Since the run-time of the lattice SCFG-based decoder is cubic in the number of nodes in the input lattice, I used max-marginal pruning (also called forward-backward pruning) to remove low-probability edges (as predicted by the segmentation model) from the input lattices before translation. Max-marginal pruning is a technique to remove edges from a WFST or WSCFG whose marginal weight under the tropical semiring (hence 'max' because of the semiring's addition operator), is some factor away from its best path or derivation. Recall that edge marginals are computed using the INSIDEOUTSIDE algorithm (§2.4). The pruning factor, α, may be constant or it may be selected for each lattice so as to ensure a particular edge density. For the experiments reported here, a constant α was used, whose value was determined on the development set.

This pruning technique was originally introduced as a means of reducing the size of recognition lattices in automatic speech recognition (Sixtus and Ortmanns, 1999), but has found applications in parsing and machine translation (Huang, 2008). Max-marginal pruning is particularly useful because although the decision to prune each edge is only

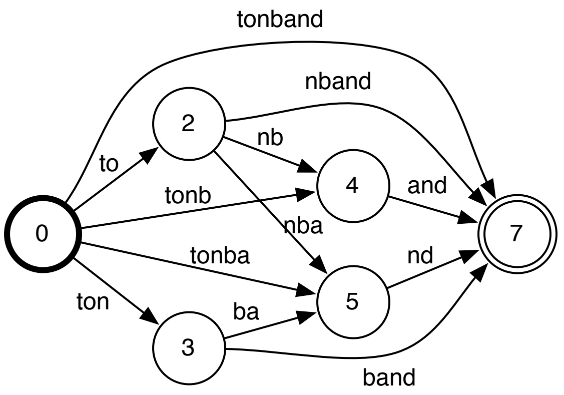


Figure 4.7: A full segmentation lattice (WFST) as of the word *tonband* with a minimum segment length of 2.

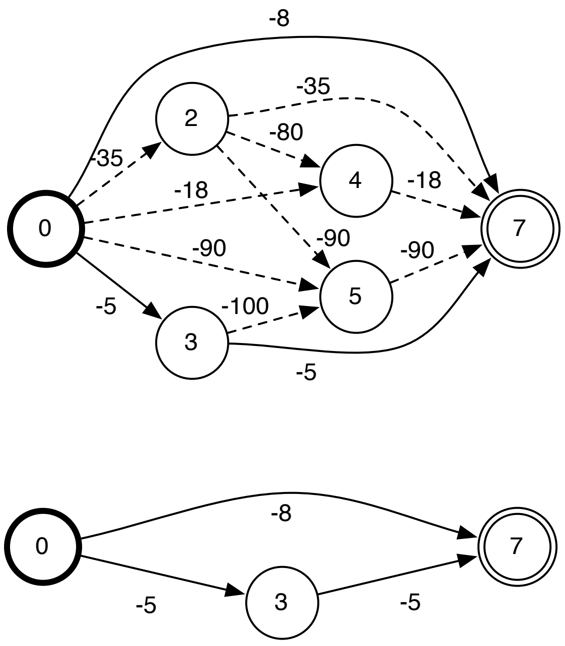


Figure 4.8: (Above) possible max marginals for the lattice in Figure 4.7; paths more than 10 away from the best path are dashed; (below) lattice after max-marginal pruning.

made looking at that edge's marginal weight, after pruning, a path from start state to final state is guaranteed to remain. Figure 4.7 illustrates all possible segmentations of the word *tonband*, with a minimum segment length of 2, and Figure 4.8 shows a possible max-marginal weighting of the edges and example result of pruning.

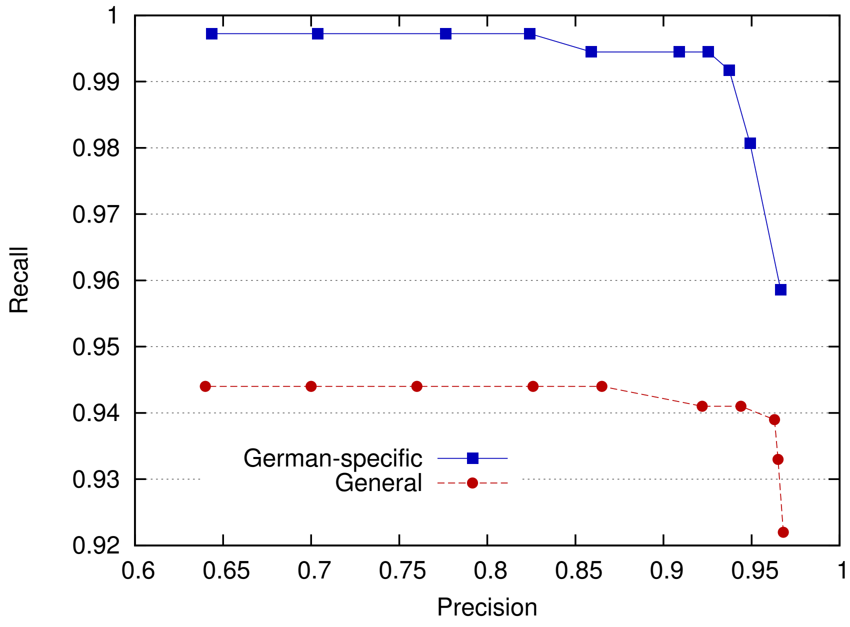


Figure 4.9: The effect of the lattice density parameter on precision and recall.

4.4.4 Intrinsic segmentation evaluation

To give some sense of the model's ability to generate lattices, I present precision and recall of segmentations for max-marginal pruning parameters ranging from $\alpha = 0$ to $\alpha = 5$ as measured on a held-out test set. Precision measures the number of paths in the hypothesized lattice that correspond to paths in the reference lattice; recall measures the number of paths in the reference lattices that are found in the hypothesis lattice. Figure 4.9 shows the effect of manipulating the density parameter on the precision and recall of the German lattices. Note that very high recall is possible; however, the German-only features have a significant impact, especially on recall, because the reference lattices include paths where *Fugenelemente* have been deleted. These can only be reached when the German-specific deletions are supported by the model.

4.4.5 Translation experiments

In this section, I report the results of experiments to verify that the segmentation lattices constructed using the CRF model trained with reference segmentation lattices yield better translations than either an unsegmented baseline or a baseline consisting of a single-best (maximum weight) segmentation.

For each language pair, let three conditions be defined: BASELINE, 1-BEST, and LATTICE. In the BASELINE condition, a lowercased and tokenized (but not segmented) version of the test data is translated using the grammar derived from a non-segmented training data. In the 1-BEST condition, the single best segmentation $\mathbf{y}$ that maximizes $\Lambda \cdot \vec{H}(\mathbf{x}, \mathbf{y})$ is chosen for each word $\mathbf{x}$ using the MERT-trained model (the German model for German, and the language-neutral model for Hungarian and Turkish). This variant is translated using a grammar induced from a parallel corpus that has also been segmented in the same way. In the LATTICE condition, segmentation lattices were constructed using the semi-CRF model and then pruned. For all languages pairs, $\alpha = 2.3$ was used as the pruning density parameter (which corresponds to the highest F-score on the held out test set). Additionally, if the unsegmented form of the word was removed from the lattice during pruning, it was restored to the lattice with a weight of 1.

Table 4.4 summarizes the results of the translation experiments comparing the three input variants. For all language pairs, substantial improvements are seen in both BLEU and TER when segmentation lattices are used. Additionally, these experiments also confirm previous findings that showed that when a large amount of training data is available, moving to a one-best segmentation does not yield substantial improvements (Yang

Table 4.4: Translation results for German-English, Hungarian-English, and Turkish-English. Scores were computed using a single reference and are case insensitive.

Source	condition	BLEU	TER
German	(baseline)	21.0	60.6
German	(1-best)	20.7	60.1
German	(lattice)	**21.6**	**59.8**
Hungarian	(baseline)	11.0	71.1
Hungarian	(1-best)	10.7	70.4
Hungarian	(lattice)	**12.3**	**69.1**
Turkish	(baseline)	26.9	61.0
Turkish	(1-best)	27.8	61.2
Turkish	(lattice)	**28.7**	**59.6**

and Kirchhoff, 2006). Perhaps most surprisingly, the improvements observed when using lattices with the Hungarian and Turkish systems were *larger* than the corresponding improvement in the German system, but German was the only language for which segmentation training data was available. The smaller effect in German is probably due to there being more in-domain training data in the German system than in the (otherwise comparably sized) Hungarian system.

Targeted analysis of the translation output shows that while both the *1-best* and *lattice* systems generally produce adequate translations of compound words that are out of vocabulary in the BASELINE system, the LATTICE system performs better since it recovers from infelicitous splits that the one-best segmenter makes. For example, one class of errors that are frequently observed is that the one-best segmenter splits an OOV proper name into two pieces when a portion of the name corresponds to a known word in the source language (e.g. *tom tancredo* $\rightarrow$ *tom tan credo* which is then translated as *tom tan belief*).[16]

[16]We note that one possible solution for this problem would be to incorporate a feature indicating whether

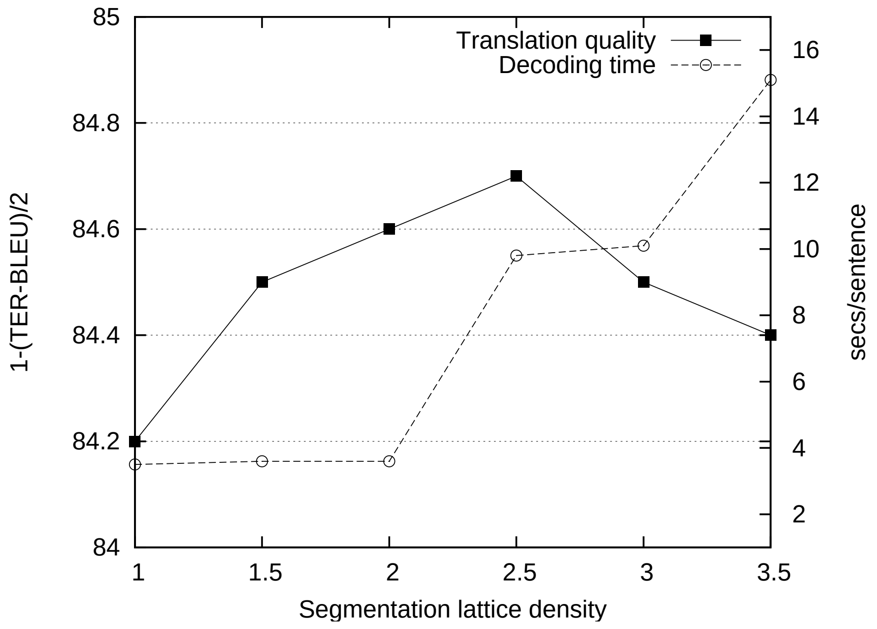


Figure 4.10: The effect of the lattice density parameter on translation quality and decoding time.

The effect of pruning on translation. Figure 4.10 shows the effect of manipulating the α parameter used in max-marginal pruning on the performance and decoding time of the Turkish-English translation system.[17] It further confirms the hypothesis that increased diversity of segmentations encoded in a segmentation lattice can improve translation performance; however, it also shows that once the density becomes too great, and too many implausible segmentations are included in the lattice, translation quality will be harmed. Thus, pruning is not only helpful for improving decoding efficiency, but important for quality.

a named entity is being segmented into the semi-CRF model.

[17]Turkish-English was used for this experiment because the BASELINE and LATTICE systems have the largest difference in scores of any of the language pairs.

4.5 Related work

The related problem of learning from a data set where there is a prior distribution over possible labels has been explored quite extensively (Dawid and Skene, 1979; Jin and Ghahramani, 2002; Smyth et al., 1994; Wiebe et al., 1999). With the rise of the so-called 'human computation paradigm' (von Ahn, 2006) there has been renewed interest in the problems associated with learning from uncertain labels and determining how accurate training data must be to facilitate learning (Dredze et al., 2009; Snow et al., 2008).

Sutton et al. (2007) describes how linear chain CRFs can be composed and trained jointly, leading to improved performance on a number of tasks where information from one classifier is used as a feature in a downstream classifier. This can be approximately understood as supervised training where there is a distribution over the *inputs* side of the pairs in the training data (whereas in this chapter the case where there is a distribution over the *outputs* was considered). Their approach relies on WFST composition to perform inference over the product of multiple CRFs.

Learning to segment words from unbroken text is a well-studied problem, in the context of both supervised and unsupervised learning. Tseng et al. (2005) describe a CRF segmenter for Chinese text, which Andrew (2006) shows can be improved with a hybrid linear CRF / semi-CRF model that is capable of using features that depend on the full word, not just characters and local boundary labels. While segmenters are often intended to match the segmentations described in style guides (for details, see Sproat and Emerson (2003)), some more recent work has attempted to learn segmentation for use in specific tasks. Chang et al. (2008) adapt the parameters of a CRF Chinese segmenter so as to maximize performance in Chinese-English machine translation, and Chung and Gildea (2009) describe an (unsupervised) Bayesian semi-HMM that learns word segmentations (for a variety of East Asian languages) tailored to the machine translation task from otherwise unannotated parallel corpora.

Koehn and Knight (2003) describes a heuristic approach to segmenting compound words in German that is based on the frequency with which hypothesized segments are found as free-standing tokens in large corpora. Based on this observation, they propose a model of word segmentation that splits compound words into pieces found in the dictionary using heuristic scoring criteria. A weakness of this approach is that the hypothesized segments their model produces must be found in their segmenter's dictionary.

4.6 Future work

The model described here can easily be generalized to incorporate a prior over label distributions, as proposed by Dredze et al. (2009) for the purpose of learning from noisy data. Therefore the technique described in this chapter is useful in the context of learning when training data is both noisy and there are multiple correct labels. This opens up a number of useful possibilities. Some tasks, like Chinese word segmentation (Sproat and Emerson, 2003), which are inherently ambiguous, are typically annotated with the help of a *style guide* that attempts to formulate rules that annotators can use to resolve ambiguities while labeling. Rather than attempting to create an exact style guides, one can simply ask a large number of unskilled annotators to segment the words based on their intuitions, which would produce a distribution over segmentations that could be used to train a model. Furthermore, training will be efficient since, unlike Dredze et al. (2009), the ‘reference distribution’ is encoded a compact WFST.

Alternative training objectives. Conditional random fields can be understood as maximum entropy models where a probabilistic structured prediction model is chosen such that its (Shannon) entropy is maximized but the expected values of feature functions match the empirical values of those functions in a set of training data (Gong and Xu, 2007).[18] That is, given a set vector of feature functions $\vec{H}(\mathbf{x},\mathbf{y})$, and a set of training data $\mathcal{D} = \{\mathbf{x}^j,\mathbf{y}^j\}_{j=1}^{\ell}$, the principle of maximum entropy says to select a model p^* from among all possible models $\mathcal{P}$ that predict an element from $\mathcal{Y}$ given an element from $\mathcal{X}$, according to the following constrained optimization problem:

$$\begin{aligned} p^* &= \arg\max_{p\in\mathcal{P}} \sum_{j=1}^{|\mathcal{D}|} \mathbf{H}(p(\mathbf{y}'|\mathbf{x}^j)) \\ & \text{s.t.} \sum_{j=1}^{|\mathcal{D}|} \mathbb{E}_{p(\mathbf{y}'|\mathbf{x}^j)} H_k(\mathbf{x}^j,\mathbf{y}') = \sum_{j=1}^{|\mathcal{D}|} H_k(\mathbf{x}^j,\mathbf{y}^j) \qquad \forall k \in [1,m] \end{aligned}$$

With the introduction of the ambiguity in references, the empirical feature value becomes an expectation under a distribution $q(\mathbf{y}'|\mathbf{x},Y^j)$:

[18]The Shannon entropy of a distribution p is $\mathbf{H}(p)$ is the negative expectation of the negative log probability under the distribution with $0\cdot\log 0 = 0$ (Cover and Thomas, 2006).

$$\begin{aligned} p^* &= \arg\max_{p \in \mathcal{P}} \sum_{j=1}^{|\mathcal{D}|} \mathbf{H}(p(\mathbf{y}'|\mathbf{x}^j)) \\ &\text{s.t.} \sum_{j=1}^{|\mathcal{D}|} \mathbb{E}_{p(\mathbf{y}'|\mathbf{x}^j)} H_k(\mathbf{x}^j, \mathbf{y}') = \sum_{j=1}^{|\mathcal{D}|} \mathbb{E}_{q(\mathbf{y}'|\mathbf{x}, Y^j)} H_k(\mathbf{x}^j, \mathbf{y}') \qquad \forall k \in [1, m] \end{aligned}$$

Specifically, I use EM, defining $q(\mathbf{y}'|\mathbf{x}, Y^j)$ to be the posterior weighting of a reference under the model (§5.2). Thus, whereas in the traditional maximum likelihood formulation, the constraints are fixed, in the latent variable case, they depend on the model parameters! Solving this problem amounts to the identification of a stable point where the entropy is maximized and the constraints are met.

Is this a good training criterion? While the empirical results clearly indicate it is useful, several things militate against it. First, the training objective does not care about the shape of q—there are absolutely no constraints on it (other than those that apply to all probability distributions). In the case of learning segmentation models from reference lattices, a solution could possibly be found that assigns all probability mass to a single segmentation in the reference lattice. This seems undesirable: reference segmentation lattices are defined to contain *all good segmentations*: the model should not get to decide to effectively disregard some of them. A better objective would ensure that some probability mass is distributed over all paths in the reference. Second, the objective is non-convex, meaning that in general, any solution may be a local optimum, and arbitrarily far from the global minimum that is sought. Although this does not appear to have been a problem in the segmentation model explored here, this could be a different story when modeling other linguistic phenomena.

Fortunately, both objections can be addressed by changing the form of q. One possible alternative definition of q that may be more appropriate is just a uniform distribution over all strings in Y^j, that is:

$$q(\mathbf{y}'|\mathbf{x},Y^j) = \frac{1}{|L(Y^j)|}$$

Not only does this objective ensure that some probability mass is assigned to every path in the reference lattice at training time, using a uniform q has other advantages. Most important, the optimization problem becomes globally convex again, meaning that local optima will not be a problem during optimization. Second, the 'empirical' term in the gradient becomes independent of Λ, meaning it can be computed just once, potentially reducing the computational effort required to train the model. Alternatively, changing the posterior can be understood as a form of posterior regularization, which is proposed by Ganchev et al. (2009) as general technique for biasing how models with latent variables are learned by imposing constraints on the posterior distributions over latent variables during learning.

Future work will explore the differences in training objectives on the word segmentation task. Additionally, tasks that have a latent variable over derivations (such as parsing (Clark and Curran, 2004), or translation (Blunsom et al., 2008a)) may also benefit from alternative definitions of q.

4.7 Summary

I have described a learning objective that permits multiple references encoded compactly in a FST to be used in the training of a semi-Markov conditional random field and demonstrated that this technique can be used effectively to generate segmentation lattices for input to a translation system. By using dense, linguistically motivated features, I was able to learn a model from a very small amount of training data that not only performs well, but generalizes to typologically similar languages. Furthermore, while the task of generating the reference lattices was rather minimally specified, it was not difficult to execute since rather than having to use some arbitrary criterion to resolve annotation ambiguities, all possibly labels were utilized, since they were 'correct' from the model's perspective.

5 Context-free representations of ambiguity

The girl the man the boy saw kissed left.

–John Kimball (1973)

In the previous two chapters, I showed that replacing single input values with weighted sets of alternatives improves translation, and that replacing single output labels with sets of correct output labels can be used effectively in learning. To avoid having to enumerate all the elements in the sets when performing inference, a (non-recursive) WFST was utilized to represent the content of the sets compactly, taking advantage of common substructure found across elements. In this chapter, the problem of translation from a set of possible inputs is revisited, focusing on the case where the input possibilities can be represented efficiently using a *context-free* structure (§2.2.2).[1]

As was mentioned in the overview of machine translation given in Chapter 3, translation models based on synchronous context free grammars (SCFGs) have become widespread in recent years (Chiang, 2007; Wu, 1997; Zollmann and Venugopal, 2006). One reason for their popularity is that SCFG models have the ability to search large numbers of reordering patterns in space and time that is polynomial in the length of the displacement, whereas an FST must generally explore a number of states that is exponen-

[1]This chapter contains material published originally in Dyer and Resnik (2010).

tial in this length.[2] As one would expect, for language pairs with substantial structural differences (and thus requiring long-range reordering during translation), SCFG models have come to outperform the best FST models (Zollmann et al., 2008). Targeted analysis indicates that these context-free models improve translation quality specifically by dealing more effectively with mid- and long-range reordering phenomena than phrase-based (WFST) models do (Birch et al., 2009).

In this chapter, I introduce a new way to take advantage of the computational benefits of CFGs during translation. Rather than using a single SCFG as the translation model, which both reorders and translates a source sentence into the target language, the translation process is factored into a two step pipeline where (1) the source language is reordered into a target-like order, with alternatives encoded in a context-free forest, and (2) the reordered source is transduced into the target language using an FST that represents phrasal correspondences.

While multi-step decompositions of the translation problem that take advantage of the closure properties of compositions of WFSTs have been proposed before (Kumar et al., 2006), such models are less practical with the rise of SCFG models, since the context free languages are not closed under composition (§2.2.3). However, the CFLs are closed under composition with regular languages. By using only a *finite state* phrase transducer and representing reorderings of the source in a *context free* forest, inference over the composition of the two models is not only decidable, but tractable. In particular, the context free reordering forest can be composed with a WFST phrase transducer

[2]The focus here is the reordering made possible by varying the arrangement of the translation units, not the local word order differences captured inside memorized translation pairs.

using the top-down composition algorithm (§2.3.2.1). The result of this composition is a WSCFG of translations, the same as encountered in translation with WSCFG-based translation models, and it can be intersected with an n-gram target language model using standard techniques (Huang and Chiang, 2007).

This chapter proceeds as follows. In the next section (§5.1), reordering forests are introduced. Since in translation it is necessary to discriminate between good reorderings of the source and bad ones, I how show to reweight the edge weights in the reordering forest by treating them as latent variables in an end-to-end translation model (§5.2). Then experimental results on language pairs requiring both small and large amounts of reordering are presented (§5.3). The chapter concludes with a discussion of related work (§5.4) and future work (§5.5).

5.1 Reordering forests

In this section, I describe *source reordering forests*, a WCFG representation of source language word order alternatives. The basic idea is that for the source sentence, $\mathbf{f}$, that is to be translated, a (monolingual) context-free grammar $\mathcal{F}$ will be created that generates strings ($\mathbf{f}'$) of words in the source language that are permutations of the original sentence. Specifically, this forest should contain derivations that put the source words into an order that approximates how they will be ordered in the grammar of the target language.

For a concrete example, consider the task of English-Japanese translation.[3] The input sentence is *John ate an apple*. Japanese is a head-final language, where the heads

[3]English is used here as the source language since the parse structure of English sentences is expected to be more familiar.

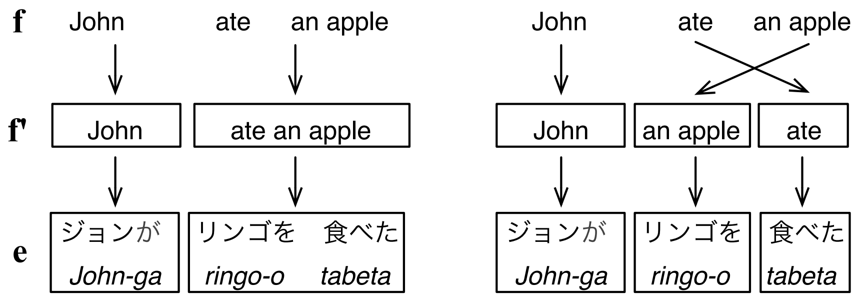


Figure 5.1: Two possible derivations of a Japanese translation of an English source sentence.

of phrases (such as the verb in a verb phrase) typically come last, and English is a *head-initial language*, where heads come first. As a result, the usual order for a declarative sentence in English is SVO (subject-verb-object), but in Japanese, it is SOV, and the desired translation into Japanese is *John-ga ringo-o* [an apple] *tabeta* [ate]. In summary, when translating from English into Japanese, it is usually necessary to move verbs from their position between the subject and object to the end of the sentence.

This reordering can happen in two ways, which is depicted in Figure 5.1. In the derivation on the left, a memorized phrase pair captures the movement of the verb (Koehn et al., 2003). In the other derivation, the source is first reordered into target word order and then translated, using smaller translation units. In addition, it is assumed that the phrase translations were learned from a parallel corpus that is in the original ordering, so the reordering forest $\mathcal{F}$ should include derivations of phrase-size units in the source order as well as the target order.

A minimal reordering forest that supports the derivations depicted needs to include both an SOV and SVO version of the source. This could be accomplished trivially with the following grammar:

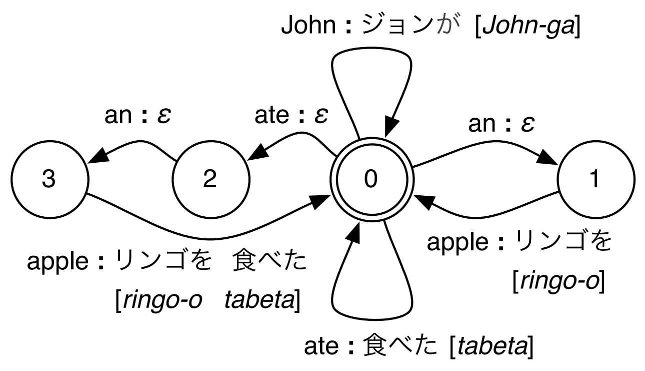

John : ジョンが [*John-ga*]
an : ε
ate : ε
an : ε
3
2
0
1
apple : リンゴを 食べた [*ringo-o tabeta*]
apple : リンゴを [*ringo-o*]
ate : 食べた [*tabeta*]

Figure 5.2: A fragment of a phrase-based English-Japanese translation model, represented as an FST. Japanese romanization is given in brackets.

$$
\begin{aligned}
\text{S} &\rightarrow \textit{John ate an apple} \\
\text{S} &\rightarrow \textit{John an apple ate}
\end{aligned}
$$

However, this grammar misses the opportunity to take advantage of the regularities in the permuted structure. A better alternative might be:

$$
\begin{aligned}
\text{S} &\rightarrow \textit{John}\ \text{VP} \\
\text{VP} &\rightarrow \textit{ate}\ \text{NP} \\
\text{VP} &\rightarrow \text{NP}\ \textit{ate} \\
\text{NP} &\rightarrow \textit{an apple}
\end{aligned}
$$

In this grammar, the phrases *John* and *an apple* are fixed and only the VP contains ordering ambiguity.

5.1.1 Reordering forests based on source parses

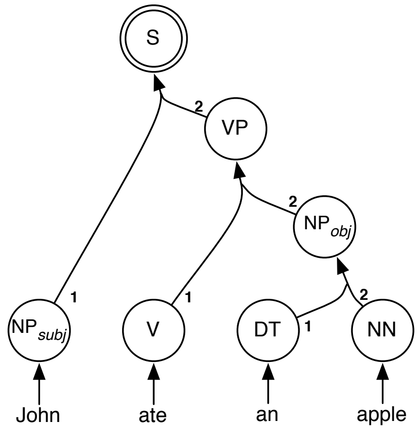
Many kinds of reordering forests are possible. In general, the best one for a particular language pair must fulfill two criteria. It must be easy to create given the resources available in the source language, and it will also be one that compactly expresses the source reorderings that are most likely to be useful for translation. In this chapter, the focus is on one possible kind of reordering forest that is inspired by the reordering model of Yamada and Knight (2001).[4] These are generated by taking a source language parse tree and 'expanding' each node so that it rewrites with different permutations of its children. For computational tractability, all permutations are included in the grammar only when the number of children of a node in the input parse is less than 5, otherwise permutations where any child moves more than 4 positions away from where it starts are excluded.

For an illustration using the example sentence, refer to Figure 5.3 for the forest representation and Figure 5.4 for its isomorphic CFG representation. It is easy to see that this forest generates the two 'good' order variants from Figure 5.1; however, the forest includes many other derivations that will not lead to good translations.

Figure 5.5 shows an example of a source reordering forest that is then composed with a finite state transducer transduction model, using the top-down algorithm (§2.3.2.1) to perform the composition. The output of the translation process thus has the same structure—a WSCFG—as the output of a standard WSCFG-based translation model (compare with Figure 3.3, which shows the translation forest output of a hierarchical phrase-

[4]One important difference is that this translation model is not restricted by the structure of the source parse tree; i.e., phrases used in transduction need not correspond to constituents in the source reordering forest. However, if a phrase does cross a constituent boundary between constituents A and B, then translations that use that phrase will have A and B adjacent.

Original parse:



Reordering forest:

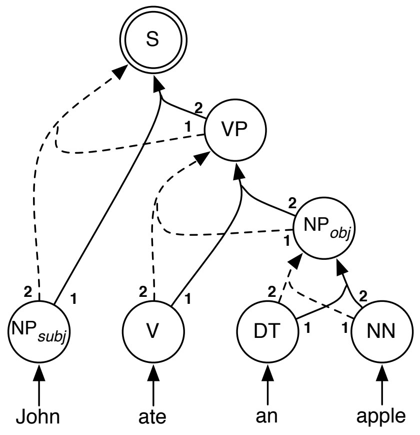


Figure 5.3: Example of a reordering forest. Linearization order of non-terminals is indicated by the index at the tail of each edge. The isomorphic CFG is shown in Figure 5.4; dashed edges correspond to reordering-specific rules.

based translation model). This forest has three derivations: two yielding the correct head-final Japanese word order, and one where the VP is in (incorrect) head-initial order. In the next section, a technique for learning the weights of the edges (i.e., rewrite rules) in the reordering forest is described. This technique seeks to rank correct translations into the target language more highly than incorrect ones. In the given example, to get the proper output word order where *tabeta* comes after *ringo-o*, there are two possibilities. Either the head-final rule S $\rightarrow$ NP V should have a higher weight than the head-initial

Original parse grammar:

$$\begin{array}{rcl} \mathrm{S} & \rightarrow & \mathrm{NP_{subj}\ VP} \\ \mathrm{VP} & \rightarrow & \mathrm{V\ NP_{obj}} \\ \mathrm{NP_{obj}} & \rightarrow & \mathrm{DT\ NN} \\ \mathrm{NP_{subj}} & \rightarrow & \textit{John} \\ \mathrm{V} & \rightarrow & \textit{ate} \\ \mathrm{DT} & \rightarrow & \textit{an} \\ \mathrm{NN} & \rightarrow & \textit{apple} \end{array}$$

Additional reordering grammar rules:

$$\begin{array}{rcl} \mathrm{S} & \rightarrow & \mathrm{VP\ NP_{subj}} \\ \mathrm{VP} & \rightarrow & \mathrm{NP_{obj}\ V} \\ \mathrm{NP_{obj}} & \rightarrow & \mathrm{NN\ DT} \end{array}$$

Figure 5.4: Context free grammar representation of the forest in Figure 5.3. The reordering grammar contains the parse grammar, plus the reordering-specific rules.

rule S $\rightarrow$ V NP, or, (alternatively) the phrase translation of *ate an apple* should be more highly weighted than the component translations, which enables the head-initial English word order to be translated by a memorized phrase pair.

5.1.2 What about finite-state equivalents?

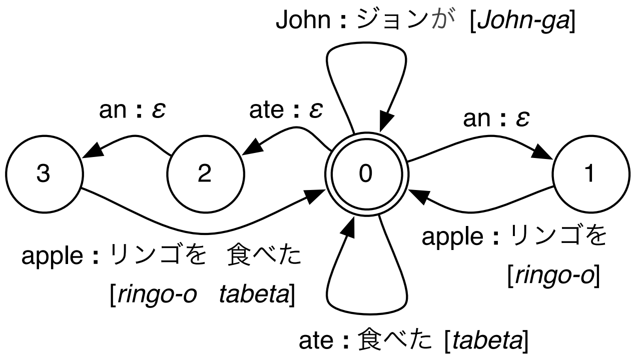
Before looking at how to learn a reordering model, I consider one possible objection to the translation process proposed here. Because reordering forests as defined are non-recursive CFGs, there must be an FSA which defines exactly the same set of strings as generated by the CFG (Hopcroft and Ullman, 1979). Since previous chapters have demonstrated that WFSTs may be used as input to the translation process, one might naturally wonder why bother to come up with a method to translate non-recursive WCFGs when the approaches described in previous chapters are apparently available. I argue that a finite-state

f – input sentence: *ate an apple*

$\mathcal{F}$ – identity WSCFG reordering forest (weights not shown; start symbol is S):

$$\begin{array}{rcl} \mathrm{S} & \rightarrow & \mathrm{V\ NP} \\ \mathrm{S} & \rightarrow & \mathrm{NP\ V} \end{array} \qquad \begin{array}{rcl} \mathrm{V} & \rightarrow & \textit{ate} \\ \mathrm{NP} & \rightarrow & \textit{an apple} \end{array}$$

G – WFST translation model:



$\mathcal{F} \circ G = (G^{-1} \circ \mathcal{F}^{-1})^{-1}$ – translation forest (start symbol is ${}_0\mathrm{S}_0$):

$$\begin{array}{rcl} {}_0\mathrm{S}_0 & \rightarrow & {}_0\mathrm{V}_0\ {}_0\mathrm{NP}_0 \\ {}_0\mathrm{S}_0 & \rightarrow & {}_0\mathrm{V}_2\ {}_2\mathrm{NP}_0 \\ {}_0\mathrm{V}_0 & \rightarrow & \textit{tabeta} \end{array} \qquad \begin{array}{rcl} {}_0\mathrm{V}_2 & \rightarrow & \varepsilon \\ {}_2\mathrm{NP}_0 & \rightarrow & \textit{ringo-o tabeta} \\ {}_0\mathrm{NP}_0 & \rightarrow & \textit{ringo-o} \end{array}$$

Figure 5.5: Example reordering forest translation forest.

representation of a reordering forest is impractical for two reasons: model parameterization and the size required to model long-range reordering patterns in a WFST. First, a WCFG can quite easily be parameterized using features that are quite natural for modeling word ordering divergences between two languages, as will be demonstrated below (§5.3.1). While there would surely be many useful finite-state features as well, the second objection is more serious. Constructing a WFST equivalent of a reordering WCFG may require, in the worst case, an exponential number of states compared to the number of non-terminals in the WCFG (Pereira and Wright, 1991). Moreover, this bound is likely to be problematic for exactly the kinds of grammars that should be considered when modeling word order alternatives in translation. The following example grammar illustrates the problem with finite-state representations of order alternatives.

$$\begin{aligned} \text{S} &\rightarrow \text{AdvP X} \mid \text{X AdvP} \\ \text{AdvP} &\rightarrow \textit{allegedly} \\ \text{X} &\rightarrow \textit{John robbed the bank} \end{aligned}$$

This CFG representation of two order possibilities is quite natural: there is exactly one representation of the phrase *John robbed the bank* in the grammar, and the adverbial phrase is permitted to locate on either side. In the equivalent (minimal) finite-state representation of this set of strings, it would be necessary to encode the phrase *John robbed the bank* twice, once when it occurs to the right of an AdvP and once when the AdvP has yet to occur. Since, in the general case X may be any length, and this kind of alternation is exactly the sort of variation that reordering forests are likely to contain when modeling word order divergences between languages, this poses a serious problem.

5.2 Modeling

A reordering forest derives many different target strings corresponding to different permutations of the input sentence. Some of these permutations, when translated using the lexical and phrasal transduction operations possible in the phrasal WFST, will lead to translations that have the correct word order (and word choice) and others will be incorrect (incorrect target word order or incorrect lexical choice). The model should distinguish these two classes such that reordering forest derivations leading to well-ordered translations have a higher weight than those which are unlikely to lead to well-ordered translations.

If a corpus of source language sentences paired with 'reference reorderings' were available, such a model could be learned directly as a relatively straightforward supervised learning task. However, creating the optimal target-language reordering $\mathbf{f}'$ for some $\mathbf{f}$ is a nontrivial task, even if when taking advantage of the ability to have multiple correct references using the techniques from Chapter 4.[5] Instead of trying to model reordering directly, I opt to treat the reordered form of the source, $\mathbf{f}'$, as a *latent variable* in a translation model, and to train the reordering model and translation model jointly. Using this approach, only a parallel corpus of translations is required to learn the reordering model. Not only does the latent variable approach obviate the necessity of creating problematic 'reference reorderings', but it is also intuitively satisfying because from a task perspec-

[5]For a discussion of methods for generating reference reorderings from automatically word aligned parallel corpora, refer to Tromble and Eisner (2009).

tive, the values of $\mathbf{f}'$ are not of interest, the task is only to produce a good translation $\mathbf{e}$.

5.2.1 A probabilistic translation model with a latent reordering variable

The forest-reordering translation model is a two phase process. First, source sentence $\mathbf{f}$ is reordered into a target-like word order $\mathbf{f}'$ according to a reordering model $r(\mathbf{f}'|\mathbf{f})$. The reordered source is then transduced into the target language according to a translation model $t(\mathbf{e}|\mathbf{f}')$. The requirement that $r(\mathbf{f}'|\mathbf{f})$ can be represented by a non-recursive WCFG, i.e. a forest as in §5.1.1, is imposed, and also that $t(\mathbf{e}|\mathbf{f}')$ can be represented by a (cyclic) finite state transducer, as in Figure 5.2.

Since the reordering forest may define multiple derivations $\mathbf{a}$ from $\mathbf{f}$ to a particular $\mathbf{f}'$, and the transducer may define multiple derivations $\mathbf{d}$ from $\mathbf{f}'$ to a particular translation $\mathbf{e}$, these variables are treated as latent and marginalized out to define the probability of a translation given the source:

$$p(\mathbf{e}|\mathbf{f}) = \sum_{\mathbf{d}} \sum_{\mathbf{f}'} t(\mathbf{e}, \mathbf{d}|\mathbf{f}') \sum_{\mathbf{a}} r(\mathbf{f}', \mathbf{a}|\mathbf{f}) \tag{5.1}$$

Crucially, since $r(\mathbf{f}'|\mathbf{f})$ is assumed to have the form of a non-recursive WSCFG and $t(\mathbf{e}|\mathbf{f}')$ that of a cyclic WFST (which has no epsilons in its output), the quantity (5.1), which sums over all reorderings (and derivations), can be computed using the top-down, left-to-right composition algorithm (§2.3.2.1) and then the INSIDE algorithm (§2.4.1). The WSCFG that results from the composition is likewise guaranteed to be non-recursive, meaning that the INSIDE algorithm can be utilized.

5.2.2 Conditional training

While it is straightforward to use expectation maximization to optimize the (marginal) joint likelihood of the parallel training data with a latent variable model, a log-linear parameterization trained to maximize conditional likelihood offers more flexibility (Blunsom et al., 2008a; Petrov and Klein, 2008). Thus, the CRF training criterion described in the previous chapter (§4.1) is reused here. Note that even though a single, unambiguous reference translation is assumed, it is nevertheless necessary to perform marginalization during computation of the empirical feature expectations: training requires summing over the different derivations (corresponding to different permutations of the source sentence and different decompositions into phrases) that lead to the target translation. Using log-linear parameterization enables a rich set of (possibly overlapping, non-independent) features to be used to discriminate among translations. The probability of a derivation from source to reordered source to target is thus written in terms of model parameters $\Lambda = \langle \lambda_1, \lambda_2, \ldots, \lambda_m \rangle$ as:

$$p(\mathbf{e}, \mathbf{d}, \mathbf{f}', \mathbf{a} | \mathbf{f}; \Lambda) = \frac{\exp \sum_k \lambda_k \cdot H_k(\mathbf{e}, \mathbf{d}, \mathbf{f}', \mathbf{a}, \mathbf{f})}{Z(\mathbf{f}; \Lambda)}$$

$$\text{where } H_k(\mathbf{e}, \mathbf{d}, \mathbf{f}', \mathbf{a}, \mathbf{f}) = \sum_{r \in \mathbf{d}} h_k(\mathbf{f}', r) + \sum_{s \in \mathbf{a}} h_k(\mathbf{f}, s)$$

The derivation probability is globally normalized by the partition function $Z(\mathbf{f}; \Lambda)$, which is just the sum of the numerator for all derivations of $\mathbf{f}$ (corresponding to any $\mathbf{e}$). As in the previous chapters, the H_k are real-valued feature functions that may be overlapping and non-independent. For computational tractability, it is assumed that the feature functions H_k decompose additively with the derivations of $\mathbf{f}'$ and $\mathbf{e}$ in terms of *local* feature functions

h_k. Details about the features used in the parameterization are discussed below (§5.3.1). Also define $Z(\mathbf{e},\mathbf{f};\lambda)$ to be the sum of the numerator over all derivations that yield the sentence pair $\langle \mathbf{e},\mathbf{f} \rangle$. A spherical Gaussian prior on the value of Λ with mean $\mathbf{0}$ and variance $\sigma^2 = 1$ is also used, which helps prevent overfitting of the model, as discussed in the previous chapter (§4.1.1). The training objective is thus to select Λ minimizing:

$$\begin{aligned} \mathcal{L} &= -\log \prod_{\langle \mathbf{e},\mathbf{f} \rangle} p(\mathbf{e}|\mathbf{f};\Lambda) - \frac{||\Lambda||_2^2}{2\sigma^2} \\ &= -\sum_{\langle \mathbf{e},\mathbf{f} \rangle} [\log Z(\mathbf{e},\mathbf{f};\Lambda) - \log Z(\mathbf{f};\Lambda)] - \frac{||\Lambda||_2^2}{2\sigma^2} \end{aligned} \tag{5.2}$$

The gradient of $\mathcal{L}$ with respect to the feature weights has a parallel form; it is the difference in feature expectations under the reference distribution and the translation distribution with a penalty term due to the prior:

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \left(\sum_{\langle \mathbf{e},\mathbf{f} \rangle} \mathbb{E}_{p(\mathbf{d},\mathbf{a}|\mathbf{e},\mathbf{f};\Lambda)}[h_k] - \mathbb{E}_{p(\mathbf{e},\mathbf{d},\mathbf{a}|\mathbf{f};\Lambda)}[h_k] \right) - \frac{\lambda_k}{\sigma^2} \tag{5.3}$$

Computing the objective and gradient. The objective and gradient that were just introduced can be computed in two steps, by constructing appropriate WSCFGs and then using standard inference algorithms (§2.4.1). For clarity, I describe precisely how to do so here. Note that this is essentially the same two step construction described by the two-parse algorithm (§2.3.4) and used to compute the objective and gradient when training with reference lattices (§4.2; Figure 4.3).

1. Given a training pair $\langle \mathbf{e},\mathbf{f} \rangle$, generate the WSCFG of reorderings $\mathcal{F}$ from $\mathbf{f}$ as described in §5.1.1.

2. Compose this grammar with T, the WFST representing the phrasal translation model, using the top-down composition algorithm (§2.3.2.1) which yields $\mathcal{F} \circ T$,[6] a translation forest that contains all possible translations of **f** into the target language, using any possible permutation of **f** in the reordering forest.

3. Run the INSIDE algorithm on $\mathcal{F} \circ T$ to compute $Z(\mathbf{f};\Lambda)$, the first term in the objective, and run the INSIDEOUTSIDE algorithm to compute $\mathbb{E}_{p(\mathbf{e},\mathbf{d},\mathbf{a}|\mathbf{f})}[h_i]$.

4. Compute $Z(\mathbf{e},\mathbf{f};\Lambda)$ and the first expectation in the gradient by finding the subset of the translation forest $\mathcal{F} \circ T$ that exactly derives the reference translation **e**. To do this, rely on the fact that $\mathcal{F} \circ T$ is a WSCFG. Since, by construction, it is non-recursive, and the reference string **e** is also recursive, composition can be performed with the bottom-up composition algorithm (§2.3.2.2). The resulting forest, $\mathcal{F} \circ T \circ \mathbf{e}$, contains all and only derivations that yield the pair $\langle \mathbf{e},\mathbf{f} \rangle$. On this forest, the INSIDE algorithm computes $Z(\mathbf{e},\mathbf{f};\Lambda)$ and the INSIDEOUTSIDE algorithm can be used to compute $\mathbb{E}_{p(\mathbf{e},\mathbf{d},\mathbf{a}|\mathbf{f})}[h_i]$.

5. Compute $\mathcal{L}$ using Equation (5.2) and its gradient using Equation (5.3).

With an objective and gradient, any first-order numerical optimization technique can be applied. For the experiments reported below, L-BFGS was used (Liu and Nocedal, 1989). Although the conditional likelihood surface of this model is non-convex (on account of the latent variables), no obvious initialization effect was noted; therefore, initial

[6]To be precise, the inversion theorem (§2.1.3.5) is used, computing $\mathcal{F} \circ T$ as $(T^{-1} \circ \mathcal{F}^{-1})^{-1}$ since the presentation of the algorithms assumes the context-free operand in a composition operation is the right-most element.

values of $\Lambda = \mathbf{0}$ were used, with $\sigma^2 = 1$. For all models, training converged to a local minimum in fewer than 1,500 function evaluations.[7]

5.3 Experiments

I now turn to an experimental validation of the reordering and translation models introduced in the preceding section. The behavior of the model is evaluated in three conditions: a small data scenario consisting of a translation task based on the BTEC Chinese-English corpus (Takezawa et al., 2002), a large data Chinese-English condition designed to be more comparable to conditions in a NIST MT evaluation, and a large data Arabic-English task.

The WFST phrase translation model was constructed from phrase tables extracted as described in Koehn et al. (2003) with a maximum phrase size of 5 (see also §3.1.2.1). The parallel training data was aligned using the Giza++ implementation of IBM Model 4 (Och and Ney, 2003). Chinese text was segmented using a CRF-based word segmenter, optimized to produce output compatible with the Chinese Treebank segmentation standard (Tseng et al., 2005). The Arabic text was segmented using the technique described in Lee et al. (2003). The Stanford parser was used to generate parses for all conditions, and these were then used to generate the reordering forests as described in §5.1.1.

Table 5.1 summarizes statistics about the corpora used. The reachability statistic indicates what percentage of sentence pairs in the training data could be regenerated us-

[7]Other optimization techniques may converge much more rapidly, such as stochastic gradient descent (Bottou, 1998); however, L-BFGS was preferred since its internal representation of the curvature of the objective surface will detect inconsistent calculations of the objective and gradient, which is quite useful for verifying the correctness of the model implementation.

ing the reordering/translation model.[8] Since L-BFGS required over 1,000 function and gradient evaluations for convergence, and each requires a full pass through the training data, the full set of reachable sentences was not used to train the reordering model, except in the small BTEC corpus. Instead, a randomly selected 20% of the reachable set in the Chinese-English condition, and all reachable sentence pairs under 40 words (source) in length in the Arabic-English condition were used.[9]

Error analysis indicates that a substantial portion of unreachable sentence pairs are due to alignment (word or sentence) or parse errors; however, in some cases the reordering forests did not contain an adequate source reordering to produce the necessary target. For example, in Arabic, which is a VSO language, the treebank annotation is to place the subject NP as the 'middle child' between the V and the object constituent. This can be reordered into an English SVO order using child-permutation; however, if the source VP is modified by a modal particle, the parser makes the particle the parent of the VP, and it is no longer possible to move the subject to the first position in the sentence. Richer reordering rules are needed to address this. Other solutions to the reachability problem include targeting reachable oracles instead of the reference translation (Li and Eisner, 2009) or making use of alternative training criteria, such as minimum risk training, which do not require being able to exactly reach a target translation (Li and Khudanpur, 2009).

[8]When training to maximize conditional likelihood, only sentences that can be generated by the model can be used in training.

[9]Despite using a reduced training data size, both the qualitative analysis of what is learned by the reordering model and the quantitative analysis of the translation results suggest that the reordering model is not undertrained.

Table 5.1: Corpus statistics

Condition	Sentences	Source words	Target words	Reachability
BTEC	44k	0.33M	0.36M	81%
Chinese-English	400k	9.4M	10.9M	25%
Arabic-English	120k	3.3M	3.6M	66%

5.3.1 Reordering and translation features

Since the conditional random field training criterion is used, it is straightforward to use numerous sparse features to parameterize the model. For the WFST translation component, the typical features used in translation are included: relative phrase translation frequencies $p(\overline{e}|\overline{f})$ and $p(\overline{f}|\overline{e})$, 'lexically smoothed' translation probabilities $p_{lex}(\overline{e}|\overline{f})$ and $p_{lex}(\overline{f}|\overline{e})$, and a phrase count feature. For the reordering model, a binary feature for each kind of rule used, for example $\phi_{\mathrm{VP}\rightarrow\mathrm{V\ NP}}(\mathbf{a})$ fires once for each time the rule VP $\rightarrow$ V NP was used in a derivation, $\mathbf{a}$. For the Arabic-English condition, it was observed that the parse trees tend to be quite flat, with many repeated non-terminal types in one rule, so the non-terminal types were augmented with an index indicating where they were located in the original parse tree. This resulted in a total of 6.7k features for IWSLT, 18k features for the large Chinese-English condition, and 516k features for Arabic-English (there were many more due to the splitting of the non-terminals).

A target language model was not used during the training of the source reordering model, but it was used during the translation experiments (see below).

5.3.2 Qualitative assessment of reordering model

Before looking at the performance of the model on a translation task, it is illuminating to look at what the model learns during training. Figure 5.6 lists the 10 most highly weighted reordering features learned by the BTEC model (above) and shows an example reordering using this model (below), with the most English-like reordering indicated with a star.[10] Keep in mind, it is expected that these features will reflect what the best *English-like* order of the input should be. All are quite intuitive, but this is not terribly surprising since Chinese and English have very similar large-scale structures (both are head initial, both have adjectives and quantifiers that precede nouns). However, two entries in the list (starred) correspond to an English word order that is ungrammatical in Chinese: PP modifiers in Chinese typically precede the VPs they modify, and CPs (relative clauses) also typically precede the nouns they modify. In English, the reverse is true, and the model has learned to prefer this ordering. It was not necessary that this be the case: since the training procedure makes use of phrases memorized from a non-reordered training set, it could have relied on those for all its reordering. These weights suggest that large-scale reordering patterns are being successfully learned.

5.3.3 Translation experiments

The section considers how to apply this model to a translation task. The maximum conditional likelihood training described in §5.2.2 is suboptimal for state-of-the-art translation systems, since (1) it optimizes likelihood rather than an MT metric and (2) it does not

[10]The italicized symbols in the English gloss are functional elements with no precise translation. Q is an interrogative particle, and DE marks a variety of attributive roles and is used here as the head of a relative clause.

Feature	λ	note
VP → VE NP	0.995	
VP → VV VP	0.939	modal + VP
VP → VV NP	0.895	
VP → VP PP*	0.803	PP modifier of VP
VP → VV NP IP	0.763	
PP → P NP	0.753	
IP → NP VP PU	0.728	PU = punctuation
VP → VC NP	0.598	
NP → DP NP	0.538	
NP → NP CP*	0.537	rel. clauses follow

Input:

我	能	赶上		去	西尔顿	饭店	的	巴士	吗	?
I	CAN	CATCH	[$_{NP}$[$_{CP}$	GO	HILTON	HOTEL	***DE***]	BUS]	***Q***	?

(Can I catch a bus that goes to the Hilton Hotel ?)

5-best reordering:

I CAN CATCH [$_{NP}$ BUS [$_{CP}$ GO HILTON HOTEL ***DE***]] ***Q*** ?

★ I CAN CATCH [$_{NP}$ BUS [$_{CP}$ ***DE*** GO HILTON HOTEL]] ***Q*** ?

I CAN CATCH [$_{NP}$ BUS [$_{CP}$ GO HOTEL HILTON ***DE***]] ***Q*** ?

I CATCH [$_{NP}$ BUS [$_{CP}$ GO HILTON HOTEL ***DE***]] CAN ***Q*** ?

I CAN CATCH [$_{NP}$ BUS [$_{CP}$ ***DE*** GO HOTEL HILTON]] ***Q*** ?

Figure 5.6: (Above) The 10 most highly-weighted features in a Chinese-English reordering model. (Below) Example reordering of a Chinese sentence (with English gloss, translation, and partial syntactic information).

include a language model. However, despite these shortcomings, the fact that exact inference was tractable made it a compelling starting point, enabling learning to start from the completely uniform distribution that occurs when $\Lambda = \mathbf{0}$.[11] To address these limitations, the maximum conditional likelihood model will be taken as a starting point for further training with a more task-appropriate objective.

[11]The approximate inference techniques suggested by Li et al. (2009b) or training using expected BLEU training (Li and Eisner, 2009) are problematic to utilize when $\Lambda = \mathbf{0}$, since they rely implicitly on the best-first heuristic search of cube pruning, which fails when all derivations have the same weight.

5.3.3.1 Training for Viterbi decoding with MERT

To be competitive with other state-of-the-art translation systems, it is useful to be able to optimize the model using Och's minimum error training algorithm (§3.1.4) to optimize BLEU on a development set; however, the model as described above cannot be used since it contains far too many features and weights. Therefore, the weights assigned to the sparse reordering features obtained using the maximum conditional likelihood training described above (§5.2.2) into a single 'dense' reordering feature, which then has a single weight assigned to it tuned against the other translation features (relative frequencies of phrase translations, lexical translation probabilities, etc.). Once the reordering weights have been collapsed into a single feature, the coefficient for this feature can be learned using the MERT algorithm (implemented using the upper envelope semiring) to optimize the remaining weights together with this new feature so to maximize BLEU of the maximum-weight derivation a held-out development set.

During this second training phase, a language model was incorporated using cube pruning (Huang and Chiang, 2007). To improve the ability of the model to match reordered phrases, the 1-best reordering of the training data under the learned reordering model was extracted and phrases that translate from the 1-best reordering into the target were extracted using the standard phrase extraction heuristic and used to supplement the translation WFST.

5.3.3.2 Translation results

Scores on a held-out test set are reported in Table 5.2 using case-insensitive BLEU with 4 reference translations (16 for BTEC) using the original definition of the brevity penalty (Papineni et al., 2002). The results of the new forest reordering model along with three standard baseline conditions are presented, one with no-reordering at all (mono), the performance of a phrase-based (PB) translation model with distance-based distortion, the performance of a hierarchical (Hiero) phrase-based translation model (Chiang, 2007), and then the forest-reordering model (CFG+FST).

Table 5.2: Translation results (BLEU)

Corpus	Mono	PB	Hiero	CFG+FST
BTEC	47.4	51.8	52.4	**54.1**
Chinese-English	29.0	30.9	32.1	**32.4**
Arabic-English	41.2	45.8	**46.6**	44.9

In the two Chinese conditions (BTEC and NIST Chinese-English), the forest reordering model attains or surpasses the baseline of the a hierarchical phrase-based translation mode. However, the model performs less well on Arabic-English translation. These results are unsurprising for two reasons. First, the forest-reordering model considers reordering patterns from local all the way to full sentence reordering. While in Chinese-English translation mid- to long-range reordering is often necessary (Birch et al., 2009), Arabic tends to require more local reordering during translation. Thus, these results confirm previous findings that show that models that support local reordering but only limited (or entirely forbidden) long-range reordering outperform models capable of support long-range reordering on Arabic-English translation (Zollmann et al., 2008).

Condition	Example output
CFG+FST	Mankind has a total of 23 pairs of chromosomes.
Hiero	A total of 23 pairs of chromosomes of human beings.
Reference	Human beings have a total of 23 pairs of chromosomes.
CFG+FST	Australian foreign minister will not be able to achieve more aid to North Korea's act of bad behavior
Hiero	Australian foreign minister to bad behavior will not be able to achieve more aid to North Korea
Reference	Australian foreign minister says North Korea will not get more aid for disgusting acts
CFG+FST	This plan will provide tax preferential treatment to the broad working masses.
Heiro	The preferential tax cut plan will provide broad working masses.
Reference	The plan will offer preferential tax-cuts to the ordinary people.
CFG+FST	The United States against a dozen allies support for Iraq
Heiro	The US war against Iraq, there are more than ten allies support
Reference	The US war with Iraq wins support from a dozen allied countries

Figure 5.7: Four example outputs from the Chinese-English CFG+FST and hierarchical phrase-based translation (Hiero) systems.

Figure 5.7 compares example outputs from the forest-reordering system (CFG+FST) and the hierarchical phrase-based translation system (Hiero). Examples were selected by examining sentences with fewer than 20 words, and the first four that had order differences were chosen. In 3 of the 4 cases, the CFG+FST system achieves a reasonable ordering. However, the fourth, where the order of entities are mangled is typical of the errors seen in the CFG+FST: elements move 'too easily', and end up in the wrong spot during translation. This problem of excessive reordering is doubly problematic in Arabic, where long-range movement is, as a rule, less often required. Figure 5.8 provides example translations. While the CFG+FST system does managed to deal properly with the notoriously problematic VSO$\rightarrow$SVO reordering in the third and fourth examples, in the first two, elements have been permuted inappropriately.

Condition	Example output
CFG+FST	That is scheduled to appear before the court four tomorrow, Monday.
Hiero	It is expected that the four appear before the court tomorrow, Monday.
Reference	It is scheduled that the four will stand before the court tomorrow (Monday).
CFG+FST	“Iraqna” mobile phone confirm the release of the two officials about security.
Hiero	“Iraqna” mobile phone to confirm the release of two of the security officials.
Reference	“Iraqna” mobile phone company confirms the release of its two security officers.
CFG+FST	The French Embassy in Baghdad maintaining silence regarding the fate of journalists.
Hiero	Abide by the French Embassy in Baghdad silence regarding the fate of journalists .
Reference	The French Embassy in Baghdad is remaining silent about the fate of the two journalists.
CFG+FST	On the final peace agreement for south Sudan signed in Nairobi.
Hiero	Signed in Nairobi on the final peace agreement for south Sudan.
Reference	Final peace accord on southern Sudan signed in Nairobi.

Figure 5.8: Four example outputs from the Arabic-English CFG+FST and hierarchical phrase-based translation (Hiero).

5.3.4 Model complexity

The difference in size of the FST+CFG translation models (which are just WFSTs) and hierarchical phrase-based translation models (§3.1.2.2) is also an interesting comparison. Table 5.3 compares the number of unique phrasal translations of the hierarchical phrase-based translation models with the number of distinct phrasal translations encoded in the WFST translation table used in the translation experiments reported in the previous section.

Table 5.3: Number of translations in the WFST model vs. rules in the hierarchical phrase-based WSCFG model.

Corpus	WFST rules	WSCFG rules
BTEC	0.26M	3.1M
Chinese-English	7.4M	43M
Arabic-English	2.9M	33M

Table 5.3 makes clear that the WFSTs in the CFG+FST model are much smaller than the corresponding hierarchical phrase-based translation grammars (and yet they attain comparable or better translation performance). Furthermore, the WFSTs were constructed from two variants of the parallel training data—one in the original order and one under the 1-best reordering of the reordering model, whereas the WSCFG model contains rules extracted from a single variant of the corpus (the original word order). Although these size statistics do not completely reflect the full complexity of the translation models (the CFG+FST model depends on a source language parser with its own grammar which was not considered at all here), the tendency is clear. Furthermore, the relative sizes are expected. A hierarchical phrase-based translation model can be understood as the

composition of a reordering model and a phrasal translation model. Thus the translation model *must* be much more complicated, since every reordering pattern for every different translation pair must have its own rule. Furthermore, each combination will have its own parameters, making data sparsity a potentially more serious issue when estimating hierarchical phrase-based translation models than when estimating the WFSTs for the CFG+FST translation model.

5.4 Related work

A variety of translation processes can be formalized as the composition of a finite state representation of input (typically just a sentence, but often a more complex structure, like a word lattice) with an SCFG (Chiang, 2007; Dyer et al., 2008; Wu, 1997; Zollmann and Venugopal, 2006). Like these, this work uses parsing algorithms to perform the composition operation, but to my knowledge, for the first time, the *input* has a context free structure.[12] Although not described in terms of operations over formal languages, the model of Yamada and Knight (2001) can be understood as an instance of this class of model with a specific input forest and phrases restricted to match syntactic constituent boundaries.

Syntax-based preprocessing approaches that have relied on hand-written rules to restructure source trees for particular translation tasks have been quite widely used (Chang et al., 2009; Collins et al., 2005; Wang et al., 2007; Xu et al., 2009). Discriminatively trained reordering models have also been extensively explored. A widely used approach

[12]Satta (submitted) discusses the theoretical possibility of this sort of model but provides no experimental results.

has been to use a classifier to predict the orientation of phrases during decoding (Chang et al., 2009; Zens and Ney, 2006). However, these classifiers must be trained independently from the translation model using training examples extracted from the word aligned data. While sparse features are most often used, Setiawan et al. (2009) shows that a bigram model over function word pairs can improve reordering.

A more ambitious approach to learning reordering models is described by Tromble and Eisner (2009), who build a global reordering model that is learned automatically from reordered training data. They also rely on parsing-like algorithms to explore an exponential number of reorderings of an input sentence. However, at decoding time they only consider a single-best reordering from their model. During training they iterate the parsing-like search algorithm so as to locate permutations that are unreachable in a single step. Somewhat surprisingly, their best results on a translation task are obtained if they only run a single iteration of the model.

The discriminative training of the latent variable model is based on the techniques proposed by Blunsom et al. (2008a,b). However, in their model translation and reordering is learned jointly, whereas the model proposed in this chapter factors these two components.

Huang and Mi (2010), which was developed concurrently to this work, describe a top-down, left-to-right translation algorithm for tree-to-string translation that includes a language model in the first decoding pass. This algorithm could be utilized to decode with the translation model presented here, making a second-pass cube-pruning step unnecessary.

5.5 Future work

The work described in this chapter can be extended in many ways. A few of the most promising possibilities here are discussed here.

Reordering forests (§5.1) can be constructed in virtually any way, not just by permuting the children of nodes identified in the 1-best output of constituency parsers, as was done in the experiments in this chapter. One natural extension, in the spirit of the models from Chapter 3, is to consider a forest of outputs from a parser, which would enable the reordering model to recover from parse errors made in the 1-best output from a statistical parser.

The work of Tromble and Eisner (2009) also suggests another alternative: abandon supervised parse structures entirely, and use a forest that recursively either directly orders or inverts adjacent spans that are determined by other means. The ITG structure that was used by Tromble and Eisner (2009), which starts with single word phrases that are merged to form larger spans, is a straightforward starting point. However, larger units determined by other means (mutual information, relative entropy, etc.) could also furnish a starting segmentation. Alternatively the parse trees generated by unsupervised parsing techniques could be used.

Another somewhat orthogonal line of work is to use dependency parses (Mel′čuk, 1988), rather than constituency parses, as the basis for the reordering forest. This could be accomplished in a relatively straightforward manner simply by using CFG encoding of a dependency parse tree (the simplest possible encoding consists of grammar rules $H \rightarrow CH$ for left attaching children and $H \rightarrow HC$ for right attaching children). The starting dependency parses could be obtained using either supervised or unsupervised parsers. However, since the quality of unsupervised dependency parsers has improved significantly in recent years (Cohen and Smith, 2009; Cohen et al., 2010; Headden III et al., 2009; Klein and Manning, 2004), a wholly unsupervised forest reordering model based on unsupervised dependency parsing is a particularly promising avenue of research. Unlike many other applications of parse trees in NLP, the parse structures used in the forest reordering model need only be useful for defining a space of plausible reorderings. Since the original word order is maintained, even if they are linguistically unusual, a reasonable translation may still be found.

One potential limitation of the model presented here is that it has no ability to reorder phrases after translation. Except for the reordering captured by memorized phrase pairs, any reordering considered by the model must be found in the input reordering forest. Since reordering is crucial for effective translation, providing *multiple* opportunities to reorder during translation seems preferable. From this perspective, the approach taken by Tromble and Eisner (2009) is preferable to the one taken here: they use a context-free reordering model to select a 1-best reordering of the source words, and then they translate this using a phrase-based translation model that is capable of performing further reordering (note that since they are selecting only a 1-best output, they could use any translation model). Therefore, one logical extension of this work is to add a post-translation reordering step that permits reordering beyond what is represented in the input forest. This could take advantage of finite-state phrase-based reordering models, such as those suggested by Kumar et al. (2006).

More ambitious still would be an approach that would enable a WCFG reordering forest to be translated with a WSCFG translation model. Li et al. (2009b) describe how variational techniques can be used to approximate the distribution over strings found in WCFG forest with a WFST. As showed earlier (§5.1.2), the exact WFST equivalent of a reordering forest that contains long-distance reordering patterns often becomes intractably large. However, variational techniques provide a principled way of selecting a WFST that is of manageable size but has a distribution over strings that is close to the distribution defined by the reordering forest. While the equivalent WFSTs will only be approximations of the reordering forest, this approach holds a number of intriguing possibilities.[13]

5.6 Summary

In this chapter, I showed how to translate input structured as a WCFG using a WFST translation model. As a demonstration of this technique's practical value, I described a technique for constructing 'reordering forests', based on the permutation of the children of nodes of source language parse trees, in a manner reminiscent of the model of Yamada and Knight (2001), but in which phrases were not restricted to line up with syntactic constituents, and in which a log-linear parameterization was used, rather than a stochastic one. The resulting 'forest reordering translation model' obtains quite competitive performance in Chinese-English translation, a language pair where reordering problems are a significant barrier to generating fluent target translations (Birch et al., 2009; Zollmann et al., 2008).

[13] I thank Jason Eisner for this suggestion.

6 Conclusion

A computer beat me in chess, but it was no match when it came to kickboxing.

–Emo Philips

One of the central problems in natural language processing—if not *the* central problem—is resolving ambiguity. This problem reappears in numerous specific instances: interpreting the sequence of words underlying a speech signal, the parse tree or logical form of a sentence, or a meaning-preserving translation of a text from one language into another. This dissertation has argued that when modeling language processing, standard computational models are often too hasty at committing to an analysis, and that it is better to preserve ambiguity for as long as possible, thereby incorporating as much knowledge across as possible, and only making a decision in response to the input at the latest possible moment.

The contributions of this dissertation can be grouped into three broad areas: formal foundations, machine learning, and applications. In the area of formal foundations, I developed a general model of ambiguity processing that subsumed two common approaches to machine translation (phrase based (Koehn et al., 2003) and hierarchical phrase based (Chiang, 2007)), extending them to support multiple inputs into the translation process. These models made use of a new, general WFST-WSCFG composition algorithm.

In the area of machine learning, I presented a new formulation of the ‘hypergraph’ minimum error rate training (MERT) algorithm in terms of the well-known INSIDE algorithm and a novel semiring (the upper envelope semiring). I also introduced a generalization of CRF training to deal with multiple references (for a single training example) compactly encoded in a FST.

I also presented several applications of the general ambiguity-preserving processing model and learning innovations. Across a variety of language pairs and domains, deferring ambiguity resolution in machine translation systems improved the translation quality. This was true not only when translating the output of noisy preprocessing components (like speech recognizers), but even when translating text inputs. Furthermore, these improvements held regardless of the form of the translation model (whether finite-state or context-free). I also demonstrated that my generalization of CRF training could be applied to the problem of learning a word segmentation model for translation. Finally, I showed this way of thinking about information processing could be used to motivate a new translation model that decomposed the problem into two independent components. Not only did this model attain state-of-the-art performance or better, but the models were far smaller than comparable hierarchical phrase based translation models.

6.1 Future work

Each chapter in this dissertation has discussed possible extensions to the individual topics covered. I now consider several more speculative, high-level extensions of this work.

Ambiguity propagation in cognitive modules. Modular architectures are not only useful in the design of natural language processing systems, but there is good evidence that cognitive faculties are also factored into largely independent modules that interact through narrow interfaces (Fodor, 1983; Steedman, 2000). In this dissertation, I have given evidence that pipelines of (artificial) modules perform more robustly (Chapter 3) and can consist of simpler components (Chapter 5) when they are able to propagate information about a distribution over possible outputs, rather than being forced to commit to a single output at each interface. Although it is quite different from the line of work I have pursued so far, an interesting extension of this project into a completely new area would seek to assess how adequate the ambiguity-preserving model is as a characterization interfaces between cognitive modules. After all, the same ambiguity that artificial systems encounter when processing language will be encountered by natural systems.

A potentially productive way of thinking about cognition in terms of ambiguity propagation is to note that the output representations of upstream modules impose constraints on the computational systems that make use of them: the encoding of the ambiguous alternatives must be computationally 'compatible' with the modules that make use of them. To take an example from the dissertation, since reordering forests (§5.1) are context free, they cannot be used as input to a context-free translation model. This suggests that, in addition to looking for particular behavioral evidence that ambiguity is propagated between cognitive modules, there may be evidence in the form of the computational complexity of the processing tasks themselves.

One domain where the ambiguity-preserving hypothesis may shed some light concerns the interface between syntax and phonology. Phonological knowledge (for example, in the form of rules that relate phonemic representations to phonetic forms) appears to be expressible using nothing more computationally powerful than (W)FSTs (Beesley and Karttunen, 2003). In contrast, syntax seems to depend on computations that are at least context free. The presence of one context-free component in the system raises the question: why should phonology not be context free as well? It is, after all, a more powerful representation. One might think about this question as follows: if the output interface of some module produces a single, best-guess analysis, it is not obvious why there should be any restrictions on the computations used internally to individual modules. For example, the output of the module that applies a phone$\rightarrow$phoneme mapping would be the best guess phonemic form of the input—whether or not the module relied on a finite-state, context-free, or other computational device internally. However, if ambiguity is propagated between components in such a way as to *share structure* between alternative possibilities, then finding a context-free structured output as input to a context-free system would be *a priori* unlikely, on account of the complexity and decidability results discussed in Chapter 2. Thus, the ambiguity-preserving hypothesis may provide a framework for interpreting empirical facts about language in new ways.

Unsupervised learning from ambiguous data. In Chapter 4, I considered how supervised learning can be adapted when the training data contains ambiguous labels. However, there are many problems that are more naturally solved using unsupervised learning techniques, such as expectation maximization (Dempster et al., 1977) or nonparametric Bayesian inference (Jordan, 2005). For example, inducing WSCFG or WFST translation models from a parallel corpus can be treated as a problem where for an observed sentence pair, the alignment and segmentation of the input words into phrases are latent variables (Blunsom et al., 2009; DeNero et al., 2008). The obvious question asked by this work is: what if there is ambiguity in the sentence pairs? Since many unsupervised techniques are probabilistic in nature, the technique that I used in Chapter 4, dealing with ambiguity in supervised learning by marginalizing over the ambiguous alternatives, can be used as a starting point. Finally, since it has been argued that nonparametric Bayesian models are useful statistical models of cognition (Griffiths et al., 2008), and since real-world learning necessarily must deal with ambiguous inputs, the applications of unsupervised learning techniques when the observation variables are noisy could be diverse indeed. As one example application from the cognitive domain, the nonparametric Bayesian word learning model of Goldwater (2006) assumes an input consisting of unambiguous strings of phonemes. While this assumption was a reasonable starting point, one may wonder what affect there would be on the model if, rather than unambiguous inputs, the learner was presented with distributions over phoneme strings. One strand of future work will be to develop unsupervised learning models where ambiguity in the observations is explicitly modeled.

Meaning as a latent variable in translation. Before the advent of the statistical machine translation paradigm, machine translation was often carried out by constructing an *interlingual representation* from the input which was then used to generate output with the same meaning (Dorr, 1993). The interlingual model is quite appealing since it is structured around the preservation of meaning across languages, which is the goal of translation. However, translation systems based on interlingual approaches to translation have traditionally been much more difficult to develop than systems based on statistical approaches, requiring significant effort by expert annotators. On the other hand, statistical systems, while inexpensive to develop and capable of achieving quite impressive performance, nevertheless fail to incorporate any direct representation of *meaning*, which seems to be a liability in a task centered on the preservation of meaning. I argue that the ability to efficiently translate a set of sentences is one way to bring a concept of meaning into statistical translation—thereby leveraging the strengths of the interlingual approach—without incurring the costs traditionally associated with it. Specifically, future work will replace an single, unambiguous input sentence with a distribution over sentences representing alternative ways of expressing the same underlying meaning. Then, using the lattice translation techniques from Chapter 3, entire sets of sentences will be translated, marginalizing out their source realization. Generating alternative realizations of the underlying meaning of a sentence can then leverage work on automatic paraphrasing (Madnani and Dorr, 2010). Initial forays into this idea have been made by Resnik et al. (2010) as well as Onishi et al. (to appear 2010).

Other grammatical formalisms. The general model of ambiguity processing introduced in Chapter 2 was used exclusively with WFST- and WSCFG-based models. Another extension of this work will consider other grammatical formalisms, such as TAG, which has already begun to be used for modeling translational equivalence (Abeillé et al., 1990; Carreras and Collins, 2009; DeNeefe and Knight, 2009), CCG, and tree-transducer based formalisms (Graehl et al., 2008). Not only will this enable a verification of how well the general ambiguity processing approach works with other formalisms, but it will provide a common mathematical language with which to discuss a variety of grammar and transducer formalisms.

Annotation. Another area of research enabled by the work in Chapter 4 on learning from ambiguous labels concerns better strategies for annotation for the purposes of supervised learning. Annotation is typically carried out by developing a style guide that is used to train annotators how to label example inputs. In addition to being a training manual, style guides have traditionally been important because they have been used to create consistency when there would otherwise be ambiguity about the 'correct' annotation of some example. Recent work has begun to demonstrate that many redundant non-expert annotators can perform as well as a few highly trained annotators on many annotation tasks, but at lower costs (Snow et al., 2008). However, this previous work still tends to focus on a head-to-head comparison on annotation tasks where there is a single correct label—that is, where a style guide exists and can be used to determine what the ideal annotation is. The multi-label paradigm, where the model discriminates bad labels from good labels, but where there may be many good labels, suggests a different annotation paradigm based on annotator intuitions rather than style guide adherence. For example, rather than instructing an annotator to label the segmentation of a Chinese sentence (which, as I discussed in Chapter 4, is an inherently ambiguous problem) according to a highly detailed style guide, the annotator might be instructed to label all possible segmentations he believes are reasonable based on a higher level characterization of the task. The goal of the annotation task is thus to capture the *intuitions* of human annotators about a particular phenomenon, which may vary between annotators, and—quite importantly—where multiple annotations may exist for each instance to be labeled. This has the potential to simplify the complexity of developing style guides (which is expensive itself, and larger style guides makes it more difficult to train annotators); although it may come at the expensive of making it more difficult to detect poor annotators who are either deliberately or inadvertently not carrying out the annotation task.

Propagating ambiguity further with 'fuzzy' human interfaces. This dissertation has provided evidence that deferring the resolution of ambiguity in machine translation improves translation quality. However, at the end of the process, the evaluation methodology that was used, adhering to the standards of the field, requires that a single translation be selected as the output of a process. Furthermore, this is the standard interface used by translation systems: the user provides an input document and receives a translated output document. However, it is reasonable to assume the effectiveness of the interface between a translation system and a human consumer of translation output will also benefit from the propagation of ambiguity. That is, the translation interface should communicate ambiguity about the translations to the user. For example, when reading translation output, one may encounter a sentence that it is uninterpretable. Current translation interfaces offer very little opportunity for a consumer of MT output to deal with the failure of a system, other than going back to the source language. Being able to explore other translation hypotheses beyond the 1-best hypothesis under the model is likely to be one of the most helpful things one can do. Furthermore, a number of human evaluation methodologies that have been developed, such as having monolingual English speakers take reading comprehension tests based on MT output (Jones et al., 2005), could be easily adapted to deal with new ambiguity-preserving interface paradigms. This proposed work is in the spirit of the proposals by Church and Hovy (1993), who argued that machine translation—even at 1993 levels—already held value for end-users, but that the technology needed to be exploited differently than a translation agency would be. Viewed in these terms, the value of the technology today is certainly much higher and holds much greater possibilities. The success of the work of Albrecht et al. (2009), Koehn (2010), and the work presented at the NIST 2005 Machine Translation Evaluation by Callison-Burch,[1] who used sophisticated interfaces to help users improve translation output and assist in the translation process, indicates that fuzzy interface designed for the acquisition of information from foreign language sources holds considerable potential.

[1]http://www.itl.nist.gov/iad/mig/tests/mt/2005/doc/mt05eval_official_results_release_20050801_v3.html

Bibliography

Anne Abeillé, Yves Schabes, and Aravind K. Joshi. Using lexicalized tags for machine translation. In *Proceedings of the 13th Conference on Computational Linguistics (COLING)*, volume 3, pages 1–6, 1990.

Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling, Volume 1: Parsing*. Prentice-Hall, Englewood Cliffs, NJ, 1972.

Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling, Volume 2: Compiling*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

Joshua Albrecht, Rebecca Hwa, and G. Elisabeta Marai. The Chinese room: Visualization and interaction to understand and correct ambiguous machine translation. *Computer Graphics Forum*, 28(3), 2009.

Cyril Allauzen and Mehryar Mohri. N-way composition of weighted finite-state transducers. *International Journal of Foundations of Computer Science*, 20(4):613–627, 2009.

Cyril Allauzen, Mehryar Mohri, and Brian Roark. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 40–47, 2003.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer, 2007. `http://www.openfst.org`.

James F. Allen. *Natural Language Understanding*. Benjamin Cummings, 1987.

Galen Andrew. A hybrid Markov/semi-Markov conditional random field for sequence segmentation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 465–472, Sydney, 2006.

Yehoshua Bar-Hillel, Micah Perles, and Eliyahu Shamir. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172, 1961.

K. Beesley and L. Karttunen. *Finite State Morphology*. CLSI Publications, 2003.

James O. Berger. *Statistical decision theory and Bayesian analysis*. Springer, 1985.

Nicola Bertoldi, Richard Zens, and Marcello Federico. Speech translation by confusion network decoding. In *Proceeding of ICASSP 2007*, Honolulu, Hawaii, April 2007.

Rajendra Bhatia. *Matrix Analysis*. Springer, 1996.

Alexandra Birch, Phil Blunsom, and Miles Osborne. A Quantitative Analysis of Reordering Phenomena. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 197–205, Athens, Greece, March 2009. Association for Computational Linguistics.

Phil Blunsom, Trevor Cohn, and Miles Osborne. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-HLT*, 2008a.

Phil Blunsom, Trevor Cohn, and Miles Osborne. Probalistic inference for machine translation. In *EMNLP*, 2008b.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. A Gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 782–790, Singapore, 2009.

Léon Bottou. *Online Algorithms and Stochastic Approximations*. Cambridge University Press, Cambridge, UK, 1998.

Pierre Boullier. Range concatenation grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 53–54, Trento, Italy, February 2000.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Razvan C. Bunescu. Learning with probabilistic features for improved pipeline models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 670–679, 2008.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. Findings of the 2009 workshop on statistical machine translation. In *Proceedings of Workshop on Statistical Machine Translation (WMT09)*, 2009.

Xavier Carreras and Michael Collins. Non-projective parsing for statistical machine translation. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 200–209, 2009.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, 2008.

Pi-Chuan Chang, Dan Jurafsky, and Christopher D. Manning. Disambiguating "DE" for Chinese-English machine translation,. In *Proceedings of the EACL 2009 Fourth Workshop on Statistical Machine Translation*, 2009.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, 1996.

J. Cheppalier, M. Rajman, R. Aragues, and A. Rozenknop. Lattice parsing for speech recognition. In *Sixth Conference sur le Traitement Automatique du Langage Naturel (TANL'99)*, pages 95–104, 1999.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2): 201–228, 2007.

Tagyoung Chung and Daniel Gildea. Unsupervised tokenization for machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, Singapore, 2009.

Kenneth W. Church and Eduard H. Hovy. Good applications for crummy machine translation. *Machine Translation*, 8:239–258, 1993.

Stephen Clark and James R. Curran. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 104–111, Barcelona, 2004.

Shay B. Cohen and Noah A. Smith. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

Shay B. Cohen, David M. Blei, and Noah A. Smith. Variational inference for adapter grammars. In *Proceedings of NAACL*, 2010.

Michael Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 184–191, 1996.

Michael Collins, Philipp Koehn, and Ivona Kucerova. Clause restructuring for statistical machine translation. In *Proceedings of ACL 2005*, 2005.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, pages 558–565. The MIT Press and McGraw-Hill Book Company, 1989.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, 2006.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, 28(1):20–28, 1979.

Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, second edition, 2000.

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 (Methodological)*, 39(1):1–38, 1977.

Steve DeNeefe and Kevin Knight. Synchronous tree adjoining machine translation. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 727–736, 2009.

John DeNero, Alex Bouchard-Côté, and Dan Klein. Sampling alignment structure under a Bayesian translation model. In *Proceedings of EMNLP*, 2008.

Steven J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14:31–39, 1988.

Mona Diab, Mahmoud Ghoneim, and Nizar Habash. Arabic diacritization in the context of statistical machine translation. In *Proceedings of the Machine Translation Summit (MT-Summit)*, 2007.

Bonnie J. Dorr. Interlingual machine translation: A parameterized approach. *Artificial Intelligence*, 63(1):429–492, 1993.

Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. Sequence learning from data with multiple labels. In *Proceedings of the 1st International Workshop on learning from Multi-Label Data (MLD09)*, 2009.

Manfred Droste and Werner Kuich. Semirings and formal power series. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science, pages 3–27. Springer, 2009.

Chris Dyer. Noisier channel translation: translation from morphologically complex languages. In *Proceedings of the Second Workshop on Statistical Machine Translation*, Prague, June 2007.

Chris Dyer. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT) 2009*, 2009.

Chris Dyer. Two monolingual parses are better than one (synchronous parse). In *Proceedings of NAACL*, 2010.

Chris Dyer and Philip Resnik. Context-free reordering, finite-state translation. In *Proceedings of NAACL*, 2010.

Chris Dyer, Smaranda Muresan, and Philip Resnik. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio, June 2008.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. `cdec`: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL (Demo session)*, Uppsala, Sweden, 2010.

John Earley. An efficient context-free parsing algorithm. *Communications of the Association for Computing Machinery*, 13(2):94–102, 1970.

Jason Eisner. Directional constraint evaluation in Optimality Theory. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 257–263, Saarbrücken, Germany, August 2000.

Jason Eisner, Eric Goldlust, and Noah A. Smith. Compiling comp ling: Weighted dynamic programming and the Dyna language. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 281–290, Vancouver, October 2005.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. Solving the problem of cascading errors: Approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2006.

Radu Florian, Hany Hassan, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. A statistical model for multilingual entity detection and tracking. In *Proceedings of HLT-NAACL 2004*, pages 1–8, 2004.

Jerry A. Fodor. *Modularity of Mind: An Essay on Faculty Psychology*. MIT Press, Cambridge, 1983.

Michel Galley and Christopher D. Manning. Accurate non-hierarchical phrase-based translation. In *Proceedings of NAACL*, 2010.

Giorgio Gallo, Giustino Longo, and Stefano Pallottino. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2):177–201, 1993.

Kuzman Ganchev, Jo ao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. Technical Report MS-CIS-09-16, University of Pennsylvania Department of Computer and Information Science, January 2009.

Sharon Goldwater. *Nonparametric Bayesian Models of Lexical Acquisition*. PhD thesis, Brown University, 2006.

Sharon Goldwater and David McClosky. Improving statistical MT through morphological analysis. In *Proceedings of HLT-EMNLP 2005*, pages 676–683, Vancouver, British Columbia, 2005.

Yihong Gong and Wei Xu. Maximum entropy model and conditional random field. In Borko Furht, editor, *Machine Learning for Multimedia Content Analysis*, chapter 9. Springer, 2007.

Joshua Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–605, 1999.

Jonathan Graehl, Kevin Knight, and Jonathan May. Training tree transducers. *Computational Linguistics*, 34(3):391–427, 2008.

D. Graff, J. Kong, K. Chen, and K. Maeda. English gigaword third edition, 2007.

Susan L. Graham, Walter L. Ruzzo, and Michael Harrison. An improved context-free recognizer. *ACM Trans. Program. Lang. Syst.*, 2(3):415–462, 1980.

Thomas L. Griffiths, Charles Kemp, and Joshua B. Tenenbaum. Bayesian models of cognition. In Ron Sun, editor, *The Cambridge handbook of computational cognitive modeling*. Cambridge University Press, 2008.

Dick Grune and Ceriel J. H. Jacobs. Parsing as intersection. In David Gries and Fred B. Schneider, editors, *Parsing Techniques*, pages 425–442. Springer, New York, 2008.

Nizar Habash and Fatiha Sadat. Arabic preprocessing schemes for statistical machine translation. In *Proceedings of NAACL*, pages 49–52, New York, 2006.

Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitch Marcus. Determining case in arabic: Learning complex linguistic behavior requires complex linguistic features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic, 2007.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. Better word alignments with supervised ITG models. In *Proceedings of ACL/IJCNLP*, pages 923–931, 2009.

Jan Hajič and Barbora Hladká. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the COLING-ACL 1998*, pages 483–490, 1998.

William P. Headden III, Mark Johnson, and David McClosky. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, 2009.

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

Liang Huang. *Forest-based Algorithms in Natural Language Processing*. PhD thesis, University of Pennsylvania, 2008.

Liang Huang and David Chiang. Better *k*-best parsing. In *In Proceedings of the 9th International Workshop on Parsing Technologies*, pages 53–64, 2005.

Liang Huang and David Chiang. Forest rescoring: Faster decoding with integrated language models. In *ACL*, 2007.

Liang Huang and Haitao Mi. Efficient incremental decoding for tree-to-string translation. In *In Proc. EMNLP*, 2010.

Liang Huang, Hao Zhang, Dan Gildea, and Kevin Knight. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4), 2009.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. Hierarchical phrase-based translation with weighted finite state transducers. In *Proceedings of NAACL*, pages 433–441, 2009.

Edwin T. Jaynes and G. Larry Bretthorst. *Probability theory: the logic of science*. Cambridge University Press, 2003.

Frederick Jelinek. *Statistical Methods for Speech Recognition (Language, Speech, and Communication)*. The MIT Press, January 1998.

Rong Jin and Zoubin Ghahramani. Learning with multiple labels. In *Advances in Neural Information Processing Systems 15 (NIPS)*, 2002.

Douglas Jones, Wade Shen, Neil Granoien, Martha Herzog, and Clifford Weinstein. Measuring translation quality by testing English speakers with a new Defense Language Proficiency Test for Arabic. In *Proceedings of the 2005 International Conference on Intelligence Analysis*, McLean, Virginia, 2005.

Michael I. Jordan. Dirichlet processes, Chinese restaurant processes and all that. Tutorial presentation at the NIPS Conference, Vancouver, 2005.

Dan Klein and Christopher D. Manning. Parsing with hypergraphs. In *Proceedings of IWPT 2001*, 2001.

Dan Klein and Christopher D. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *ACL '04: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485, 2004.

Reinhard Kneser and Hermann Ney. Improved backing-off for *m*-gram language modeling. In *Proceedings of IEEE Internation Conference on Acoustics, Speech, and Signal Processing*, pages 181–184, 1995.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2009.

Philipp Koehn. Enabling monolingual translators: Post-editing vs. options. In *Proceedings of NAACL*, 2010.

Philipp Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation models, 2004.

Philipp Koehn and Kevin Knight. Empirical methods for compound splitting. In *Proceedings of the EACL 2003*, pages 187–193, Budapest, 2003.

Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of NAACL*, pages 48–54, 2003.

Philipp Koehn, Abhishek Arun, and Hieu Hoang. Towards better machine translation quality for the German-English language pairs. In *ACL Workshop on Statistical Machine Translation*, 2008.

Terry Koo and Michael Collins. Hidden-variable models for discriminative reranking. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing*, pages 507–514, 2005.

Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Springer, 1985.

Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In *Processings of HLT-NAACL*, 2004.

Shankar Kumar, Yongang Deng, and William Byrne. A weighted finite state transducer translation template model for statistical machine translation. *Journal of Natural Language Engineering*, 12(1):35–75, 2006.

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 Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 163–171, 2009.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, San Francisco, 2001.

Lillian Lee. Fast context-free grammar parsing requires fast Boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15, 2002.

Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. Language model based Arabic word segmentation. In *ACL '03: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 399–406, 2003.

Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 1981.

P. M. Lewis, II and R. E. Stearns. Syntax-directed transduction. *Journal of the ACM*, 15 (3):465–488, 1968.

Zhifei Li. *Novel Inference, Training, and Decoding Methods over Translation Forests*. PhD thesis, The Johns Hopkins University, 2010.

Zhifei Li and Jason Eisner. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proc. EMNLP*, 2009.

Zhifei Li and Sanjeev Khudanpur. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of ACL SSST Workshop*, 2008.

Zhifei Li and Sanjeev Khudanpur. Efficient extraction of oracle-best translations from hypergraphs. In *Proc. NAACL*, 2009.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N. G. Thornton, Jonathan Weese, and Omar F. Zaidan. Joshua: an open source toolkit for parsing-based machine translation. In *StatMT '09: Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, 2009a.

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur. Variational decoding for statistical machine translation. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 593–601, 2009b.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming B*, 45(3):503–528, 1989.

Adam Lopez. Statistical machine translation. *ACM Computing Surveys*, 40(3):1–49, 2008a.

Adam Lopez. Translation as weighted deduction. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 532–540, 2009.

Adam Lopez. *Machine Translation by Pattern Matching*. PhD thesis, University of Maryland, March 2008b.

Yanjun Ma, Nicolas Stroppa, and Andy Way. Bootstrapping word alignment via word packing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 304–311, Prague, Czech Republic, June 2007.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. Lattice-based minimum error rate training for statistical machine translation. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 725–734, 2008.

Nitin Madnani. *The Circle of Meaning: From Translation to Paraphrasing and Back*. PhD thesis, University of Maryland, May 2010.

Nitin Madnani and Bonnie Dorr. Generating phrasal & sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3), 2010.

Lidia Mangu, Eric Brill, and Andreas Stolcke. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Speech and Language*, 14(4):373–400, 2000.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Boston, MA, 1999.

Lambert Mathias and William Byrne. Statistical phrase-based speech translation. In *IEEE Conference on Acoustics, Speech and Signal Processing*, 2006.

I. Dan Melamed. Statistical machine translation by parsing. In *ACL*, pages 653–660, 2004.

Igor′ Mel′čuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany, NY, 1988.

Mehryar Mohri. Weighted automata algorithms. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, Monographs in Theoretical Computer Science, pages 213–254. Springer, 2009.

Mark-Jan Nederhof. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143, Mar 2003.

Mark-Jan Nederhof and Giorgio Satta. Probabilistic parsing as intersection. In *8th International Workshop on Parsing Technologies*, pages 137–148, Nancy, France, April 2003.

Mark-Jan Nederhof and Giorgio Satta. The language intersection problem for non-recursive context-free grammars. *Information and Computation*, 192:172–184, 2004.

Sonja Niessen and Hermann Ney. Morpho-syntactic analysis for reordering in statistical machine translation. In *Proceedings of MT Summit VIII*, Santiago de Compostela, Galicia, Spain, 2001.

Franz Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan, July 2003.

Franz Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Kemal Oflazer and Ilknur Durgar El-Kahlout. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 25–32, Prague, Czech Republic, June 2007.

Takashi Onishi, Masao Utiyama, and Eiichiro Sumita. Paraphrase lattice for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, to appear 2010.

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 ACL*, pages 311–318, 2002.

Barbara H. Partee, Alice G.B. ter Meulen, and Robert E. Wall. *Mathematical Methods in Linguistics*. Springer, 1990.

Fernando C. N. Pereira and Rebecca N. Wright. Finite-state approximation of phrase structure grammars. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 246–255, Berkeley, 1991.

Slav Petrov and Dan Klein. Discriminative log-linear grammars with latent variables. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1153–1160, Cambridge, MA, 2008. MIT Press.

Matt Post and Daniel Gildea. Parsers as language models for statistical machine translation. In *Proceedings of AMTA*, Honolulu, HI, 2008.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In Lawrence K. Saul, Yair Weiss, and Leon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1097–1104, 2004.

Lance Ramshaw, Elizabeth Boschee, Sergey Bratus, Scott Miller, Rebecca Stone, Ralph Weischedel, and Alex Zamanian. Experiments in multi-modal automatic content extraction. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–5, 2001.

Philip Resnik, Olivia Buzek, Chang Hu, Yakov Kronrod, Alex Quinn, and Benjamin B. Bederson. Improving translation via targeted paraphrasing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 127–137, 2010.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. On dual decomposition and linear programming relaxations for natural language processing. In *In Proc. EMNLP*, 2010.

Sunita Sarawagi and William W. Cohen. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 1185–1192, 2004.

Giorgio Satta. Translation algorithms by means of language intersection, submitted.

Giorgio Satta and Enoch Peserico. Some computational complexity results for synchronous context-free grammars. In *Proceedings of NAACL*, 2005.

Hendra Setiawan, Min-Yen Kan, Haizhou Li, and Philip Resnik. Topological ordering of function words in hierarchical phrase-based translation. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 324–332, 2009.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 213–220, 2003.

Stuart Shieber, Yves Schabes, and Fernando C. N. Pereira. Priniples and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36, 1995.

Achim Sixtus and Stefan Ortmanns. High quality word graphs using forward-backward pruning. In *Proceedings of ICASSP*, pages 593–596, Phoenix, AZ, 1999.

Noah Smith. Log-linear models. http://www.cs.cmu.edu/~nasmith/papers/smith.tut04.pdf, 2004.

Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of Venus images. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS)*, pages 1085–1092, Denver, CO, 1994. MIT Press.

Matthew Snover, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, 2006.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast–but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*, pages 254–263, 2008.

Hagen Soltau, George Saon, Daniel Povy, Lidia Mangu, Brian Kingsbury, Jeff Kuo, Mohamed Omar, and Geoffrey Zweig. The IBM 2006 GALE Arabic ASR system. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, 2007.

Richard Sproat and Thomas Emerson. The first international Chinese word segmentation bakeoff. In *Proceedings of the second SIGHAN workshop on Chinese language processing*, pages 133–143, 2003.

Mark Steedman. *The Syntactic Process*. MIT Press, Cambridge, MA, USA, 2000.

Andreas Stolcke. SRILM – an extensible language modeling toolkit. In *Intl. Conf. on Spoken Language Processing*, 2002.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64, Boulder, Colorado, June 2009.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8:693–723, 2007.

Toshiyuku Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world. In *Proceedings of LREC 2002*, pages 147–152, Las Palmas, Spain, 2002.

David Talbot and Miles Osborne. Modelling lexical redundancy for machine translation. In *Proceedings of ACL 2006*, Sydney, Australia, 2006.

Roy Tromble and Jason Eisner. Learning linear ordering problems for better translation. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1007–1016, 2009.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. Lattice minimum Bayes-risk decoding for statistical machine translation. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 620–629, 2008.

Huihsin Tseng, Pi-Chuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. A conditional random field word segmenter. In *Fourth SIGHAN Workshop on Chinese Language Processing*, 2005.

Grigorios Tsoumakas and Ioannis Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

Nicola Ueffing, Franz J. Och, and Hermann Ney. Generation of word graphs in statistical machine translation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 156–163, 2002.

Gertjan van Noord. The intersection of finite state automata and definite clause grammars. In *Proceedings of ACL*, 1995.

Ashish Venugopal, Andreas Zollmann, and Vogel Stephan. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 500–507, Rochester, New York, 2007.

K. Vijay-Shanker and David J. Weir. The use of shared forests in tree adjoining grammar parsing. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*, pages 384–393, Utrecht, The Netherlands, April 1993.

Luis von Ahn. Games with a purpose. *IEEE Computer Magazine*, pages 96–98, June 2006.

Chao Wang, Michael Collins, and Philipp Koehn. Chinese syntactic reordering for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 737–745, Prague, 2007.

Janyce M. Wiebe, Rebecca F. Bruce, and Thomas P. O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 246–253, 1999.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, 1997.

Jia Xu, Richard Zens, and Hermann Ney. Do we need Chinese word segmentation for statistical machine translation? In *Proceedings of the Third SIGHAN Workshop on Chinese Language Learning*, pages 122–128, Barcelona, Spain, 2004.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. Using a dependency parser to improve SMT for subject-object-verb languages. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 245–253, Boulder, CO, 2009.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530, 2001.

Mei Yang and Katrin Kirchhoff. Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of the EACL 2006*, pages 41–48, 2006.

Omar F. Zaidan and Chris Callison-Burch. Feasibility of human-in-the-loop minimum error rate training. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 52–61, 2009.

Richard Zens and Hermann Ney. Discriminative reordering models for statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 55–63, 2006.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of ACL*, 2008.

Tiejun Zhao, Lu Yajuan, Yang Muyun, and Yu Hao. Increasing accuracy of Chinese segmentation with strategy of multi-step processing. In *Journal of Chinese Information Processing (Chinese Version)*, volume 1, pages 13–18, 2001.

Andreas Zollmann and Ashish Venugopal. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on SMT*, 2006.

Andreas Zollmann, Ashish Venugopal, Franz Och, and Jay Ponte. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of 22nd International Conference on Computational Linguistics (Coling)*, Manchester, U.K., 2008.