Statistical Machine Translation

ADAM LOPEZ

University of Edinburgh

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 new ideas are constantly introduced. This survey presents a tutorial overview of the state of the art. We describe the context of the current research and then move to a formal problem description and an overview of the main subproblems: translation 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 a discussion of future directions.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: Statistical computing; I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Machine translation*; I.5.1 [**Pattern Recognition**]: Models—*Statistical*

General Terms: Algorithms

Additional Key Words and Phrases: Natural language processing, machine translation

ACM Reference Format:

Lopez, A. 2008. Statistical machine translation. ACM Comput. Surv., 40, 3, Article 8 (August 2008), 49 pages DOI = 10.1145/1380584.1380586 http://doi.acm.org/10.1145/1380584.1380586

1. INTRODUCTION

Machine translation (MT) is the automatic translation from one natural language into another using computers. Interest in MT is nearly as old as the electronic computer—popular accounts trace its modern origins to a letter written by Warren Weaver in 1949, only a few years after ENIAC came online.[1] It has since remained a key application in the field of natural language processing (NLP). A good historical overview is given by Hutchins [2007], and a comprehensive general survey is given by Dorr, Jordan, and Benoit [1999].

[1]This letter is reproduced as Weaver [1955].

This research was supported in part by ONR MURI Contract FCPO.810548265, the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-02-001, and the EuroMatrix Project funded by the European Commission (6th Framework Programme). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author and do not necessarily reflect the view of DARPA. This research was carried out while the author was at the University of Maryland.

Author's address: A. Lopez, Department of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom; email: alopez@inf.ed.ac.uk.

Statistical machine translation (SMT) is an approach to MT that is characterized by the use of machine learning methods. In less than two decades, SMT has come to dominate academic MT research, and has gained a share of the commercial MT market. 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 article are to characterize the core ideas of SMT and provide a taxonomy of various approaches. We have tried to make the survey as self-contained as possible. However, SMT draws from many fundamental research areas in computer science, so some knowledge of automata theory, formal languages, search, and data structures will be beneficial. Familiarity with statistical theory and mathematical optimization techniques used in machine learning will also be helpful, but we will focus on the main ideas and intuitions behind the mathematics rather than full mathematical rigor, which is in any case beyond the scope of this work. We will touch briefly on a few linguistic concepts, but for our purposes SMT can be understood in pure computer science terms.

Knight [1997, 1999b] has written excellent tutorial introductions, but they do not cover many later developments in the field. Knight and Marcu [2005] give a very brief survey. At the time of this writing, other materials in preparation include Koehn [2008]. Chapter 25 of Jurafsky [2008] contains a brief introduction to the larger field of machine translation. For greater coverage of related fundamental research, refer to textbooks on NLP [Manning and Schütze 1999; Jurafsky and Martin 2008], artificial intelligence [Russell and Norvig 2003] machine learning [Mitchell 1997], or formal language theory [Hopcroft and Ullman 1979; Sipser 2005].

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 a 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. Workshops have shown that translation systems can be built for a wide variety of language pairs within similar time frames [Koehn and Monz 2005; Koehn and Monz 2006; Callison-Burch et al. 2007]. 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].

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 text [Koehn 2005], bilingual news text, 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 data and billions of statistics. Advances in processor speed, random access memory size, secondary storage, and grid computing 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.[2] Academic workshops coordinate similar evaluations [Koehn and Monz 2005; Koehn and Monz 2006; Callison-Burch et al. 2007].

(5) Several projects have focused on the development of freely available SMT toolkits [Al-Onaizan et al. 1999; Germann et al. 2001; Och and Ney 2003; Koehn 2004a; Burbank et al. 2005; Olteanu et al. 2006; Koehn et al. 2007]. Many 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 V_F, and transform it into a sequence of tokens in the *target language* with vocabulary V_E.[3] 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 that 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 V_E and V_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 $f_1^J \in {V_F}^J$, and a sequence of I target words as $e_1 e_2 ... e_I$ or $e_1^I \in {V_E}^I$. 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. An excercise familiar to those who have learned a foreign language is to draw a line between the words in the sentence that are translations of each other. For instance, we can see that Chinese word 北 is translated as the English word *north*, and we could draw a line between them. We say that such words are *aligned*. An example word alignment is shown in Figure 2. This illustrates that translational equivalence can be decomposed into a number of smaller equivalence problems at the word level.

Word translation is often *ambiguous*. For instance, we might reasonably translate 北 as *northern* without loss of meaning. However, it is not uncommon for the different possible translations of a word to have very different meanings. Often, the correct

[2]These evaluations are described at http://www.nist.gov/speech/tests/mt/index.htm.

[3]We follow the widely used notation of Brown et al. [1993], 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 .

Fig. 1. An example of translationally equivalent sentences. We give an English gloss for each Chinese word.

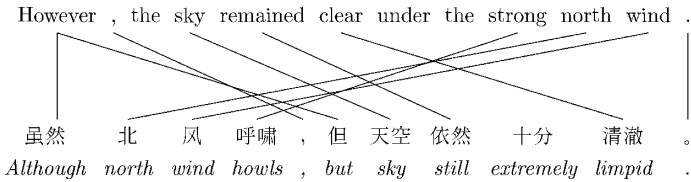

Fig. 2. An alignment of the the sentence in Figure 1.

choice will depend on context. Therefore, our system will need some mechanism to choose between several possible options for each translation 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 need some mechanism to correctly *reorder* the words. Reordering is typically dependent on the syntactic structure of the target language. For instance, in English sentences, the typical sentence order is *subject-verb-object* (SVO). In Japanse, it is *subject-object-verb* (SOV). As with word translation, reordering decisions often entail resolving some kind of ambiguity.

Translation can therefore be thought of as making a sequence of word translation and reordering decisions. We rewrite the source sentence one decision at a time, 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.

(1) First, we must describe the series of steps that transform a source sentence into a target sentence. We can think of this as creating a story about how a human translator might perform this task. This story is called a *translational equivalence model*, or more simply a *model*. All of the translational equivalence models that we will consider derive from concepts from automata and language theory. We describe them in Section 2.

(2) Next, we want to enable our model to make good choices when faced with a decision to resolve some ambiguity. We need to develop a *parameterization* of the model that will enable us to assign a score to every possible source and target sentence pair that our model might consider. We describe parameterization in Section 3. Taken together, translational equivalence modeling and parameterization are often combined under the rubric of *modeling*.[4]

(3) The parameterization defines a set of statistics called *parameters* used to score the model, but we need to associate values to these parameters. This is called *parameter estimation*, and it is based on machine learning methods. We describe parameter estimation in Section 4.

[4]Although these two steps are conflated under this term in the literature, following Brown et al. [1990], for didactic reasons we find it helpful to separate them.

(4) Finally, when we are presented with input sentence, we must search for the highest-scoring translation according to our model. This is called *decoding*. We describe decoding in Section 5.

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

2. MODELING PART I: TRANSLATIONAL EQUIVALENCE

Broadly speaking, a model is simply the set of all rules employed by an MT system to transform a source sentence into a target sentence. In principle these rules may come from anywhere. In most SMT systems they are automatically extracted from a parallel corpus. The extraction process is described in more detail in Section 4.2. In this section, we describe the various types of models.

Most popular models can be described by one of two formalisms: finite-state transducers (FST) or synchronous context-free grammars (SCFG).[5] 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.[6] 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 Section 2.1, and SCFG models are described in Section 2.2. We briefly touch on other model types in Section 2.3.

2.1. Finite-State Transducer Models

Finite-state transducers are extensions of finite-state automata (FSA). 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. Each transition in $D \subseteq \{S \times S \times L\}$ 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. Finite-state methods are widely used in NLP for problems such as automatic speech recognition, optical character recognition, and part-of-speech tagging.

Finite-state transducers extend FSAs with a second label set. Each transition includes a label from both sets. We can imagine that the transducer operates on an input string and an output string. When a label from the first set is read from the input while traversing the transition, the label for the other is written to the output. A transition labelled with x from set L_1 and y from set L_2 therefore 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. We can compose FSTs by making the output of one FST the input to another, giving us

[5]This distinction may be confusing, since finite state transducers come to us 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. However, we note that there is growing interest in *tree transducers*, a class of objects in the automata literature with computational properties similar (but not identical) to those of context-free grammar [Knight and Graehl 2005; Galley et al. 2006; Marcu et al. 2006].

[6]We can generalize these formalisms even further to handle an arbitrary number of dimensions [Melamed 2003]. This is useful for *multi-source translation*, wherein we have a document already translated in multiple languages, and we would like to translate into another language [Och and Ney 2001].

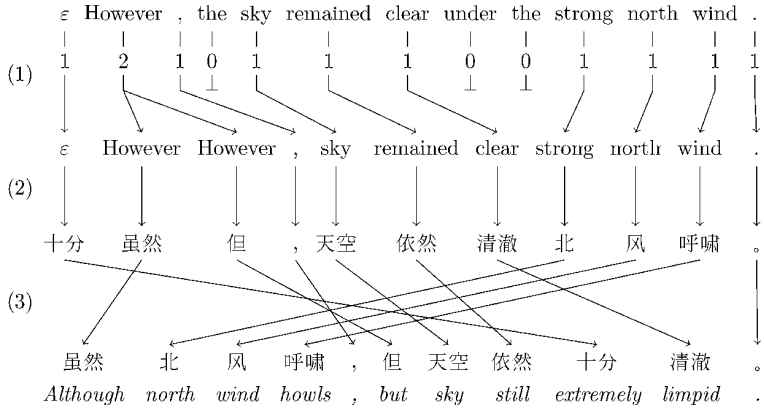

Fig. 3. Visualization of IBM Model 4. This model of translation takes three steps. (1) Each English word (and the null word) selects a fertility—the number of Chinese words to which it corresponds. (2) Each English word produces a number of Chinese words corresponding to its fertility. Each Chinese word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

a useful modeling tool. For each of the model types that we describe in this section, we will show how it can be implemented with composed FSTs.

We first describe *word-based models* (Section 2.1.1), which introduce many of the common problems in translation modeling. They are followed by *phrase-based models* (Section 2.1.2).

2.1.1. Word-Based Models. SMT continues to be influenced by the groundbreaking IBM approach [Brown et al. 1990; Brown et al. 1993; Berger et al. 1994]. The IBM Models are *word-based models* and represent the first generation of SMT models. They illustrate many common modeling concepts. We focus on a representative example, IBM Model 4.

For reasons that we will explain in Section 3.1, IBM Model 4 is 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.

The model entails three steps (Figure 3). Each step corresponds to a single transducer in a composed set [Knight and Al-Onaizan 1998]. 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 enables us to define a translational equivalence between source and target sequences of different lengths.

(2) Each copy of each target word produces a single source word. This represents the translation of individual words.

(3) The translated words are permuted into their final order.

These steps are also applied to a special empty token ε, called the *null word*—or more simply *null*—and denoted e_0. *Null translation* accounts for target words that are dropped in translation, as is often the case with function words.

Note that the IBM Model 4 alignment is asymmetric. Each source word can align to exactly one target word or the null word. However, a target word can link to an

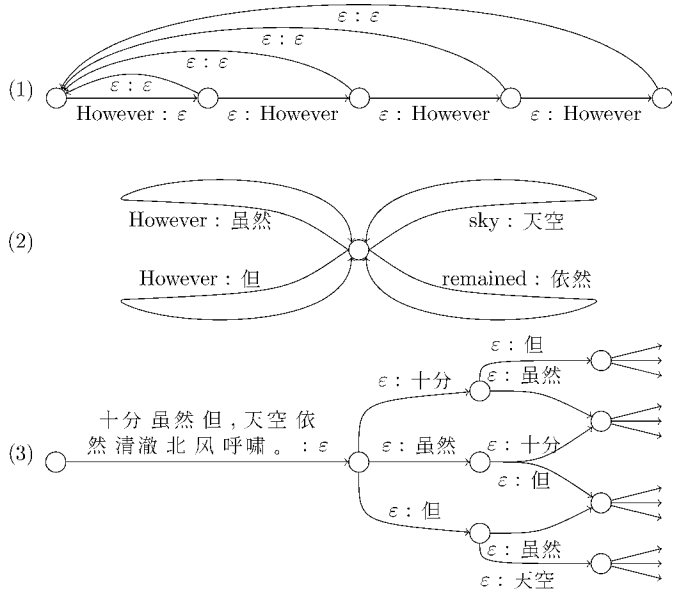

Fig. 4. Visualization of the finite-state transducer conception of IBM Model 4. We show only a portion of each transducer. Transducer (1) copies the input word to its output a number of times according to its fertility; (2) corresponds to word-to-word translation, and (3) corresponds to reordering. Transducer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

arbitrary number of source words, as defined by its fertility. This will be important when we discuss word alignment (Section 4.2.2).

The reordering step exposes a key difficulty of finite-state transducers for translation. There is no efficient way to represent reordering in a standard finite-state transducer. They are designed to represent relationships between strings with a monotonic alignment—in other words, if an input label at position i is aligned to an output label at position j, then an input label at position $i' > i$ will be aligned to an output label at position $j' > j$. This is fine for problems such as automatic speech recognition and part-of-speech tagging, where monotonicity is the natural relationship between two sequences. However, as we have seen, words are typically reordered in real translation.

One solution to this discrepancy is to simply ignore the reordering problem and require monotonic alignments [Tillmann et al. 1997; Zens and Ney 2004; Banchs et al. 2005]. This enables very fast decoding, but generally leads to less accurate translation, particularly for languages with naturally different word orders.

At the other end of the spectrum is full permutation of the J source words. An FST to do this contains $O(J!)$ paths and $O(2^J)$ states [Och and Ney 2003]. However, search for the best permutation is NP-complete, as Knight [1999a] shows by reduction to the Traveling Salesman Problem.

Most models take a middle ground, using a mostly monotonic approach but allowing any of the k leftmost uncovered words to be translated. The setting $k = 4$ is sometimes called the *IBM constraint*, following Berger et al. [1996]. This method enables local reordering, which accounts for much language variation. However, it still prevents *long-distance reordering*, such as the movement of "north" (北) in our example. Accounting for long-distance reordering without resorting to arbitrary permutation requires additional modeling. For instance, Tillman and Ney [2003] describe targeted

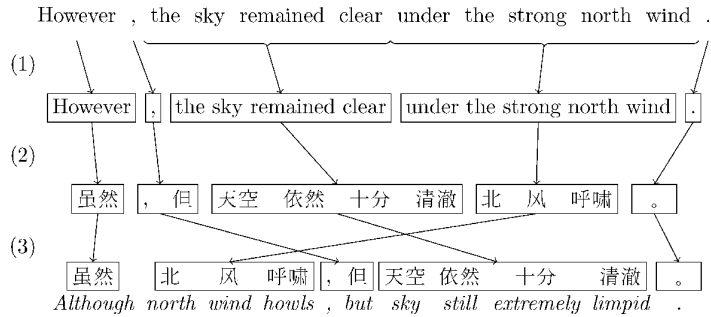

Fig. 5. Visualization of the phrase-based model of translation. The model involves three steps. (1) The English sentence is segmented into "phrases"—arbitrary contiguous sequences of words. (2) Each phrase is translated. (3) The translated phrases are reordered. As in Figure 3, each arrow corresponds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

long-distance reordering for verb movement in German to English translation, using language-specific heuristics.

As we can see, a dominant theme of translation modeling is the constant tension between expressive modeling of reordering phenomena, and model complexity.

2.1.2. Phrase-Based Models. In real translation, it is common for contiguous sequences of words to translate as a unit. For instance, the expression "北 风" is usually translated as "north wind." However, in word-based translation models, a substitution and reordering decision is made separately for each individual word. For instance, our model would have to translate 北 as "north" and 风 as "wind," and then order them monotonically, with no intervening words. Many decisions invite many opportunities for error. This often produces "word salad"—a translation in which many words are correct, but their order is a confusing jumble.

Phrase-based translation addresses this problem [Och et al. 1999; Marcu and Wong 2002; Koehn et al. 2003; Och and Ney 2004]. In phrase-based models the unit of translation may be any contiguous sequence of words, called a *phrase*.[7] Null translation and fertility are gone. Each source phrase is nonempty and translates to exactly one nonempty target phrase. However, we do not require the phrases to have equal length, so our model can still produce translations of different length. Now we can characterize the substitution of "north wind" with "北 风" as an atomic operation. If in our parallel corpus we have only ever seen "north" and "wind" separately, we can still translate them using one-word phrases.

The translation process takes three steps (Figure 5).

(1) The sentence is first split into phrases.

(2) Each phrase is translated.

(3) The translated phrases are permuted into their final order. The permutation problem and its solutions are identical to those in word-based translation.

A cascade of transducers implementing this is shown in Figure 6. A more detailed implementation is described by Kumar et al. [2006].

The most general form of phrase-based model makes no requirement of individual word alignments [Marcu and Wong 2002]. Explicit internal word alignments are

[7]In this usage, the term phrase has no specific linguistic sense.

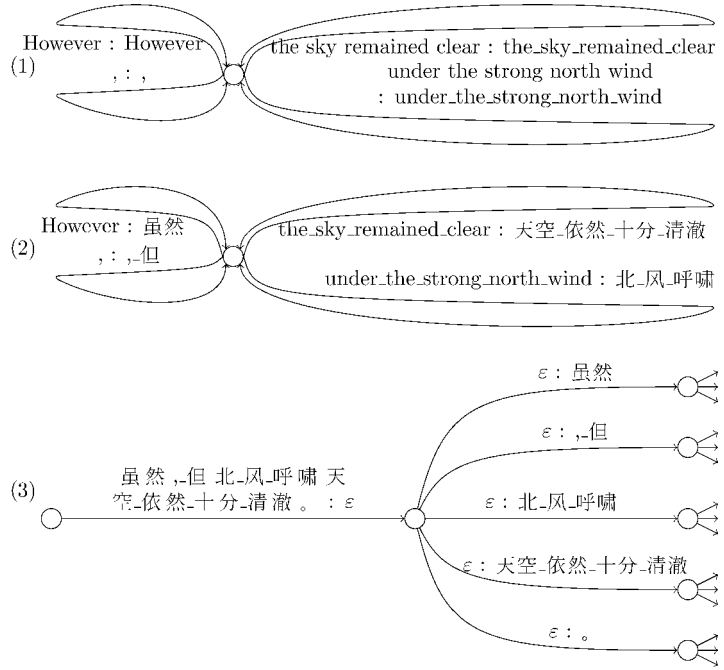

Fig. 6. Visualization of the finite-state transducer conception of phrase-based translation. We show only a portion of each transducer. Transducer (1) segments the target sentence into phrases; (2) performs word-to-word translation; (3) reorders the phrases in the same way that the reordering transducer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

sometimes assumed for the purposes of parameterization [Koehn et al. 2003; Kumar et al. 2006].

A variant of phrase-based models is the *alignment template model* [Och et al. 1999; Och and Ney 2004]. 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 another variant of phrase-based translation in which phrases can be discontiguous, and gaps must be filled in with other phrases.

Phrase-based translation is implemented in the Pharaoh toolkit [Koehn 2004a] and its open-source successor Moses [Koehn et al. 2007].[8] They are widely used in the SMT research community.

Phrase-based models produce better translations than word-based models, and they are widely used. They successfully model many local reorderings, and individual passages are often fluent. However, they cannot easily model long-distance reordering without invoking the expense of arbitrary permutation. This sometimes leads to *phrase salad*. It is sometimes argued that linguistic notions of syntax are needed to adequately model these phenomena [Burbank et al. 2005]. However, incorporation of syntax into phrase-based models has had mixed results, with some failures [Koehn et al. 2003; Och et al. 2004b] and some successes [Collins et al. 2005; Wang et al. 2007]. In

[8]The Moses toolkit is available at http://www.statmt.org/moses/.

general, phrase-based models seem to be a poor fit for syntax, which usually depends on hierarchical models of string generation.

Formal language theory tells us that the set of *regular languages* that can be generated using finite-state automata is a strict subset of the set of *context-free languages* that can be generated using push-down automata. To gain modeling power, we will now look at synchronous formalisms from this theoretically more powerful class.

2.2. Synchronous Context-Free Grammar Models

Compared with the regular languages generated by FSAs, *context-free grammars* (CFG) confer a couple of potential benefits that relate to our modeling problem. First, they are closely tied to some linguistic representations of syntax. Second, in their synchronous form, they can easily represent long-distance reordering without the exponential complexity of permutation. However, added modeling power comes with added modeling challenges, and meeting these challenges is currently an area of much active research. Indeed, there is some skepticism over the superiority of synchronous grammar models over finite-state methods. For the remainder of this section, we describe translation models based on *synchronous context-free grammars* (SCFG), and describe some of their modeling challenges.

SCFGs are known in different guises as *syntax-directed translation* [Lewis and Stearns 1968], *inversion transduction grammar* [Wu 1995b], *head transducers* [Alshawi et al. 2000], and a number of other names. A formalism that generalizes these is multitext grammar [Melamed 2003]. Chiang [2006] provides a good overview of SCFG and several variants. In the following discussion, we will use the notation of SCFG, which is relatively easy to understand.

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 nonterminal symbols N, terminal symbols T, and productions $D = \{N \rightarrow \{N \cup T\}^*\}$. Each production describes the inplace replacement of the productions *right-hand side* nonterminal with the *left-hand side* sequence of terminals and nonterminals. We begin by writing a special *root* nonterminal symbol to the output. This symbol is rewritten using a rule $d \in D$. Rewriting of nonterminal symbols continues recursively until the output contains only terminal symbols. CFG is popular in natural language parsing, where terminal symbols are words (i.e., $T = V_E$) and nonterminal symbols represent syntactic categories. Consider the following fragment of a CFG grammar.[9]

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

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.

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

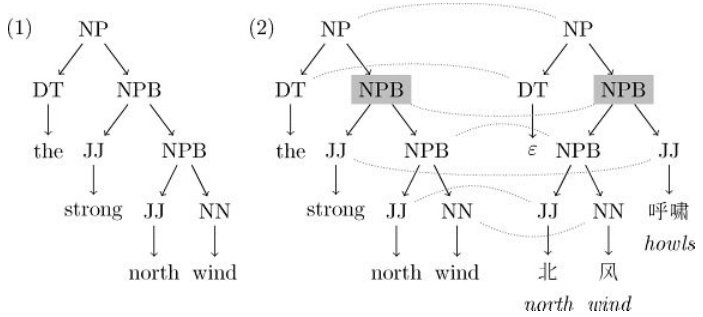

Fig. 7. Visualization of the CFG derivation and SCFG derivations. Derivation happens in exactly the same way in CFG (1) and SCFG (2). Each nonterminal symbol is replaced by the contents of the right-hand-side of a rule whose left-hand-side matches the symbol. In our illustration, each arrow represents a single production. The difference in SCFG is that we specify two outputs rather than one. Each of the nonterminal 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 nonterminal 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).

In SCFG, the grammar specifies *two* output strings for each production. It defines a correspondence between strings via the use of *co-indexed* nonterminals. Consider a fragment of SCFG grammar.

$$\text{NP} \longrightarrow \text{DT}_{\boxed{1}}\text{NPB}_{\boxed{2}} \;/\; \text{DT}_{\boxed{1}}\text{NPB}_{\boxed{2}} \tag{S1}$$

$$\text{NPB} \longrightarrow \text{JJ}_{\boxed{1}}\text{NN}_{\boxed{2}} \;/\; \text{JJ}_{\boxed{1}}\text{NN}_{\boxed{2}} \tag{S2}$$

$$\text{NPB} \longrightarrow \text{NPB}_{\boxed{1}}\text{JJ}_{\boxed{2}} \;/\; \text{JJ}_{\boxed{2}}\text{NPB}_{\boxed{1}} \tag{S3}$$

$$\text{DT} \longrightarrow \text{the} \;/\; \varepsilon \tag{S4}$$

$$\text{JJ} \longrightarrow \text{strong} \;/\; \text{呼啸} \tag{S5}$$

$$\text{JJ} \longrightarrow \text{north} \;/\; \text{北} \tag{S6}$$

$$\text{NN} \longrightarrow \text{wind} \;/\; \text{风} \tag{S7}$$

The two outputs are separated by a slash (/). The boxed numbers are co-indexes. When two nonterminals share the same co-index, they are aligned. We can think of SCFG as generating isomorphic trees, in which the nonterminal nodes of each tree are aligned. One tree can be transformed into another by rotating its nonterminal nodes, as if it were an Alexander Calder mobile (Figure 7). Note that if we ignore the source string dimension, the rules in this SCFG correspond to the rules that appear in our CFG example. Also notice that we can infer an alignment between source and target words based on the alignment of their parent nonterminals. In this way SCFG defines a mapping between both strings and trees, and has a number of uses depending on the relationship that we are interested in [Wu 1995b; Melamed 2004b].

Normal forms and complexity analysis for various flavors of SCFG are presented in Aho and Ullman [1969] and Melamed [2003]. The number of possible reorderings is quite large. Church and Patil [1982] show that the number of binary-branching trees that can generate a string is related to a combinatorial function, the *Catalan number*. Zens and Ney [2003] show that the number of reorderings in a binary SCFG are related to another combinatorial function, the *Shröder number*. However, due to recursive sharing of subtrees among many derivations, we can parse SCFG in polynomial time

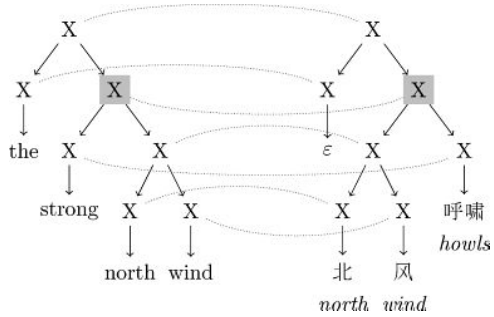

Fig. 8. Visualization of a bracketing grammar derivation. Each arrow corresponds to a grammar production. Rules that involve reordering are highlighted, and the order of the target language sentence is illustrated beneath the syntax tree.

with dynamic programming algorithms [Melamed 2003]. This makes SCFG models of translational equivalence less computationally expensive than full permutation, even though they can model long-distance reordering [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 (Section 2.2.1), *syntax-based translation* that exploits linguistic syntax (Section 2.2.2), and *hierarchical phrase-based translation* that combines the insights of phrase-based models with syntactic structure (Section 2.2.3).

2.2.1. Bracketing Grammars. One reason to use SCFG is efficient expression of reordering. In FST models, long-distance reordering is difficult to model. The most permissive approach—arbitrary permutation—is exponential in sentence length. In contrast, SCFG can represent long-distance reordering while remaining polynomial in sentence length. This motivates the use of bracketing grammars. They represent all possible reorderings consistent with a binary bracketing of the input string [Wu 1996]. Bracketing grammars use a single undifferentiated nonterminal symbol and three rules.

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

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

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

In Rule B1 we define symbols $e \in V_E \cup \varepsilon$ and $f \in V_F \cup \varepsilon$. Instances of this rule model word-to-word alignments.

The polynomial runtime of bracketing grammar comes at a cost. In contrast with permutation, the bracketing grammar cannot represent certain reorderings. A canonical example is the so-called inside-outside alignment (Figure 9). It was originally argued that this reordering does not occur in real translation [Wu 1995a]. However, Wellington et al. [2006] provided counterexamples from real data. On the other hand, Zens and Ney [2003] show empirically that bracketing grammars can model more of the reorderings observed in real data than a word-based model using the IBM constraint.

One way to increase the reordering power of an SCFG is to increase the number of coindexed nonterminal symbols in each rule. In general, any alignment can be expressed in a grammar with enough coindexed nonterminals [Aho and Ullman 1969]. The tradeoff is increased complexity. In particular, upper bound complexity is

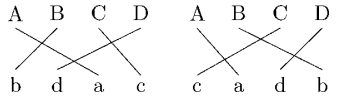

Fig. 9. Visualization of the so-called inside-outside alignments that are not possible using bracketing transduction grammar. Due to the interleaving words in these configurations, we cannot construct a binary-branching SCFG that is isomorphic for these strings, although a SCFG with four nonterminals can produce them. 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.

exponential in the maximum number of coindexed nonterminals. As this number increases, the reordering power and the complexity of the grammar approach those of permutation.

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].

Xiong et al. [2006] adapted bracketing grammars to phrase translation.

2.2.2. Syntax-Based Translation. In addition to long-distance reordering, a perceived benefit of 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.[10] Monolingual syntax resembles our example fragment CFG (Rules C1-C8). 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 keep linguistic phrases intact, a property that is often (but not universally) observed in real translation [Fox 2002; Wellington et al. 2006]. This is called a *syntax-based translation*. The example derivation from Figure 7 is an illustration of this.

The hypothesis investigated by syntax-based translation was that reorderings would respect linguistic syntax in translation. Empirical evidence only partially supports this. Fox [2002] shows that reordering tends to respect the boundaries of syntactic phrases, but also describes some systematic exceptions. Additional discrepancies are described by Wellington et al. [2006]. The rigidity of full isomorphism at the level of SCFG productions harms syntax-based translation. These difficulties have prompted investigation into more powerful formalisms for syntactic translation [Galley et al. 2004; Knight and Graehl 2005; Galley et al. 2006].

2.2.3. Hierarchical Phrase-Based Translation. The SCFG models that we have described share a key weakness of word-based FST-models: they enable only word-to-word translation, which requires a reordering decision for each word. We know that phrase-based models solve this. 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].

In this grammar, no linguistic syntax is required. A single undifferentiated nonterminal X is used in the main productions, and a maximum of two nonterminals are permitted in the right-hand size of any rule, just as in bracketing grammar. However, unlike a bracketing grammar, the right-hand side may also contain a number of

[10]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.

Fig. 10. Visualization of hierarchical phrase-based translation.

terminal symbols in both languages. This corresponds to the basic insight of phrase-based translation, in that each rule can represent a mapping between sequences of words. Essentially, the rules represent phrases that may be reordered recursively. Consider the following grammar fragment.

$$X \longrightarrow \text{However , } X_{\boxed{1}} X_{\boxed{2}} \text{ . } / \text{虽然 } X_{\boxed{2}} \text{ , 但 } X_{\boxed{1}} \text{ 。} \quad \text{(H1)}$$

$$X \longrightarrow \text{under the strong north wind } / \text{北 风 呼啸} \quad \text{(H2)}$$

$$X \longrightarrow \text{the sky remained clear } / \text{天空 依然 十分 清澈} \quad \text{(H3)}$$

In this grammar, recursivity is captured in Rule H1. A derivation is illustrated in Figure 10.

Chiang [2005, 2007] showed that this model was competitive with a standard phrase-based models, outperforming Pharaoh on standard test sets.

2.3. Other Models of Translational Equivalence

FST and SCFG models represent a good cross section of popular models. However, a number of models do not fit these characterizations.

2.3.1. More Powerful Formalisms. 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 [Shieber and Schabes 1990]. Formally, tree-adjoining grammars are part of a large class of formalisms known as linear context-free rewriting systems [Vijay-Shanker et al. 1987; Joshi et al. 1991]. 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], which is equivalent to a linear context-free rewriting system, is the basis of an SMT system [Burbank et al. 2005].

2.3.2. Syntactic Phrase-Based Models. Much active research aims to combine the advantages of hierarchical reordering, syntax, and phrases [Marcu et al. 2006; Galley et al. 2006; Quirk et al. 2005]. Most of these models employ syntactic reordering models more complex than those that can be described with SCFG. The formal description for several of these models is based on *tree transducers*, which describe operations on tree fragments rather than strings. SCFG translation can be modeled with tree transducers, although in general they are strictly more powerful than SCFG. Systems

described using tree transducers are increasingly common [Graehl and Knight 2004; Galley et al. 2006; Marcu et al. 2006; Wellington et al. 2006]. For a good introduction to tree transducers, refer to Knight and Graehl [2005].

2.3.3. 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 [Collins et al. 1999]. Therefore, we can construct SCFGs that correspond to dependency grammar [Melamed 2003]. Dependency grammar translation models are described by Gildea [2004] and Quirk et al. [2005].

3. MODELING PART II: PARAMETERIZATION

Translational equivalence models allow us to enumerate possible structural relationships between pairs of strings. However, even within the constraints of a strict model, the ambiguity of natural language results in a very large number of possible target sentences for any input source sentence. Our translation system needs a mechanism to choose between them.

This mechanism comes from our second topic in modeling: *parameterization*. We design a function that allows us to assign a real-valued score to any pair of source and target sentences. The general forms of these models are similar to those in 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 multidimensional. 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. 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. The model may be probabilistic, meaning that we constrain it in one of two ways. In a *joint model*, denoted $P(\mathbf{x}, \mathbf{y})$, we introduce two 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 as $P(x, y)$.

In a *conditional model*, denoted $P(\mathbf{x}|\mathbf{y})$, we introduce the following 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 $\mathbf{y}$ 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 $\mathbf{f}$ ranges over ${V_F}^*$ and our output $\mathbf{e}$ ranges over ${V_E}^*$. 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 $\mathbf{e} = e_1^I$. We 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 (Section 2.1) and a set of grammar productions in the case of SCFG models (Section 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 use (e_1^I, d_1^M) to remind us of our ultimate goal. For notational purposes we also define a predicate $Y(e_1^I, d_1^M)$ that is true when the derivation 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 derivation of the output. However, these models usually define multiple structures that can relate the same pair (e_1^I, f_1^J). The value of $P(\mathbf{e}|\mathbf{f})$ is therefore obtained by summing the probabilities of all derivations that yield $\mathbf{e}$.

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

Unfortunately, computation of this sum involves exponential complexity in both FST [Brown et al. 1993] and SCFG models [Melamed 2004b]. 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 parameterization of the model determines the rank of candidates. Usually these tasks are integrated, but not necessarily (see Section 5.3).

Although the function $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$ ranges over discrete sets, these sets are very large or even infinite. This poses both practical and theoretical problems. There is no efficient way to enumerate the function. Furthermore, we will never have enough training data to reliably learn what its values should be. The goal of mathematical modeling, then, is to *parameterize* the function in such a way that we can efficiently and reliably learn it. There are a number of ways to accomplish this. We describe *generative models* in Section 3.1 and *discriminative models* in Section 3.2.

3.1. Generative Models

We begin our discussion of generative models by introducing some statistical machinery. Later in the section, we will illustrate this with an example.

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

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

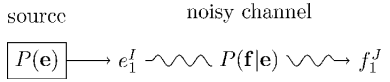

Fig. 11. The noisy channel model of sentence pair generation. The source model $P(\mathbf{e})$ produces e_1^I, which is then transformed by the channel model $P(\mathbf{f}|\mathbf{e})$ to produce f_1^J. If we are given only f_1^J, we can try to deduce an e_1^I using our knowledge of both models.

Generative models 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 (3).

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

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

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

In decoding we can ignore the denominator $P(\mathbf{f})$ because it is constant for any input f_1^J. Therefore, we do not 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 was inspired by its successful use in automatic speech recognition [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 as we saw in Section 2.1.1. 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 errorful. By applying them together we hope to counterbalance their errors.

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. This idea was suggested 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*.

3.1.1. Language Models. In language modeling, our goal is to find a tractable representation of the function $P(e_1^I)$. Generative models use probabilistic tools for this. One of these is chain rule, which allows us to write the following.

$$P\left(e_1^I\right) = \prod_{j=1}^{I} P\left(e_i \mid e_1^{i-1}\right). \tag{5}$$

This equation tells us that the conditional probability of the sentence e_1^I is simply the product of many small probabilities, each of which corresponds to a single word.

Equation (5) helps to simplify our problem, but not completely. For instance, the distribution $P(e_I|e_1^{I-1})$ assigned to the last word of the sentence contains nearly as many terms as $P(e_1^I)$ itself. In order to simplify the model even further we introduce the idea of *conditional independence*. When we say that a variable $\mathbf{x}$ is conditionally independent of $\mathbf{y}$, we mean that $P(\mathbf{x}|\mathbf{y}) = P(\mathbf{x})$. 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.

In language modeling, the simplest assumption we can make is that the probability of word e_i is conditionally independent of all but the $n-1$ preceding words e_{i-n}^{i-1}. We call e_{i-n}^{i} an *n-gram* and the language model based on this independence assumption is an *n-gram language model*. We assume without loss of generality that the first word e_1 is preceded by $n-1$ distinguished start symbols not in V_E.

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 an n-gram model.

$$P\left(e_1^I\right) = \prod_{j=1}^{I} P_\delta\left(e_i|e_1^{i-1}\right) = \prod_{j=1}^{I} P\left(e_i|e_{i-n}^{i-1}\right). \tag{6}$$

Most language models take this form, inherited from SMT's roots in speech recognition. The discussion of n-grams barely scratches the surface of language modeling, which is a full topic in its own right, well beyond the scope of this paper. For a good overview, refer to a text on speech recognition, such as Jelinek [1998].

Language modeling has not received much special attention in the SMT community, which has preferred to focus on the more specialized translation models. However, it continues to be an active area of research, especially in the speech recognition community. Most SMT systems simply borrow models that were popularized for speech. However, there is little doubt that better language modeling leads to better translation [Eck et al. 2004; Kirchhoff and Yang 2005; Och 2005; Zhang et al. 2006; Brants et al. 2007].

A few syntax-based models are constructed to take advantage of syntactic language models based on context-free grammars [Wu and Wong 1998; Charniak et al. 2003; Marcu et al. 2006].

3.1.2. Translation Models. Taking advantage of the statistical machinery that we have introduced, we now turn to the *translation model* $P(f_1^J, d_1^M|e_1^I)$. As we have seen, we can use the chain rule to decompose this into a series of smaller models.

$$P\left(f_1^J, d_1^M|e_1^I\right) = \prod_{j=1}^{J} P\left(f_j|f_1^{j-1}, d_1^M, e_1^I\right) \times \prod_{m=1}^{M} P\left(d_m|d_1^{m-1}, e_1^I\right). \tag{7}$$

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. Loosely speaking, we can say that there is a probability associated with every arrow in the figures in Section 2. Using our notation for conditional independence, we can now rewrite Equation (7).

$$P(f_1^J, d_1^M | e_1^I) = \prod_{j=1}^{J} P_\delta(f_j | f_1^{j-1}, d_1^M, e_1^I) \times \prod_{m=1}^{M} P_\delta(d_m | d_1^{m-1}, e_1^I). \quad (8)$$

Assuming that we have made strong enough independence assumptions, each distribution on the right side of this equation contains a sufficiently small number of terms 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* of our model. 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 already saw a parameter in the section on language modeling—$p(e_i|e_{i-n}^{i-1})$ was a parameter.

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 Section 2. However, we will use a slightly simplified version of IBM Model 4 as an example to illustrate parameterization of generative models. Recall the steps of this model.

(1) Each target word e_i selects a fertility ϕ_i and copies itself ϕ_i times.

(2) Each copy of each target word is translated to a single source word.

(3) The source words are reordered into their final positions.

Let us now take a look a parameterization of this model. We know that we want to assign a probability to each step taken by the model. Therefore, we will need a *fertility probability*, a *word translation probability*, and some probability to control the reordering, which we call a *distortion probability*.

Fertility is simple. We can make it conditionally dependent on the word identity. We define the fertility probability for word e_i by the parameter $p(\phi_i|e_i)$. We can think of this as the probability that a human translator would choose ϕ_i source words to translate the target word e_i.

Word translation is equally simple if we limit ourselves to the identities of the words being translated. Let $\tau_{i,k} \in V_F$ be the translation of the kth copy of e_i. The word translation probability is then $p(\tau_{i,k}|e_i)$. This is quite intuitive. We can think of it as the probability that a translator, when presented with word e_i, will choose to translate it using word $\tau_{i,k}$.

Let T denote the set of all random variables $T = \{\tau_{i,k} : 0 \leq i \leq I, 0 \leq k \leq \phi_i\}$ representing target word translations.

Modeling the reordering step is a bit more complicated. We will use two parameters to control this. The first parameter controls the placement of the *first* word generated by e_i, $\tau_{i,1}$. Let random variable $\pi_{i,1} \in [1, J]$ denote its final position. IBM Model 4 models this according to the distribution $p(\pi_{i,1} - \odot_{i-1}|C_E(e_{i-1}), C_F(\tau_{i,1}))$. Here, $\pi_{i,1} \in [1, M]$ represents the final absolute location of the translated word and $\odot_{i-1} = \frac{1}{\phi_{i-1}} \lceil \sum_{k=1}^{\phi_{i-1}} \pi_{i-1,k} \rceil$ represents the average location of all translations of e_{i-1}. In other words, we make its

position dependent on the positions of the previously translated word. The functions $C_E : E \rightarrow [1, K]$ and $C_F : F \rightarrow [1, K]$ partition the vocabularies V_E and V_F onto suitably small sets of K classes to avoid the sparseness that would arise from conditioning on the words themselves [Brown et al. 1992; Och 1999].

Just as we condition the positioning of the first translated word $\tau_{i,1}$ on the position of the translations of the adjacent target word, we condition the positioning of $\tau_{i,k}$ on $\pi_{i,k-1}$. Let this be controlled by the distribution $p(\pi_{i,k} - \pi_{i,k-1}|C(\tau_{i,k}))$. We use Π to denote the set of all random variables $\Pi = \{\pi_{i,k} : 0 \leq i \leq I, 0 \leq k \leq \phi_i\}$ representing word positions.

We can now show the complete parameterization of the IBM Model 4.[11]

$$
\begin{aligned}
P(\mathbf{f}|\mathbf{e}) = \sum_{T,\Pi} & \prod_{i=0}^{I} p(\phi_i|e_i)\times && \text{fertility} \\
& \prod_{i=0}^{I}\prod_{k=1}^{\phi_i} p(\tau_{i,k}|e_i)\times && \text{translation} \\
& \prod_{i=0}^{I} p(\pi_{i,1} - \odot_{i-1}|C_E(e_{i-1}), C_F(\tau_{i,1}))\times && \text{distortion for } \tau_{i,1} \\
& \prod_{i=0}^{I}\prod_{k=2}^{\phi_i} p(\pi_{i,k} - \pi_{i,k-1}|C(\tau_{i,k}))\times && \text{distortion for } \tau_{i,k}, k > 1
\end{aligned}
$$

As we can see from this discussion, the parameterization of generative models is closely tied to the form of the underlying translational equivalence model. However, many of these parameters have obvious analogues in other models. For instance, phrase-based models are parameterized using *phrase translation probabilities*, which apply to pairs of phrases, and are analogous to word translation probabilities [Marcu and Wong 2002; Koehn et al. 2003]. Numerous other parameterizations of distortion have been proposed for FST models [Al-Onaizan and Papineni 2006; Vogel et al. 1996; Och and Ney 2000; Toutanova et al. 2002; Lopez and Resnik 2005; DeNero and Klein 2007]. The parameterization of SCFG models follows a similar pattern of diversity.

3.2. Discriminative Models

Generative models are useful in decoding (Section 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 popular form for this is *log-linear* modeling [Berger et al. 1996; Och and Ney 2002].[12] The introduction of log-linear models to SMT follows from

[11] For simplicity, we leave out the parameterization of null translation.

[12] Without the normalization term in Equation (9), these are simply linear models.

their increasing use in NLP [Berger et al. 1996; Ratnaparkhi 1998; Smith 2006], 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 2004b]; 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 (9).

$$P(\mathbf{e}, \mathbf{d}|\mathbf{f}) = \frac{\exp \sum_{k=1}^{K} \lambda_k h_k(\mathbf{e}, \mathbf{d}, \mathbf{f})}{\sum_{\mathbf{e}', \mathbf{d}' : Y(\mathbf{e}', \mathbf{d}')} \exp \sum_{k=1}^{K} \lambda_k h_k(\mathbf{e}', \mathbf{d}', \mathbf{f})}. \tag{9}$$

The daunting normalization factor in the denominator is required only to make the function a well-formed probability. Fortunately, we can ignore it during decoding because it constant for any given f_1^J. Its computation may or may not be required during parameter estimation, depending on the algorithm.

The log-linear model defined in Equation (9) has K parameters, λ_1^K.[13] These are called *feature weights* or *model scaling factors*. They determine the contribution of a feature to the overall value of $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$. Ideally, each parameter would indicate the pairwise correspondence between the feature and the output probability. A positive value λ_k should indicate that the feature $h_1^K(\mathbf{e}, \mathbf{d}, \mathbf{f})$ correlates with $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$; a negative value should indicate an inverse correlation; and a value near zero should indicate that the feature is not a useful predictor of $P(\mathbf{e}, \mathbf{d}|\mathbf{f})$. However, in practice, if two features are highly correlated with each other, an estimator might distribute the weight in any fashion between them. If a feature is uncorrelated with the output, then the estimator might assign an arbitary weight to it. This complicates parameter estimation and motivates the task of *feature selection*, which seeks to identify the smallest set of the most predictive features. Feature selection is a very open problem in SMT.

Notice that Equation (4) corresponds to a special case of Equation (9) when the following conditions hold.

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

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 (9) 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

[13] Confusingly, in the general literature this is sometimes called a *parameter-free* model. It is also known as a *distribution-free* model, which can be understood from the fact that the normalization is not strictly required.

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.[14] A popular technique is to simply take the summed logarithm of all instances of a particular distribution in an underlying generative model. This yields a small number of features, usually less than ten. Another important feature of most models is the *word count* or *word penalty* feature, which is the number of words in the target sentence. Its weight therefore controls target sentence length.[15]

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 some exception this area has not been fully explored [Och et al. 2004a, 2004b; Marcu et al. 2006; Liang et al. 2006; Venugopal et al. 2007]. Many models use little more than a word penalty feature small set of generative features that can be traced directly to the IBM Models.

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 method for iteratively estimating both is presented by Fraser and Marcu [2006].

We describe parameter estimation for generative models in Section 4.1. We will then discuss the important concept of word alignment in Section 4.2. Finally, we describe parameter estimation for log-linear models in Section 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." To estimate this parameter, we turn to our parallel corpus.

One way to capitalize on our 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.

---

[14] 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 et al. 1999; Och and Ney 2002]. Beginning with [Koehn 2004a], many systems use both [Simard et al. 2005; Marcu et al. 2006; Chiang 2007]. It is not clear if this is beneficial [Lopez and Resnik 2006].

[15] Lopez and Resnik [2006] found this feature to be surprisingly important. This may be due to the use of evaluation metrics that are very sensitive to length.

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 (10).

$$\Theta = \underset{\hat{\Theta}}{\operatorname{argmax}}\, P_{\hat{\Theta}}(C). \tag{10}$$

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 (10) is the *maximum likelihood estimate*. This is not the only possible objective function, but it is the one that is typically used for generative models. We will discuss a different objective function in Section 4.3.1.

4.1.1. *Learning Word Translation Probabilities.* Recall that in our generative models, each probability is tied to a single decision taken by the model. MLE is easy when we can observe all of these decisions. 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 do not 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. The accuracy of MLE depends crucially on the number of examples—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 far from the true value of the parameter. 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.[16,17]

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 count the fertility, substitution, and distortion outcomes 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(风,wind)) times, then we would compute p(风|wind) = #(a(风,wind))/#(wind).

It is not that easy, however. The data that we collect from the real world contains only sentence pairs, not alignments.[18] So, while we can see that "wind" and "风" both *cooccur* in many of these sentence pairs, we cannot see how many times they are actually aligned to each other. 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. How can we estimate the probability for words that *do* cooccur?

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

[16] A discussion of smoothing is highly relevant, as it is to most NLP problems, but well beyond the scope of this paper. Chen and Goodman [1998], Manning and Schütze [1999], and Jurafsky and Martin [2008] and are good starting points.

[17] In fact, the unsmoothed values used by most phrase-based models are so imprecise that they can be stored in four bits without loss of performance [Och 2005; Federico and Bertoldi 2006].

[18] In fact, we have even glossed over the conversion of raw text data to sentence pairs, which is not an entirely trivial problem [Gale and Church 1993; Smith 2002].

describe word alignment methods in Section 4.2. Alternatively, we need a method to estimate our parameters that will work even when we cannot explicitly count all of the decisions that we are interested in. 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*. A method that is commonly used to solve this problem in SMT is the Expectation-Maximization (EM) algorithm [Dempster et al. 1977]. It 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 have some initial value for Θ, which we will call Θ_0. Let Θ_0 be random, uniform, or initialized from some simpler model. We can then compute the expected count of the event $E(a(风,\text{wind}))$ in any particular sentence pair $(\mathbf{e}, \mathbf{f})$ as follows.

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

In other words, we compute all possible alignments between f_1^J and e_1^I, and their probabilities under Θ_0. We then sum the probability of those alignments that contain the decision that we are interested in, and divide this by the summed probability of all possible alignments. This gives a fractional expected probability that the event occurred. If we apply this method to all parameters over the entire corpus, we can produce a new estimate, Θ_1. Under certain conditions this offers the minimal guarantee that $P_{\Theta_1}(C) > P_{\Theta_0}(C)$. Thus, it works by *hill-climbing*, or constantly improving its estimate of Θ. We can define EM according to the following recursion, for our observed training data C and unknown alignment data D.

$$\Theta_i = \underset{\hat{\Theta}}{\text{argmax}}\, P_{\hat{\Theta}}(C, E_{\Theta_{i-1}}(D)). \tag{11}$$

The EM algorithm does not, in general—and in particular for most SMT models—guarantee convergence to a globally optimal value for Θ. In fact, it depends crucially on a good initial estimate Θ_0 to avoid a poor local maximum. A method for generating this estimate for the IBM Models is sketched in Section 4.2. Despite the various difficulties in using EM, it has been applied to a variety of other NLP problems [Lari and Young 1990; Merialdo 1994].

A full discussion of the EM algorithm is beyond the scope of this article. 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*. EM for phrase-based models involves many approximations and tradeoffs [Marcu and Wong 2002; DeNero et al. 2006; Birch et al. 2006]. A pragmatic solution is to generate a word alignment (Section 4.2), and then count all phrases that are consistent with the alignment [Och et al. 1999; Koehn et al. 2003]. We 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—in other words, the phrase contains the transitive closure of some set of nodes in the bipartite alignment graph. This is illustrated in Figure 12.

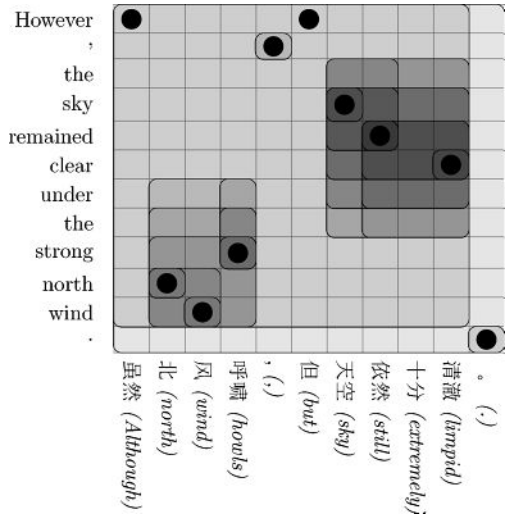

Fig. 12. Supervised learning of phrases from word alignments. Here, we view each sentence pair on a grid. Word alignment is indicated by the presence of dark circles in the grid point corresponding to the word pair. The rectangles outline bilingual phrase pairs that are consistent with this word alignment.

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 (Section 4.2) and a supervised learning method. Melamed [2004b] 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

We need a method for word alignment as a precursor to most of our learning methods in generative models. Word alignment is a microcosm of translation: we need a model, parameter estimation, and a search algorithm. 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 [Yarowsky et al. 2001; Hwa et al. 2005; Smith and Smith 2004]. 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 Section 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

$A \subset [1, I] \times [1, J]$. 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 (Section 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 $\mathbf{a}$ 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 (12), which tells us how to find the optimal alignment $\mathbf{a}$.

$$\mathbf{a} = \underset{a_1^J}{\operatorname{argmax}} \prod_{j=1}^{J} p(a_j) \cdot p\left(f_j \middle| e_{a_j}\right). \tag{12}$$

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 $p(f_j|e_i)$. Notice that our independence assumptions reduce the model to a set of J independent decisions each 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:

$$P(\mathbf{f}|\mathbf{e}) = \prod_{j=1}^{J} \sum_{a_j=0}^{I} p(a_j) \cdot p\left(f_j \middle| e_{a_j}\right). \tag{13}$$

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 source and target words [Och et al. 2004b; Lopez and Resnik 2006].

We obtain better alignments when we move to a first-order dependency between the alignment variables, as in Equation (14) [Vogel et al. 1996].

$$\mathbf{a} = \underset{a_1^J}{\operatorname{argmax}} \prod_{j=1}^{J} p(a_j|a_{j-1}) \cdot p\left(f_j \middle| e_{a_j}\right). \tag{14}$$

Equation (14) is in the basic form of a 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 that standard algorithms for EM parameter estimation [Baum 1972] and maximization [Viterbi 1967] are widely known. HMMs have been the subject of several studies in word alignment [Och and Ney 2000; Toutanova et al. 2002; Lopez and Resnik 2005; DeNero and Klein 2007]. In general, they are very accurate, significantly outperforming IBM Models 1, 2, and 3 in detailed empirical studies [Och and Ney 2000, 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 [Och and Ney 2003; Lopez and Resnik 2005]. 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.

The IBM Models for alignment are implemented in the open-source toolkit GIZA and its successor GIZA++ [Al-Onaizan et al. 1999; Och and Ney 2003].[19] They are widely used in the SMT research community for a variety of purposes, including parameter learning for other models [Och et al. 1999; Yarowsky and Ngai 2001; Koehn et al. 2003; Smith and Smith 2004]. Various improvements to Model 4 have been proposed [Dejean et al. 2003; Fraser and Marcu 2006, 2007a].

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 models are symmetric. We would like symmetric 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 symmetrize by combining these two alignments. This is done via set union, set intersection, or a number of heuristic methods, which usually begin with the intersection and proceed by iteratively adding links from the union [Och et al. 1999; Koehn et al. 2003]. Matusov et al. [2004] present a symmetric word alignment method based on linear combination of complementary asymmetric word alignment probabilities. Ayan and Dorr [2006a] investigate the effect of various symmetrization heuristics on the performance of phrase-based translation.

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 et al. 2005a, 2005b; Ayan and Dorr 2006b; Liang et al. 2006]. 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

[19] GIZA++ is available from http://www.fjoch.com/GIZA++.html.

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 [Gale and Church 1991; Melamed 2000; Cherry and Lin 2003; Moore 2005a].

Fraser and Marcu [2007a] extended the IBM Models to produce symmetric many-to-many alignments that can be viewed as phrase alignments.

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. 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 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.

Numerous discriminative alignment models have been proposed, based on a wide variety of machine learning methods. Most of these methods depend on supervised training, and depend on the availability of a small set of manually produced alignments. Learning methods include transformation-based learning [Ayan et al. 2005a], neural networks [Ayan et al. 2005b], maximum margin estimation [Taskar et al. 2005], perceptron learning [Moore 2005b], and log-linear models [Ittycheriah and Roukos 2005; Fraser and Marcu 2006]. When annotated alignments are available, these methods outperform unsupervised methods according to common alignment metrics, and sometimes in downstream translation results.

4.2.5. *Evaluation of Word Alignment.* The ultimate measure of word alignment is in its contribution to parameter estimation of our translation models. If one alignment method produces a better translation system than another, we might conclude that it is more accurate overall.

Word alignment is used for tasks other than SMT parameter estimation, so other task-based evaluations might be applicable. Although this is preferable, it is common to evaluate word alignment intrinsically, by comparison with alignments prepared by human annotators. Most of these test sets contain a few hundred sentences. They are available in several languages [Melamed 1998; Och and Ney 2000; Mihalcea and Pedersen 2003]. Ideally, each sentence is aligned by multiple annotators and the results are combined in some way. In much of the reported literature, the annotations contain two sets of links. The *sure* set S contains links about which all annotators agreed. The *probable* set P is a superset of S that additionally contains links about which annotators disagreed or expressed uncertainty about, such as "idiomatic expressions, free translations, and missing function words" [Och and Ney 2000]. However, Fraser and Marcu [2007b] found that the use of probable links reduced the ability of alignment metrics to predict translation accuracy. They recommend an annotation style that does not contain them [Melamed 1998].

Given the set of hypothesized alignment links A, we 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.[20] A widely used metric that combines these statistics is the *alignment error rate* (AER) given in Equation 15 [Och and Ney 2000].

$$AER = 1 - \frac{|S \cap A| + |P \cap A|}{|S| + |A|}. \tag{15}$$

A similar metric was proposed by Ahrenberg et al. [2000].

Although intrinsic evaluation of word alignments is popular, the exact relationship between alignment evaluations and SMT performance is not entirely clear. Several studies report poor correlation between alignment performance and MT performance [Koehn et al. 2003; Callison-Burch et al. 2004; Ittycheriah and Roukos 2005], and a number of researchers have investigated the relationship directly [Ayan and Dorr 2006a; Fraser and Marcu 2007b; Lopez and Resnik 2006]. In particular, Fraser and Marcu [2007b] advocate *unbalanced* F-measure as a better predictor of SMT performance than AER. The F-measure is given in Equation (16).

$$F = \frac{|S \cap A|}{\alpha|S| + (1-\alpha)|A|}. \tag{16}$$

The parameter α is used to move balance towards either precision or recall. Fraser and Marcu [2007b] show that α can be tuned to more reliably predict translation accuracy.

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 (Section 3.2). This is usually done on a training set separate from the one used to learn underlying generative model probabilities.

As in generative models, the maximum likelihood objective (Equation (10)) can be used to train the feature weights. A nice property of log-linear models is the availability of a convenient objective function obtained via the *maximum entropy principle* [Berger et al. 1996].[21] It corresponds to the maximum likelihood objective function and has a single optimum point, which we can find using an iterative search method called *generalized iterative scaling* [Darroch and Ratcliff 1972]. 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. Unfortunately, the normalization factor represented by the denominator of Equation (9) 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. Och and Ney [2002] show that the N-best output of the previous parameter setting can be used to approximate this sum.

4.3.1. *Minimum Error-Rate Training.* Automatic evaluation metrics for MT have become widespread (Section 6). They facilitate a different method of parameter estimation: *minimum error-rate training* (MERT) [Och 2003]. In MERT, we assume that the best

[20]Precision and recall metrics are common in NLP evaluation.

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

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 simply by devising an error function that more closely corresponds to human judgments 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:[22]

$$\lambda_1^K = \underset{\lambda_1^K}{\operatorname{argmin}} \sum_{(\mathbf{e},\mathbf{f}) \in C} E(\underset{\hat{\mathbf{e}}}{\operatorname{argmax}} P_{\lambda_1^K}(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e}). \tag{17}$$

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 solution using the method sketched in Och [2003], which we describe in greater detail here due to its widespread use. Pseudocode appears 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 $\mathbf{e}$ and $\mathbf{f}$ is $P(\mathbf{e}|\mathbf{f}) = \lambda_k h_k(\mathbf{e}, \mathbf{f}) + (\sum_{k' \neq k} \lambda_{k',m} h_{k'}(\mathbf{e}, \mathbf{f}))$. 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 $E(\operatorname{argmax}_{\hat{\mathbf{e}}} P(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e}))$. 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.[23] We then select the midpoint in the interval that minimizes the function.

4.3.2. *Purely Discriminative Training.* Most current state-of-the-art SMT systems use log-linear models with a small number of generative submodels and use MERT in order to optimize whatever error function is chosen for evaluation. An overview of 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,

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

[23] Sometimes this function is not additive, as is the case with the commonly used BLEU score [Papineni et al. 2002]. 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.

features are usually binary or integral. For instance, we might define a word pair feature $h(e, f)$ as follows:

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

Algorithm 1. Minimum Error Rate Training

Input initial estimate $\lambda_{1,0}^K$ ▷Uniform or random
Input training corpus C
$\lambda_1^K = \lambda_{1,0}^K$
$E_{best} = \sum_{(\mathbf{e},\mathbf{f})\in C} E(\text{argmax}_{\hat{\mathbf{e}}}\, P_{\lambda_1^K}(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$ ▷Decode
repeat
  Generate M random estimates $\lambda_{1,1}^K, ..., \lambda_{1,M}^K$ ▷To avoid poor local maximum
  for $m = \{0, 1, ..., M\}$ **do**
    for $k = \{1, 2, ..., K\}$ **do**
      $\lambda'_{k,m} = \text{LINE-MINIMIZE}(k, \lambda_{1,m}^K, C)$
      $E_{k,m} = \sum_{(\mathbf{e},\mathbf{f})\in C} E(\text{argmax}_{\hat{\mathbf{e}}}\, P_{\lambda_{1,m}^{k-1}\lambda'_{k,m}\lambda_{k+1,m}^K}(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$ ▷N-best list
      if $E_{k,m} < E_{best}$ **then**
        $\lambda_1^K = \lambda_{1,m}^{k-1}\lambda'_{k,m}\lambda_{k+1,m}^K$
        $E_{best} = E_{k,m}$
      end if
    end for
  end for
  $\lambda_{1,0}^K = \lambda_1^K$
  $E_{best} = \sum_{(\mathbf{e},\mathbf{f})\in C} E(\text{argmax}_{\hat{\mathbf{e}}}\, P_{\lambda_1^K}(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$ ▷Decode
until no change in λ_1^K
return $\lambda_{1,0}^K$

function LINE-MINIMIZE(k, λ_1^K, C)
  $E_{\lambda_k}(C) = 0$
  for all $(\mathbf{e}, \mathbf{f}) \in C$ **do**
    for all $\hat{\mathbf{e}} \in$ DECODER-N-BEST($\mathbf{f}$) **do**
      $m_{\hat{\mathbf{e}}} = h_k(\hat{\mathbf{e}}, \mathbf{f})$ ▷slope of $P_{\lambda_k}(\hat{\mathbf{e}}, \mathbf{f})$
      $b_{\hat{\mathbf{e}}} = \sum_{k'=1}^{k-1} \lambda_{k'} \cdot h_{k'}(\hat{\mathbf{e}}, \mathbf{f}) + \sum_{k'=k+1}^{K} \lambda_{k'} \cdot h_{k'}(\hat{\mathbf{e}}, \mathbf{f})$ ▷intercept of $P_{\lambda_k}(\hat{\mathbf{e}}, \mathbf{f})$
    end for
    $i = 0$
    $\Delta[i] = -\infty$ ▷left interval boundary
    $e[i] = \text{argmin}_{\hat{e}}\, m_{\hat{\mathbf{e}}}$ ▷equivalent to $\text{argmax}_{\hat{\mathbf{e}}} \lim_{\lambda_k \to -\infty} P(\hat{\mathbf{e}}, \mathbf{f})$
    repeat
      $i = i + 1$
      $\Delta[i] = \min_{\hat{\mathbf{e}}}$ X-INTERSECT$(m_{\hat{e}}, m_{e[i-1]}, b_{\hat{e}}, b_{e[i-1]}) > \Delta[i-1]$
      $e[i] = \text{argmin}_{\hat{\mathbf{e}}}$ X-INTERSECT$(m_{\hat{e}}, m_{e[i-1]}, b_{\hat{e}}, b_{e[i-1]}) > \Delta[i-1]$
    until No more intersection points found
    $\Delta_{i+1} = \infty$
    $E_{\lambda_k}(\text{argmax}_{\hat{\mathbf{e}}}\, P(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e}) = \{\lambda_k \to E(\hat{\mathbf{e}}, \mathbf{e}) : \hat{\mathbf{e}} = e[i], \Delta[i] \le \lambda_k \le \Delta_{i+1}\}$
    $E_{\lambda_k}(C) += E_{\lambda_k}(\text{argmax}_{\hat{\mathbf{e}}}\, P(\hat{\mathbf{e}}|\mathbf{f}), \mathbf{e})$
  end for
  return $\lambda_k = \text{argmin}\, E_{\lambda_k}(C)$
end function

Under this defintion, the weight given to this feature by the combined generative and discriminative training procedure outlined above is $\lambda \log p(f|e)$. However, as we have

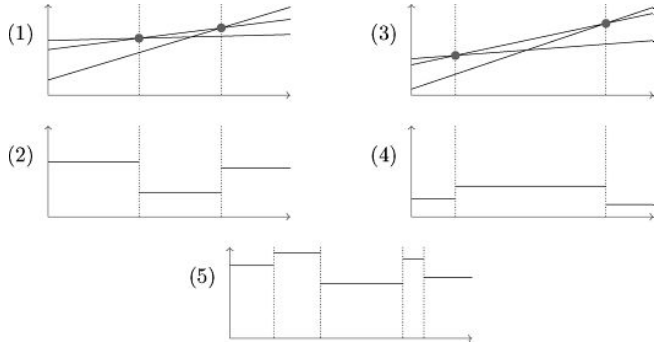

Fig. 13. Illustration of the MERT line minimization function for optimizing a single parameter λ_k. In (1) we compute $P_{\lambda_k}(\hat{\mathbf{e}}|\mathbf{f})$ as a line in λ_k for each candidate translation $\hat{\mathbf{e}}$ of a single source sentence $\mathbf{f}$. We then find the intervals at 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.

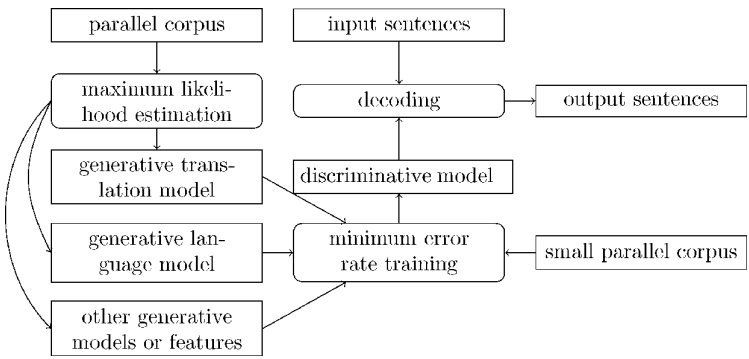

Fig. 14. Putting it all together: the flow of data, models, and processes commonly involved in the deployment of an SMT system.

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. Examples include the perceptron algorithm [Liang et al. 2006], large margin learning [Tillmann and Zhang 2006; Watanabe et al. 2007], decision tree learning [Wellington et al. 2006], and transductive learning [Ueffing et al. 2007]. 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 are quite slow compared to generative model estimation and MERT.

	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)	1	2	3	4	5	6	7	8	9	10	11

(2)

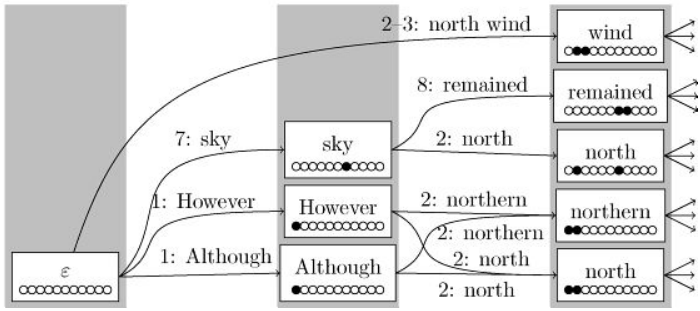

Fig. 15. Illustration of search in a finite-state decoder. The input sentence (1) generates a large search graph, partially illustrated in (2). In this illustration, each arrow represents extension of a hypothesis by appending the words on the arrow. To recover the best translation, we traverse the highest scoring path. In each state, we show the coverage set and most recently generated target word, which is needed for computation of a bigram language model. Notice that states can only be combined if the coverage set and the most recently produced words match. Items with the same number of covered words are stored in the same stack.

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 (18).

$$\mathbf{e} = \underset{(\hat{\mathbf{e}}:Y(\hat{\mathbf{e}},\mathbf{d}))}{\operatorname{argmax}} P(\hat{\mathbf{e}}, \mathbf{d}|\mathbf{f}). \tag{18}$$

We call this the *decision rule*. Equation (18) 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, Zollmann, and Waibel [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 by Wang and Waibel [1997] and Koehn [2004a]. It is a generalization of speech recognition algorithms dating back to Jelinek [1969].

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 $C \subseteq \{1, 2, \ldots, J\}$ enumerates the positions of the source string f_1^J that have been translated.

(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. We implement model-specific extension operators to apply this algorithm to IBM Model 4 [Tillman and Ney 2003; Germann et al. 2004], phrase-based models [Koehn 2004a], or any number of other finite-state translation models [Wang and Waibel 1997; Nießen et al. 1998].

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].[24] 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 full 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 [Och et al. 2001; Koehn 2004a]. 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 [Ueffing et al. 2002; Koehn 2004a]. 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 Section 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.

A variant algorithm allows the target language hypothesis to be extended to either left or right [Watanabe and Sumita 2002].

[24] 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. The terminology dates back to Jelinek [1969].

5.1.1. *Optimality and Pruning.* Using A^* heuristics, we can solve the optimization in Equation (18) 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. Due to the large number of translations for each word or phrase, even with limited reordering this can be very slow. Fortunately, optimal search is not strictly necessary, because there are many good translations of any sentence. If many of these receive high probability under our model, then we may safely permit a certain amount of *search error*. Search error occurs when the decoder does not choose the globally highest-scoring hypothesis according to the model, but rather some other high-scoring hypothesis that can be found more quickly. We can optimize for speed by *pruning* the search graph [Tillman and Ney 2003; Koehn 2004a]. 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 pruning is sometimes called *beam search*, where t or n is the size of the *beam*. With a well-tuned beam size, we gain large speedups with very little loss of accuracy [Tillman and Ney 2003; Zens and Ney 2004; Koehn 2004a; Germann et al. 2004].

5.1.2. *Greedy Decoding.* An alternative to standard finite-state decoding is greedy decoding [Marcu and Wong 2002; Germann 2003; Germann et al. 2004]. 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, it algorithm runs in time nearly linear in target sentence length [Germann 2003]. A tradeoff of is that the search error rate is much higher than stack decoding [Germann et al. 2004].

5.2. SCFG Decoding

Decoding with SCFG models is equivalent to CFG parsing [Melamed 2004b]. The goal is to infer the highest-scoring tree that generates the input sentence using the source side of the grammar, and then read off the tree in target order. Most practical syntax-based decoders are straightforward extensions of dynamic programming algorithms for parsing monolingual context-free grammars [Wu and Wong 1998; Yamada and Knight 2002; Zens and Ney 2003; Chiang 2007; Marcu et al. 2006; Venugopal et al. 2007]. A benefit of this is that the standard algorithms and optimizations that have been developed for CFG parsing can be applied to SMT [Melamed 2004b].

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 [Melamed 2004a; Chiang 2007; Zhang et al. 2006]. In order to construct larger spans, we find SCFG productions whose right-hand sides match a 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, and they can be computed in polynomial time. It is easy to see from this that SCFG decoding is, *in principle*, much less computationally expensive than FST decoding with full reordering. However, most

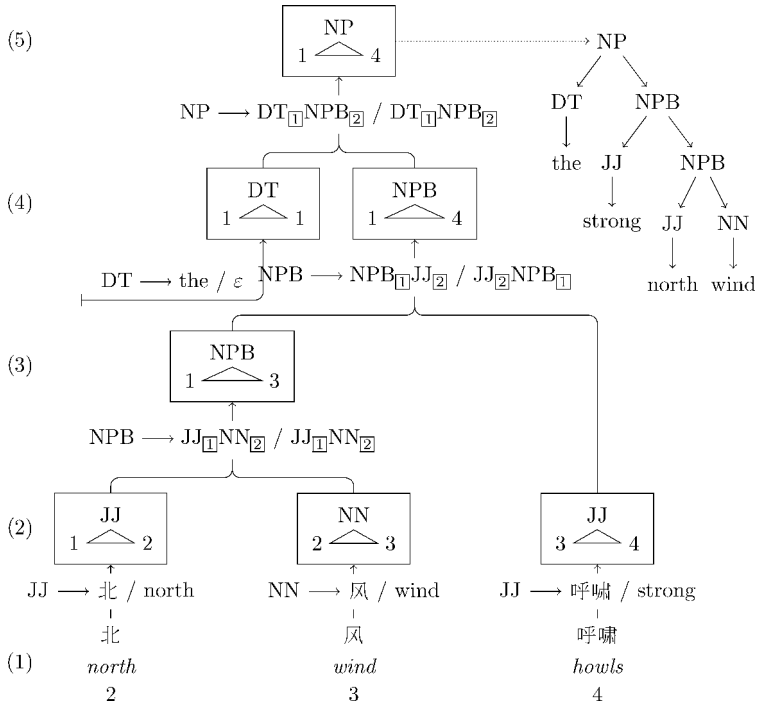

Fig. 16. An illustration of SCFG decoding, which is equivalent to parsing the source language. (1) Scan each source word and associate it with a span. (2) Apply SCFG rules that match the target spans. (3) Recursively infer larger spans from smaller spans. (4) We can also infer target language words with 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.

practical FST decoders allow only a limited amount of reordering, and in practice they are often much faster than SCFG decoders.[25] Search optimization for these models is therefore an active area of research.

Chiang [2007] describes a optimization called *cube pruning* that prevents excessive combination of hypotheses in adjacent subspans. Zhang et al. [2006] describe a method for *binarizing* rules containing more than two nonterminals, which helps reduce grammar constants for parsing and simplifies n-gram language model integration. Venugopal et al. [2007] present a 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. A number of other optimizations have also been investigated [Huang and Chiang 2005; Huang and Chiang 2007].

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

[25]It is possible to apply reordering constraints of FST models to SCFG models. Chiang [2005, 2007] restricts hierarchical reordering to spans that are shorter than ten words. Spans longer than this are required to be monotonic orderings of smaller hierarchical phrases. This prevents some long-distance reordering.

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 returning the highest-scoring translation, we return 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. [2004b] and Shen et al. [2004]. Och et al. [2004b] show using oracle studies on decoder N-best lists that large gains in accuracy are possible with rescoring, although so far these are unrealized.

5.4. Data Structures for Model Representation

In addition to efficient algorithms for decoding, we also need efficient data structures and strategies to store a model with millions of parameters.

A common data structure in phrase-based models is the *phrase table*. This structure enumerates all of the phrase pairs in the model and maps them to their phrase translation probabilities. In practice, these phrase pairs may be extracted from parallel corpora containing tens or hundreds of millions of words. Even the most conservative extraction heuristics will produce a huge number of phrases under this condition. Efficient algorithms for storage and access are necessary for practical decoding algorithms. Since the quantity of available parallel data is always increasing, this is particularly important.

The obvious approach of storing phrase pairs and probabilities in a hash table or trie requires making tradeoffs as model size increases. A common practice in research systems is to first filter the table for a specific test set. However, this is impractical for anything but batch systems. A different method of filtering is to identify rules that are statistically improbable and remove them [Johnson et al. 2007]. Another common approach is to restrict the model in some way. In phrase based systems, we can simply restrict the length of the phrases [Koehn et al. 2003; Ayan and Dorr 2006a]. If the phrase table is still large to fit in memory, it can be accessed from disk [Zens and Ney 2007].

A substantially different approach is to do away with the phrase table altogether and simply store the aligned training data itself in memory. In this approach, source phrases are looked up at runtime using fast pattern matching algorithms, and their translations are extracted and scored on the fly [Callison-Burch et al. 2005; Zhang and Vogel 2005; Lopez 2007]. This method has the advantage that the training data is more compact than a phrase table, and we do not need to limit phrase length. Therefore we can scale to quite large corpora and arbitrary phrase lengths. However, finding, extracting, and scoring phrase pairs at translation time can add considerable computational expense.

Language models also require a very large number of parameters. Although many systems rely on standard language model toolkits designed and optimized in the speech recognition domain, there are several alternative strategies. Brants et al. [2007] and Zhang et al. [2006] describe distributed language models for grid computing. Talbot and Osborne [2007a, 2007b] describe an alternative method based on the *Bloom filter*, a randomized data structure that enables set membership queries with some small probability of errors. They exploit this to reduce the memory footprint of a standard language model by an order of magnitude.

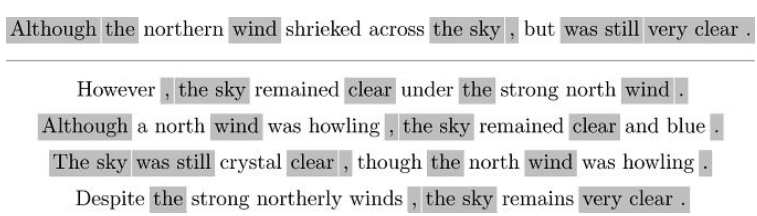

Fig. 17. Example of partial string matching used for most evaluation methods. Here we show a single output hypothesis compared with four reference translations. Sequences of words in the hypothesis that match sequences in any of the reference translations are highlighted. Likewise, sequences of words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

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. [White et al. 1994]. More ideal than this are measures that determine how well some human task can be performed when the human subject is provided with machine-translated text. If possible, we would optimize for task related-metrics directly. Unfortunately, human evaluation requires time and money. This usually rules out its use in 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 that 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 (Section 4.3).

A common element of automatic metrics is their use of a set of test sentences for which we already have human translations, called *reference translations*. They 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 intuition behind metrics based on reference sentences is that MT must be good if it closely resembles human translation of the same sentence [Papineni et al. 2002]. These metrics are based on partial string matching between the output and the 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, we may use multiple references. This requires the use of human translators to produce the additional references, but it is 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].

The most widely used metric is the *bilingual evaluation understudy* (BLEU) [Papineni et al. 2002]. BLEU considers not only single word matches between the output and the reference sentence, but also n-gram matches, up to some maximum n. This allows it to reward sentences where local word order is closer to the local word order in the reference. BLEU 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. Let $\#(g)$ be the count of an n-gram g in a particular hypothesis sentence $\hat{e}$, and $\#_{clip}(g)$ be the minimum number of times that g appears in the corresponding reference sentence. We can compute the n-gram precision p_n for a set of hypothesis translations H.

$$p_n = \frac{\sum_{\hat{e} \in H} \sum_{g \in ngrams(\hat{e})} \#_{clip}(g)}{\sum_{\hat{e} \in H} \sum_{g' \in ngrams(\hat{e})} \#(g')}.$$

To get a better idea of the accuracy, we combine multiple n-gram precisions, up to some maximum n, by taking the geometric average $\sum_n \log p_n$. This biases the metric towards translations with fewer words, because denominator contains the total number of hypothesis n-grams. To correct this defect, the metric includes a *brevity penalty*, which penalizes output sentences that are much shorter than the reference. It compares the overall number of words h of the entire hypothesis set with *effective reference length* r, created by summing the lengths of the closest reference sentences to each candidate sentence. This gives us BLEU.

$$BP = \begin{cases} 1 & \text{if } h > r \\ e^{(1-r/h)} & \text{otherwise} \end{cases}$$

$$BLEU = BP \cdot \exp\left(\sum_n \log p_n\right).$$

Automatic evaluation is an active research area. 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]. The METEOR metric enhances token matching with weighted matching based on morphological or semantic similarity [Banerjee 2005]. Translation edit rate (TER) [Snover et al. 2006] computes an edit distance between hypotheses and human-corrected versions of those hypotheses. The intuition is that it measures "the amount of work needed to correct the translations." It is an fully automatic approximation to human TER (hTER), a task-based metric which measures the amount of work done by actual human post-editors.

It is important to note when interpreting metrics such as BLEU that they can be used to rank systems relative to each other, but the scores are generally uninterpretable as absolute measures of correctness. A key element of most research in this area is the identification of metrics that correlate with human judgement in controlled studies [Papineni et al. 2002; Callison-Burch et al. 2007]. Since this correlation is important, a natural line of research involves the use of machine learning to optimize metrics for correlation [Kulesza and Shieber 2004; Lita et al. 2005; Liu and Gildea 2007; Albrecht and Hwa 2007].

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

BLEU has been highly influential in SMT research. It extensively used SMT literature, and it has been used as the basis for a number of comparative evaluations [Doddington 2002; Koehn and Monz 2005, 2006; Callison-Burch et al. 2007]. It is commonly used in the objective function for minimum error-rate training [Och 2003].

Use of BLEU is controversial. Turian et al. [2003] and Callison-Burch et al. [2006] provide counterexamples to its claimed correlation with human judgement. Other problems have been illustrated by construction [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.

With the proliferation of available metrics, it is not always clear which one to use. Practical considerations such as comparison of with previous benchmarks encourages continued use of BLEU, despite criticism. The use of discriminative training depends on computationally simple metrics, including BLUE, METEOR, and TER. Correlation with human judgement is also a desirable characteristic. For a good contemporary evaluation of metrics across several language pairs, refer to Callism-Burch [2007].

7. CURRENT DIRECTIONS AND FUTURE RESEARCH

There are many common elements in the best systems, although there is also growing diversity. Most can be characterized as follows: phrase-based models (in either the FST or SCFG framework); log-linear models with a small set of generative features; and discriminative training. The success of these methods is seen in academic workshops [Koehn and Monz 2005, 2006; Callison-Burch et al. 2007] and the yearly NIST evaluations.

All of these methods were popularized very quickly after their initial introduction. SMT has made swift progress and there is great optimism for future success. Nonetheless, there are many hurdles and open questions in the field.

Most of the community evaluations in SMT focus 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 underutilized. Since statistical methods are sensitive to both domain differences and noise, the move to informal text and Internet data will present many interesting challenges.

Application of SMT to language pairs with very little parallel text presents an interesting challenge. Bannard and Callison-Burch [2005] and Callison-Burch et al. [2006] describe a novel method for solving this problem by learning paraphrases of the source language using a parallel text in a third language, and applying these paraphrases to generate sentences that can be translated by an impoverished SMT system.

Another understudied problem is the translation of English into other languages. In the United States, research focuses almost exclusively on translation from other languages into English. This is dictated by government funding, but has the effect of obscuring deficiencies in the current approaches. For instance, it is easier to map morphologically rich languages such as German and Arabic onto a relatively morphologically simple language such as English. This can be seen as a movement from a higher-dimensional to a lower dimensional space, where some loss of meaning and nuance is harmless. Translation in the other direction requires much more attention to this issue [Nießen and Ney 2004; Goldwater and McClosky 2005; Schafer and Drabek 2005; Minkov et al. 2007]. Koehn and Hoang [2007] and Koehn et al. [2007] describe *factored models*, a framework for modeling with morphology and other annotations.

Evaluation of MT systems will continue to be a focus, since discriminative training illustrates the importance of metrics that correspond to human judgment. However, as

we have seen, most popular metrics provide very little insight into the typical errors made by any particular system. They are especially useless for identifying sentence-level errors since they provide only an aggregate measure of accuracy. 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 [DeNeefe et al. 2005; Chiang et al. 2005; Popovic et al. 2006] and confidence estimation [Ueffing and Ney 2005] have been investigated, but in general this area remains underexplored.

The fundamental issues in SMT will remain a focus of all future research. Refinements to modeling techniques and parameter estimation methods will no doubt continue. New developments in machine learning will increasingly be applied to machine translation, although additional work is needed to scale them up to data sizes commonly used in SMT. There is also increasing interest in the incorporation of linguistic knowledge into models and parameter estimation. As we described, syntactic modeling is an area of active research. There are also some steps toward semantic modeling [Carpuat and Wu 2005, 2007; Chan et al. 2007].

Finally, although fully automatic MT is often treated as the objective in many SMT studies, it is not the only objective—perhaps not even the primary objective. Understanding the translation needs of users is 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.

ACKNOWLEDGMENTS

I would like to thank David Chiang, Bonnie Dorr, Chris Dyer, Nitin Madnani, Doug Oard, Austin Parker, Michael Subotin, and especially the anonymous reviewers for very helpful and detailed comments. This paper has benefitted immensely from illuminating discussions with Necip Fazil Ayan, David Chiang, Chris Dyer, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. Any errors are entirely my own.

REFERENCES

Aho, A. V. and Ullman, J. D. 1969. Syntax directed translations and the pushdown assembler. *J. Comput. Syst. Sci. 3*, 37–57.

Ahrenberg, L., Merkel, M., Hein, A. S., and Tiedmann, J. 2000. Evaluation of word alignment systems. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*. Vol. 3. 1255–1261.

Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, D., Och, F. J., Purdy, D., Smith, N. A., and Yarowsky, D. 1999. Statistical machine translation: Tech. rep., Center for Speech and Language Processing, Johns Hopkins University.

Al-Onaizan, Y. and Papineni, K. 2006. Distortion models for statistical machine translation. In *Proceedings of ACL-COLING*. 529–536.

Albrecht, J. and Hwa, R. 2007. Regression for sentence-level MT evaluation with pseudo references. In *Proceedings of the Association for Computational Linguistics (ACL)*. 296–303.

Alshawi, H., Bangalore, S., and Douglas, S. 2000. Learning dependency translation models as collections of finite state head transducers. *Computat. Linguist. 26*, 1, 45–60.

Ayan, N. F. and Dorr, B. 2006a. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proceedings of ACL-COLING*. 9–16.

Ayan, N. F., Dorr, B., and Monz, C. 2005a. Alignment link projection using transformation-based learning. In *Proceedings of HLT-EMNLP*. 185–192.

Ayan, N. F., Dorr, B., and Monz, C. 2005b. Neuralign: Combining word alignments using neural networks. In *Proceedings of HLT-EMNLP*. 65–72.

AYAN, N. F. AND DORR, B. J. 2006b. A maximum entropy approach to combining word alignments. In *Proceedings of HLT-NAACL*. 96–103.

BANCHS, R. E., CREGO, J. M., DE GISPERT, A., LAMBERT, P., AND MARIÑO, J. B. 2005. Statistical machine translation of euparl data by using bilingual n-grams. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*. 133–136.

BANNARD, C. AND CALLISON-BURCH, C. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the Association for Computational Linguistics (ACL)*. 597–604.

BAUM, L. E. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. In *Proceedings of the 3rd Symposium on Inequalities*. Inequalities, vol. 3. Academic Press, 1–8.

BERGER, A. L., BROWN, P. F., PIETRA, S. A. D., PIETRA, V. J. D., GILLETT, J. R., LAFFERTY, J. D., MERCER, R. L., PRINTZ, H., AND UREŠ, L. 1994. The Candide system for machine translation. In *Proceedings of the ARPA Workshop on Human Language Technology*. 157–162.

BERGER, A. L., BROWN, P. F., PIETRA, S. A. D., PIETRA, V. J. D., KEHLER, A. S., AND MERCER, R. L. 1996. Language translation apparatus and method using context-based translation models. United States Patent 5510981.

BERGER, A. L., PIETRA, S. A. D., AND PIETRA, V. J. D. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist. 22*, 1, 39–71.

BIRCH, A., CALLISON-BURCH, C., OSBORNE, M., AND KOEHN, P. 2006. Constraining the phrase-based, joint probability statistical translation model. In *Proceedings of HLT-NAACL Workshop on Statistical Machine Translation*. 154–157.

BLUNSOM, P. AND COHN, T. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of ACL-COLING*. 65–72.

BRANTS, T., POPAT, A. C., XU, P., OCH, F. J., AND DEAN, J. 2007. Large language models in machine translation. In *Proceedings of EMNLP-CoNLL*. 858–867.

BROWN, P. F., COCKE, J., PIETRA, S. D., PIETRA, V. J. D., JELINEK, F., LAFFERTY, J. D., MERCER, R. L., AND ROOSSIN, P. S. 1990. A statistical approach to machine translation. *Comput. Linguist. 16*, 2, 79–85.

BROWN, P. F., DESOUZA, P. V., MERCER, R. L., PIETRA, V. J. D., AND LAI, J. C. 1992. Class-based *n*-gram models of natural language. *Comput. Linguist. 18*, 4, 467–479.

BROWN, P. F., PIETRA, S. A. D., PIETRA, V. J. D., AND MERCER, R. L. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist. 19*, 2, 263–311.

BURBANK, A., CARPUAT, M., CLARK, S., DREYER, M., FOX, P., GROVES, D., HALL, K., HEARNE, M., MELAMED, I. D., SHEN, Y., WAY, A., WELLINGTON, B., AND WU, D. 2005. Final report of the 2005 language engineering workshop on statistical machine translation by parsing. Tech. rep., Johns Hopkins University Center for Speech and Language Processing.

CALLISON-BURCH, C., BANNARD, C., AND SCHROEDER, J. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the Association for Computational Linguistics (ACL)*. 255–262.

CALLISON-BURCH, C., FORDYCE, C., KOEHN, P., MONZ, C., AND SCHROEDER, J. 2007. (Meta-) evaluation of machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*. 136–158.

CALLISON-BURCH, C., KOEHN, P., AND OSBORNE, M. 2006. Improved statistical machine translation using paraphrases. In *Proceedings of HLT-NAACL*.

CALLISON-BURCH, C., OSBORNE, M., AND KOEHN, P. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of European Chapter of the Association for Computational Linguistics (EACL)*. 249–256.

CALLISON-BURCH, C., TALBOT, D., AND OSBORNE, M. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of the Association for Computational Linguistics (ACL)*. 176–183.

CARPUAT, M. AND WU, D. 2005. Word sense disambiguation vs. statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 387–394.

CARPUAT, M. AND WU, D. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 61–72.

CHAN, Y. S., NG, H. T., AND CHIANG, D. 2007. Word sense disambiguation improves statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 33–40.

CHARNIAK, E., KNIGHT, K., AND YAMADA, K. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX*.

CHELBA, C. AND JELINEK, F. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of ACL-COLING*. 225–231.

CHEN, S. F. AND GOODMAN, J. 1998. An empirical study of smoothing techniques for language modeling. Tech. rep. TR-10-98, Computer Science Group, Harvard University.

CHERRY, C. AND LIN, D. 2003. A probability model to improve word alignment. In *Proceedings of the Association for Computational Linguistics (ACL)*.

CHIANG, D. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 263–270.

CHIANG, D. 2006. An introduction to synchronous grammars. *Part of a ACL Tutorial*.

CHIANG, D. 2007. Hierarchical phrase-based translation. *Comput. Linguist. 33*, 2.

CHIANG, D., LOPEZ, A., MADNANI, N., MONZ, C., RESNIK, P., AND SUBOTIN, M. 2005. The Hiero machine translation system: Extensions, evaluation, and analysis. In *Proceedings of HLT-EMNLP*. 779–786.

CHURCH, K. AND HOVY, E. 1993. Good applications for crummy machine translation. *Mach. Transl. 8*, 239–258.

CHURCH, K. AND PATIL, R. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *Comput. Linguist. 8*, 3–4, 139–149.

COLLINS, M., HAJIČ, J., RAMSHAW, L., AND TILLMAN, C. 1999. A statistical parser for Czech. In *Proceedings of the Association for Computational Linguistics (ACL)*. 505–512.

COLLINS, M., KOEHN, P., AND KUCEROVA, I. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 531–540.

DARROCH, J. N. AND RATCLIFF, D. 1972. Generalized iterative scaling for log-linear models. *Annals Math. Statist. 43*, 5, 1470–1480.

DEJEAN, H., GAUSSIER, E., GOUTTE, C., AND YAMADA, K. 2003. Reducing parameter space for word alignment. In *Proceedings of HLT-NAACL Workshop on Building and Using Parallel Texts*. 23–26.

DEMPSTER, A. P., LAIRD, N. M., AND RUBIN, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. 39*, 1, 1–38.

DENEEFE, S., KNIGHT, K., AND CHAN, H. H. 2005. Interactively exploring a machine translation model. In *Proceedings of the Association for Computational Linguistics (ACL)* (Companion Vol.). 97–100.

DENERO, J., GILLICK, D., ZHANG, J., AND KLEIN, D. 2006. Why generative phrase models underperform surface heuristics. In *Proceedings of HLT-NAACL Workshop on Statistical Machine Translation*. 31–38.

DENERO, J. AND KLEIN, D. 2007. Tailoring word alignments to syntactic machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 17–24.

DODDINGTON, G. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Human Language Technology Conference (HLT)*.

DORR, B. J., JORDAN, P. W., AND BENOIT, J. W. 1999. A survey of current paradigms in machine translation. In *Advances in Computers*, M. Zelkowitz, Ed. Vol. 49. Academic Press, 1–68.

ECK, M., VOGEL, S., AND WAIBEL, A. 2004. Language model adaptation for statistical machine translation based on information retrieval. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

FEDERICO, M. AND BERTOLDI, N. 2006. How many bits are needed to store probabilities for phrase-based translation? In *Proceedings of NAACL Workshop on Statistical Machine Translation*. 94–101.

FOSTER, G., KUHN, R., AND JOHNSON, H. 2006. Phrasetable smoothing for statistical machine translation. In *Proceedings of EMNLP*. 53–61.

FOX, H. J. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP*. 304–311.

FRASER, A. AND MARCU, D. 2006. Semi-supervised training for statistical word alignment. In *Proceedings of the Association for Computational Linguistics (ACL)*. 769–776.

FRASER, A. AND MARCU, D. 2007a. Getting the structure right for word alignment: LEAF. In *Proceedings of EMNLP-CoNLL*. 51–60.

FRASER, A. AND MARCU, D. 2007b. Measuring word alignment quality for statistical machine translation. *Comput. Linguist. 33*, 3.

GALE, W. A. AND CHURCH, K. W. 1991. Identifying word correspondences in parallel text. In *Proceedings of Darpa Workshop on Speech and Natural Language*. 152–157.

GALE, W. A. AND CHURCH, K. W. 1993. A program for aligning sentences in bilingual corpora. *Comput. Linguist. 19*, 1, 75–102.

GALLEY, M., GRAEHL, J., KNIGHT, K., MARCU, D., DENEEFE, S., WANG, W., AND THAYER, I. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the Association for Computational Linguistics (ACL)*. 961–968.

GALLEY, M., HOPKINS, M., KNIGHT, K., AND MARCU, D. 2004. What's in a translation rule? In *Proceedings of HLT-NAACL*. 273–280.

GERMANN, U. 2003. Greedy decoding for statistical machine translation in almost linear time. In *Proceedings of HLT-NAACL*. 72–79.

GERMANN, U., JAHR, M., KNIGHT, K., MARCU, D., AND YAMADA, K. 2001. Fast decoding and optimal decoding for machine translation. In *Proceedings of ACL-EACL*.

GERMANN, U., JAHR, M., KNIGHT, K., MARCU, D., AND YAMADA, K. 2004. Fast and optimal decoding for machine translation. *Artif. Intellig. 154*, 1–2, 127–143.

GILDEA, D. 2004. Dependencies vs. constituencies for tree-based alignment. In *Proceedings of EMNLP*. 214–221.

GOLDWATER, S. AND MCCLOSKY, D. 2005. Improving statistical MT through morphological analysis. In *Proceedings of HLT-EMNLP*. 676–683.

GRAEHL, J. AND KNIGHT, K. 2004. Training tree transducers. In *Proceedings of HLT-NAACL*. 105–112.

HOPCROFT, J. E. AND ULLMAN, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

HOVY, E., KING, M., AND POPESCU-BELIS, A. 2002. Principles of context-based machine translation evaluation. *Mach. Transl. 17*, 1, 43–75.

HUANG, L. AND CHIANG, D. 2005. Better *k*-best parsing. In *Proceedings of the International Workshop on Pausing Technologies (IWPT)*. 53–64.

HUANG, L. AND CHIANG, D. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the Association for Computational Linguistics (ACL)*. 144–151.

HUTCHINS, J. 2007. Machine translation: A concise history. In *Computer Aided Translation: Theory and Practice*, C. S. Wai, Ed. Chinese University of Hong Kong.

HWA, R., RESNIK, P., WEINBERG, A., CABEZAS, C., AND KOLAK, O. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Lang. Engin. 11*, 3, 311–325.

ITTYCHERIAH, A. AND ROUKOS, S. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of HLT-EMNLP*. 89–96.

JELINEK, F. 1969. A stack algorithm for faster sequential decoding of transmitted information. Tech. rep. RC2441, IBM Research Center, Yorktown Heights, NY.

JELINEK, F. 1998. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA.

JOHNSON, H., MARTIN, J., FOSTER, G., AND KUHN, R. 2007. Improving translation quality by discarding most of the phrasetable. In *Proceedings of EMNLP-CoNLL*. 967–975.

JOSHI, A. K. AND SCHABES, Y. 1997. Tree-adjoining grammars. In *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Vol. 3. Springer, Berlin, Germany, 69–124.

JOSHI, A. K., VIJAY-SHANKER, K., AND WEIR, D. 1991. The convergence of mildly context-sensitive grammar formalisms. In *Foundational Issues in Natural Language Processing*, P. Sells, S. Shieber, and T. Wasow, Eds. MIT Press, Cambridge, MA, Chapter 2, 31–81.

JURAFSKY, D. AND MARTIN, J. H. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd Ed. Prentice-Hall.

KIRCHHOFF, K. AND YANG, M. 2005. Improved language modeling for statistical machine translation. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*. 125–128.

KNIGHT, K. 1997. Automating knowledge acquisition for machine translation. *AI Mag. 18*, 4, 81–96.

KNIGHT, K. 1999a. Decoding complexity in word-replacement translation models. *Comput. Linguist. 25*, 4, 607–615.

KNIGHT, K. 1999b. A statistical MT tutorial workbook. Unpublished. http://www.cisp.jhu.edu/ws99/projects/mt/wkbk.rtf.

KNIGHT, K. AND AL-ONAIZAN, Y. 1998. Translation with finite-state devices. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*. 421–437.

KNIGHT, K. AND GRAEHL, J. 2005. An overview of probabilistic tree transducers for natural language processing. In *Proceedings of CICLing*.

KNIGHT, K. AND MARCU, D. 2005. Machine translation in the year 2004. In *Proceedings of ICASSP*.

KOEHN, P. 2004a. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

KOEHN, P. 2004b. *PHARAOH, A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models, User Manual and Description for Version 1.2 USC Information Sciences Institute*.

KOEHN, P. 2004c. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*. 388–395.

KOEHN, P. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

KOEHN, P. 2008. *Statistical Machine Translation*. Cambridge University Press. To appear.

KOEHN, P. AND HOANG, H. 2007. Factored translation models. In *Proceedings of EMNLP-CoNLL*. 868–876.

KOEHN, P., HOANG, H., BIRCH, A., CALLISON-BURCH, C., FEDERICO, M., BERTOLDI, N., COWAN, B., SHEN, W., MORAN, C., ZENS, R., DYER, C., BOJAR, O., CONSTANTIN, A., AND HERBST, E. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL Demo and Poster Sessions*. 177–180.

KOEHN, P. AND MONZ, C. 2005. Shared task: Statistical machine translation between european languages. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*. 119–124.

KOEHN, P. AND MONZ, C. 2006. Manual and automatic evaluation of machine translation between european languages. In *Proceedings of NAACL Workshop on Statistical Machine Translation*. 102–121.

KOEHN, P., OCH, F. J., AND MARCU, D. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*. 127–133.

KULESZA, A. AND SHIEBER, S. M. 2004. A learning approach to improving sentence-level MT evaluation. In *Proceedings of the International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*.

KUMAR, S. AND BYRNE, W. 2004. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of HLT-NAACL*. 169–176.

KUMAR, S., DENG, Y., AND BYRNE, W. 2006. A weighted finite state transducer translation template model for statistical machine translation. *Natural Lang. Engin. 12*, 1, 35–75.

LARI, K. AND YOUNG, S. J. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Comput. Speech Lang. 4*, 1.

LEWIS, P. M. I. AND STEARNS, R. E. 1968. Syntax-directed transductions. *J. ACM 15*, 465–488.

LIANG, P., BOUCHARD-CÔTÉ, A., TASKAR, B., AND KLEIN, D. 2006. An end-to-end discriminative approach to machine translation. In *Proceedings of ACL-COLING*. 761–768.

LIANG, P., TASKAR, B., AND KLEIN, D. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*. 104–111.

LIN, C.-Y. AND OCH, F. J. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the Association for Computational Linguistics (ACL)*. 606–613.

LITA, L., ROGATI, M., AND LAVIE, A. 2005. BLANC: Learning evaluation metrics for MT. In *Proceedings of HLT-EMNLP*. 740–747.

LIU, D. AND GILDEA, D. 2007. Source-language features and maximum correlation training for machine translation evaluation. In *Proceedings of HLT-NAACL*. 41–48.

LOPEZ, A. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of EMNLP-CoNLL*. 976–985.

LOPEZ, A. AND RESNIK, P. 2005. Improved HMM alignment models for languages with scarce resources. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*. 83–86.

LOPEZ, A. AND RESNIK, P. 2006. Word-based alignment, phrase-based translation: What's the link? In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*. 90–99.

MANNING, C. D. AND SCHÜTZE, H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

MARCU, D., WANG, W., ECHIHABI, A., AND KNIGHT, K. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of EMNLP*. 44–52.

MARCU, D. AND WONG, W. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP*. 133–139.

MARCUS, M. P., MARCINKIEWICZ, M. A., AND SANTORINI, B. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comput. Linguist. 19*, 2, 314–330.

MATUSOV, E., ZENS, R., AND NEY, H. 2004. Symmetric word alignments for statistical machine translation. In *Proceedings of COLING*. 219–225.

MELAMED, I. D. 1996. Automatic construction of clean broad-coverage translation lexicons. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

MELAMED, I. D. 1998. Manual annotation of translational equivalence: The blinker project. Tech. rep. 98-07, University of Pennsylvania Institute for Research in Cognitive Science.

MELAMED, I. D. 2000. Models of translational equivalence among words. *Comput. Linguist. 26*, 2, 221–249.

MELAMED, I. D. 2003. Multitext grammars and synchronous parsers. In *Proceedings of HLT-NAACL*. 79–86.

MELAMED, I. D. 2004a. Algorithms for syntax-aware statistical machine translation. In *Proceedings of Theoretical and Methodological Issues in Machine Translation (TMI)*.

MELAMED, I. D. 2004b. Statistical machine translation by parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*. 654–661.

MELAMED, I. D., GREEN, R., AND TURIAN, J. P. 2003. Precision and recall of machine translation. In *Proceedings of HLT-NAACL* (Companion Vol.). 61–63.

MELAMED, I. D., SATTA, G., AND WELLINGTON, B. 2004. Generalized multitext grammars. In *Proceedings of the Association for Computational Linguistics (ACL)*. 662–669.

MERIALDO, B. 1994. Tagging English text with a probabilistic model. *Comput. Linguist. 20*, 2, 155–172.

MIHALCEA, R. AND PEDERSEN, T. 2003. An evaluation exercise for word alignment. In *Proceedings of HLT-NAACL Workshop on Building and Using Parallel Texts*. 1–10.

MINKOV, E., TOUTANOVA, K., AND SUZUKI, H. 2007. Generating complex morphology for machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 128–135.

MITCHELL, T. M. 1997. *Machine Learning*. McGraw-Hill.

MOORE, R. C. 2004. Improving IBM word-alignment model 1. In *Proceedings of the Association for Computational Linguistics (ACL)*. 519–526.

MOORE, R. C. 2005a. Association-based bilingual word alignment. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*. 1–8.

MOORE, R. C. 2005b. A discriminative framework for bilingual word alignment. In *Proceedings of HLT-EMNLP*. 81–88.

NIEßEN, S. AND NEY, H. 2004. Statistical machine translation with scarce resources using morpho-syntactic information. *Comput. Linguist. 30*, 2, 182–204.

NIEßEN, S., VOGEL, S., NEY, H., AND TILLMAN, C. 1998. A DP based search algorithm for statistical machine translation. In *Proceedings of ACL-COLING*. 960–967.

OARD, D. W., DOERMANN, D., DORR, B., HE, D., RESNIK, P., WEINBERG, A., BYRNE, W., KHUDANPUR, S., YAROWSKY, D., LEUSKI, A., KOEHN, P., AND KNIGHT, K. 2003. Desperately seeking Cebuano. In *Proceedings of HLT-NAACL* (Companion Vol.). 76–78.

OARD, D. W. AND OCH, F. J. 2003. Rapid-response machine translation for unexpected languages. In *Proceedings of MT Summit IX*.

OCH, F. J. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*. 71–76.

OCH, F. J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*.

OCH, F. J. 2005. Statistical machine translation: The fabulous present and future. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*. Invited talk.

OCH, F. J., GILDEA, D., KHUDANPUR, S., SARKAR, A., YAMADA, K., FRASER, A., KUMAR, S., SHEN, L., SMITH, D., ENG, K., JAIN, V., JIN, Z., AND RADEV, D. 2004a. Final report of Johns Hopkins 2003 summer workshop on syntax for statistical machine translation. Tech. rep., Johns Hopkins University.

OCH, F. J., GILDEA, D., KHUDANPUR, S., SARKAR, A., YAMADA, K., FRASER, A., KUMAR, S., SHEN, L., SMITH, D., ENG, K., JAIN, V., JIN, Z., AND RADEV, D. 2004b. A smorgasbord of features for statistical machine translation. In *Proceedings of HLT-NAACL*. 161–168.

OCH, F. J. AND NEY, H. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of COLING*. 1086–1090.

OCH, F. J. AND NEY, H. 2001. Statistical multi-source translation. In *Proceedings of MT Summit*.

OCH, F. J. AND NEY, H. 2002. Discriminative training and maximum entropy models for machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 156–163.

OCH, F. J. AND NEY, H. 2003. A systematic comparison of various statistical alignment models. *Comput. Linguist. 29*, 1, 19–51.

OCH, F. J. AND NEY, H. 2004. The alignment template approach to machine translation. *Comput. Linguist. 30*, 4, 417–449.

OCH, F. J., TILLMAN, C., AND NEY, H. 1999. Improved alignment models for statistical machine translation. In *Proceedings of EMNLP-VLC*. 20–28.

OCH, F. J., UEFFING, N., AND NEY, H. 2001. An efficient A* search algorithm for statistical machine translation. In *Proceedings of ACL Workshop on Data-Driven Methods in Machine Translation*. 55–62.

OLTEANU, M., DAVIS, C., VOLOSEN, I., AND MOLDOVAN, D. 2006. Phramer — an open source statistical phrase-based translator. In *Proceedings of HLT-NAACL Workshop on Statistical Machine Translation*. 146–149.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 311–318.

Popovic, M., de Gispert, A., Gupta, D., Lambert, P., Ney, H., Mariño, J. B., Federico, M., and Banchs, R. 2006. Morpho-syntactic information for automatic error analysis of statistical machine translation output. In *Proceedings of NAACL Workshop on Statistical Machine Translation*. 1–6.

Quirk, C., Menezes, A., and Cherry, C. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the Association for Computational Linguistics (ACL)*. 271–279.

Ratnaparkhi, A. 1998. Maximum entropy models for natural language ambiguity resolution. Ph.D. thesis, University of Pennsylvania.

Resnik, P., Olsen, M. B., and Diab, M. 1997. Creating a parallel corpus from the Book of 2000 Tongues. In *Proceedings of the Text Encoding Initiative 10th Anniversary User Conference (TEI-10)*.

Resnik, P. and Smith, N. A. 2003. The Web as parallel corpus. *Comput. Linguist. 29*, 3, 349–380.

Russell, S. and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach* 2nd Ed. Prentice-Hall.

Schafer, C. and Drabek, E. F. 2005. Models for inuktitut-english word alignment. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*. 79–82.

Shen, L., Sarkar, A., and Och, F. J. 2004. Discriminative reranking for machine translation. In *Proceedings of HLT-NAACL*. 177–184.

Shieber, S. M. and Schabes, Y. 1990. Synchronous tree-adjoining grammars. In *Proceedings of COLING*. 253–258.

Simard, M., Cancedda, N., Cavestro, B., Dymetman, M., Gaussier, E., Goutte, C., Yamada, K., Langlais, P., and Mauser, A. 2005. Translating with non-contiguous phrases. In *Proceedings of HLT-EMNLP*. 755–762.

Sipser, M. 2005. *Introduction to the Theory of Computation* 2nd Ed. PWS Publishing.

Smith, D. A. and Smith, N. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceedings of EMNLP*. 49–56.

Smith, N. A. 2002. From words to corpora: Recognizing translation. In *Proceedings of EMNLP*. 95–102.

Smith, N. A. 2006. Novel estimation methods for unsupervised discovery of latent structure in natural language text. Ph.D. thesis, Johns Hopkins University.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*. 223–231.

Talbot, D. and Osborne, M. 2007a. Randomised language modelling for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 512–519.

Talbot, D. and Osborne, M. 2007b. Smoothed bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the Association for Computational Linguistics (ACL)*. 468–476.

Taskar, B. 2004. Learning structured prediction models: A large-margin approach. Ph.D. thesis, Stanford University.

Taskar, B., Lacoste-Julien, S., and Klein, D. 2005. A discriminative matching approach to word alignment. In *Proceedings of HLT-EMNLP*. 73–80.

Tillman, C. and Ney, H. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Comput. Linguist. 29*, 1, 98–133.

Tillmann, C., Vogel, S., Ney, H., and Zubiaga, A. 1997. A DP-based search using monotone alignments in statistical translation. In *Proceedings of ACL-EACL*. 289–296.

Tillmann, C. and Zhang, T. 2006. A discriminative global training algorithm for statistical MT. In *Proceedings of ACL-COLING*. 721–728.

Toutanova, K., Ilhan, H. T., and Manning, C. D. 2002. Extensions to HMM-based statistical word alignment models. In *Proceedings of EMNLP*. 87–94.

Turian, J. P., Shen, L., and Melamed, I. D. 2003. Evaluation of machine translation and its evaluation. In *Proceedings of MT Summit IX*.

Ueffing, N., Haffari, G., and Sarkar, A. 2007. Transductive learning for statistical machine translation. In *Proceedings of the Association for Computational Linguisties (ACL)*. 25–32.

Ueffing, N. and Ney, H. 2005. Word-level confidence estimation for machine translation using phrase-based translation models. In *Proceedings of HLT-EMNLP*. 763–770.

Ueffing, N., Och, F. J., and Ney, H. 2002. Generation of word graphs in statistical machine translation. In *Proceedings of EMNLP*. 156–163.

Venugopal, A., Zollmann, A., and Vogel, S. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *Proceedings of HLT-NAACL*.

VENUGOPAL, A., ZOLLMANN, A., AND WAIBEL, A. 2005. Training and evaluating error minimization rules for statistical machine translation. In *Proceedings of ACL Workshop on Building and Using Parallel Texts*. 208–215.

VIJAY-SHANKER, K., WEIR, D., AND JOSHI, A. K. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the Association for Computational Linguistics (ACL)*. 104–111.

VITERBI, A. J. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory 13*, 2, 260–269.

VOGEL, S., NEY, H., AND TILLMAN, C. 1996. HMM-based word alignment in statistical machine translation. In *Proceedings of COLING*. 836–841.

WANG, C., COLLINS, M., AND KOEHN, P. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP-CoNLL*. 737–745.

WANG, J. 2005. Matching meaning for cross-language information retrieval. Ph.D. thesis, University of Maryland.

WANG, Y.-Y. AND WAIBEL, A. 1997. Decoding algorithm in statistical machine translation. In *Proceedings of ACL-EACL*. 366–372.

WATANABE, T. AND SUMITA, E. 2002. Bidirectional decoding for statistical machine translation. In *Proceedings of COLING*. 1079–1085.

WATANABE, T., SUZUKI, J., TSUKADA, H., AND ISOZAKI, H. 2007. Online large-margin training for statistical machine translation. In *Proceedings of EMNLP-CoNLL*. 764–773.

WEAVER, W. 1955. Translation. In *Machine Translation of Languages: Fourteen Essays*, W. N. Locke and A. D. Booth, Eds. MIT Press, Chapter 1, 15–23.

WELLINGTON, B., TURIAN, J., PIKE, C., AND MELAMED, I. D. 2006. Scalable purely-discriminative training for word and tree transducers. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*. 251–260.

WELLINGTON, B., WAXMONSKY, S., AND MELAMED, I. D. 2006. Empirical lower bounds on the complexity of translational equivalence. In *Proceedings of ACL-COLING*. 977–984.

WHITE, J. S., O'CORNELL, T., AND O'NAVA, F. 1994. The ARPA MT evaluation methodologies: Evolution, lessons and future approaches. In *Proceedings of the Association for Machine Translation in the Americas*.

WU, D. 1995a. Grammarless extraction of phrasal translation examples from parallel texts. In *Proceedings of the Theoretical and Methodological Issues in Machine Translation (TMI)*. 354–372.

WU, D. 1995b. Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *Proceedings of IJCAI*. 1328–1335.

WU, D. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 152–158.

WU, D. AND WONG, H. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of ACL-COLING*. 1408–1415.

XIONG, D., LIU, Q., AND LIN, S. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of ACL-COLING*. 521–528.

YAMADA, K. AND KNIGHT, K. 2001. A syntax-based statistical translation model. In *Proceedings of ACL-EACL*.

YAMADA, K. AND KNIGHT, K. 2002. A decoder for syntax-based statistical MT. In *Proceedings of the Association for Computational Linguistics (ACL)*. 303–310.

YAROWSKY, D. AND NGAI, G. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of NAACL*. 200–207.

YAROWSKY, D., NGAI, G., AND WICENTOWSKI, R. 2001. Inducing mulilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the Human Language Technology Conference (HLT)*. 109–116.

ZENS, R. AND NEY, H. 2003. A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*. 144–151.

ZENS, R. AND NEY, H. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of HLT-NAACL*. 257–264.

ZENS, R. AND NEY, H. 2007. Efficient phrase-table representation for machine translation with applications to online MT and speech translation. In *Proceedings of HLT-NAACL*.

ZHANG, H. AND GILDEA, D. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *Proceedings of the Association for Computational Linguistics (ACL)*. 475–482.

ZHANG, H., HUANG, L., GILDEA, D., AND KNIGHT, K. 2006. Synchronous binarization for machine translation. In *Proceedings of HLT-NAACL*. 256–263.

ZHANG, Y., HILDEBRAND, A. S., AND VOGEL, S. 2006. Distributed language modeling for N-best list re-ranking. In *Proceedings of EMNLP*. 216–223.

ZHANG, Y. AND VOGEL, S. 2005. An efficient phrase-to-phrase alignment model for arbitrarily long phrase and large corpora. In *Proceedings of EAMT*.

Received March 2006; revised August 2007; accepted October 2007