LAMP-TR-135
CS-TR-4831
UMIACS-TR-2006-47

April 2007

A Survey of Statistical Machine Translation

Adam Lopez

Computational Linguistics and Information Processing Laboratory
Institute for Advanced Computer Studies
Department of Computer Science
University of Maryland
College Park, MD 20742
alopez@cs.umd.edu

Abstract

Statistical machine translation (SMT) treats the translation of natural language as a machine learning problem. By examining many samples of human-produced translation, SMT algorithms automatically learn how to translate. SMT has made tremendous strides in less than two decades, and many popular techniques have only emerged within the last few years. This survey presents a tutorial overview of state-of-the-art SMT at the beginning of 2007. We begin with the context of the current research, and then move to a formal problem description and an overview of the four main subproblems: translational equivalence modeling, mathematical modeling, parameter estimation, and decoding. Along the way, we present a taxonomy of some different approaches within these areas. We conclude with an overview of evaluation and notes on future directions.

This is a revised draft of a paper currently under review. The contents may change in later drafts. Please send any comments, questions, or corrections to alopez@cs.umd.edu. Feel free to cite as University of Maryland technical report UMIACS-TR-2006-47. The support of this research by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-0001, ONR MURI Contract FCPO.810548265, and Department of Defense contract RD-02-5700 is acknowledged.

Contents

1 Introduction

Machine translation (MT) is the automatic translation from one natural language into another using computers. It is a key application in the field of natural language processing (NLP), and interest in MT is nearly as old as the electronic computer – popular accounts trace its origins to a letter written by Warren Weaver in 1947, only a year after ENIAC came online. A good historical overview is given by Hutchins (2007), and comprehensive general survey is given by Dorr et al. (1999).

Statistical machine translation (SMT) refers to a subset of MT systems that are characterized by their use of machine learning methods. Although the first systems were developed less than two decades ago, SMT currently dominates the research field. Progress is rapid and the state-of-the-art is a moving target. However, as the field has matured, some common themes have emerged.

The goals of this paper are to outline the essential elements of SMT and provide a taxonomy of popular approaches. Our objective is to provide the reader with a basic understanding of how SMT works and some of the common choices that are made, without focusing on the minute details of any specific system. The reader should be familiar with topics from a senior undergraduate or first-year graduate computer science curriculum, such as discrete mathematics, formal language theory, logic, graph theory, search, and data structures. Some knowledge of statistics and machine learning will also be helpful, although we focus on the main ideas and intuitions rather than full mathematical rigor, which is in any case beyond the scope of this work. We will also briefly touch on some linguistic concepts, although we will write mainly from a computer science perspective.

Good SMT tutorials appear in Knight (1997), Knight (1999b), and Manning and Schütze (1999, chap. 13), but they are now somewhat dated. Knight and Marcu (2005) give a recent but brief survey. At the time of this writing, other introductory materials in preparation include Koehn (2007), and a chapter in a planned future edition of Jurafsky and Martin (2000). For greater coverage of fundamental research areas, refer to textbooks on NLP (Jurafsky and Martin, 2000; Manning and Schütze, 1999), artificial intelligence (Russell and Norvig, 2003) machine learning (Mitchell, 1997), or formal language theory (Hopcroft and Ullman, 1979).

1.1 Background and Context

SMT treats translation as a machine learning problem. This means that we apply a learning algorithm to a large body of previously translated text, known variously as a *parallel corpus*, *parallel text*, *bitext*, or *multitext*. The learner is then able translate previously unseen sentences. With an SMT toolkit and enough enough parallel text, we can build an MT system for an new language pair within a very short period of time – perhaps as little as a day (Al-Onaizan et al., 1999; Oard and Och, 2003; Oard et al., 2003). For example, Oard and Och (2003) report constructing a Cebuano-to-English SMT system in a matter of weeks. Recent exercises have shown that translation systems can be built for a wide variety of language pairs within similar time frames (Koehn and Monz, 2005, 2006). The accuracy of these systems depends crucially on the quantity, quality, and domain of the data, but there are many tasks for which even poor translation is useful (Church and Hovy, 1993).

Recent interest in SMT can be attributed to the convergence of several factors.

(1) The growth of the internet has strongly affected two constituencies of translation consumers. The first of these is interested in the *dissemination* of information in multiple languages. Examples are multilingual governments and news agencies and companies operating in the global marketplace. The internet enables them to easily publish information in multiple languages. Due to this widespread dissemination, SMT researchers now have access to Biblical texts (Resnik et al., 1997), bilingual government and news text (Koehn, 2005), and other data mined from the Internet (Resnik and Smith, 2003). These data are the fundamental resource in SMT research. Because they are the product of day-to-day

human activities, they are constantly growing. Multilingual governments interested in dissemination, such as the European Union, have increased MT research funding to further their domestic policy interests.

(2) The other consumers of translation are those interested in the *assimilation* of information not in their native language. These include intelligence agencies, researchers, and casual internet users. The internet has made such information much more readily accessible, and increasing demand from these users helps drive popular interest in MT. The United States government is interested in assimilation, and has increased MT research funding to further its international policy interests.

(3) Fast, cheap computing hardware has enabled applications that depend on large datasets and billions of statistics. Advances in processor speed, random access memory size and speed, secondary storage, and parallel processing have all helped to enable SMT.

(4) The development of automatic translation metrics – although controversial – has enabled rapid iterative development of MT systems and fostered competition between research groups. Objective measurable goals have naturally led to objective measurable progress. The National Institute of Standards has used these metrics since 2002 in a yearly competition at its MT Evaluation conference.

(5) Several projects have focused on the development of freely available SMT toolkits (Al-Onaizan et al., 1999; Burbank et al., 2005; Germann et al., 2001; Koehn, 2004a; Koehn et al., 2006; Och and Ney, 2003; Olteanu et al., 2006). Most are open-source. These implementations help lower the barrier for entry into SMT research.

1.2 Formal Description

Formally, our task is to take a sequence of tokens in the *source language* with vocabulary F, and transform it into a sequence of tokens in the *target language* with vocabulary E. [1] Without loss of generality, we will assume that tokens are words and sequences are sentences. Agglutinative languages such as German and Inuktitut, or languages with no clearly marked word boundaries, such as Chinese, may require special preprocessing. The most important consideration is all data are preprocessed consistently, since statistical systems are sensitive to discrepancies. There is often no special treatment of morphological variants – for instance, the English words *translate* and *translation* are treated as unrelated, indivisible tokens. Therefore, it is possible for the size of the vocabularies E and F to reach into the tens or hundreds of thousands, or even millions in the case of morphologically complex languages such as Arabic.

We denote a sequence of J source words as $f_1 f_2 ... f_J$ or $f_1^J \in F^J$, and a sequence of I target words as $e_1 e_2 ... e_I$ or $e_1^I \in E^I$. The goal of a translation system, when presented with an input sequence f_1^J, is to find a sequence e_1^I that is *translationally equivalent*.

An example of translationally equivalent sequences is shown in Figure 1. It is natural to imagine that translational equivalence can be decomposed into a number of smaller equivalence problems. For instance, we can see that Chinese word 北 is translated as the English word *north*. We say that such words are *aligned*. An example word alignment is shown in Figure 2.

Word translation is often *ambiguous*. For instance, we might reasonably translate 北 as *northern* without loss of meaning. Often, the correct word translation will depend on context.

Word alignment itself is an imprecise concept, and in fact human judges will often disagree on the alignment of a sentence (Melamed, 1998; Och and Ney, 2000a). If our system translates word by word, it needs some mechanism to choose between several possible options for each decision.

In addition to word translation, the other main problem that can be seen from the figure is that words with equivalent meanings do not appear in the same order in both sentences. Therefore, our system will

[1]We follow the widely used notation of Brown et al. (1990), who use E for English and F for French (or foreign).

However , the sky remained clear under the strong north wind .

虽然	北	风	呼啸	,	但	天空	依然	十分	清澈	。
Although	north	wind	howls	,	but	sky	still	extremely	limpid	.

Figure 1: An example of translationally equivalent sentences. We show English glosses for each Chinese word.

need some mechanism to *reorder* the words into an appropriate order for the target language. As with word translation, reordering decisions often entail resolving some kind of ambiguity.

Translation can therefore be thought of as making a set of decisions. We rewrite the source sentence, making word translation and reordering decisions, until we have replaced it completely with a target sentence. At each decision point, a large number of possible rules may apply. We will need some mechanism to disambiguate these rules. Furthermore, both rules and methods of disambiguation must be learned from our parallel data.

Following these core ideas, there are four problems that we must solve in order to build a functioning SMT system. [2]

(1) Define the translational equivalence model that relates sequences in F^* to sequences in E^*. All of the translational equivalence models that we will consider in this paper are generalizations of formal languages that produce two outputs rather than one. We describe translational equivalence models in §2.

(2) Define a mathematical model, which is a function that assigns a score to every pair in the set $\{E^* \times F^*\}$. We describe mathematical models in §3.

(3) Search for reasonable values for our mathematical model with the aid of a parallel corpus. This is called *parameter estimation*, and it is based on machine learning methods. The search space is defined by our mathematical model. We describe parameter estimation in §4.

(4) Finally, when we are presented with input f_1^J, search for the e_1^I that conforms to our model of translational equivalence and is assigned a high score under our mathematical model. This step is called *decoding*. Just as the search space in parameter estimation is defined by our mathematical model, the search space in decoding is defined by our translational equivalence model. We describe decoding in §5.

In addition to these four problems, we will discuss the important topic of evaluation in §6. The article concludes with notes on current directions in §7.

2 Translational Equivalence Modeling

We can classify SMT systems according to the model of translational equivalence describing the conversion of f_1^J into e_1^I. Broadly speaking, a translational equivalence model is simply the set of all

[2]Most descriptions of the SMT problem will refer to only three subproblems, following Brown et al. (1990). This is because early systems involved a tight coupling between the translational equivalence model and the mathematical model (§3.1). Recent advances that we describe in this paper have shown that the choices of translational equivalence model and mathematical model are largely orthogonal. What we call here a mathematical model might be more accurately called a *parameterization*. However, since in the literature the overloaded term *model* is often used interchangeably to refer to both the models of translational equivalence and their parameterizations, we use *mathematical model* to acknowledge this usage, while at the same time distinguishing it from translational equivalence.

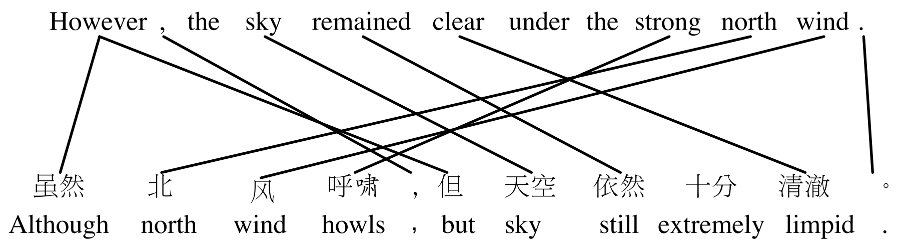

However , the sky remained clear under the strong north wind .
虽然 北 风 呼啸 , 但 天空 依然 十分 清澈 。
Although north wind howls , but sky still extremely limpid .

Figure 2: A possible alignment of the the sentence illustrated in Figure 1.

(unweighted) rules employed by an MT system to transform a source sentence into a target sentence. Although in principle these rules may come from anywhere, in most SMT systems they are automatically extracted from the training corpus. The extraction process is described in more detail in §4.2. In this section, we describe the various types of rules that are typically used.

Most popular translational equivalence models can be described by one of two formalisms: finite-state transducers (FST) or synchronous context-free grammar (SCFG). [3] These formalisms are generalizations of finite-state automata (FSA) and context-free grammar (CFG), respectively. Rather than producing single output strings as in those formalisms, they produce two output strings, and define an alignment between them. Translational equivalence models are important in decoding, where they constrain the search space in which we will attempt to find translations.

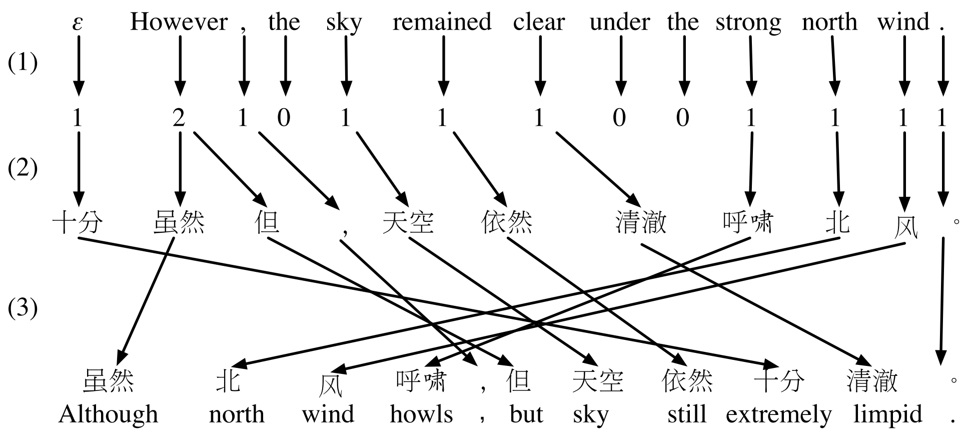
The remainder of this section discusses translational equivalence models. FST models are described in §2.1, and SCFG models are described in §2.2. We briefly touch on some other types of translational equivalence model in §2.3.

2.1 Finite-State Transducer Models

Finite-state transducers are straightforward extensions of the familiar finite-state automata (FSA) (Hopcroft and Ullman, 1979). Recall that we can define a finite-state automaton (S, L, D) as a set of states S, a set of labels L, and a set of transitions $D \subseteq \{S \times S \times L\}$, where each transition is defined as a pair of states and a label that must be output (or read, depending on the use of the FSA) as we move from the first state to the second. In NLP, there are a number of applications of FSAs and FSTs. A common use of FSAs comes from automatic speech recognition (ASR), where they are the basis of so-called *n-gram language models* (see e.g. Jelinek, 1998). We can define an n-gram language model for a language with vocabulary V as $(V^{n-1}, V, D = \{\mathbf{s} \times \mathbf{s}' \times \mathbf{v} : \mathbf{v} \in V, \mathbf{s} \in V^{n-1}, \mathbf{s}' = \mathbf{s}_2^{n-1}\mathbf{v}\})$ In this FSA, each transition represents a word in the language, and each state represents the $n - 1$ most recently encountered words. By applying probabilistic weights to this FSA, we can use it to compute the probability of any string in a language. Although FSA language models bear little resemblance to our understanding of how natural language works, their use in probabilistic modeling is widespread, often with good results.

Finite-state transducers extend the concept of FSAs by using two sets of labels; each transition will have a label from each set. We can imagine that the transducer operates on an input string and an output string. One label is read from the input, and the other is written to the output. A transition labelled

[3]This distinction may be a bit confusing, since finite state transducers come from automata theory and synchronous context-free grammars from formal language theory. Although concepts from each theory have dual representations in the other, it is a historical accident that in the first case, the automata theory concept is generally used, and in the second case, the language theory concept is generally used. We simply follow the prevailing terminology in the literature (although this is changing, with recent popular interest in *tree transducers*, e.g. in Galley et al., 2006; Marcu et al., 2006).



ε However , the sky remained clear under the strong north wind .
(1)
1 2 1 0 1 1 1 0 0 1 1 1 1
(2)
十分 虽然 但 ， 天空 依然 清澈 呼啸 北 风 。
(3)
虽然 北 风 呼啸 ， 但 天空 依然 十分 清澈 。
Although north wind howls , but sky still extremely limpid .

Figure 3: Visualization of IBM Model 4. This model of translation takes three steps. (1) Each English word (and the null word) selects a fertility – the number of Chinese words to which it corresponds. (2) Each English word produces a number of Chinese words corresponding to its fertility. Each Chinese word is generated independently. (3) The Chinese words are reordered.

with x from set X and y from set Y signifies a correspondence between x and y. Additionally, either or both label may consist of the empty string ε, which indicates that there is no change in that output for that particular transition. A powerful technique that we can use with FSTs is composition. FSTs are composed by making the output of one the input to the next.

2.1.1 Word-Based Models

SMT continues to be influenced by the groundbreaking IBM approach (Berger et al., 1994; Brown et al., 1990, 1993). The IBM Models are *word-based models* and represent the first generation of SMT models. Since they illustrate many concepts common to other SMT models they make a useful starting point for our exposition. We illustrate word-based models with the most commonly used example, IBM Model 4.

For historical reasons that we will explain in §3.1, IBM Model 4 is usually described as a target-to-source model, that is, a model that produces the source sentence f_1^J from the target sentence e_1^I. We follow this convention in our description.

In IBM Model 4, the process that produces f_1^J from e_1^I takes three steps, which are illustrated in Figure 3. Each of these steps corresponds to a single transducer in a cascade. The transducers are illustrated in Figure 4.

(1) Each target word chooses the number of source words that it will generate. We call this number ϕ_i the *fertility* of e_i. One way of thinking about fertility is that when the transducer encounters the word e_i in the input, it outputs ϕ_i copies of e_i in the output. The length J of the source sentence is determined at this step since $J = \sum_{i=0}^{I} \phi_i$. This allows us to define a translational equivalence between source and target sequences of different lengths.

(2) In the second transducer, each input word $e_i \in E$ produces an output word $f_j \in F$. This represents the translation of individual words.

(3) The translated words are reordered.

The final step exposes the key weakness of finite-state transducers for translation. There is no efficient way to represent reordering in a finite-state transducer. They are designed to represent relationships between strings with a monotonic ordering relationship – in other words, if an input label at position

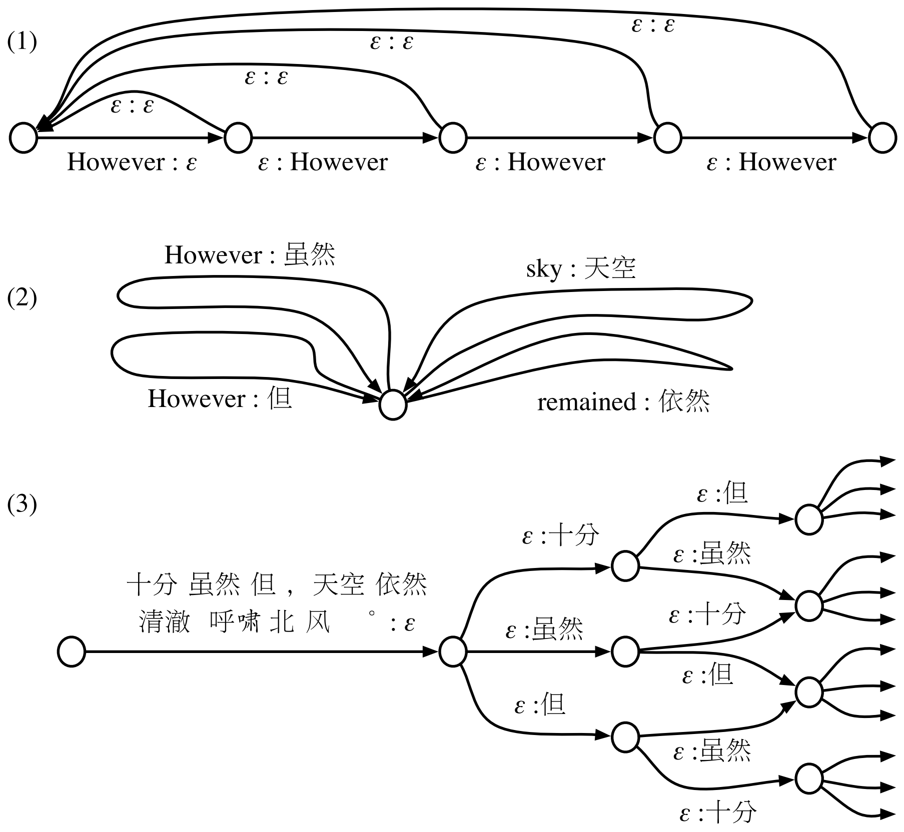


Figure 4: Visualization of the finite-state transducer conception of IBM Model 4. We show only a portion of each transducer. Transducer (1) copies the input word to its output a number of times according to its fertility; (2) corresponds to word-to-word translation, and (3) corresponds to reordering. Transducer (3) is the most complex, because it must represent all possible reorderings. A cascade of these transducers can represent the process shown in Figure 3.

i corresponds to an output label at position j, then an input label at position $i' > i$ will correspond to an output label at position $j' > j$. This is ideal for problems such as automatic speech recognition and part-of-speech tagging, where monotonicity is the natural relationship between streams of data. One solution to this problem is to require monotonic alignments (Tillmann et al., 1997). Monotonicity is beneficial for the complexity of decoding and for integration with ASR for speech-to-speech or speech-to-text translation. However, it will rule out many good translations such as the one in Figure 1, and the results may be awkward, especially in the translation of languages that have naturally different word orders. At the other end of the spectrum, full reordering requires an FST that accepts all possible permutations of the J source words. This FST contains $O(J!)$ states (Och and Ney, 2003). Most models take a middle ground, using a generally monotonic approach but allowing a fixed number of words to be skipped. This approach originated with the IBM models, which allowed a lookahead of up to four words. This is often called the *IBM constraint*.

In addition to applying this process to each target word e_i in the sequence e_1^I, it is also applied to a special empty token ε, called the *null word* – or more simply *null* – and denoted e_0. The purpose of modeling *null translation* is to account for words in f_1^J which are often dropped in translation, as is often the case with function words.

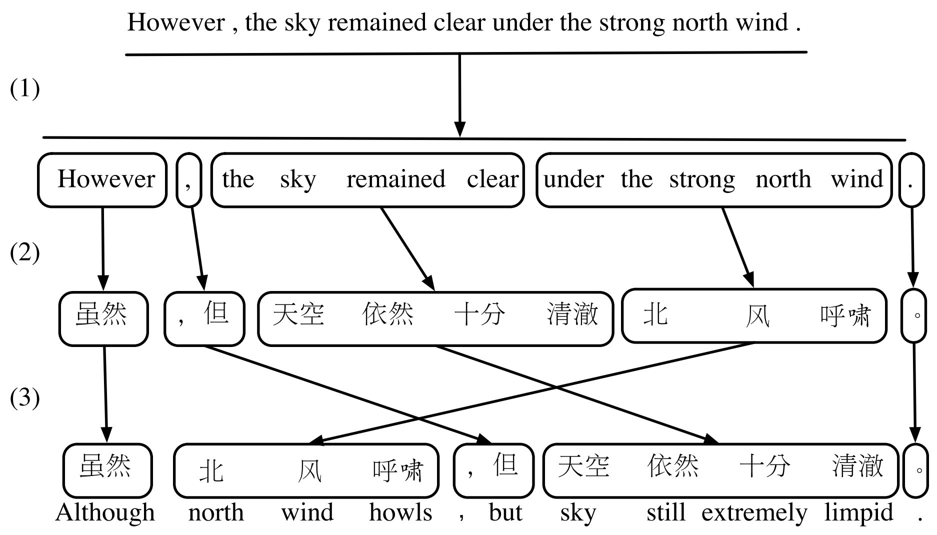

However , the sky remained clear under the strong north wind .
(1)
However
,
the sky remained clear
under the strong north wind
.
(2)
虽然
，但
天空 依然 十分 清澈
北 风 呼啸
。
(3)
虽然
北 风 呼啸
，但
天空 依然 十分 清澈
。
Although north wind howls , but sky still extremely limpid .

Figure 5: Visualization of the phrase-based model of translation. The model involves three steps. (1) The English sentence is segmented into “phrases” – arbitrary contiguous sequences of words. (2) Each phrase is translated. (3) The translated phrases are reordered. As in Figure 3, each arrow corresponds to a single decision.

Note that for IBM Model 4, the alignment is asymmetric. In particular, each source word can align to exactly one target word, or the null word. A target word can link to an arbitrary number of source words, as defined by its fertility.

2.1.2 Phrase-Based Models

One problem with word-based models is that the concept of a word must be precisely defined in order to correctly tokenize the sentence. Although this is adequate for languages such as English, it is somewhat more problematic for morphologically complex languages such as German, or languages with ambiguous word boundaries, such as Chinese. When translating such languages with word-based models, tokenization becomes a critical issue.

A second problem is that the concepts of null translation and fertility require substantial special modeling and engineering and are difficult to estimate accurately. They are also unappealing on an aesthetic level.

Finally, in real translation, monotonic reordering is common (though not absolute) and contiguous sequences of words often translate as a unit. However, in word-based translation models, substitution and reordering decisions are all made independently for each individual word. This results in a greater chance for error, usually resulting in “word salad” – a translation in which many words are correct, but their order is thoroughly wrong.

These observations motivate *phrase-based* translation (Koehn et al., 2003; Marcu and Wong, 2002; Och and Ney, 2004; Och et al., 1999). In phrase-based models we discard the assumption that the unit of translation is a single word. Instead, a unit of translation may be any contiguous sequence of words, called a *phrase*. In this usage, the term “phrase” has no particular linguistic sense, although it is straightforward to restrict ourselves to phrases licensed by a syntactic parser (Koehn et al., 2003). Phrase-based models do away with the concepts of null translation and fertility; each phrase in e_1^I is

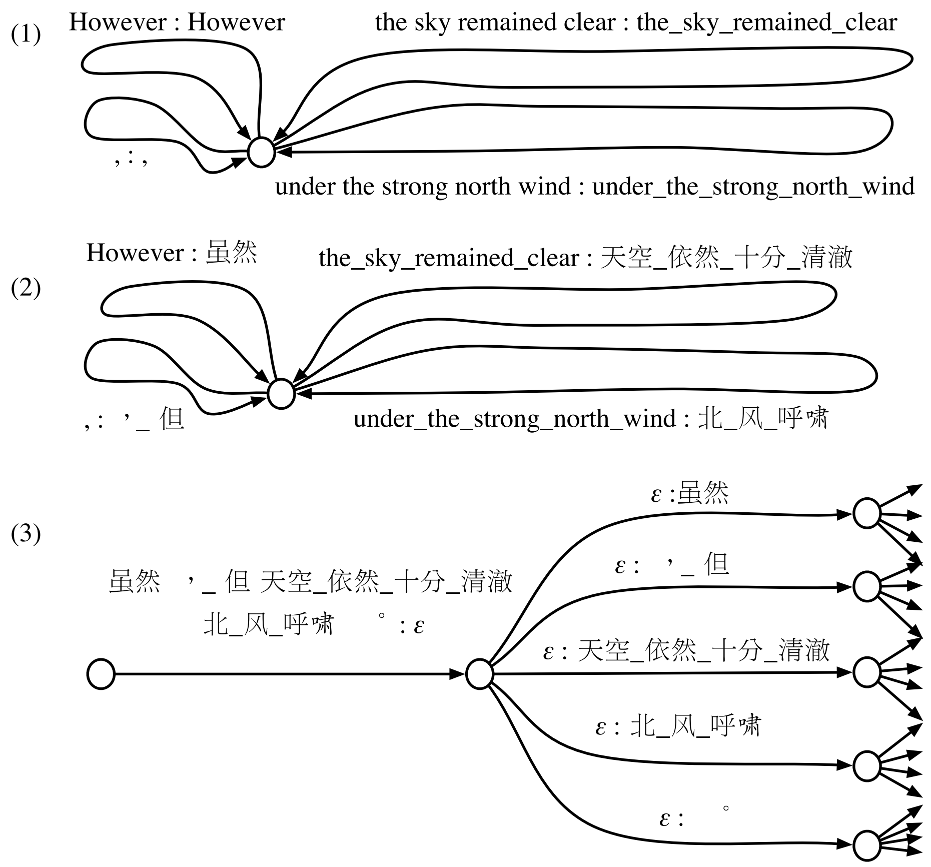

(1)
However : However
the sky remained clear : the_sky_remained_clear
, : ,
under the strong north wind : under_the_strong_north_wind
(2)
However : 虽然
the_sky_remained_clear : 天空_依然_十分_清澈
, : ，_ 但
under_the_strong_north_wind : 北_风_呼啸
(3)
虽然 ，_ 但 天空_依然_十分_清澈
北_风_呼啸 。: ε
ε :虽然
ε : ，_ 但
ε : 天空_依然_十分_清澈
ε : 北_风_呼啸
ε : 。

Figure 6: Visualization of the finite-state transducer conception of phrase-based translation. We show only a portion of each transducer. Transducer (1) segments the target sentence into phrases; (2) performs word-to-word translation; (3) reorders the phrases in the same way that the reordering transducer of Figure 4 reorders words.

nonempty and translates to exactly one nonempty phrase in f_1^J. We can also defer the tokenization step in languages such as Chinese, allowing the phrase boundaries to define tokens (Wu, 1995a; Xu et al., 2004). As a consequence we can characterize the substitution of "north wind" with "北 风" as an atomic operation. In the case where we have only seen the words "north" and "wind" separately, we can still translate in the same manner as a word-based model – that is, we translate with phrases that consist only of single tokens.

In phrase-based models, the process of transforming e_1^I to f_1^J takes the following steps, illustrated in Figure 5.

(1) The sentence e_1^I is first split into K phrases $\tilde{e}_1^K$. Each phrase $\tilde{e}_k = e_{i_1}^{i_2} \in E^{i_2-i_1+1}$.

(2) Each phrase $\tilde{e}_k$ is replaced with a phrase $\tilde{f}_k = f_{k_1}^{k_2} \in F^{k_2-k_1+1}$. Because of the one-to-one restriction, this means that we will end up with exactly K phrases. The length J of f_1^J is then $J = \sum_{k=1}^{K} k_1 - k_2$.

(3) The translated phrases are permuted into their final order. The permutation problem is identical to the one that we encounter in word-based models, and as in that case, one possible solution is monotonicity (Banchs et al., 2005).

A cascade of transducers which does this is shown in Figure 6.

We might assume that the words within any particular phrase pair are internally aligned in some way, but that is not strictly necessary in the most general form of phrase-based models (Marcu and Wong, 2002). Explicit internal word alignments are assumed in some phrase-based models, although this is usually for the purposes of mathematical modeling (e.g. Koehn et al., 2003; Kumar et al., 2006).

A variant of phrase-based models is the *alignment template model* (Och and Ney, 2004; Och et al., 1999). In this model explicit phrase-to-phrase translations are not used. Instead, each phrase is first associated with an *alignment template*, which is a reordering of the words in the phrase based on word categories rather than specific word identities. The words in the phrase are then translated using word-to-word translation, and the alignment templates are reordered as in phrase-based models.

Simard et al. (2005) present a version of phrase-based translation in which phrases are not required to be continuous, and gaps must be filled in with other phrases.

2.2 Synchronous Context-Free Grammar Models

Phrase-based FST models are widely used. Nonetheless, these models present a number of drawbacks for MT. Arbitrary permutation is an especially crude representation of the type of reordering that happens in real translation. There is no hope of describing the movement of hierarchical structures within sentences using such a model. These behaviors are more likely to be characterized by linguistic notions of syntax. Syntactic cues may provide several hints to a translation model.

(1) Knowledge of a word's syntactic collocates could be used to influence the translation choice of the word (García Varea et al., 2001).

(2) Non-overlapping syntactic units rarely overlap in translation. We call this property *phrasal cohesion* (Fox, 2002).

(3) Target language syntax can hopefully guide reordering more accurately than arbitary permutation. Just as word-based models produce word salad, phrase-based models often produce phrase salad

Recent empirical studies have confirmed that these conditions hold in the majority of cases, although there are plenty of exceptions. Fox (2002) presents empirical evidence that reordering tends to respect the boundaries of syntactic phrases, and Galley et al. (2004, 2006) expand on this to illustrate that deviations can be explained with the benefit of some sensitivity to additional syntactic context. Other empirical studies have also supported the hypothesis that substantial syntactic information is preserved in translation (Hwa et al., 2005; Yarowsky et al., 2001).

There have been a few attempts to apply syntactic knowledge to FST-based models. These include limiting phrase-based translation to phrases licensed by a syntactic parser (Koehn et al., 2003), reranking FST decoder output using probabilities from a syntactic parser (Och et al., 2004); and preprocessing input by performing language-specific reordering of phrases found by a syntactic parser (Collins et al., 2005). Only the last of these resulted in modest improvements.

An alternative to the FST is the synchronous context-free grammar (SCFG), which is a generalization of context-free grammar (CFG). SCFGs can be applied in concert with syntactic parsing of one or both languages in translation, although this is not strictly necessary.

SCFGs are known under different guises as syntax-directed translation (Aho and Ullman, 1969), inversion transduction grammar (Wu, 1995a), or head transducers (Alshawi et al., 2000). A formalism that generalizes these is multitext grammar (Melamed, 2003). Chiang (2006) provides a good overview of SCFG and several variants.

A related formalism is the *tree transducer*, which describes operations on tree fragments. In many cases, *regular* tree transducers are equivalent SCFG models. The use of tree transducers to describe these translation models is increasingly popular (e.g. Galley et al., 2006; Graehl and Knight, 2004;

Marcu et al., 2006; Wellington et al., 2006a), although grammar-based descriptions are more common in earlier descriptions.

In the discussion that follows, we will use the notation of SCFG, which is relatively easy to understand. Although the general literature on SCFG does not draw any formal distinction between the models we describe here, in SMT a finer-grained classification is used (Chiang, 2004).

SCFG is a generalization of CFG to the case of two output strings. Recall that a CFG (N, T, D) consists of a set of non-terminal symbols N, terminal symbols T, and productions $D = \{N \longrightarrow \{N^* \times T^*\}\}$. We begin by writing a special *root* non-terminal symbol to the output, This symbol is rewritten using a rule $d \in D$. Rewriting of non-terminal symbols continues recursively until the output contains only terminal symbols. In natural language parsing, terminal symbols are words (i.e. $T = E$) and non-terminal symbols represent syntactic categories, as in the following fragment of a CFG grammar:

$$\text{NP} \longrightarrow \text{DT NPB} \tag{C1}$$
$$\text{NPB} \longrightarrow \text{JJ NPB} \tag{C2}$$
$$\text{NPB} \longrightarrow \text{NP} \tag{C3}$$
$$\text{DT} \longrightarrow \text{the} \tag{C4}$$
$$\text{JJ} \longrightarrow \text{strong} \tag{C5}$$
$$\text{JJ} \longrightarrow \text{north} \tag{C6}$$
$$\text{NN} \longrightarrow \text{wind} \tag{C7}$$

This grammar is similar to ones that are currently used in natural language parsing. [4] In this grammar, the syntactic category NP can be rewritten as "the strong north wind" via a series of productions, as shown in Figure 7.

In SCFG, we must specify two output strings for each production. We show a fragment of an SCFG grammar:

$$\text{NP} \longrightarrow \text{DT}_{\boxed{1}}\text{NPB}_{\boxed{2}} \;/\; \text{DT}_{\boxed{1}}\text{NPB}_{\boxed{2}} \tag{S1}$$
$$\text{NPB} \longrightarrow \text{JJ}_{\boxed{1}}\text{NPB}_{\boxed{2}} \;/\; \text{JJ}_{\boxed{1}}\text{NPB}_{\boxed{2}} \tag{S2}$$
$$\text{NPB} \longrightarrow \text{JJ}_{\boxed{1}}\text{NPB}_{\boxed{2}} \;/\; \text{NPB}_{\boxed{2}}\text{JJ}_{\boxed{1}} \tag{S3}$$
$$\text{NPB} \longrightarrow \text{NP}_{\boxed{1}} \;/\; \text{NP}_{\boxed{1}} \tag{S4}$$
$$\text{DT} \longrightarrow \text{the} \;/\; \varepsilon \tag{S5}$$
$$\text{JJ} \longrightarrow \text{strong} \;/\; \text{呼啸} \tag{S6}$$
$$\text{JJ} \longrightarrow \text{north} \;/\; \text{北} \tag{S7}$$
$$\text{NN} \longrightarrow \text{wind} \;/\; \text{风} \tag{S8}$$

Here, we separate the two outputs using a slash (/). Alignment is indicated by coindexes on the nonterminals, which are required to appear in both outputs. We can think of SCFG as generating isomorphic trees, in which the non-terminal nodes of each tree are aligned. One tree can be transformed into another by rotating its non-terminal nodes, as if it were an Alexander Calder mobile. The relationship is illustrated in Figure 7. Note that if we ignore the source string dimension, the rules in this SCFG

[4]The nonterminal categories are borrowed from those used in an annotation of the Wall Street Journal corpus, and represent familiar English syntactic categories such as noun phrase (NP), adjective (JJ), preposition (IN), determiner (DT), and noun (NN) (Marcus et al., 1993).

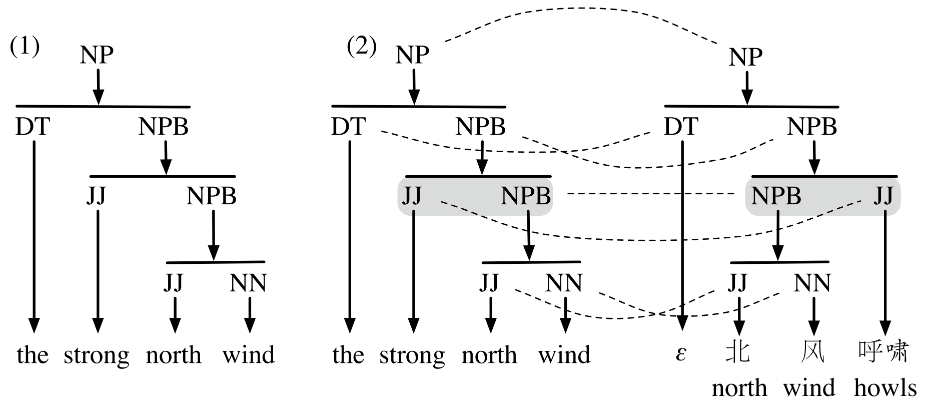


Figure 7: Visualization of the CFG derivation and SCFG derivations. Derivation happens in exactly the same way in CFG (1) and SCFG (2). Each nonterminal symbol is replaced by the contents of the right-hand-side of a rule whose left-hand-side matches the symbol. In our illustration, each arrow represents a single production. The difference in SCFG is that we specify two outputs rather than one. Each of the non-terminal nodes in one output is linked to exactly one node in the other; the only difference between the outputs is the order in which these nodes appear. Therefore, the trees are isomorphic. Although terminal nodes are not linked, we can infer a word alignment between words that are generated by the same non-terminal node. In this illustration, the only reordering production is highlighted. Note that if we ignore the Chinese dimension of the output, the SCFG derivation in the English dimension is exactly the same as in (1).

correspond to the rules that appears in our CFG example. Alternatively, if we ignore the tree structures, we obtain an alignment on the words of the source and target language strings. Thus SCFG defines a mapping between both strings and trees, and has a number of uses depending on the relationship that we are interested in (Melamed, 2004a; Wu, 1995a).

Normal forms and complexity analysis for various flavors of SCFG are presented in Aho and Ullman (1969) and Melamed (2003). Generally speaking, the number of possible reorderings is quite large. Church and Patil (1982) shows that the number of parses in a binary CFG is related to a combinatorial function, the *Catalan number*, and Zens and Ney (2003) shows that the number of reorderings in a binary SCFG are related to another combinatorial function, *Shröder number*. Despite this, it is possible to process SCFG in polynomial time using standard chart parsing algorithms (Melamed, 2003). In principle, this makes SCFG models of translational equivalence less computationally expensive than FST models (Wu, 1996).

Numerous different approaches to SMT can be expressed in the SCFG formalism. In the following sections we will illustrate three applications of SCFG that are representative of their use in SMT. We will consider *bracketing grammars* used to constrain reordering (§2.2.1); *syntax-based grammars* that exploit linguistic syntax (§2.2.2); *hierarchical phrase-based translation* that combines the insights of phrase-based models with syntactic structure, (§2.2.3); and *syntactic phrase-based translation* that combines phrases with linguistic syntax (§2.2.4).

2.2.1 Bracketing Grammars

One reason to use SCFG is to curtail the number of reorderings described by the model. This is important during decoding – even if we knew the correct translation for each individual word in the sentence, we

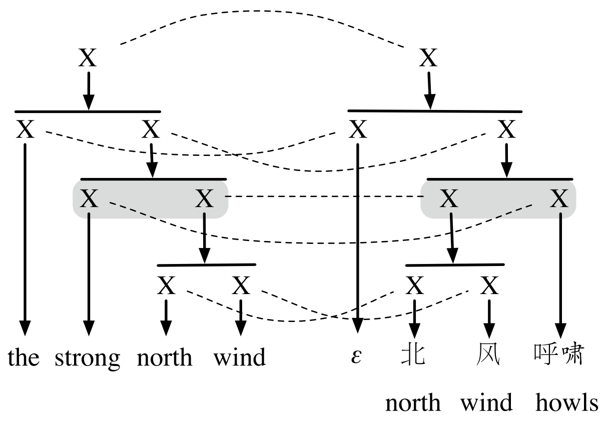


Figure 8: Visualization of a bracketing grammar derivation. Each arrow corresponds to a grammar production. Rules that involve reordering are highlighted, and the order of the target language sentence is illustrated beneath the syntax tree.

will still need to search for a good reordering. In FST models, reordering is modeled as arbitrary permutation unless constraints are used. Search exponential in sentence length. However, as described above, the search in SCFG is polynomial in sentence length. This observation motivates the use of bracketing grammars, which are designed to compactly represent all possible reorderings consistent with a binary bracketing of the input string (Wu, 1996). In bracketing grammars, we use a single undifferentiated nonterminal symbol, and three rules:

$$X \longrightarrow X_{\boxed{1}}X_{\boxed{2}} \;/\; X_{\boxed{1}}X_{\boxed{2}} \tag{B1}$$

$$X \longrightarrow X_{\boxed{1}}X_{\boxed{2}} \;/\; X_{\boxed{2}}X_{\boxed{1}} \tag{B2}$$

$$X \longrightarrow e \;/\; f \tag{B3}$$

In Rule B1 we define symbols $e \in E \cup \varepsilon$ and $f \in F \cup \varepsilon$. This is really a template rule; instances of it capture word-to-word alignments.

A potential drawback of using bracketing grammar as our model of translational equivalence is that we may encounter a sentence pair for which the model cannot express the reordering relationship. A canonical example of this is the so-called inside-outside alignment, which is illustrated in Figure 9. Wu (1995b) argued that such reorderings do not occur in real data. However, Wellington et al. (2006b) recently examples in real data. Zens and Ney (2003) show empirically that bracketing grammars can represent most reorderings found in real data, and specifically, that they can represent more reorderings than a finite state model using typical constraints. In general, any alignment can be expressed in a grammar with a sufficient number of nonterminal (Aho and Ullman, 1969). The tradeoff for this is increased complexity – in particular, complexity is exponential in the number of coindexed nonterminals.

A *lexicalized* bracketing grammar, in which nonterminal symbols are annotated with words, is described in Zhang and Gildea (2005). A related formalism is the head-transduction grammar (Alshawi et al., 2000).

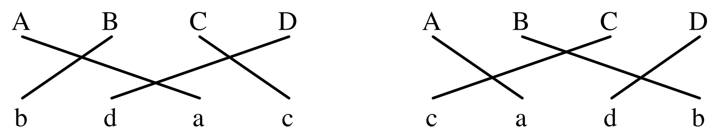


Figure 9: Visualization of the so-called inside-outside alignments that are not possible using bracketing transduction grammar. Due to the interleaving words in these configurations, we cannot construct a binary-branching SCFG that is isomorphic for these strings, although a SCFG with four nonterminals can produce them (Aho and Ullman, 1969). These are the smallest impossible configurations in this grammar; as the number of words in each string increases, so does the number of impossible configurations.

2.2.2 Syntax-Based Grammars

One possible advantage of using SCFG is that it allows us to easily incorporate knowledge based on natural language syntax. This follows from developments in syntactic modeling for ASR (Chelba and Jelinek, 1998).

Often, we will have meaningful linguistic grammars only for one language. [5] The monolingual syntax will look much like our example fragment CFG (Rules C1-C8). In order to use this monolingual syntax in an SMT model, we construct an SCFG where productions mirror the known syntax; in the other language, we allow arbitrary reorderings of these symbols (Wu and Wong, 1998; Yamada and Knight, 2001). The objective of such a SCFG is to require that reordering observe the phrasal cohesion constraint imposed by linguistic syntax. This is called a *syntax-based grammar*. The example derivation from Figure 7 is an illustration of this.

2.2.3 Hierarchical Phrase-Based Translation

The SCFGs that we have so far described differ radically from FST models in their characterization of reordering between strings. However, they share a key weakness of word-based FST-models: they only allow word-to-word translation, which requires a reordering decision for each word. Each decision leads to more possibility for error. Ideally, we would like to benefit from the insights behind both hierarchical models and phrase-based models. This is accomplished in hierarchical phrase-based translation (Chiang, 2005, 2007; Chiang et al., 2005).

In this grammar, no linguistic syntax is required. A single undifferentiated non-terminal X is used in the main productions, and only two such nonterminals may appear on the right-hand side of any production, just as in a bracketing grammar. However, unlike a bracketing grammar, the right-hand side may also contain a number of terminal symbols in both languages. This corresponds to the basic insight of phrase-based translation, in that each rule presents a mapping between strings of words. Essentially, the rules represent phrases that may be reordered recursively. This is illustrated in the following grammar fragment.

[5]Usually this is the target language (Wu and Wong, 1998; Yamada and Knight, 2002). This imbalance reflects the fact that many of the translation systems reported in the literature are designed to translate into well-studied languages, such as English, for which we already have high-quality syntactic parsers.

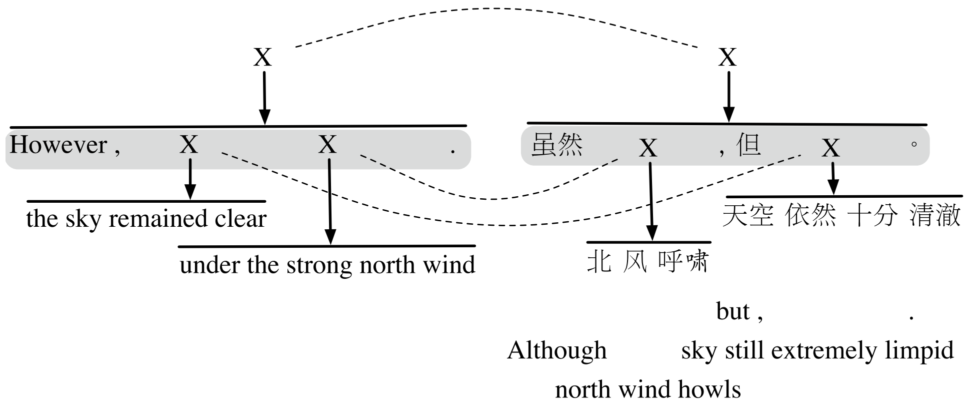


Figure 10: Visualization of hierarchical phrase-based translation. Each arrow corresponds to a grammar production. Rules that involve reordering are highlighted. Glosses of the Chinese sentence fragments are shown beneath the Chinese derivation.

$$X \longrightarrow \text{However , } X_{\boxed{1}}X_{\boxed{2}} \text{ . /虽然 } X_{\boxed{2}} \text{ , 但 } X_{\boxed{1}} \text{ 。} \tag{H1}$$
$$X \longrightarrow \text{under the strong north wind /北 风 呼啸} \tag{H2}$$
$$X \longrightarrow \text{the sky remained clear /天空 依然 十分 清澈} \tag{H3}$$

In this grammar, recursivity is captured in Rule H1. Note that, as with FST phrase-based models, explicit word alignments are not specified in this model, although it implies that there may be some internal alignment between words appearing in the same grammar rule. A derivation is illustrated in Figure 10.

2.2.4 Syntactic Phrase-Based Models

Recently, several models have been proposed that combine the advantages of hierarchical reordering, syntax, and phrases (Galley et al., 2006; Marcu et al., 2006; Quirk et al., 2005). These models are similar to hierarchical phrase-based models. However, in addition to using the syntactic structure of SCFG, they employ lingustic syntax as in syntax-based models (§2.2.2). Although there are differences between various models, they all involve multi-word translation rules, as in phrase-based translation models. These are decorated with fragments of linguistic syntax, which are used to constrain the possible reorderings of the phrases.

2.2.5 Alternative Linguistic Models

A wide variety of linguistic theories are computationally equivalent to CFG, and these can be used as the basis for translation using SCFG. Head transducers may be seen as a form of synchronous *dependency grammar* Alshawi et al. (2000). In dependency grammar, the nodes of the rooted tree which describes the sentence structure are also the words of the sentence. It is possible to derive transformations that will convert many dependency grammars to context-free grammars, and vice versa (e.g. Collins et al., 1999). Therefore, we can construct SCFGs that correspond to dependency grammar (Melamed, 2003). Translation models based on dependency grammar are described in Gildea (2004) and Quirk et al. (2005).

2.3 Other Models of Translational Equivalence

Moving up the hierarchy of formal languages, there are synchronous models based on language formalisms more powerful than context-free languages. A good example is tree-adjoining grammar (Joshi and Schabes, 1997), which we can generalize to synchronous tree-adjoining grammar (STAG) (Shieber and Schabes, 1990). Formally, TAG is a member of a large class of formalisms known as linear context-free rewriting systems (LCFRS) (Joshi et al., 1991; Vijay-Shanker et al., 1987). These formalisms can parse a restricted subset of context-sensitive languages in polynomial time. Much of the work in this area is currently theoretical. However, *Generalized Multitext Grammar* (Melamed et al., 2004), and LCFRS, is used as the basis for an SMT system (Burbank et al., 2005).

3 Mathematical Modeling

Translational equivalence models allow us to enumerate possible structural relationships between pairs of strings. However, even with the constraints of a strict translational equivalence model, the ambiguity of natural language results in a very large number of possible target strings for any input source string. Obviously, our translation system will need some mechanism to choose the correct target string.

This mechanism comes in the form of a *mathematical model*, which is a function that assigns a real-valued score to any pair of source and target sequences. The general forms of these models are familiar from other machine learning problems. There is a vast number of approaches; we will only briefly describe the most common ones here. For more detail, the reader is referred to a general text on machine learning, such as Mitchell (1997).

In typical machine learning problems, we are given an input $y \in Y$, and the goal is to find the best output $x \in X$. Note that the values x and y may be complex. We introduce a function $f : X \times Y \rightarrow \mathbb{R}$ that maps input and output pairs to a real-valued score that is used to rank possible outputs. This model may be probabilistic, meaning that we constrain it in one of two ways. We introduce the *random variables* $\mathbf{x}$ and $\mathbf{y}$ which range over the sets X and Y, respectively. Let $x \in X$ and $y \in Y$ be specific values drawn from these sets. In a *joint model*, denoted $P(\mathbf{x}, \mathbf{y})$, we introduce the constraints

$$\sum_{(x,y)\in\{X\times Y\}} P(\mathbf{x} = x, \mathbf{y} = y) = 1$$
$$\forall_{(x,y)\in\{X\times Y\}} P(\mathbf{x} = x, \mathbf{y} = y) \in [0, 1]$$

The value $P(\mathbf{x} = x, \mathbf{y} = y)$ is the *joint probability* of the assignments $\mathbf{x} = x$ and $\mathbf{y} = y$ occurring, out of all possible combinations of assignments to these variables. We will often abbreviate this $P(x, y)$.

In a *conditional model*, denoted $P(\mathbf{x}|\mathbf{y})$, we introduce the constraints

$$\forall_{y\in Y} \sum_{x\in X} P(\mathbf{x} = x|\mathbf{y} = y) = 1$$
$$\forall_{(x,y)\in\{X\times Y\}} P(\mathbf{x} = x|\mathbf{y} = y) \in [0, 1]$$

The conditional probability $P(\mathbf{x} = x|\mathbf{y} = y)$ – abbreviated $P(x|y)$ – is simply the probability of the assignment $\mathbf{x} = x$, given that the assignment $\mathbf{y} = y$ is fixed. In this case, we assume that knowledge of the value assigned to $\mathbf{x}$ will help us determine the assignment to $\mathbf{x}$.

These constraints represent the *distribution* of finite probability mass across all combinations of assignments to $\mathbf{x}$ and $\mathbf{y}$. In many machine learning problems, it is not unusual for the input set Y to be complex. Often, the set X of possible outputs is a small, finite set of *labels* or *classes* and our goal is simply to find the best *classification* of the input. This is not the case in SMT, where our input **f** ranges over F^* and our output **e** ranges over E^*. We will usually expand our definition of the output to include the decisions made by our translational equivalence model defining the relationship between $\mathbf{f} = f_1^J$ and $\mathbf{e} = e_1^I$. Let's denote this structure using the variable $\mathbf{d} = d_1^M \subset D$. Recall that D is a set of transitions in the case of FST models (§2.1) and a set of grammar productions in the case of SCFG models (§2.2) – in other words, D is simply a set of rules. In the SMT problem, we are given an input f_1^J and we are interested in finding a "class" (e_1^I, d_1^M) with complex internal structure, which comes from a set that is exponential in the size of the input. This problem is known as *structured classification* or *structured prediction* (Taskar, 2004). Note that the set d_1^M of derivations exactly defines e_1^I for a given input f_1^J, and we could simply denote the label using d_1^M; we denote the label using (e_1^I, d_1^M) because this more intuitive. For notational purposes we also define a predicate $Y(e_1^I, d_1^M)$ that is true when the set d_1^M yields e_1^I, and false otherwise.

The mathematical function that we are truly interested in is $P(\mathbf{e}|\mathbf{f})$. This function ignores the structure given by our translational equivalence models. However, these models usually define multiple structures that can relate the same pair (e_1^I, f_1^J). This gives us:

$$P(\mathbf{e}|\mathbf{f}) = \sum_{\mathbf{d}:Y(\mathbf{d},\mathbf{e})} P(\mathbf{e}, \mathbf{d}|\mathbf{f}) \tag{1}$$

Unfortunately, this sum is exponential in both FST (Brown et al., 1993) and SCFG models (Melamed, 2004a). Therefore we will use the simpler function $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ for classification. We can view the classification problem as one in which the decoder produces candidate labels according to our translational equivalence model, and the mathematical model is used to rank these candidates. Usually these tasks are integrated, but not necessarily (see §5.3).

Although the function $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ ranges over discrete sets, these sets are too large for us to directly enumerate them. This poses both practical and theoretical problems: we cannot enumerate the function to store its value on disk; and furthermore, we will not have enough training data to reliably learn what these values should be. The goal of mathematical modeling, then, is to define the function in such a way that we can enumerate its values, efficiently learn them, and store them on a disk or in memory. In SMT, there are two broad classes of models that accomplish this: *generative models*, which we describe in §3.1, and *discriminative models*, which we describe in §3.2.

3.1 Generative Models

One method of manipulating probabilistic models is through use of the chain rule:

$$P(\mathbf{x}, \mathbf{y}) = P(\mathbf{x}|\mathbf{y})P(\mathbf{y}) \tag{2}$$

In generative models, we decompose $P(\mathbf{x}|\mathbf{y})$ using Bayes' rule, which we derive using the chain rule and a bit of algebra as shown in Equation 4.

$$P(\mathbf{x}|\mathbf{y}) = \frac{P(\mathbf{x}, \mathbf{y})}{P(\mathbf{y})} = \frac{P(\mathbf{y}|\mathbf{x})P(\mathbf{x})}{P(\mathbf{y})} \tag{3}$$

Applying Bayes' rule to our structured classification problem, we arrive at the following decomposition:

$$P(\mathbf{e}, \mathbf{d}|\mathbf{f}) = \frac{P(\mathbf{f}, \mathbf{d}|\mathbf{e})P(\mathbf{e})}{P(\mathbf{f})} \tag{4}$$

During the decoding step, we will be able to ignore the denominator $P(\mathbf{f})$ in Equation 4 because it is constant for any input f_1^J. Therefore, we don't need to consider this model any further and we can focus on $P(\mathbf{f}, \mathbf{d}|\mathbf{e})$ and $P(\mathbf{e})$. The decomposition is identical to the successful approach used in automatic speech recognition (ASR) (Brown et al., 1990).

In SMT we call $P(\mathbf{e})$ the *language model* and $P(\mathbf{f}, \mathbf{d}|\mathbf{e})$ the *translation model*. Note that while our objective is to discover e_1^I given f_1^J, we actually model the reverse. This originated with the IBM system, and it is for this reason that IBM Model 4 is described as a translation from e_1^I to f_1^J (§2.1.1) (Brown et al., 1993). The advantage of this over modeling $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ directly is that we can apply two independent models to the disambiguation of $\mathbf{e}$ (Brown et al., 1990). This is beneficial because our estimates for each model are likely to contain errors; and by applying them together we hope that each can counterbalance the errors of the other. [6]

3.1.1 Translation Models

In translation modeling, our goal is to find a tractable representation of the function $P(f_1^J, d_1^M|e_1^I)$. Generative models use probabilistic tools for this. One of these is chain rule, which allows us to write:

$$P(f_1^J, d_1^M|e_1^I) = \prod_{j=1}^{J} P(f_j|f_1^j, d_1^M, e_1^I) \times \prod_{m=1}^{M} P(d_m|d_1^m, e_1^I) \tag{5}$$

This equation tells us that the conditional probability of the pair (f_1^J, d_1^M) with respect to e_1^I is simply the product of many small probabilities, each of which corresponds to a single action taken by our translational equivalence model. Thus, in our FST model, there is a probability for each transition in each transducer; in our SCFG model, there is a probability for each production. In general, we can say that there is a probability associated with every arrow in the figures in §2. Such models are called *weighted*, giving us weighted FST (WFST) and weighted SCFG (WSCFG), respectively.

We can also see from Equation 5 why such models are called generative models. In a generative model, we are given a probability for every event that occurs in our data – including our input data. We can think of the model as a stochastic process that generated the data. In fact, we can think of the language model $P(\mathbf{e})$ as a stochastic model that generates target language sentences, and the translation model $P(\mathbf{f}, \mathbf{d}|\mathbf{e})$ as a second stochastic process that "corrupts" the target language to produce source language sentences. Although first implemented in the IBM system, this idea was in fact anticipated much earlier by Weaver:

> 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." (Weaver, 1955)

Weaver makes analogy to information theoretic work on signal transmission over a physical medium, called the *noisy channel* problem. It is illustrated in Figure 11. Following Weaver, the process to recover e_1^I is called *decoding*.

Equation 5 helps to simplify our problem, but not completely. Even a simpler function such as $P(d_1|e_1^I)$ still has a domain that is too large for us to deal with. In order to simplify these functions, we introduce the idea of *conditional independence*. When we say that a variable $\mathbf{x}$ is conditionally

[6] In fact, recent empirical work has shown that the values learned by common parameter estimation methods are much less precise than common floating point representations – Och (2005) and Federico and Bertoldi (2006) show that they can be stored in four bits without loss of accuracy. There are probably learning-theoretic reasons for this, but this issue is as of yet unexplored in the SMT literature.

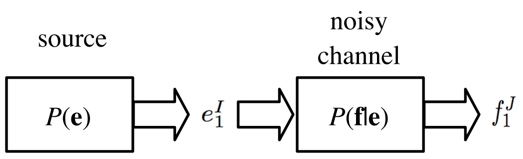


Figure 11: The source-channel model of sentence pair generation. The source model $P(\mathbf{e})$ produces e_1^I, which is then transformed by the channel model $P(\mathbf{f}|\mathbf{e})$ to produce f_1^J. If we are given only f_1^J, we can try to deduce an e_1^I using our knowledge of both models.

independent of $\mathbf{y}$, we mean that $P(\mathbf{x}|\mathbf{y}) = P(\mathbf{y})$. In other words, conditional independence means that knowing the value of $\mathbf{y}$ does not affect the probability distribution of $\mathbf{x}$. By making independence assumptions about our data, we can drop enough terms from our functions that they become tractable. The obvious danger, of course, is that if we make too many or wrong independence assumptions, our resulting probability distributions will be incorrect. This is a constant danger in SMT modeling. In general, nearly any independence assumption will be unrealistic; our hope is that they are approximately close enough to that it will not harm our results. Making conditional independence assumptions is a matter of art and pragmatism.

We will use the notation $P_\delta(\mathbf{x}|\mathbf{y})$ to represent the distribution $P(\mathbf{x}|\mathbf{y})$ that has been rewritten to include only elements of $\mathbf{y}$ on which $\mathbf{x}$ is conditionally dependent. Using this notation, we can rewrite Equation 5 as:

$$P(f_1^J, d_1^M|e_1^I) = \prod_{j=1}^{J} P_\delta(f_j|f_1^j, d_1^M, e_1^I) \times \prod_{m=1}^{M} P_\delta(d_m|d_1^m, e_1^I) \tag{6}$$

Assuming that we have made sufficient independence assumptions, each term on the right side of this equation ranges over a sufficiently small sets that we can actually learn them. At runtime, we will simply look up the values associated with the terms and use them to compute the function. Each of these values is a *parameter*; and the associated model is a *parameterization* of $P(f_1^J, d_1^M|e_1^I)$. In other words, a parameter is an element of the smallest distribution that we represent in our models – a distribution that we do not represent as a function of other distributions. We will use $p(x, y)$ or $p(x|y)$ to denote parameters.

We do not have space to describe all of the parameterizations that have been proposed for even the small selection of translational equivalence models we described in §2. However, there are a few parameters that are found in many generative models, so we will describe them here.

One parameter that appears in most translation models is the *word translation probability* (Brown et al., 1993). If the ith word e_i of e_1^I is aligned to the jth word f_j of f_1^J in our translational equivalence model, then the associated parameter is $p(f_j|e_i)$. This is relatively intuitive – we can think of this parameter representing the probability that a translator, when presented with word e_i, will choose to translate it using word f_j. Notice that the distribution ranges over $\{E \times F\}$. Although our parameter estimation methods (§4) will guarantee that many of the values in this distribution are 0, meaning that we do not need to store them, it should be apparent that in most implementations the associated the table will dominate disk and memory usage in our implementation.

The analogue to the word translation probability in phrase-based models is the *phrase translation probability*. It is popular to represent this distribution directly as $p(\tilde{f}_i|\tilde{e}_j)$ for the phrase pair $(\tilde{f}_i|\tilde{e}_j)$ (Koehn et al., 2003; Marcu and Wong, 2002). This poses representational and computational problems. In principle, it means that we must store a table ranging over $\{E^* \times F^*\}$ – the problem we were trying to avoid in the first place with our conditional independence assumptions. A common solution is to limit phrase length to some arbitrary limit L, although this still leaves us with a very large potential table size $O(E^L F^L)$, so L is usually chosen to be small (Koehn et al., 2003; Marcu and Wong, 2002). A recently proposed alternative is to compactly represent the training data in memory using a suffix array and extract phrase pairs and compute probabilities as needed, although this results in a speed tradeoff during decoding (Callison-Burch et al., 2005; Zhang and Vogel, 2005). Zens and Ney (2007) present an alternative data structure that allows phrase tables to be efficiently accessed even when they are too large to fit in main memory and must be kept on disk.

Another parameter based on the IBM Models is the *fertility probability* (Brown et al., 1993). Recall from our discussion of IBM Model 4 that each target word e_i chooses a number of source words ϕ_i to which it will correspond. We can represent this using the parameter $p(\phi_i|e_i)$. This implies that we must arbitrarily bound the fertility in order to store the associated parameters (Brown et al., 1993). The concept of fertility persists in some more recent translation models.

In generative models based on FST, we must parameterize the permutation decisions in some way. A common way to do this is to parameterize based on the distance between the target words that are aligned to adjacent source words – the so-called *distortion*. If we denote using a_j the location of the target word e_{a_j} that is aligned to f_j in our translational equivalence model, then we say that the probability of word f_j appearing in position j of the permutation is $P(a_j = i) = p(a_j - a_{j-1})$. Numerous other parameterizations of this *distortion probability* have been proposed (Al-Onaizan and Papineni, 2006; Lopez and Resnik, 2005; Och and Ney, 2000b; Toutanova et al., 2002; Vogel et al., 1996). One variation is to make the probability conditionally dependent on a word class learned using unsupervised clustering methods (Brown et al., 1992; Och, 1999). IBM Model 4 uses a particularly complex variation based on target word distortion instead of source word distortion. Some phrase-based models also use a variant of distortion (Koehn et al., 2003; Marcu and Wong, 2002).

In generative models based on SCFG, we must assign probabilities for each synchronous grammar rule. Any number of parameterizations are available, often using methods from CFG parsing models (see, e.g. Marcu et al., 2006).

3.1.2 Language Models

Language modeling for ASR is surveyed in Chen and Goodman (1998) and Rosenfeld (2000). Even more background can be found in (Jelinek, 1998). Language modeling has not received much attention in the SMT community, which has preferred to focus on the more specialized translation models. Most systems use smoothed n-gram models (§2.1), although syntax-based SCFG models of translational equivalence are usually constructed to take advantage of a CFG language models (Charniak et al., 2003; Wu and Wong, 1998). An example is given in Marcu et al. (2006).

Language models continue to be the subject of substantial active research, particularly for ASR. Although many SMT systems use models that were popular in ASR several years ago, recent empirical results show that SMT can benefit from the use of state-of-the-art methods in language modeling (Eck et al., 2004; Kirchhoff and Yang, 2005; Och, 2005; Zhang et al., 2006b).

3.2 Discriminative Models

Generative models are useful in decoding (§5) because they correspond so closely to the translational equivalence models that define the search space. However, there are tradeoffs. As we have seen, to make them both theoretically well-founded and tractable, we must make very strong independence assumptions. This means that the information that we can bring to bear at each individual decision point is very limited. For instance, we are generally limited to translation between small numbers of words in each sentence, although we expect in principle that knowledge of all words in a source sentence may help us translate any particular word. Using generative models, there is no tractable mechanism to represent this.

We can bring additional context into modeling by moving from generative to *discriminative* models. In SMT, a convenient form for this is *log-linear* modeling (Berger et al., 1996b; Och and Ney, 2002). The introduction of log-linear models to SMT follows from their increasing use in NLP (Berger et al., 1996b; Ratnaparkhi, 1998), and reflects general trends in machine learning.

Log-linear models define a relationship between a set of K fixed *features* $h_1^K(\mathbf{e}, \mathbf{d}, \mathbf{f})$ of the data and the function $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ that we are interested in. A feature can be any function $h : E^* \times D^* \times F^* \longrightarrow [0, \infty)$, that maps every pair of input and output strings to a non-negative value. An example of a feature might be the number of times a particular word pair (e_i, f_j) appears in the data (Och and Ney, 2002); the number of phrases in a segmentation of e_1^I (Koehn, 2004c); or the logarithm of the probability defined by a generative model (or even one of its distributions) from the previous section. Most features used in SMT to date have taken the latter form. Log-linear models take the form of Equation 7.

$$P(\mathbf{e}, \mathbf{d}|\mathbf{f}) = \frac{\exp \sum_{k=1}^{K} \lambda_k h_k(\mathbf{e}, \mathbf{d}, \mathbf{f})}{\sum_{\mathbf{e}',\mathbf{d}':Y(\mathbf{e}',\mathbf{d}')} \exp \sum_{k=1}^{K} \lambda_k h_k(\mathbf{e}', \mathbf{d}', \mathbf{f})} \tag{7}$$

The daunting normalization factor in the denominator is required only to make the function a well-formed probability. This will be important during parameter estimation, but not during decoding, where we can ignore it because it constant for any given f_1^J.

The log-linear model defined in Equation 7 has K parameters, λ_1^K. These are called *feature weights* or *model scaling factors*. They determine the contribution of a feature to the overall value of $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$. Essentially, each parameter tells us the pairwise correspondence between the feature and the output probability. A positive value λ_k indicates that the feature $h_1^K(\mathbf{e}, \mathbf{d}, \mathbf{f})$ correlates with $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$; a negative value indicates an inverse correlation; and a value near zero indicates that the feature is not a useful predictor of $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$. Notice that Equation 4 is a special case of Equation 7 when the following conditions hold:

$$\begin{aligned}
K &= 2 \\
\lambda_1 &= \lambda_2 = 1 \\
h_1(\mathbf{e}, \mathbf{f}) &= \log P(\mathbf{e}, \mathbf{d}|\mathbf{f}) \\
h_2(\mathbf{e}, \mathbf{f}) &= \log P(\mathbf{e})
\end{aligned}$$

Log-linear models *discriminate* between different possible values e_1^I when presented with a particular f_1^J. In contrast with generative models, there is no requirement that we assign a single probability to every element of data. We may assign multiple probabilities to an element, or none at all. In fact, the values that we assign are not required to be well-formed probabilities at all – the normalization factor in Equation 7 takes care of this for us. In particular, we are not required to define probabilities for our input data as we do in generative models. Because we are freed from the constraints of Bayes' rule, features can be overlapping – we could, for instance, use several of the generative models discussed previously, even though each of them will have a different explanation of each word in the target language. In principle, any other model of $P(\mathbf{e})$, $P(\mathbf{f}|\mathbf{e})$, $P(\mathbf{e}|\mathbf{f})$, or $P(\mathbf{e}, \mathbf{f})$, or combination thereof, can be a feature. [7]

[7] A justification for using log-linear models in this way was that a system based on $P(\mathbf{e}) \cdot P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ worked nearly as well as $P(\mathbf{e}) \cdot P(\mathbf{f}, \mathbf{d}|\mathbf{e})$ in empirical studies, even though the former cannot be theoretically motivated using Bayes' rule (Och and Ney, 2002; Och et al., 1999). Beginning with (Koehn, 2004a), several recent systems use both (e.g. Chiang, 2007; Simard et al., 2005). It is not clear if this is beneficial (Lopez and Resnik, 2006).

Discriminative modeling is powerful because it frees us from the generative modeling requirement that each term must conform to an event in our translational equivalence model, which is often chosen for computational reasons rather than for its ability to distinguish between good translations. This allows us to define arbitrary features that may help to improve translation. The primary art in discriminative modeling is defining useful features. However, with a few exceptions (e.g. Och et al., 2004) this area has been largely ignored or has been a secondary focus (Liang et al., 2006a; Marcu et al., 2006; Venugopal et al., 2007).

4 Parameter Estimation

Once we have defined $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$, we need to assign values to its parameters, the actual values that are used to compute it. We call this *parameter estimation*. In SMT, we use a parallel corpus as input to a machine learning algorithm in order to learn the parameter values. Broadly speaking, we can say that SMT relies on *supervised learning*, because we are given samples of input/output pairs.

Most SMT systems use a log-linear model of $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ that incorporates generative models as feature functions. Before we can learn the parameters of the log-linear model, we must fix values of the feature functions, including any generative models used as features. This means that we must first estimate any underlying generative models independently, and then separately estimate the parameters of the log-linear models. An alternative approach is presented by Fraser and Marcu (2006a).

We describe parameter estimation for generative models in §4.1. We will then discuss the important concept of word alignment in §4.2. Finally, we describe parameter estimation for log-linear models in §4.3.

4.1 Parameter Estimation in Generative Models

An example parameter from our generative models is the translation probability $p(\text{风}|\text{wind})$. Our model says that we will use this value whenever we translate the word "风" as "wind". Before we can use the value, we must determine what it is. We must do this for all of the parameters in our model – and as we have seen, there are likely to be millions or billions of them.

Fortunately, we have some information available to us in the form of a parallel corpus. One way to capitalize on this data comes to us from statistical estimation theory. We assume that the parallel corpus was produced by our model using the unknown, true parameter values. Our goal, then, is to *estimate* those values so that our estimates are as close as possible to the true ones.

If we denote our training data as $C \subset \{E^* \times F^*\}$, the complete set of parameters as Θ, and the probability (or likelihood) of C under parameter set Θ as $P_\Theta(C)$, then our goal is to choose Θ that satisfies Equation 8.

$$\Theta = \underset{\hat{\Theta}}{\operatorname{argmax}}\, P_{\hat{\Theta}}(C). \tag{8}$$

Parameter estimation, then, is equivalent to finding the maximum of a function (the *objective function*) – in this case, the *likelihood* function $P_\Theta(C)$. We call this *maximum likelihood estimation* (MLE). The parameter set Θ that satisfies the maximization problem in Equation 8 is the *maximum likelihood estimate*. It is important to note here that this is not the only possible objective function, although it is the one that is typically used for generative models. We will discuss a different objective function in §4.3.1.

4.1.1 Learning Word Translation Probabilities

MLE is easy when we can observe all of the events for which we wish to estimate probabilities. Consider a very simple model, the coin-flip model. In this model, we have a coin that comes up heads with probability $p(h)$. We don't know $p(h)$ and we would like to guess what it is. If we have access to the coin itself, we can flip it a number of times and see how many times each side comes up. Suppose that we flip the coin a number of times, and we count the number of times it comes up heads, which we denote $\#(h)$. The total number of flips is the sum of the number of heads and tails, $\#(h+t)$. Most people know intuitively that the value for $p(h)$ should be $\#(h)/\#(h+t)$. In fact, we can show analytically that this relative frequency estimate corresponds to the MLE. It is worth noting that the MLE requires a sufficient number of samples for this method to be accurate – we can see that if we flip the coin only once, then the only possible outcomes are $p(h) = 1$ and $p(h) = 0$, either of which is likely to be wrong. This issue has not received much attention in SMT, although Foster et al. (2006) show that methods to *smooth* poorly estimated probabilities can improve performance. [8]

Now suppose that we wish to estimate the parameters of a word-based generative translation model. If we had access to an alignment –such as the one depicted in Figure 2 – for every sentence in our corpus, then it would be easy to observe the fertility, substitution, and distortion for each word, and we could estimate our parameters as easily as we did in the coin-flipping model. For instance, if we saw the word "wind" #(wind) times, and it was aligned to the word "风" $\#(a(\text{风},\text{wind}))$ times, then we would compute $p(\text{风}|\text{wind}) = \#(a(\text{风},\text{wind}))/\#(\text{wind})$.

Unfortunately, we will discover that the human translators who produced our training data have neglected to include alignment information. So, while we can see that "wind" and "风" both *cooccur* in many sentence pairs, we cannot see how many times they are actually aligned to each other. With only the parallel text available to us, our initial translational equivalence model will define many ways in which any sentence can be related, and only in some of these will "wind" be aligned to "风", so we cannot simply assume this alignment. We can make some estimates based only on coocurrence – for instance, we will estimate $p(f|e) = 0$ for words f and e that never cooccur in our training data.

One solution to this problem is to automatically generate an alignment, and then to use this alignment as our training data for maximum likelihood estimation. We will describe word alignment methods in §4.2. Alternatively, we need a method to estimate our parameters that will work even when we cannot explicitly count all of the events that occur in the translational equivalence model relating e_1^I and f_1^J. Since we will not be able to directly observe the outcome of each decision made by our model, we can view the learning of the associated parameters as a form of *unsupervised learning*. [9] A method that is commonly used to solve this problem in SMT is the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). EM is a hill-climbing algorithm that works by substituting observed counts of events with *expected counts*. Although we cannot actually observe the number of times that "wind" aligns to "风", we can compute the expected number of times that this happens if we assume some initial value for Θ, which we will call Θ_0. We can then compute the expected count of the event $E(a(\text{风},\text{wind}))$ in any particular sentence pair $(\mathbf{e}, \mathbf{f})$ as follows.

$$E(a(\text{风}, \text{wind})) = \frac{P_{\Theta_0}(a(\text{风}, \text{wind}), f_1^J|e_1^I)}{P_{\Theta_0}(f_1^J|e_1^I)}$$

In short, we compute all possible alignments between f_1^J and e_1^I, and their probabilities under Θ_0. We

[8]A discussion of smoothing is highly relevant to most NLP problems, but well beyond the scope of this paper. Chen and Goodman (1998), Manning and Schütze (1999, Chap. 6), and Jurafsky and Martin (2000, §6.3) and are good starting points.

[9]In practice, we will usually have access to a small set of manually aligned data, developed for evaluation purposes (§4.2.5). It is typical to test our parameter learning algorithm on an even smaller subset of this data, called the *development data*. Often a small set of tuning parameters is modified based on evaluations against this data. This constitutes a form of weak supervision.

then sum the probability of only the alignments that contain the event that we are interested in, and divide this by the sum of all possible alignments. If we apply this method to all parameters over the entire corpus, we can produce a new estimate, Θ_1, which can be used as a starting point for a new round of estimation. Thus, EM can be defined according to the following recursion, for our observed training data C and unknown alignment data D.

$$\Theta_i = \underset{\hat{\Theta}}{\operatorname{argmax}}\, P_{\hat{\Theta}}(C, E_{\Theta_{i-1}}(D)) \tag{9}$$

The EM algorithm does not, in general – and in particular for most SMT models – guarantee convergence to a globally optimal value for Θ. However, it is guaranteed to produce monotonically non-decreasing values for $P_\Theta(C)$, and thus, convergence to a local maximum. As a consequence, it depends crucially on a good initial estimate Θ_0 to avoid a poor local maximum. A method for generating this estimate is sketched in §4.2. Despite the various difficulties in using EM, it has been applied to a variety of other NLP problems (e.g. Lari and Young, 1990; Merialdo, 1994).

A full discussion of the EM algorithm is beyond the scope of this paper. For a more detailed overview, refer to Dempster et al. (1977). For a full account of the analytical solution to the EM algorithm in the case the IBM Models, refer to Brown et al. (1993).

4.1.2 Learning Phrase Translation Probabilities

In order to train the parameters of a phrase-based model we must have access to a *phrase-to-phrase alignment*. Unfortunately, the task of learning a phrase-to-phrase model directly is compounded by the dictionary size; while a word-based dictionary would contain potentially $O(E \cdot F)$ entries, a phrase-based SMT dictionary contains $O(E^L \cdot F^L)$ entries for some maximum phrase length L. Computing EM for phrase-based models requires the use of approximations and other substantial tradeoffs for reasons of tractability (Marcu and Wong, 2002). In addition, generalizability of such a model is much worse due to the small number of training samples relative to the size of the parameter space. An alternative solution which has emerged is to first automatically generate a word alignment (§4.2), and then count all phrases that are consistent with the alignment (Koehn et al., 2003; Och et al., 1999). We can then compute the MLE using the hypothesized phrases as our observed events. We say that a bilingual phrase is consistent with a word alignment if no word inside the phrase is aligned to any word outside the phrase. This is illustrated in Figure 12.

4.1.3 Learning Parameters of Generative SCFG Models

As we have seen, SCFG models usually contain a word translation probability, which can be learned using EM under the specific model (Wu, 1996; Yamada and Knight, 2001), or using word-translation probabilities learned from other alignments (§4.2) and a supervised learning method.

If we are using a syntax-based method, we may also need to estimate the monolingual syntactic probabilities. This can be done using standard methods from the field of natural language parsing. Melamed (2004a) presents a series of steps whereby parallel corpora and parsing statistics can be coordinated to learn the parameters of an arbitrary SCFG, using MLE (including EM) where necessary. Hierarchical phrase-based grammars can be learned using a supervised method similar to the one used for finite-state phrase-based models (Chiang, 2005, 2007).

4.2 Interlude: Word Alignment

As we have seen, we need a method of automatically generating word alignments as a precursor to some of our supervised learning methods in generative models. Even in the case of unsupervised learning, we

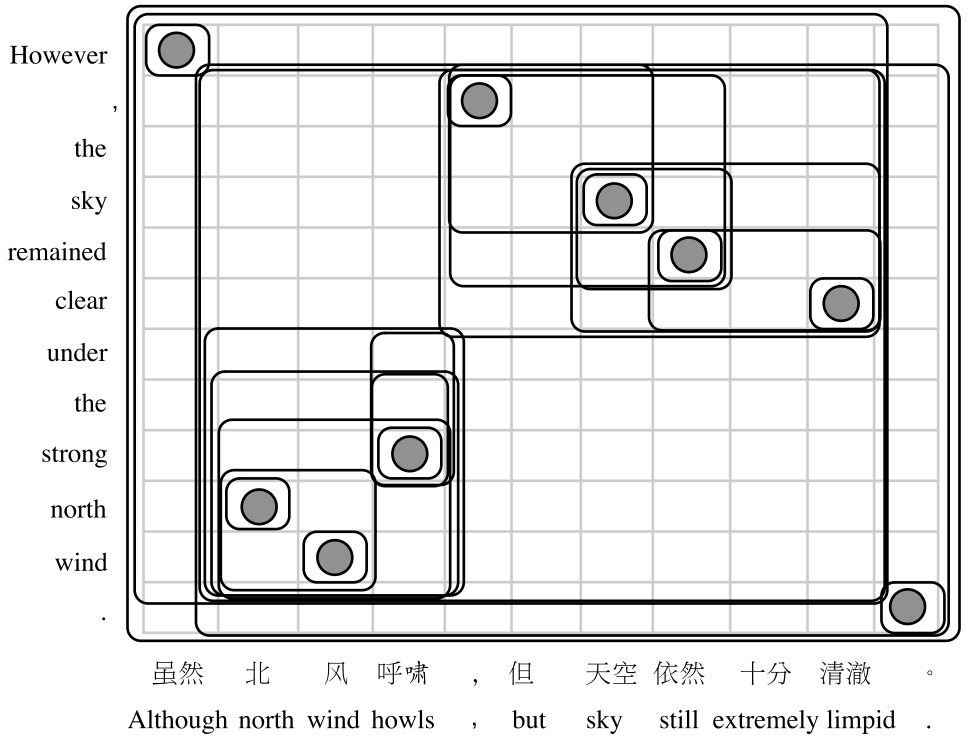

However
,
the
sky
remained
clear
under
the
strong
north
wind
.
虽然 北 风 呼啸 ， 但 天空 依然 十分 清澈 。
Although north wind howls , but sky still extremely limpid .

Figure 12: Supervised learning of phrases from word alignments. Here, we view each sentence pair on a grid. Word alignment is indicated by the presence of dark circles in the grid point corresponding to the word pair. The rectangles outline bilingual phrase pairs that are consistent with this word alignment.

find that a problem with applying the EM algorithm to complex models such as IBM Model 4 is that there is no efficient method to enumerate all of the possible alignments for a given sentence pair, which is required by the expectation step of EM. A solution to this is to generate some smaller, efficiently enumerable set of highly probable alignments as an approximation to the complete set, and sum over these instead. One way to generate this set is to take the neighborhood of alignments that is most similar to a high-probability alignment that we find using a simpler model. The neighborhood can be defined by a set of heuristic functions (Brown et al., 1993; Och and Ney, 2003). This solution is dependent on our ability to automatically find a good word alignment.

Word alignment is, in many ways, a microcosm of the translation problem: we must constrain the search space using some equivalence model, define a mathematical model, estimate parameters, and search for a good solution. The word alignment task can be viewed as a warm-up for decoding, since it is more constrained – in word alignment, we need only find a correspondence between sequences, whereas in decoding we will be required to find both the correspondence and the target sequence.

Over the past decade, a number of additional uses have been proposed for word alignment, including the automatic acquisition of bilingual dictionaries (Melamed, 1996) which can be used in cross-language information retrieval (Wang, 2005); and cross-lingual syntactic learning (Hwa et al., 2005; Lopez et al., 2002; Smith and Smith, 2004; Yarowsky et al., 2001). For this reason, word alignment has become a topic of significant study in its own right. The remainder of this section provides a brief overview of the word alignment task. We will return to the more general topic of parameter estimation in §4.3.

4.2.1 Formal Definition

Formally, we say that the objective of the word alignment task is to discover the word-to-word correspondences in a sentence pair (e_1^I, f_1^J). The alignment A of this pair is simply a set of these correspondences. We say that $A \subset \{1, 2, ..., I\} \times \{1, 2, ..., J\}$. If $(i, j) \in A$, then word e_i is aligned to word f_j. Models for word alignment depend on the way in which they decompose this problem.

4.2.2 Asymmetric Models

Recall that in our word-based translational equivalence model (§2.1.1) an asymmetry exists in the alignment between e_1^I and f_1^J. In particular, each source word f_j corresponds to one and only one target word (or null). The target words are unconstrained, and each can link to an arbitrary number of words (even zero), defined by its fertility. When we are generating an initial alignment to train this model, we must observe the same constraints.

In fact, we can exploit this asymmetry to produce an efficient alignment algorithm by modeling the alignment directly. To do this we introduce the alignment variable **a** to which we must assign a value a_1^J. In this representation each element a_j is a value in the range $\{0, 1, ..., I\}$. The value of a_j represents the position of the target word e_{a_j} to which f_j corresponds. By making the very strong assumption that each variable a_j is independent, we arrive at Equation 10, which tells us how to find the optimal alignment **a**.

$$\mathbf{a} = \underset{a_1^J}{\text{argmax}} \prod_{j=1}^{J} p(a_j) \cdot p(f_j|e_{a_j}) \tag{10}$$

This is the form of IBM Model 1 (Brown et al., 1993). If we make the additional simplifying assumption that the distribution $p(a_j)$ is uniform, the only parameters that are required in order to compute the optimal alignment are the word translation parameters $p(f_j|e_i)$. On first glance, this may not seem to be much of an improvement over the situation in IBM Model 4, since we still must learn the values in the largest table ranging over $\{E \times F\}$. However, notice that our independence assumptions reduce the model to a set of J independent decisions with $I + 1$ possible outcomes. In this simple form, the space of all possible alignments can be compactly represented, and the EM search is guaranteed to converge to a single solution (Brown et al., 1993). Although this convergence will guarantee an optimal value for $P_\Theta(C)$, this optimal value may not produce the best alignments in practice, because maximizing likelihood does not necessarily guarantee a reduction in error. This is particularly true if the model makes too many independence assumptions, as Model 1 does. Moore (2004) proposes an alternative method of Model 1 parameter estimation that produces better results in practice.

Notice that we can compute $P(\mathbf{f}|\mathbf{e})$ as a sum over Model 1 alignments as follows:

$$P(\mathbf{f}|\mathbf{e}) = \prod_{j=1}^{J} \sum_{a_j=0}^{I} p(a_j) \cdot p(f_j|e_{a_j}) \tag{11}$$

Thus Model 1 is a translation model, although it will not produce very good translations on its own (Knight, 1999a). However, it is useful as a feature function in log-linear models, most likely because it computes a correspondence between all words in **e** and all words in **f** (Lopez and Resnik, 2006; Och et al., 2004).

We obtain better alignments when we move to a first-order dependency between the alignment variables, as in Equation 12 (Vogel et al., 1996).

$$a = \operatorname*{argmax}_{a_1^J} \prod_{j=1}^{J} p(a_j|a_{j-1}) \cdot p(f_j|e_{a_j}) \tag{12}$$

Equation 12 is in the basic form of the well-known Hidden Markov Model (HMM). HMMs have been applied to numerous problems in NLP, such as part-of-speech tagging (Merialdo, 1994). A key benefit of HMMs is the availability of standard algorithms for EM parameter estimation (Baum, 1972) and maximization (Viterbi, 1967). HMMs have been the subject of several studies in word alignment (Lopez and Resnik, 2005; Och and Ney, 2000b; Toutanova et al., 2002). In general, they are very accurate, significantly outperforming IBM Models 1, 2, and 3 in detailed empirical studies (Och and Ney, 2000b, 2003). HMMs are a common form of a *sequence model* that assigns a label to each element of a sequence. In the case of alignment, the sequence is the source sentence and the labels are the target words to which each source word corresponds. HMMs are generative models. A discriminative relative, the *conditional random field* has also been used for alignment (Blunsom and Cohn, 2006).

Finally, we can use IBM Model 4 itself to perform alignment. Search is done by first generating a good alignment with a simpler model, and then modifying it using hill-climbing techniques in conjunction with the IBM Model 4 parameters (Brown et al., 1993; Och and Ney, 2003). The translation parameters can also be imported from a simpler model; this makes IBM Model 4 highly dependent on the models used to bootstrap it (Lopez and Resnik, 2005; Och and Ney, 2003). Och and Ney (2003) note that the likely reason for the good performance of Model 4 is the first-order reordering dependence in the distortion parameters, and proposes combining it with the HMM, which has a complementary first-order dependence. This is accomplished by using both models as features in a log-linear framework.

Free implementations of IBM Model 4 are available for research purposes (Al-Onaizan et al., 1999; Och and Ney, 2003). Although its use as a translation model has diminished, its use as an alignment model remains widespread in many applications, including parameter learning for other models (Koehn et al., 2003; Och et al., 1999; Smith and Smith, 2004; Yarowsky and Ngai, 2001). Various improvements to Model 4 have been proposed (Dejean et al., 2003; Fraser and Marcu, 2006a).

4.2.3 Symmetric Alignment Models

The alignment models that we have described so far are asymmetric, following IBM Model 4. This is a necessity if we plan to train a translation model with a corresponding asymmetry. However, many applications of alignment allow symmetry, so we would like to allow symmetry in our alignments as well.

One approach to symmetric alignment is to align our training corpus twice using an asymmetric method, applying the asymmetry to each side in turn. We can then symmetrize by combining these alignments using one of several methods.These include set union, set intersection, and a number of heuristic methods, which usually begin with the intersection and proceed by iteratively adding links (Koehn et al., 2003; Och et al., 1999). Matusov et al. (2004) present a symmetric word alignment method based on linear combination of complementary asymmetric word alignment probabilities.

An alternative is to simply use an alignment algorithm that explicitly generates symmetric alignments. In this case, the alignment task corresponds to solving $I \cdot J$ binary decision problems: one for each potential link in the set A. The complexity of this space depends on any constraints we put on the links. With no constraints, the problem reduces to a set of binary decision problems and is tractable under a wide variety of models and learning algorithms (Ayan and Dorr, 2006b; Ayan et al., 2005a,b; Liang et al., 2006b). A common constraint is to require that each word in either sentence be linked exactly once, or to null (Melamed, 2000). This constraint produces an exponential space of allowable alignments because decisions are not independent of each other. A solution to this is to use a greedy search algorithm called competitive linking (Melamed, 2000). A number of cooccurrence-based correlation metrics have been used to score each link in this algorithm (Cherry and Lin, 2003; Gale and Church, 1991; Melamed, 2000; Moore, 2005b).

4.2.4 Supervised Learning for Alignment

Although the alignment learning methods that we have described so far depend on unsupervised learning of the alignment model parameters, it is possible to learn alignment models using supervised learning. Callison-Burch et al. (2004) construct an experiment showing that alignment with the IBM Models could be significantly improved with supervised learning. However, a primary limitation of supervised learning for alignment is that the number of sentences that have been aligned by human annotators is nearly always several orders of magnitude smaller than the number of unannotated sentences. Supervised learning algorithms must learn from a few hundred or thousand annotated sentences, in contrast with unsupervised learning where we typically have access to hundreds of thousands or millions of sentences. Therefore supervised learning of alignments is highly dependent on models which are sufficiently general, with a compact set of parameters. The solution to this is to use discriminative models with rich feature sets that do not depend heavily (or at all) on the specific identities of the words being aligned. In particular, it is unrealistic to expect such models to learn distributions with large discrete ranges, such as word-to-word probabilities, since we not have enough training data to populate the tables. However, we can use probabilities learned using unsupervised methods as features in a discriminative model.

In the last two years, numerous supervised, discriminative alignment models have been proposed, based on a wide variety of machine learning methods. These include transformation-based learning (Ayan et al., 2005b), neural networks (Ayan et al., 2005a), large margin methods (Taskar et al., 2005), perceptron learning (Moore, 2005a), and log-linear models (Fraser and Marcu, 2006a; Ittycheriah and Roukos, 2005). When annotated alignments are available, these methods outperform unsupervised methods.

4.2.5 Evaluation of Word Alignment

In SMT, the ultimate measure of word alignment is in its contribution to parameter estimation of our translation models. If the use of a particular word alignment improves our downstream translation results, we can conclude that the time spent developing our word alignment system was worthwhile.

Word alignment is also used for tasks other than SMT parameter estimation, and any number of other task-based evaluations could be established based on these applications. Although task-based evaluations are preferable, it is possible to evaluate word alignment intrinsically, by comparison with alignments prepared by human annotators. It is typical to select a few hundred sentences of a training corpus for this purpose; several examples in different language pairs have been produced (Melamed, 1998; Mihalcea and Pedersen, 2003; Och and Ney, 2000b). Ideally, each sentence is aligned by multiple annotators and the results are merged. Annotations may contain two sets of links: the *sure* set S, which contains only links about which all annotators are certain, and the *probable* set P, which may include links that were identified only by some annotators, or which represent links about which annotators were uncertain, in particular "idiomatic expressions, free translations, and missing function words" (Och and Ney, 2000b). It has recently been observed that probable annotations introduce undesirable bias in most standard alignment metrics (Fraser and Marcu, 2006b).

Given the set of hypothesized alignment links A, it is straightforward to compute the *precision* $|A \cap P|/|A|$ corresponding to the fraction of accurate links in the hypothesized alignment, and the *recall* $|A \cap S|/|S|$ corresponding to the fraction of "true" links were discovered by the alignment algorithm. [10]

[10] Precision and recall metrics are commonly used to evaluate NLP tasks in which the outcome is a set.

The most commonly used metric, which combines these statistics, is the *alignment error rate* (AER) given in Equation 13 (Och and Ney, 2000b).

$$AER = 1 - \frac{|S \cap A| + |P \cap A|}{|S| + |A|} \tag{13}$$

A similar metric was proposed by Ahrenberg et al. (2000). This type of evaluation may be used to ensure that quality alignments are used as input to supervised training methods.

Although intrinsic evaluation of word alignments has become popular, it unclear what the exact relationship is between such evaluations and SMT performance. Several recent studies have reported poor correlation between AER and commonly used MT performance metrics (§6) (Callison-Burch et al., 2004; Ittycheriah and Roukos, 2005; Koehn et al., 2003). This relationship has recently become the subject of increased scrutiny (Ayan and Dorr, 2006a; Fraser and Marcu, 2006b; Lopez and Resnik, 2006). In particular, Fraser and Marcu (2006b) reports that *unbalanced* F-measure is a better predictor of SMT performance. The F-measure is given in Equation 14.

$$F = \frac{|S \cap A|}{\alpha|S| + (1 - \alpha)|A|} \tag{14}$$

In this metric, the parameter α is used to move balance towards either precision or recall.

4.3 Estimation in Log-Linear Models

We now return to the subject of estimating translation model parameters. One we have estimated the parameters of all of our generative models, we can turn our attention to the estimation of the log-linear feature weights λ_1^K (§3.2). It is advisable to use separate training data from the data that is used for training the generative models, in order to avoid overfitting.

As in generative models, the maximum likelihood objective (Equation 8) can be used to train the feature weights. A nice property of log-linear models is that the search problem defined by the maximum likelihood objective function has a single optimum point, which we can find using an iterative search method called *generalized iterative scaling* (Darroch and Ratcliff, 1972). In log-linear models this MLE is the dual of the *maximum entropy estimate*. [11] The training of log-linear SMT models is a supervised learning problem, since we are given inputs and the corresponding best output, and all features are known (some features may need to be computed by our decoder prior to application, if for instance they rely on a particular model of translational equivalence as in the case with underlying generative models). Unfortunately, the the normalization factor represented by the denominator of Equation 7 must be computed for the MLE, and this is expensive to compute even in the supervised case because it involves a sum over all possible translations. The typical solution is to use an approximation to this sum. Och and Ney (2002) use the n-best output of an SMT decoder for this purpose.

4.3.1 Minimum Error-Rate Training

Recently, automatic evaluation metrics for MT have become widespread (§6). They facilitate a new method of parameter estimation: *minimum error-rate training* (MERT) (Och, 2003). In MERT, we assume that the best model is the one that produces the smallest overall error with respect to a given error function. Unfortunately, determining the amount of error in a translation is not a well-defined problem with an objective answer, and numerous error metrics have been proposed. However, Och (2003) shows empirically that we achieve best results for any particular error function when we use that function in our objective function under MERT. This suggests that we can improve the accuracy of our SMT systems

[11] For this reason, log-linear models are often called maximum entropy models in the NLP literature.

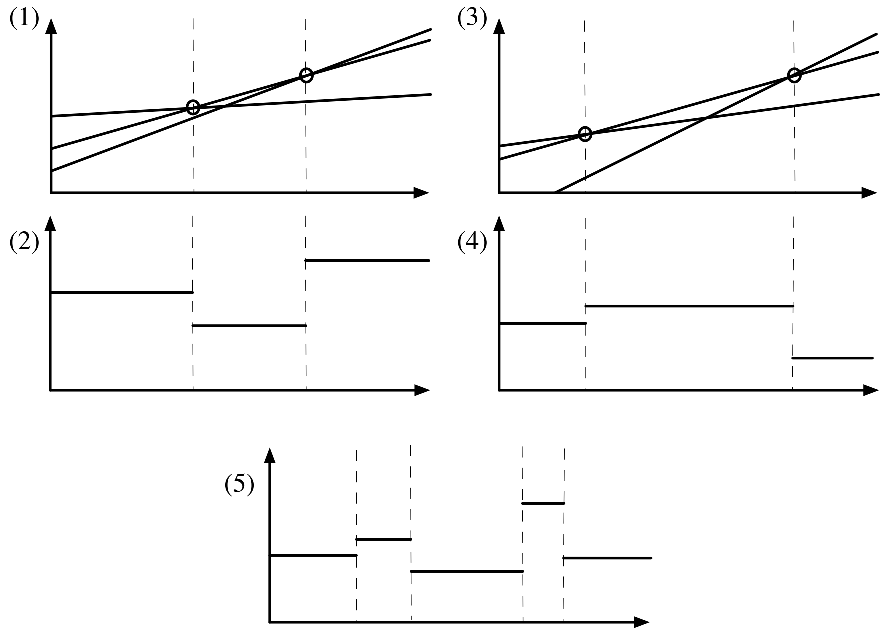


Figure 13: Illustration of the MERT line minimization function for optimizing a single parameter λ_k. In (1) we compute $P_{\lambda_k}(\hat{\mathbf{e}}|\mathbf{f})$ as a line in λ_k for each candidate translation $\hat{\mathbf{e}}$ of a single source sentence $\mathbf{f}$. We then find the intervals at which the optimal candidate $\hat{\mathbf{e}}$ changes by computing the intersection points of these lines. Once the best candidates and intervals are known, we can compute $E_{\lambda_k}(\text{argmax}_{\hat{\mathbf{e}}}\, P(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$ as a function of λ_k. This function is shown in (2). We repeat this procedure for a new source sentence in (3) and (4). Finally, we add the single-sentence error functions (2) and (4) to compute the aggregate error function for both input sentences. This function is shown in (5). Applying this process iteratively to all sentence pairs in the training corpus, we can compute the full error function $\sum_{(\mathbf{e},\mathbf{f})\in C} E_{\lambda_k}(\text{argmax}_{\hat{\mathbf{e}}}\, P(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$. To optimize, we simply walk along all intervals of this function until we determine the minimum.

simply by devising an error function that more closely corresponds to human judgements of translation error, or with some task-based notion of accuracy. Ideally, this means that SMT researchers can focus on the question of what makes a good translation, instead of what makes a good translation model (a task fraught with many orthogonal considerations). With MERT, better evaluation functions should lead directly to better translation.

Formally, we say that if we are given an error function $E(\hat{\mathbf{e}}, \mathbf{e})$ defining the amount of error in some hypothesized translation $\hat{\mathbf{e}}$ with respect to a known good actual translation $\mathbf{e}$, then the objective function is: [12]

$$\lambda_1^K = \underset{\hat{\lambda}_1^K}{\text{argmin}} \sum_{(\mathbf{e},\mathbf{f})\in C} E(\underset{\hat{\mathbf{e}}}{\text{argmax}}\, P_{\hat{\lambda}_1^K}(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e}) \tag{15}$$

The optimization contains an argmax operator, which precludes calculation of a gradient. Although there is no way to find a guaranteed optimal solution under these circumstances, we can find a good

[12]As we will see in §6, we sometimes have access to multiple good translations of $\mathbf{f}$. It is straightforward to modify our functions to accommodate this, and has no real impact on the mathematics.

Algorithm 1 Minimum Error Rate Training

```
 1: Input initial estimate $\lambda_{1,0}^K$                                   ▷ From MLE or prior knowledge
 2: Input training corpus $C$
 3: $\lambda_1^K = \lambda_{1,0}^K$
 4: $E_{best} = \sum_{(\mathbf{e},\mathbf{f})\in C} E(\text{argmax}_{\hat{\mathbf{e}}} P_{\lambda_1^K}(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$
 5: repeat
 6:     Generate $M$ random estimates $\lambda_{1,1}^K, ..., \lambda_{1,M}^K$          ▷ To avoid poor local maximum
 7:     for $m = \{0, 1, ..., M\}$ do
 8:         for $k = \{1, 2, ..., K\}$ do
 9:             $\lambda'_{k,m} = \text{LINE-MINIMIZE}(k, \lambda_{1,m}^K, C)$
10:             $E_{k,m} = \sum_{(\mathbf{e},\mathbf{f})\in C} E(\text{argmax}_{\hat{\mathbf{e}}} P_{\lambda_{1,m}^{k-1}\lambda'_{k,m}\lambda_{k+1,m}^K}(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$
11:             if $E_{k,m} < E_{best}$ then
12:                 $\lambda_1^K = \lambda_{1,m}^{k-1}\lambda'_{k,m}\lambda_{k+1,m}^K$
13:                 $E_{best} = E_{k,m}$
14:             end if
15:         end for
16:     end for
17:     $\lambda_{1,0}^K = \lambda_1^K$
18: until no change in $\lambda_1^K$
19: return $\lambda_{1,0}^K$
20:
21: function LINE-MINIMIZE($k, \lambda_1^K, C$)
22:     $E_{\lambda_k}(C) = 0$
23:     for all $(\mathbf{e}, \mathbf{f}) \in C$ do
24:         for all $\hat{\mathbf{e}} \in$ DECODER-N-BEST($\mathbf{f}$) do
25:             $m_{\hat{\mathbf{e}}} = h_k(\hat{\mathbf{e}}, \mathbf{f})$                                          ▷ slope of $P_{\lambda_k}(\hat{\mathbf{e}}, \mathbf{f})$
26:             $b_{\hat{\mathbf{e}}} = \sum_{k'=1}^{k-1} \lambda_{k'} \cdot h_{k'}(\hat{\mathbf{e}}, \mathbf{f}) + \sum_{k'=k+1}^{K} \lambda_{k'} \cdot h_{k'}(\hat{\mathbf{e}}, \mathbf{f})$          ▷ intercept of $P_{\lambda_k}(\hat{\mathbf{e}}, \mathbf{f})$
27:         end for
28:         $i = 0$
29:         $\Delta[i] = -\infty$                                                  ▷ left interval boundary
30:         $e[i] = \text{argmin}_{\hat{e}}\, m_{\hat{\mathbf{e}}}$                       ▷ equivalent to $\text{argmax}_{\hat{\mathbf{e}}} \lim_{\lambda_k \to -\infty} P(\hat{\mathbf{e}}, \mathbf{f})$
31:         repeat
32:             $i = i + 1$
33:             $\Delta[i] = \min_{\hat{\mathbf{e}}} \text{X-INTERSECT}(m_{\hat{e}}, m_{e[i-1]}, b_{\hat{e}}, b_{e[i-1]}) > \Delta[i-1]$
34:             $e[i] = \text{argmin}_{\hat{\mathbf{e}}} \text{X-INTERSECT}(m_{\hat{e}}, m_{e[i-1]}, b_{\hat{e}}, b_{e[i-1]}) > \Delta[i-1]$
35:         until No more intersection points found
36:         $\Delta_{i+1} = \infty$
37:         $E_{\lambda_k}(\text{argmax}_{\hat{\mathbf{e}}} P(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e}) = \{\lambda_k \to E(\hat{\mathbf{e}}, \mathbf{e}) : \hat{\mathbf{e}} = e[i], \Delta[i] \le \lambda_k \le \Delta_{i+1}\}$
38:         $E_{\lambda_k}(C) += E_{\lambda_k}(\text{argmax}_{\hat{\mathbf{e}}} P(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$
39:     end for
40:     return $\lambda_k = \text{argmin}\, E_{\lambda_k}(C)$
41: end function
```

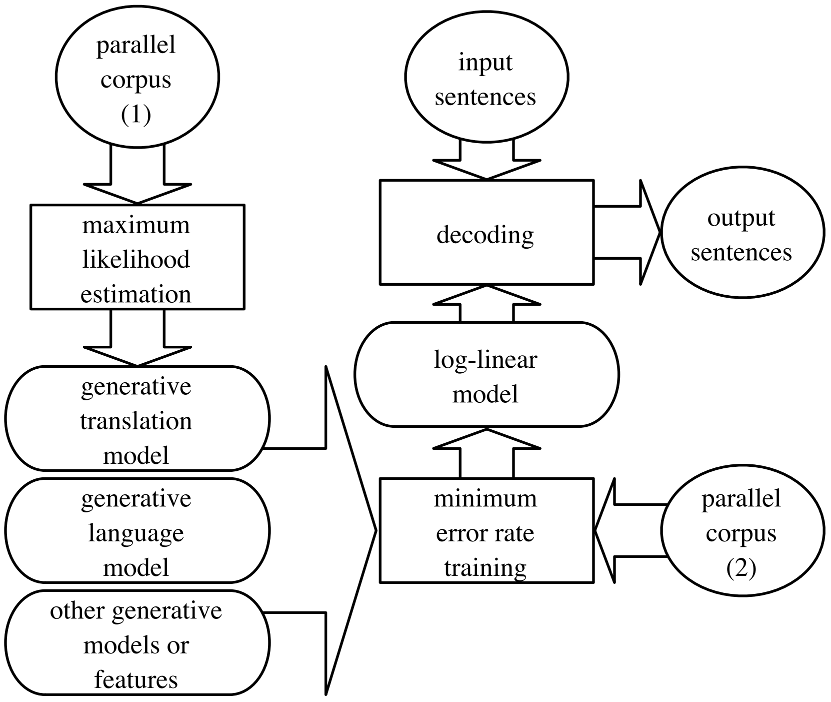

parallel corpus (1)
maximum likelihood estimation
generative translation model
generative language model
other generative models or features
input sentences
decoding
output sentences
log-linear model
minimum error rate training
parallel corpus (2)

Figure 14: Putting it all together: the flow of data, models, and processes commonly involved in the deployment of an SMT system.

solution using the method sketched in Och (2003), which we describe in greater detail here due to its extremely widespread use. Pseudocode is given in Algorithm 1.

The MERT algorithm works by iteratively generating random values for λ_1^K, which it then tries to improve by minimizing each parameter λ_k in turn while holding the others constant. At the end of this optimization step, the optimized λ_1^K yielding the greatest error reduction is used as input to the next iteration.

The single-parameter line minimization algorithm at the core of MERT is illustrated in Figure 13. It is based on the observation that if we hold all but one parameter λ_k constant, then $P(\mathbf{e}|\mathbf{f})$ for any given pair **e** and **f** is $P(\mathbf{e}|\mathbf{f}) = \lambda_k h_k(\mathbf{e}, \mathbf{f}) + (\sum_{k'=1}^{k-1} + \sum_{k'=k+1}^{K})\lambda_{k',m} h_{k'}(\mathbf{e}, \mathbf{f})$. Notice that the second term of the sum is constant, making the function linear in λ_k. Using the intersections of these lines for all candidate translations in a decoder's N-best list for a single input sentence, the algorithm exhaustively computes a representation of the piecewise linear function $E(\text{argmax}_{\hat{\mathbf{e}}} P(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e}))$. Assuming that our error function is additive, we simply sum over all input $(\mathbf{e}, \mathbf{f}) \in C$ to compute the complete function that we are trying to minimize. [13] We then select the midpoint in the interval which minimizes the function.

4.3.2 Purely Discriminative Training

Most current state-of-the-art SMT systems use log-linear models with generative submodels in combination with MERT in order to optimize whatever error function is chosen for evaluation. An overview of

[13]Sometimes this function is not additive, as is the case with the commonly used BLEU score (§6). Usually, however, the function is computed in terms of aggregate values over the training set which are additive. If this is the case, we simply keep track of all of the additive values which are used to compute the error function over each interval, and then perform the computation once all intervals are known.

the architecture used in these systems is shown in Figure 14. This approach is not *purely* discriminative; it uses generative model estimates as input to a discriminative learner that optimizes a small number of feature weights. In pure discriminative learning, features are usually binary or integral. For instance, we might define a word pair feature $h(e, f)$ as follows:

$$h(e, f) = \begin{cases} 1 \text{ if the input contains } f \text{ and the output contains } e \\ 0 \text{ otherwise} \end{cases}$$

It is easy to see that under this defintion, the weight given to this feature by the combined generative/discriminative training procedure outlined above is $\log p(f|e)^{\lambda}$. However, as we have noted, $p(f|e)$ is estimated to maximize likelihood, not translation performance. We might instead wish to assign a weight to this feature that is estimated to directly optimize translation performance. This is the goal of pure discriminative learning, which can be accomplished by a number of different algorithms. Recent examples have included the perceptron algorithm (Liang et al., 2006a), maximum margin estimation (Tillmann and Zhang, 2006), and decision tree learning (Wellington et al., 2006a). Pure discriminative learning is promising, but there are still a number of significant obstacles to overcome, most notably the ability to scale to the very large datasets and billions of paramters required for SMT. The present approaches take many days to train on computing clusters.

5 Decoding

Now that we have a model and estimates for all of our parameters, we can translate new input sentences. This is called decoding. In principle, decoding corresponds solving the maximization problem in Equation 16.

$$\mathbf{e} = \underset{(\hat{\mathbf{e}}:Y(\mathbf{e},\mathbf{d}))}{\operatorname{argmax}} P(\hat{\mathbf{e}}, \mathbf{d}|\mathbf{f}) \tag{16}$$

We call this the *decision rule*. Equation 16 is not the only possible decision rule, although it is by far the most common. Alternative decision rules are presented in Kumar and Byrne (2004) and Venugopal et al. (2005).

This is a difficult optimization. Recall that $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ ranges over $\{E^* \times D^* \times F^*\}$. Even though $\mathbf{f}$ is fixed, and even though the number of possible outputs $(\mathbf{e}, \mathbf{d})$ is finite due to the constraints of our translational equivalence model, there is still a very large number of them to consider in order to maximize the function. Therefore, a primary objective of decoding is to search this space as efficiently as possible.

There are two types of decoders, corresponding to our two broad types of translational equivalence models: FST and SCFG.

5.1 FST Decoding

Nearly all approaches to finite-state decoding follow a general framework described in Berger et al. (1996a) and Wang and Waibel (1997), which is a generalization of the ASR search algorithm that has been modified to accommodate reordering.

In this algorithm, search proceeds through a directed acyclic graph of states representing partial or completed translation hypotheses, which are constructed from left-to-right in the target language word order. An example graph is depicted in Figure 15. Each state consists of the following elements.

(1) A coverage set $C \subseteq \{1, 2, ..., J\}$ enumerates the positions of the source string f_1^J that have been translated.

(1)

虽然	北	风	呼啸	，	但	天空	依然	十分	清澈	。
Although	north	wind	howls	,	but	sky	still	extremely	limpid	.
1	2	3	4	5	6	7	8	9	10	11

(2)

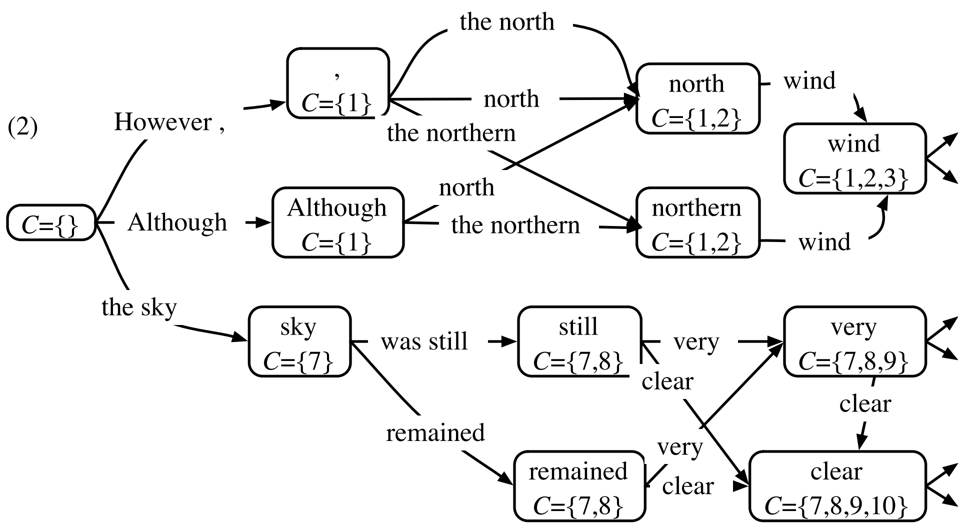


Figure 15: Illustration of search in a finite-state decoder. The input sentence (1) generates a large search graph, partially illustrated in (2). In this illustration, each arrow represents extension of a hypothesis by appending the words on the arrow. In each state, we depict the coverage set and the most recently generated target word, which is needed for computation of the language model (assuming a bigram language model for simplicity). Notice that states can only be combined if the coverage set and the most recently produced words match. The cost of each transition is a combination of the model costs incurred by appending the corresponding target words to the hypothesis.

(2) If using an n-gram language model, the $n-1$ most recently generated target words are kept for computing the n-gram language model component of the probability. These words and the subset C constitute the state's signature.

(3) The cost h of our partial hypothesis is computed as the combination of model costs associated with the hypothesis. This will be fairly straightforward for any generative model based on the underlying translational equivalence model, since we will be reconstructing the events that occur in that model, and we can simply apply the associated probabilities. It may or may not be difficult for a discriminative model, depending on the specific feature functions.

(4) The estimated cost g of completing the partial hypothesis is computed heuristically. Because this computation must be done quickly, we usually use only the single-best word-to-word (or phrase-to-phrase) costs in this heuristic function (Koehn, 2004a).

Hypotheses in this space are extended by adding one or more source word indices to the coverage set and appending one or more target words to the hypothesis string to produce a new state. This corresponds to the translation of the newly covered source words by the newly generated target words. We apply model probabilities accordingly to update the partial cost h. It is straightforward to implement specific extension operators by which we can apply this algorithm to IBM Model 4 (Germann et al., 2004; Tillman and Ney, 2003), phrase-based models (Koehn, 2004a), or any number of other finite-state translation models (Nießen et al., 1998; Wang and Waibel, 1997).

In order to organize the search space, hypotheses may be stored in one or more priority queues, usually corresponding to either the cardinality $|C|$ of the coverage set, or to the coverage sets themselves (Tillman and Ney, 2003). [14] This is done to ensure that comparisons between hypotheses – used for sorting and pruning purposes within each priority queue – are done on hypotheses of relatively equal depth in the search space. (Wang and Waibel, 1997) If we were to compare hypotheses of unequal length, our heuristic functions, which favor shorter hypotheses, will cause more complete hypotheses to be pruned from the priority queue prior to complete evaluation.

Each hypothesis contains a backpointer to the hypothesis that generated it. If two hypotheses have matching signatures, only the higher-scoring hypothesis is kept (Koehn, 2004a; Och et al., 2001). This is a risk-free optimization because the set of all extensions to these two hypotheses will be the same; therefore the higher-scoring partial hypothesis is guaranteed to generate a higher-scoring completed hypothesis.

The search space defines a finite-state word lattice, in which we can find the score of any particular hypothesis by traversing the lattice (Koehn, 2004a; Ueffing et al., 2002). We can use standard finite-state methods for finding the best path (or paths) through this lattice. It is possible to directly implement such decoders as a cascade of weighted finite-state transducers (Knight and Al-Onaizan, 1998; Kumar et al., 2006). These transducers will differ from the ones we describe in §2.1. However, the decoding algorithm we have described does, in principle, reverse the set of transductions represented by those models; we can see, for instance, that it reconstructs the English sentence in the order that it was fed into the transducer, at each step consuming the source words that were created by transductions over the associated target word or words.

Watanabe and Sumita (2002) present a variation on this algorithm in which the target language hypothesis may be extended by adding words to either the left or right.

5.1.1 Optimality and Pruning

Using A^* heuristics, we can solve the optimization in Equation 16 exactly (Och et al., 2001). Germann et al. (2004) illustrate how we can also do this by converting the problem to a linear integer programming problem and using standard tools to solve it. Unfortunately, optimal search in this space is NP-complete, as Knight (1999a) illustrates by reduction to the Traveling Salesman Problem. This boils down to the fact that, in general, finite-state models allow all possible permutations of the input string in the reordering model. Fortunately, optimal search is not strictly necessary, because there are likely to be many good translations of a sentence. If many high-probability translations are good translations, then a certain amount of *search error* is acceptable, particularly if it allows us to improve search speed. A number of strategies are used to do this.

(1) We can restrict the allowable reorderings. In the search space we have described, any uncovered source word may be translated at each step. Since this is largely responsible for the combinatorial explosion, and because it is likely to be unrealistic, one way that we can limit this effect is to permit only the first k uncovered words to be translated (Berger et al., 1996a). A language-specific variation on this approach is illustrated by Tillman and Ney (2003). In the extreme case, only the first uncovered word (or contiguous sequence of words in the case of phrase-based models) may be translated, which leads to a monotonic alignment between the source and target strings (Banchs et al., 2005; Tillmann et al., 1997; Zens and Ney, 2004).

[14]This priority queue is often called a *stack* in literature, and the algorithm that we describe is called *stack decoding*, although its central object is technically not a stack. This is due to historical reasons, since the algorithm traces its lineage back to so-called stack decoding algorithms of ASR.

(2) We can prune hypotheses from the search space. Two pruning methods are commonly used: *threshold pruning* and *histogram pruning*. (Koehn, 2004a; Tillman and Ney, 2003) In threshold pruning, any hypothesis with a probability less than t times the probability of the best estimate in the same priority queue is removed. In histogram pruning, only the n best hypotheses are kept in any priority queue. Search with these methods is also known as *beam search*, and we refer to t or n as the size of the beam.

When the parameters of these optimization methods are properly tuned, we gain large speedups at the cost of relatively little accuracy (Germann et al., 2004; Koehn, 2004a; Tillman and Ney, 2003; Zens and Ney, 2004).

5.1.2 Greedy Decoding

An alternative to standard finite-state decoding is greedy decoding (Germann, 2003; Germann et al., 2004; Marcu and Wong, 2002). In greedy decoding, we generate an initial hypothesis by substituting each source word with the highest-probability target word, using the original target word order. This gives us a complete word-for-word gloss of the source sentence. We then use hill-climbing heuristics in an attempt to find higher-scoring hypotheses by considering neighboring translations produced by changing the order or translation of one or two words at a time, and choosing the highest-scoring neighbor. This new hypothesis becomes the starting point for the next iteration of the algorithm. The algorithm terminates when no higher-scoring hypothesis can be found. With some optimizations, this algorithm runs in time nearly linear in target sentence length (Germann, 2003). The tradeoff is that the algorithm considers a much smaller space of possible translations, and its error rate is correspondingly higher (Germann et al., 2004).

5.2 SCFG Decoding

Decoding with SCFG models is equivalent to CFG parsing (Melamed, 2004a). As the CFG tree of the input is reconstructed, the corresponding target language tree is produced in parallel according to the SCFG rules that are applied. Most practical syntax-based decoders are straightforward extensions of the CKY or Earley algorithms for parsing monolingual context-free grammars in one dimension (Chiang, 2007; Marcu et al., 2006; Venugopal et al., 2007; Wu and Wong, 1998; Yamada and Knight, 2002; Zens and Ney, 2003). A benefit of this is that the standard algorithms and optimizations that have been developed for CFG parsing can be applied to SMT (Melamed, 2004a). Furthermore, it has been shown that generative parameter estimation, decoding, and most other applications can be performed using a single algorithm (Goodman, 1999), which means that we do not need to spend development time on separate algorithms for these tasks, as is typically done for FST models.

SCFG decoding works by attempting to cover larger and larger *spans* of the input sentence. A span is simply a contiguous sequence of words. States in the search space consist of a span, a nonterminal symbol which covers the span, and any language model information needed to combine spans Chiang (2007); Melamed (2004b); Zhang et al. (2006a). In order to construct larger spans, we find SCFG rules which construct the exact sequence of nonterminals that we have already inferred to cover a set of smaller, adjacent spans. Once we have constructed the full source language parse, we produce output using an in-order traversal based on target language ordering of the tree. This is illustrated in Figure 16.

There are $O(J^2)$ possible spans in the source sentence. It is easy to see from this that SCFG decoding is, in principle, much less computationally expensive than finite-state decoding. In finite-state decoding, we must enumerate all possible coverings of the input sentence, whereas in SCFG decoding we enumerate only spans. SCFG decoding is, in general, polynomial in sentence length (Melamed, 2003). We can apply beam search methods to improve search speed. However, it is often slower than

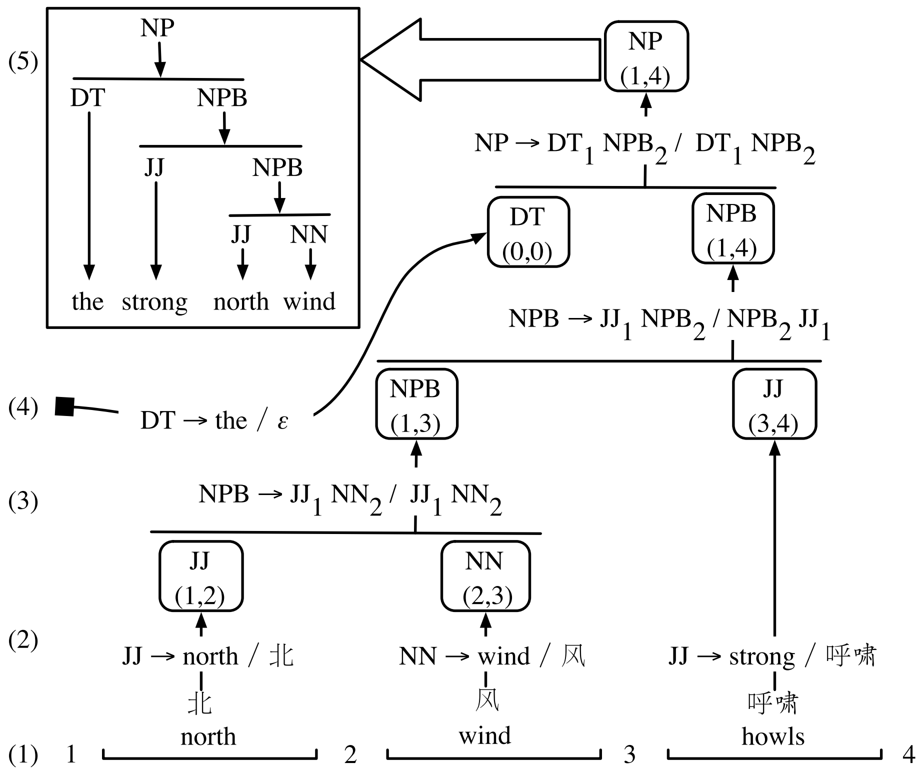


Figure 16: An illustration of SCFG decoding, which is equivalent to parsing the source language. (1) Scan each source word and associate it with a span. (2) Apply SCFG rules that match the target spans. (3) Recursively infer larger spans from smaller spans. (4) We can also infer target language words with no matching span in the source language, if our grammar contains SCFG rules that produce these words correspondent with ε in the source language. (5) Read off the tree in target-language order.

FST decoding with tight reordering constraints. A number of optimizations are used to speed up search. Chiang (2007) describes a clever optimization called *cube pruning* that prevents excessive combination of hypotheses in adjacent subspans. Zhang et al. (2006a) describe a clever method for *binarizing* rules containing more than two nonterminals, which helps reduce grammar constants for parsing and enables n-gram language model integration. Venugopal et al. (2007) present an alternative method based on delayed language model integration, in which the parse graph is first constructed quickly with simplified language model statistics, and then expanded in a second pass using a full language model, following only the most promising paths.

5.3 Reranking

Even if there are no search errors and we produce the translation that exactly optimizes our decision rule, the translations produced by our decoder may not be the actual best translations according to human judgement. It is possible that the search space explored by the decoder contained a better translation, and our decoder assigned a lower score for this hypothesis because its estimate of $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ was incorrect. This is called *model error*.

One approach to reducing model error is *reranking* or *rescoring*. In reranking, we first run our decoder, and rather than merely outputting the highest-scoring translation, we output N highest-scoring translations for some value N. These translations are then input to an alternative model with access to more feature functions than may be efficiently computed in our decoder, or which are otherwise difficult to incorporate. Hopefully, this alternative model can give us more accurate scores than the one used in decoding.

Reranking approaches to SMT are described in Och et al. (2004), Shen et al. (2004), and Kumar and Byrne (2004). Och et al. (2004) show using oracle studies on decoder n-best lists that large gains in accuracy are possible with rescoring, although so far these are unrealized.

6 Evaluation

How can we know if the output of our SMT system is any good? Many methods have been proposed to evaluate MT output. Hovy et al. (2002) attribute to Yorick Wilks the remark that "more has been written about MT evaluation over the past 50 years than about MT itself". In the discussion that follows, we will narrowly focus on methods that have figured prominently in the evaluation of statistical systems.

Traditionally accepted measures of MT evaluation have required examination of MT system output by human judges, who rank the *adequacy* of the translation in conveying the source language meaning and the *fluency* of expression in the target language syntax. In general, we can regard these or other task-related measures as ideal; if it was possible, we would optimize for these directly. Unfortunately, this would require human judges to rank the vast number of translations considered in the optimization process. In general, human evaluation is expensive in both the time taken to perform the evaluation, and in monetary cost, since we will usually need to pay for the expertise of bilingual evaluators. This is often out of the question even for iterative system development, where we will need to perform regular evaluation to determine if changes are beneficial to performance. The next best thing is to develop automatic metrics which closely correlate with human judgement. The closer that these metrics are to the real objective, the better our performance on that objective will be after we apply discriminative training (§4.3).

One important element of automatic metrics is the use of a set of test sentences for which we already have human translations. These can come from a parallel corpus, although we must take care to use a separate set of sentences from the set we used for training. The automatic evaluation methods are based on partial string matching between the output and these *reference translations*, as illustrated in Figure 17. However, the use of a single reference may bias the evaluation towards a particular translation style. In order to mitigate against this and reflect the diversity of possible good translations, it is common to use multiple references. This requires the use of human translators to produce the additional references, but this is still a one-time cost.

One metric for evaluation is the well-known Levenshtein or edit distance, which is borrowed from ASR evaluation, where it is known as the *word error rate* (WER) (Och et al., 1999). The WER sums the number of insertions, deletions, and substitutions required to transform an output sentence into the reference sentence. Unfortunately, this metric is less appropriate for MT than ASR, because it does not recognize word reorderings. A word that is translated correctly but in the wrong location will be penalized as a deletion (in the output location) and an insertion (in the correct location). This problem motivates the use of *position-independent word error rate* (PER), which is similar to WER but does not penalize reorderings, because it regards the output and reference sentences as unordered sets rather than totally ordered strings (Och et al., 1999).

For the last several years, the most commonly used metric has been the *bilingual evaluation understudy* or BLEU metric (Papineni et al., 2002). The BLEU metric considers not only single word matches between the output and the reference sentence, but also n-gram matches, up to some maximum n. This

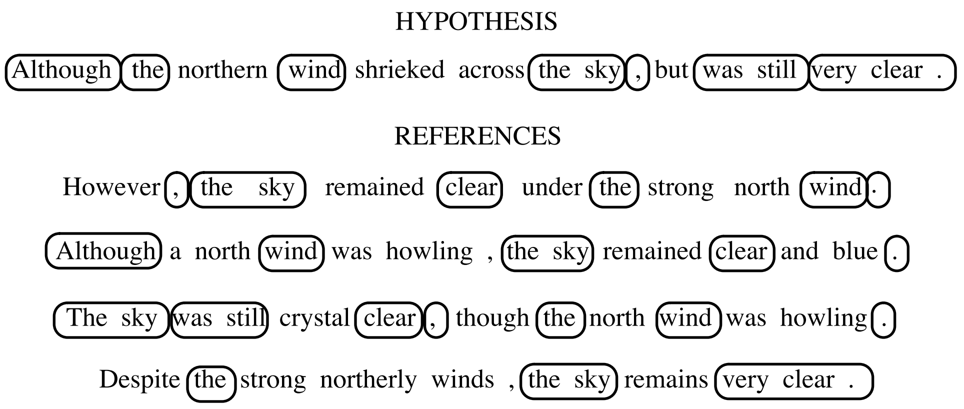

HYPOTHESIS
Although the northern wind shrieked across the sky , but was still very clear .
REFERENCES
However , the sky remained clear under the strong north wind .
Although a north wind was howling , the sky remained clear and blue .
The sky was still crystal clear , though the north wind was howling .
Despite the strong northerly winds , the sky remains very clear .

Figure 17: Example of partial string matching used for most evaluation methods. Here we show a single output hypothesis compared with four reference translations. Sequences of words in the hypothesis that match sequences in any of the reference translations are circled. Likewise, sequences of words in each reference that are found in the hypothesis are circled. Most evaluation metrics are based on functions of counts of these matches.

allows it to reward sentences where local word order is closer to the local word order in the reference. Furthermore, the BLEU metric is a *precision*-oriented metric; that is, it considers the number of n-gram matches as a fraction of the number of total n-grams in the output sentence. The fraction is computed separately for each n, and a geometric average is taken. This biases the metric towards translations that drop words for which the translation is uncertain, so the metric also includes a weighted *brevity penalty*, which penalizes output sentences that are much shorter than the reference. A number of other metrics based on word matching include *precision* and *recall* (Melamed et al., 2003), and length of the longest common subsequence (Lin and Och, 2004). All of these algorithms have variants which work with multiple reference sentences. A key element of most research in this area is the identification of metrics that correlate with human judgement in controlled studies.

The BLEU has been highly influential in SMT research. It is reported in most recent SMT literature, and it has been used as the basis for a number of comparative evaluations (Doddington, 2002; Koehn and Monz, 2005, 2006). It is commonly used in the objective function for minimum error-rate training (Och, 2003).

The use of the BLEU score has always been controversial. Habash (2003) presents an early critique of the metric. A number of counterexamples have been published regarding the metric's correlation with human judgement (Callison-Burch et al., 2006; Turian et al., 2003), and other potential problems have been constructively demonstrated (Callison-Burch et al., 2006).

Despite controversy, automatic evaluation has had a profound impact on progress in SMT research, and it is likely to continue in some form. The identification of new, better metrics for MT evaluation is the subject of ongoing research. Recent proposals include the METEOR metric, which enhances token matching with weighted matching based on morphological or semantic similarity (Banerjee and Lavie, 2005). An alternative approach is the *translation edit rate* (TER), which constructs reference sentences based on system output, and computes the number of edits required to generate these *targeted references* from a hypothesis sentence. In this case, movement of a whole phrase is counted as a single error. This task-oriented metric represents "the amount of work needed to correct the translations." (Snover et al., 2006). There is also work in using machine learning methods to produce better metrics (Kulesza and Shieber, 2004; Lita et al., 2005).

It is not always clear whether a difference in scores between two systems represents a significant difference in their output. Koehn (2004b) describes a method to compute statistical confidence intervals for most automatic metrics using bootstrap resampling.

7 Current Directions and Future Research

At the time of this writing, there are many common elements in the best systems, although there is also growing diversity. The following attributes are characteristic of nearly all the best systems: phrase-based translational equivalence models (in either the FST or SCFG framework); log-linear models with a small set of generative features; and minimum-error rate training. Most of the evidence comes from various evaluation settings (Doddington, 2002; Koehn and Monz, 2005, 2006).

All of these methods were introduced less than ten years ago, some considerably more recently. SMT has made swift progress, and there is considerable optimism for future successes. Nonetheless, there remain a large number of hurdles and open questions in the field. Refinements to modeling techniques and parameter estimation methods will no doubt continue. Recent developments in machine learning, which have recently been applied to alignment, will increasingly be applied to machine translation, although additional work is needed to scale them up to data sizes commonly used in SMT. To complement this work, better feature engineering will be required. However, other interesting issues are also likely to increase in importance.

Most of the standard evaluations in SMT have focused on the translation of news and government texts. There is very little work on open-domain translation, particularly for informal genres – which describes much of the information found on the internet, and for which translation is in demand. Although it is possible to mine data from the web (Resnik and Smith, 2003), this resource is so far underutilized, with most studies using news and government texts in training. Statistical methods are notoriously sensitive to domain differences, however, so the move to informal text is likely to present many interesting challenges.

Another understudied problems in SMT is the translation of English into other languages. Most studies have focused on translation from other languages into English. This probably obscures some deficiencies in the current approaches, since it is likely easier to map morphologically rich languages such as German and Arabic onto a relatively morphologically impoverished language such as English. This can be seen as a movement from a higher-dimensional to a lower dimensional space, and some data loss is not harmful. Translation in the other direction is likely to require much more attention to this issue. Some experiments in morphologically rich translation modeling have recently been reported (Goldwater and McClosky, 2005; Nießen and Ney, 2004; Schafer and Drabek, 2005). Koehn et al. (2006) describe *factored models*, a framework for modeling with morphology and other annotations.

Evaluation of MT systems will probably continue to be a focus, since the minimum error-rate training technique has emphasized the importance of evaluation metrics that correspond to human judgement. However, as we have seen, the methods currently used for evaluation of SMT systems are holistic, and provide very little insight into the systematic errors made by a system. They are especially useless for identifying sentence-level errors made by a system because the scores are not reliable at that granularity – they are only meaningful in aggregate. For this reason, the relative merits and drawbacks of different models with respect to different types of translation error are not well understood. Error analysis techniques have not been substantially explored, although it has recently been identified as an important task (Och, 2005). A few techniques for error analysis (Chiang et al., 2005; DeNeefe et al., 2005; Popovic et al., 2006) and confidence estimation (Ueffing and Ney, 2005) have begun to emerge, but in general this area remains underexplored.

Finally, although fully automatic MT is often treated as the objective in many SMT studies, it is certainly not the only objective – perhaps not even the primary objective. Understanding the translation needs of users will be critical to the continued improvement of MT services. Integration with with speech recognition, information retrieval, document summarization, question-answering, and other NLP applications will no doubt become increasingly important as SMT takes its place alongside a suite of tools for global information access.

Acknowledgements

I would like to thank David Chiang, Nitin Madnani, Austin Parker and Reiner Schulz for helpful feedback on previous drafts. This paper has benefitted immensely from illuminating discussions with Necip Fazil Ayan, David Chiang, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. Any errors are entirely my own.

References

Alfred V. Aho and Jeffrey D. Ullman. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3:37–57, 1969.

Lars Ahrenberg, Magnus Merkel, Anna Sågvall Hein, and Jörg Tiedmann. Evaluation of word alignment systems. In *Proc. of LREC*, volume 3, pages 1255–1261, May 2000.

Yaser Al-Onaizan and Kishore Papineni. Distortion models for statistical machine translation. In *Proc. of ACL-COLING*, pages 529–536, Jul 2006.

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. Statistical machine translation: Final report. Technical report, Johns Hopkins University Center for Speech and Language Processing, 1999.

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1):45–60, Mar 2000.

Necip Fazil Ayan and Bonnie Dorr. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proc. of ACL-COLING*, pages 9–16, Jul 2006a.

Necip Fazil Ayan and Bonnie J. Dorr. A maximum entropy approach to combining word alignments. In *Proc. of HLT-NAACL*, pages 96–103, Jun 2006b.

Necip Fazil Ayan, Bonnie Dorr, and Christof Monz. Neuralign: Combining word alignments using neural networks. In *Proc. of HLT-EMNLP*, pages 65–72, Oct 2005a.

Necip Fazil Ayan, Bonnie Dorr, and Christof Monz. Alignment link projection using transformation-based learning. In *Proc. of HLT-EMNLP*, pages 185–192, Oct 2005b.

Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, and José B. Mariño. Statistical machine translation of euparl data by using bilingual n-grams. In *Proc. of ACL Workshop on Building and Using Parallel Texts*, pages 133–136, Jun 2005.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Jun 2005.

Leonard E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. In *Proceedings of the Third Symposium on Inequalities*, volume 3 of *Inequalities*, pages 1–8. Academic Press, 1972.

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, John R. Gillett, John D. Lafferty, Robert L. Mercer, Harry Printz, and Luboš Ureš. The Candide system for machine translation. In *Proc. of the ARPA Workshop on Human Language Technology*, pages 157–162, Mar 1994.

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Andrew S. Kehler, and Robert L. Mercer. Language translation apparatus and method using context-based translation models, Apr 1996a. United States Patent 5510981.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, Mar 1996b.

Phil Blunsom and Trevor Cohn. Discriminative word alignment with conditional random fields. In *Proc. of ACL-COLING*, pages 65–72, Jul 2006.

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

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, Dec 1992.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, Jun 1993.

Andrea Burbank, Marine Carpuat, Stephen Clark, Markus Dreyer, Pamela Fox, Declan Groves, Keith Hall, Mary Hearne, I. Dan Melamed, Yihai Shen, Andy Way, Ben Wellington, and Dekai Wu. Final report of the 2005 language engineering workshop on statistical machine translation by parsing. Technical report, Johns Hopkins University Center for Speech and Language Processing, Nov 2005.

Chris Callison-Burch, David Talbot, and Miles Osbourne. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proc. of ACL*, pages 176–183, Jul 2004.

Chris Callison-Burch, Colin Bannard, and Josh Shroeder. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proc. of ACL*, pages 255–262, Jun 2005.

Chris Callison-Burch, Miles Osbourne, and Philipp Koehn. Re-evaluating the role of BLEU in machine translation research. In *Proc. of EACL*, pages 249–256, Apr 2006.

Eugene Charniak, Kevin Knight, and Kenji Yamada. Syntax-based language models for statistical machine translation. In *Proc. of MT Summit IX*, Sept 2003.

Ciprian Chelba and Frederick Jelinek. Exploiting syntactic structure for language modeling. In *Proc. of ACL-COLING*, pages 225–231, Aug 1998.

Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, Aug 1998.

Colin Cherry and Dekang Lin. A probability model to improve word alignment. In *Proc. of ACL*, Jul 2003.

David Chiang. *Evaluation of Grammar Formalisms for Applications to Natural Language Processing and Biological Sequence Analysis*. PhD thesis, University of Pennsylvania, 2004.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL*, pages 263–270, June 2005.

David Chiang. An introduction to synchronous grammars. Part of a tutorial given at ACL 2006, Jul 2006.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2), 2007. In press.

David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. The Hiero machine translation system: Extensions, evaluation, and analysis. In *Proc. of HLT-EMLP*, pages 779–786, Oct 2005.

Kenneth Church and Eduard Hovy. Good applications for crummy machine translation. *Machine Translation*, 8: 239–258, 1993.

Kenneth Church and Ramesh Patil. Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics*, 8(3–4):139–149, Jul 1982.

Michael Collins, Jan Hajič, Lance Ramshaw, and Christoph Tillman. A statistical parser for Czech. In *Proc. of ACL*, pages 505–512, Jun 1999.

Michael Collins, Philipp Koehn, and Ivona Cučerová. Clause restructuring for statistical machine translation. In *Proc. of ACL*, pages 531–540, Jun 2005.

J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480, Oct 1972.

Herve Dejean, Eric Gaussier, Cyril Goutte, and Kenji Yamada. Reducing parameter space for word alignment. In

Proc. of HLT-NAACL Workshop on Building and Using Parallel Texts, pages 23–26, May 2003.

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

Steve DeNeefe, Kevin Knight, and Hayward H. Chan. Interactively exploring a machine translation model. In *Proc. of ACL (Companion Vol.)*, pages 97–100, Jun 2005.

George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. of HLT*, 2002.

Bonnie J. Dorr, Pamela W. Jordan, and John W. Benoit. A survey of current paradigms in machine translation. In M. Zelkowitz, editor, *Advances in Computers*, volume 49, pages 1–68. Academic Press, 1999.

Matthias Eck, Stephan Vogel, and Alex Waibel. Language model adaptation for statistical machine translation based on information retrieval. In *Proc. of LREC*, May 2004.

Marcello Federico and Nicola Bertoldi. How many bits are needed to store probabilities for phrase-based translation? In *Proc. of NAACL Workshop on Statistical Machine Translation*, pages 94–101, Jun 2006.

George Foster, Roland Kuhn, and Howard Johnson. Phrasetable smoothing for statistical machine translation. In *Proc. of EMNLP*, pages 53–61, Jul 2006.

Heidi J. Fox. Phrasal cohesion and statistical machine translation. In *Proc. of EMNLP*, pages 304–311, Jul 2002.

Alexander Fraser and Daniel Marcu. Semi-supervised training for statistical word alignment. In *Proc. of ACL*, pages 769–776, Jun 2006a.

Alexander Fraser and Daniel Marcu. Measuring word alignment quality for statistical machine translation. Technical Report ISI-TR-616, ISI-University of Southern California, 2006b.

William A. Gale and Kenneth W. Church. Identifying word correspondences in parallel text. In *Proc. of Darpa Workshop on Speech and Natural Language*, pages 152–157, 1991.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proc. of HLT-NAACL*, pages 273–280, May 2004.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. Scalable inference and training of context-rich syntactic translation models. In *Proc. of ACL*, pages 961–968, Jun 2006.

Ismael García Varea, Franz J. Och, Hermann Ney, and Francisco Casacuberta. Refined lexicon models for statistical machine translation using a maximum entropy approach. In *Proc. of ACL-EACL*, pages 204–211, Jul 2001.

Ulrich Germann. Greedy decoding for statistical machine translation in almost linear time. In *Proc. of HLT-NAACL*, pages 72–79, May 2003.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast decoding and optimal decoding for machine translation. In *Proc. of ACL-EACL*, Jul 2001.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. Fast and optimal decoding for machine translation. *Artificial Intelligence*, 154(1–2):127–143, Apr 2004.

Daniel Gildea. Dependencies vs. constituencies for tree-based alignment. In *Proc. of EMNLP*, pages 214–221, Jul 2004.

Sharon Goldwater and David McClosky. Improving statistical MT through morphological analysis. In *Proc. of HLT-EMNLP*, pages 676–683, Oct 2005.

Joshua Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–605, Dec 1999.

Jonathan Graehl and Kevin Knight. Training tree transducers. In *Proc. of HLT-NAACL*, pages 105–112, May 2004.

Nizar Habash. *Generation-Heavy Hybrid Machine Translation*. PhD thesis, University of Maryland, 2003.

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

Eduard Hovy, Margaret King, and Andrei Popescu-Belis. Principles of context-based machine translation evaluation. *Machine Translation*, 17(1):43–75, Mar 2002.

John Hutchins. Machine translation: A concise history. In Chan Sin Wai, editor, *Computer aided translation: theory and practice*. Chinese University of Hong Kong, 2007.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325, Sep 2005.

Abraham Ittycheriah and Salim Roukos. A maximum entropy word aligner for Arabic-English machine translation. In *Proc. of HLT-EMNLP*, pages 89–96, Oct 2005.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1998.

Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–124. Springer, Berlin, 1997.

Aravind K. Joshi, K. Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. In Peter Sells, Stuart Shieber, and Tom Wasow, editors, *Foundational Issues in Natural Language Processing*, chapter 2, pages 31–81. MIT Press, Cambridge, MA, 1991.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice-Hall, 2000.

Katrin Kirchhoff and Mei Yang. Improved language modeling for statistical machine translation. In *Proc. of ACL Workshop on Building and Using Parallel Texts*, pages 125–128, Jun 2005.

Kevin Knight. Automating knowledge acquisition for machine translation. *AI Magazine*, 18(4):81–96, 1997.

Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4): 607–615, Dec 1999a.

Kevin Knight. A statistical MT tutorial workbook. Unpublished, 1999b.

Kevin Knight and Yaser Al-Onaizan. Translation with finite-state devices. In *Proc. of AMTA*, pages 421–437, Oct 1998.

Kevin Knight and Daniel Marcu. Machine translation in the year 2004. In *Proc. of ICASSP*, Mar 2005.

Philipp Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proc. of AMTA*, Sep 2004a.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*, pages 388–395, Jul 2004b.

Philipp Koehn. *PHARAOH, a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models, User Manual and Description for Version 1.2*, Jul 2004c.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*, 2005.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2007. In press.

Philipp Koehn and Christof Monz. Shared task: Statistical machine translation between european languages. In *Proc. of ACL Workshop on Building and Using Parallel Texts*, pages 119–124, Jun 2005.

Philipp Koehn and Christof Monz. Manual and automatic evaluation of machine translation between european languages. In *Proc. of NAACL Workshop on Statistical Machine Translation*, pages 102–121, 2006.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 127–133, May 2003.

Philipp Koehn, Marcello Federico, Wade Shen, Nicola Bertoldi, Ondrej Bojar, Chris Callison-Burch, Brooke Cowan, Christopher J. Dyer, Hieu Hoang, Richard Zens, Alexandra Constantin, Christine Corbett Moran, and Evan Herbst. Open source toolkit for statistical machine translation: Factored translation models and confusion network decoding. Final report of the 2006 language engineering workshop, Johns Hopkins University Center for Speech and Language Processing, 2006. In preparation.

Alex Kulesza and Stuart M. Shieber. A learning approach to improving sentence-level MT evaluation. In *Proc. of TMI*, Oct 2004.

Shankar Kumar and William Byrne. Minimum bayes-risk decoding for statistical machine translation. In *Proc. of HLT-NAACL*, pages 169–176, May 2004.

Shankar Kumar, Yonggang Deng, and William Byrne. A weighted finite state transducer translation template model for statistical machine translation. *Natural Language Engineering*, 12(1):35–75, Mar 2006.

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4(1), 1990.

Percy Liang, Alexandre Bouchard-Côté, Ben Taskar, and Dan Klein. An end-to-end discriminative approach to machine translation. In *Proc. of ACL-COLING*, pages 761–768, Jul 2006a.

Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proc. of HLT-NAACL*, pages 104–111, Jun 2006b.

Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proc. of ACL*, pages 606–613, Jul 2004.

Lucian Lita, Monica Rogati, and Alon Lavie. BLANC: Learning evaluation metrics for MT. In *Proc. of HLT-EMNLP*, pages 740–747, Oct 2005.

Adam Lopez and Philip Resnik. Improved HMM alignment models for languages with scarce resources. In *Proc. of ACL Workshop on Building and Using Parallel Texts*, pages 83–86, Jun 2005.

Adam Lopez and Philip Resnik. Word-based alignment, phrase-based translation: What's the link? In *Proc. of AMTA*, pages 90–99, Aug 2006.

Adam Lopez, Michael Nossal, Rebecca Hwa, and Philip Resnik. Word-level alignment for multilingual resource acquisition. In *Proc. of LREC Workshop on Linguistic Knowledge Acquisition and Representation – Bootstrapping Annotated Language Data*, pages 34–42, Jun 2002.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, May 1999.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP*, pages 133–139, Jul 2002.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP*, pages 44–52, Jul 2006.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):314–330, Jun 1993.

Evgeny Matusov, Richard Zens, and Hermann Ney. Symmetric word alignments for statistical machine translation. In *Proc. of COLING*, pages 219–225, Jul 2004.

I. Dan Melamed. Automatic construction of clean broad-coverage translation lexicons. In *Proc. of AMTA*, 1996.

I. Dan Melamed. Manual annotation of translational equivalence: The blinker project. Technical Report 98-07, University of Pennsylvania Institute for Research in Cognitive Science, 1998.

I. Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, Jun 2000.

I. Dan Melamed. Multitext grammars and synchronous parsers. In *Proc. of HLT-NAACL*, pages 79–86, May 2003.

I. Dan Melamed. Statistical machine translation by parsing. In *Proc. of ACL*, pages 654–661, Jul 2004a.

I. Dan Melamed. Algorithms for syntax-aware statistical machine translation. In *Proc. of TMI*, 2004b.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. Precision and recall of machine translation. In *Proc. of HLT-NAACL (Companion Vol.)*, pages 61–63, May 2003.

I. Dan Melamed, Giorgio Satta, and Benjamin Wellington. Generalized multitext grammars. In *Proc. of ACL*, pages 662–669, Jul 2004.

Bernard Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172, 1994.

Rada Mihalcea and Ted Pedersen. An evaluation exercise for word alignment. In *Proc. of HLT-NAACL Workshop on Building and Using Parallel Texts*, pages 1–10, May 2003.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

Robert C. Moore. Improving IBM word-alignment model 1. In *Proc. of ACL*, pages 519–526, Jul 2004.

Robert C. Moore. A discriminative framework for bilingual word alignment. In *Proc. of HLT-EMNLP*, pages 81–88, Oct 2005a.

Robert C. Moore. Association-based bilingual word alignment. In *Proc. of ACL Workshop on Building and Using*

Parallel Texts, pages 1–8, Jun 2005b.

Sonja Nießen and Hermann Ney. Statistical machine translation with scarce resources using morpho-syntactic information. *Computational Linguistics*, 30(2):182–204, Jun 2004.

Sonja Nießen, Stephan Vogel, Hermann Ney, and Christoph Tillman. A DP based search algorithm for statistical machine translation. In *Proc. of ACL-COLING*, pages 960–967, Aug 1998.

Douglas W. Oard and Franz Josef Och. Rapid-response machine translation for unexpected languages. In *Proc. of MT Summit IX*, Sept 2003.

Douglas W. Oard, David Doermann, Bonnie Dorr, Daqing He, Philip Resnik, Amy Weinberg, William Byrne, Sanjeev Khudanpur, David Yarowsky, Anton Leuski, Philipp Koehn, and Kevin Knight. Desperately seeking Cebuano. In *Proc. of HLT-NAACL (Companion Vol.)*, pages 76–78, May 2003.

Franz Josef Och. An efficient method for determining bilingual word classes. In *Proc. of EACL*, pages 71–76, Jun 1999.

Franz Josef Och. Minimum error rate training in statistical machine translation. In *Proc. of ACL*, Jul 2003.

Franz Josef Och. Statistical machine translation: The fabulous present and future. In *Proc. of ACL Workshop on Building and Using Parallel Texts*, Jun 2005. Invited talk.

Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *Proc. of ACL*, pages 440–447, Oct 2000a.

Franz Josef Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *Proc. of COLING*, pages 1086–1090, Jul 2000b.

Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for machine translation. In *Proc. of ACL*, pages 156–163, Jul 2002.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, Mar 2003.

Franz Josef Och and Hermann Ney. The alignment template approach to machine translation. *Computational Linguistics*, 30(4):417–449, Jun 2004.

Franz Josef Och, Christoph Tillman, and Hermann Ney. Improved alignment models for statistical machine translation. In *Proc. of EMNLP-VLC*, pages 20–28, Jun 1999.

Franz Josef Och, Nicola Ueffing, and Hermann Ney. An efficient A* search algorithm for statistical machine translation. In *Proc. of ACL Workshop on Data-Driven Methods in Machine Translation*, pages 55–62, Jul 2001.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. A smorgasbord of features for statistical machine translation. In *Proc. of HLT-NAACL*, pages 161–168, May 2004.

Marian Olteanu, Chris Davis, Ionut Volosen, and Dan Moldovan. Phramer - an open source statistical phrase-based translator. In *Proc. of HLT-NAACL Workshop on Statistical Machine Translation*, pages 146–149, 2006.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318, Jul 2002.

Maja Popovic, Adrià de Gispert, Deepa Gupta, Patrik Lambert, Hermann Ney, José B. Mariño, Marcello Federico, and Rafael Banchs. Morpho-syntactic information for automatic error analysis of statistical machine translation output. In *Proc. of NAACL Workshop on Statistical Machine Translation*, pages 1–6, Jun 2006.

Chris Quirk, Arul Menezes, and Colin Cherry. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*, pages 271–279, Jun 2005.

Adwait Ratnaparkhi. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, 1998.

Philip Resnik and Noah A. Smith. The web as parallel corpus. *Computational Linguistics*, 29(3):349–380, Sep 2003.

Philip Resnik, Mari Broman Olsen, and Mona Diab. Creating a parallel corpus from the "book of 2000 tongues". In *Proceedings of the Text Encoding Initiative 10th Anniversary User Conference (TEI-10)*, 1997.

Roni Rosenfeld. Two decades of statistical language modeling: where do we go from here? *Proc. of IEEE*, 88 (8):1270–1278, Aug 2000.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition, 2003.

Charles Schafer and Elliott Franco Drabek. Models for inuktitut-english word alignment. In *Proc. of ACL Workshop on Building and Using Parallel Texts*, pages 79–82, Jun 2005.

Libin Shen, Anoop Sarkar, and Franz Josef Och. Discriminative reranking for machine translation. In *Proc. of HLT-NAACL*, pages 177–184, May 2004.

Stuart M. Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proc. of COLING*, pages 253–258, 1990.

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. Translating with non-contiguous phrases. In *Proc. of HLT-EMNLP*, pages 755–762, Oct 2005.

David A. Smith and Noah Smith. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proc. of EMNLP*, pages 49–56, Jul 2004.

Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*, pages 223–231, Aug 2006.

Ben Taskar. *Learning Structured Prediction Models: A Large-Margin Approach*. PhD thesis, Stanford University, Dec 2004.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. A discriminative matching approach to word alignment. In *Proc. of HLT-EMNLP*, pages 73–80, Oct 2005.

Christoph Tillman and Hermann Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):98–133, Mar 2003.

Christoph Tillmann and Tong Zhang. A discriminative global training algorithm for statistical MT. In *Proc. of ACL-COLING*, pages 721–728, Jul 2006.

Christoph Tillmann, Stephen Vogel, Hermann Ney, and Alex Zubiaga. A DP-based search using monotone alignments in statistical translation. In *Proc. of ACL-EACL*, pages 289–296, 1997.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. Extensions to HMM-based statistical word alignment models. In *Proc. of EMNLP*, pages 87–94, Jul 2002.

Joseph P. Turian, Luke Shen, and I. Dan Melamed. Evaluation of machine translation and its evaluation. In *Proc. of MT Summit IX*, Sep 2003.

Nicola Ueffing and Hermann Ney. Word-level confidence estimation for machine translation using phrase-based translation models. In *Proc. of HLT-EMNLP*, pages 763–770, Oct 2005.

Nicola Ueffing, Franz Josef Och, and Hermann Ney. Generation of word graphs in statistical machine translation. In *Proc. of EMNLP*, pages 156–163, Jul 2002.

Ashish Venugopal, Andreas Zollmann, and Alex Waibel. Training and evaluating error minimization rules for statistical machine translation. In *Proc. of ACL Workshop on Building and Using Parallel Texts*, pages 208–215, Jun 2005.

Ashish Venugopal, Andreas Zollmann, and Stephan Vogel. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *Proc. of HLT-NAACL*, 2007. To appear.

K. Vijay-Shanker, David Weir, and Aravind K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. of ACL*, pages 104–111, 1987.

Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

Stephan Vogel, Hermann Ney, and Christoph Tillman. HMM-based word alignment in statistical machine translation. In *Proc. of COLING*, pages 836–841, Aug 1996.

Jianqiang Wang. *Matching Meaning for Cross-Language Information Retrieval*. PhD thesis, University of Maryland, 2005.

Ye-Yi Wang and Alex Waibel. Decoding algorithm in statistical machine translation. In *Proc. of ACL-EACL*, pages 366–372, Jul 1997.

Taro Watanabe and Eiichiro Sumita. Bidirectional decoding for statistical machine translation. In *Proc. of COLING*, pages 1079–1085, Aug 2002.

Warren Weaver. Translation. In William N. Locke and A. Donald Booth, editors, *Machine Translation of Languages: Fourteen Essays*, chapter 1, pages 15–23. MIT Press, 1955. Reprint of 1949 memorandum.

Benjamin Wellington, Joseph Turian, Chris Pike, and I. Dan Melamed. Scalable purely-discriminative training for word and tree transducers. In *Proc. of AMTA*, pages 251–260, Aug 2006a.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. Empirical lower bounds on the complexity of translational equivalence. In *Proc. of ACL-COLING*, pages 977–984, Jun 2006b.

Dekai Wu. Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *Proc. of IJCAI*, pages 1328–1335, Aug 1995a.

Dekai Wu. Grammarless extraction of phrasal translation examples from parallel texts. In *Proc. of TMI*, pages 354–372, Jul 1995b.

Dekai Wu. A polynomial-time algorithm for statistical machine translation. In *Proc. of ACL*, pages 152–158, Jun 1996.

Dekai Wu and Hongsing Wong. Machine translation with a stochastic grammatical channel. In *Proc. of ACL-COLING*, pages 1408–1415, Aug 1998.

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, Jul 2004.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proc. of ACL-EACL*, 2001.

Kenji Yamada and Kevin Knight. A decoder for syntax-based statistical MT. In *Proc. of ACL*, pages 303–310, Jul 2002.

David Yarowsky and Grace Ngai. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proc. of NAACL*, pages 200–207, Jun 2001.

David Yarowsky, Grace Ngai, and Richard Wicentowski. Inducing mulilingual text analysis tools via robust projection across aligned corpora. In *Proc. of HLT*, pages 109–116, 2001.

Richard Zens and Hermann Ney. A comparative study on reordering constraints in statistical machine translation. In *Proc. of ACL*, pages 144–151, Jul 2003.

Richard Zens and Hermann Ney. Improvements in phrase-based statistical machine translation. In *Proc. of HLT-NAACL*, pages 257–264, May 2004.

Richard Zens and Hermann Ney. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *Proc. of HLT-NAACL*, 2007. To appear.

Hao Zhang and Daniel Gildea. Stochastic lexicalized inversion transduction grammar for alignment. In *Proc. of ACL*, pages 475–482, Jun 2005.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. Synchronous binarization for machine translation. In *Proc. of HLT-NAACL*, pages 256–263, Jun 2006a.

Ying Zhang and Stephan Vogel. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proc. of EAMT*, May 2005.

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. Distributed language modeling for N-best list reranking. In *Proc. of EMNLP*, pages 216–223, Jul 2006b.