Translation as Weighted Deduction

Adam Lopez
University of Edinburgh
10 Crichton Street
Edinburgh, EH8 9AB
United Kingdom
alopez@inf.ed.ac.uk

Abstract

We present a unified view of many translation algorithms that synthesizes work on deductive parsing, semiring parsing, and efficient approximate search algorithms. This gives rise to clean analyses and compact descriptions that can serve as the basis for modular implementations. We illustrate this with several examples, showing how to build search spaces for several disparate phrase-based search strategies, integrate non-local features, and devise novel models. Although the framework is drawn from parsing and applied to translation, it is applicable to many dynamic programming problems arising in natural language processing and other areas.

1 Introduction

Implementing a large-scale translation system is a major engineering effort requiring substantial time and resources, and understanding the tradeoffs involved in model and algorithm design decisions is important for success. As the space of systems described in the literature becomes more crowded, identifying their common elements and isolating their differences becomes crucial to this understanding. In this work, we present a common framework for model manipulation and analysis that accomplishes this, and use it to derive surprising conclusions about phrase-based models.

Most translation algorithms do the same thing: dynamic programming search over a space of weighted rules (§2). Fortunately, we need not search far for modular descriptions of dynamic programming algorithms. Deductive logic (Pereira and Warren, 1983), extended with semirings (Goodman, 1999), is an established formalism used in parsing. It is occasionally used to describe formally syntactic translation models, but these treatments tend to be brief (Chiang, 2007; Venugopal et al., 2007; Dyer et al., 2008; Melamed, 2004). We apply weighted deduction much more thoroughly, first extending it to phrase-based models and showing that the set of search strategies used by these models have surprisingly different implications for model and search error (§3, §4). We then show how it can be used to analyze common translation problems such as non-local parameterizations (§5), alignment, and novel model design (§6). Finally, we show that it leads to a simple analysis of cube pruning (Chiang, 2007), an important approximate search algorithm (§7).

2 Translation Models

A translation model consists of two distinct elements: an *unweighted* ruleset, and a parameterization (Lopez, 2008). A **ruleset** licenses the steps by which a source string $f_1...f_I$ may be rewritten as a target string $e_1...e_J$, thereby defining the finite set of all possible rewritings of a source string. A **parameterization** defines a weight function over every sequence of rule applications.

In a phrase-based model, the ruleset is simply the unweighted phrase table, where each phrase pair $f_i...f_{i'}/e_j...e_{j'}$ states that phrase $f_i...f_{i'}$ in the source is rewritten as $e_j...e_{j'}$ in the the target.

The model operates by iteratively applying rewrites to the source sentence until each source word has been consumed by exactly one rule. We call a sequence of rule applications a **derivation**. A target string $e_1...e_J$ yielded by a derivation D is obtained by concatenating the target phrases of the rules in the order in which they were applied. We define $Y(D)$ to be the target string yielded by D.

Now consider the Viterbi approximation to a noisy channel parameterization of this model, $P(f|D) \cdot P(D)$.[1] We define $P(f|D)$ in the standard way.

$$P(f|D) = \prod_{f_i...f_{i'}/e_j...e_{j'} \in D} p(f_i...f_{i'}|e_j...e_{j'}) \quad (1)$$

Note that in the channel model, we can replace any rule application with any other rule containing the same source phrase without affecting the partial score of the rest of the derivation. We call this a **local** parameterization.

Now we define a standard n-gram model $P(D)$.

$$P(D) = \prod_{e_j \in Y(D)} p(e_j|e_{j-n+1}...e_{j-1}) \quad (2)$$

This parameterization differs from the channel model in an important way. If we replace a single rule in the derivation, the partial score of the rest of derivation is also affected, because the terms $e_{j-n+1}...e_j$ may come from more than one rule. In other words, this parameterization encodes a dependency between the steps in a derivation. We call this a **non-local** parameterization.

3 Translation As Deduction

For the first part of the discussion that follows, we consider deductive logics purely over unweighted rulesets. As a way to introduce deductive logic, we consider the CKY algorithm for context-free parsing, a common example that we will revisit in §6.2. It is also relevant since it can form the basis of a decoder for inversion transduction grammar (Wu, 1996). In the discussion that follows, we use A, B, and C to denote arbitrary nonterminal symbols, S to denote the start nonterminal symbol, and a to denote a terminal symbol. CKY works on grammars in Chomsky normal form: all rules are either binary as in $A \rightarrow BC$, or unary as in $A \rightarrow a$.

The number of possible binary-branching parses of a sentence is defined by the Catalan number, an exponential combinatoric function (Church and Patil, 1982), so dynamic programming is crucial for efficiency. CKY computes all parses in cubic time by reusing subparses. To parse a sentence $a_1...a_K$, we compute a set of **items** in the form $[A, k, k']$, where A is a nonterminal category, k and k' are both integers in the range $[0, n]$. This item represents the fact that there is some parse of span $a_{k+1}...a_{k'}$ rooted at A (span indices are on the spaces between words). CKY works by creating items over successively longer spans. First it creates items $[A, k-1, k]$ for any rule $A \rightarrow a$ such that $a = a_k$. It then considers spans of increasing length, creating items $[A, k, k']$ whenever it finds two items $[B, k, k'']$ and $[C, k'', k']$ for some grammar rule $A \rightarrow BC$ and some midpoint k''. Its goal is an item $[S, 0, K]$, indicating that there is a parse of $a_1...a_K$ rooted at S.

A CKY logic describes its actions as **inference rules**, equivalent to Horn clauses. The inference rule is a list of **antecedents**, items and rules that must all be true for the inference to occur; and a single **consequent** that is inferred. To denote the creation of item $[A, k, k']$ based on existence of rule $A \rightarrow BC$ and items $[B, k, k'']$ and $[C, k'', k']$, we write an inference rule with antecedents on the top line and consequent on the second line, following Goodman (1999) and Shieber et al. (1995).

$$\frac{R(A \rightarrow BC) \;\; [B, k, k''] \;\; [C, k'', k']}{[A, k, k']}$$

We now give the complete Logic CKY.

item form: $[A, k, k']$ **goal:** $[S, 0, K]$

rules:
$$\left\{ \begin{array}{c} \dfrac{R(A \rightarrow a_k)}{[A, k-1, k]} \\ \\ \dfrac{R(A \rightarrow BC) \;\; [B, k, k''] \;\; [C, k'', k']}{[A, k, k']} \end{array} \right.$$

(Logic CKY)

A benefit of this declarative description is that complexity can be determined by inspection (McAllester, 1999). We elaborate on complexity in §7, but for now it suffices to point out that the number of possible items and possible deductions depends on the product of the domains of the free variables. For example, the number of possible CKY items for a grammar with G nonterminals is $O(GK^2)$, because k and k' are both in range $[0, K]$. Likewise, the number of possible inference rules that can fire is $O(G^3K^3)$.

3.1 A Simple Deductive Decoder

For our first example of a translation logic we consider a simple case: monotone decoding (Mariño et al., 2006; Zens and Ney, 2004). Here, rewrite rules are applied strictly from left to right on the source sentence. Despite its simplicity, the search

[1] The true noisy channel parameterization $p(f|e) \cdot p(e)$ would require a marginalization over D, and is intractable for most models.

space can be very large—in the limit there could be a translation for every possible segmentation of the sentence, so there are exponentially many possible derivations. Fortunately, we know that monotone decoding can easily be cast as a dynamic programming problem. For any position i in the source sentence $f_1 ... f_I$, we can freely combine any partial derivation covering $f_1 ... f_i$ on its left with any partial derivation covering $f_{i+1} ... f_I$ on its right to yield a complete derivation.

In our deductive program for monotone decoding, an item simply encodes the index of the rightmost word that has been rewritten.

item form: $[i]$ **goal:** $[I]$ **rule:** $\dfrac{[i] \;\; R(f_{i+1} ... f_{i'} / e_j ... e_{j'})}{[i']}$

(Logic MONOTONE)

This is the algorithm of Zens and Ney (2004). With a maximum phrase length of m, i' will range over $[i+1, \min(i+m, I)]$, giving a complexity of $O(Im)$. In the limit it is $O(I^2)$.

3.2 More Complex Decoders

Now we consider phrase-based decoders with more permissive reordering. In the limit we allow arbitrary reordering, so our item must contain a coverage vector. Let V be a binary vector of length I; that is, $V \in \{0,1\}^I$. Le 0^m be a vector of m 0's. For example, bit vector 00000 will be abbreviated 0^5 and bit vector 000110 will be abbreviated $0^3 1^2 0^1$. Finally, we will need bitwise AND ($\wedge$) and OR ($\vee$). Note that we impose an additional requirement that is not an item in the deductive system as a **side condition** (we elaborate on the significance of this in §4).

item form: $[\{0,1\}^I]$ **goal:** $[1^I]$

rule:

$$\frac{[V] \;\; R(f_{i+1} ... f_{i'} / e_j ... e_{j'})}{[V \vee 0^i 1^{i'-i} 0^{I-i'}]} \; V \wedge 0^i 1^{i'-i} 0^{I-i'} = 0^I$$

(Logic PHRASE-BASED)

The runtime complexity is exponential, $O(I^2 2^I)$. Practical decoding strategies are more restrictive, implementing what is frequently called a *distortion limit* or *reordering limit*. We found that these terms are inexact, used to describe a variety of quite different strategies.[2] Since we did not feel that the relationship between these various strategies was obvious or well-known, we give logics for several of them and a brief analysis of the implications. Each strategy uses a parameter d, generically called the distortion limit or reordering limit.

The **Maximum Distortion d strategy** (MDd) requires that the first word of a phrase chosen for translation be within d words of the the last word of the most recently translated phrase (Figure 1).[3] The effect of this strategy is that, up to the last word covered in a partial derivation, there must be a covered word in every d words. Its complexity is $O(I^3 2^d)$.

MDd can produce partial derivations that cannot be completed by any allowed sequence of jumps. To prevent this, the **Window Length d strategy** (WLd) enforces a tighter restriction that the last word of a phrase chosen for translation cannot be more than d words from the leftmost untranslated word in the source (Figure 1).[4] For this logic we use a bitwise shift operator ($\ll$), and a predicate (α_1) that counts the number of leading ones in a bit array.[5] Its runtime is exponential in parameter d, but *linear* in sentence length, $O(d^2 2^d I)$.

The **First d Uncovered Words strategy** (FdUW) is described by Tillman and Ney (2003) and Zens and Ney (2004), who call it the *IBM Constraint*.[6] It requires at least one of the leftmost d uncovered words to be covered by a new phrase. Items in this strategy contain the index i of the rightmost covered word and a vector $U \in [1, I]^d$ of the d leftmost uncovered words (Figure 1). Its complexity is $O(dI\binom{I}{d+1})$, which is roughly exponential in d.

There are additional variants, such as the **Maximum Jump d strategy** (MJd), a polynomial-time strategy described by Kumar and Byrne (2005), and possibly others. We lack space to describe all of them, but simply depicting the strategies as logics permits us to make some simple analyses.

First, it should be clear that these reordering strategies define overlapping but not identical search spaces: for most values of d it is impossible to find d' such that any of the other strategies would be identical (except for degenerate cases

[2] Costa-jussà and Fonollosa (2006) refer to the reordering limit and distortion limit as two distinct strategies.

[3] Moore and Quirk (2007) give a nice description of MDd.

[4] We do not know if WLd is documented anywhere, but from inspection it is used in Moses (Koehn et al., 2007). This was confirmed by Philipp Koehn and Hieu Hoang (p.c.).

[5] When a phrase covers the first uncovered word in the source sentence, the new first uncovered word may be further along in the sentence (if the phrase completely filled a gap), or just past the end of the phrase (otherwise).

[6] We could not identify this strategy in the IBM patents.

$$(1)\quad \textbf{item form: } [i, \{0,1\}^I] \quad \textbf{goal: } [i \in [I-d, I], 1^I] \quad \textbf{rule: } \frac{[i'', V] \;\; R(f_{i+1}...f_{i'}/e_j...e_{j'})}{[i', V \vee 0^i 1^{i'-i} 0^{I-i'}]} \; V \wedge 0^i 1^{i'-i} 0^{I-i'} = 0^I, |i - i''| \leq d$$

$$(2)\quad \textbf{item form: } [i, \{0,1\}^d] \quad \textbf{goal: } [I, 0^d] \quad \textbf{rules: } \begin{cases} \dfrac{[i, C] \;\; R(f_{i+1}...f_{i'}/e_j...e_{j'})}{[i'', C \ll i'' - i]} & C \wedge 1^{i'-i} 0^{d-i'+i} = 0^d, i' - i \leq d, \\ & \alpha_1(C \vee 1^{i'-i} 0^{d-i'+i}) = i'' - i \\ \dfrac{[i, C] \;\; R(f_{i'}...f_{i''}/e_j...e_{j'})}{[i, C \vee 0^{i'-i} 1^{i''-i'} 0^{d-i''+i}]} & C \wedge 0^{i'-i} 1^{i''-i'} 0^{d-i''+i} = 0^d, i'' - i \leq d \end{cases}$$

$$(3)\quad \textbf{item form: } [i, [1, I+d]^d] \quad \textbf{goal: } [I, [I+1, I+d]]$$

$$\textbf{rules: } \begin{cases} \dfrac{[i, U] \;\; R(f_{i'}...f_{i''}/e_j...e_{j'})}{[i'', U - [i', i''] \vee [i'', i'' + d - |U - [i', i'']|]]} & i' > i, f_{i+1} \in U \\ \dfrac{[i, U] \;\; R(f_{i'}...f_{i''}/e_j...e_{j'})}{[i, U - [i', i''] \vee [\max(U \vee i) + 1, \max(U \vee i) + 1 + d - |U - [i', i'']|]]} & i' < i, [f_{i'}, f_{i''}] \subset U \end{cases}$$

Figure 1: Logics (1) MDd, (2) WLd, and (3) FdUW. Note that the goal item of MDd (1) requires that the last word of the last phrase translated be within d words of the end of the source sentence.

$d = 0$ and $d = I$). This has important ramifications for scientific studies: results reported for one strategy may not hold for others, and in cases where the strategy is not clearly described it may be impossible to replicate results. Furthermore, it should be clear that the strategy can have significant impact on decoding speed and pruning strategies (§7). For example, MDd is more complex than WLd, and we expect implementations of the former to require more pruning and suffer from more search errors, while the latter would suffer from more model errors since its space of possible reorderings is smaller.

We emphasize that many other translation models can be described this way. Logics for the IBM Models (Brown et al., 1993) would be similar to our logics for phrase-based models. Syntax-based translation logics are similar to parsing logics; a few examples already appear in the literature (Chiang, 2007; Venugopal et al., 2007; Dyer et al., 2008; Melamed, 2004). For simplicity, we will use the MONOTONE logic for the remainder of our examples, but all of them generalize to more complex logics.

4 Adding Local Parameterizations via Semiring-Weighted Deduction

So far we have focused solely on unweighted logics, which correspond to search using only rule-sets. Now we turn our focus to parameterizations. As a first step, we consider only local parameterizations, which make computing the score of a derivation quite simple. We are given a set of inferences in the following form (interpreting side conditions $\mathcal{B}_1 ... \mathcal{B}_M$ as boolean constraints).

$$\frac{\mathcal{A}_1 ... \mathcal{A}_L}{\mathcal{C}} \; \mathcal{B}_1 ... \mathcal{B}_M$$

Now suppose we want to find the highest-scoring derivation. Each antecedent item $\mathcal{A}_\ell$ has a probability $p(\mathcal{A}_\ell)$: if $\mathcal{A}_\ell$ is a rule, then the probability is given, otherwise its probability is computed recursively in the same way that we now compute $p(\mathcal{C})$. Since $\mathcal{C}$ can be the consequent of multiple deductions, we take the max of its current value (initially 0) and the result of the new deduction.

$$p(\mathcal{C}) = \max(p(\mathcal{C}), (p(\mathcal{A}_1) \times ... \times p(\mathcal{A}_L))) \quad (3)$$

If for every $\mathcal{A}_\ell$ that is an item, we replace $p(\mathcal{A}_\ell)$ recursively with this expression, we end up with a maximization over a product of rule probabilities. Applying this to logic MONOTONE, the result will be a maximization (over all possible derivations D) of the algebraic expression in Equation 1.

We might also want to calculate the total probability of all possible derivations, which is useful for parameter estimation (Blunsom et al., 2008). We can do this using the following equation.

$$p(\mathcal{C}) = p(\mathcal{C}) + (p(\mathcal{A}_1) \times ... \times p(\mathcal{A}_L)) \quad (4)$$

Equations 3 and 4 are quite similar. This suggests a useful generalization: semiring-weighted deduction (Goodman, 1999).[7] A **semiring** $\langle \mathbb{A}, \otimes, \oplus \rangle$ consists of a domain $\mathbb{A}$, a multiplicative operator $\otimes$ and an additive operator $\oplus$.[8] In Equation 3 we use the Viterbi semiring $\langle [0,1], \times, \max \rangle$, while in Equation 4 we use the inside semiring $\langle [0,1], \times, + \rangle$. The general form of Equations 3 and 4 can be written for weights $w \in \mathbb{A}$.

$$w(\mathcal{C}) \oplus\!= w(\mathcal{A}_1) \otimes ... \otimes w(\mathcal{A}_\ell) \tag{5}$$

Many quantities can be computed simply by using the appropriate semiring. Goodman (1999) describes semirings for the Viterbi derivation, k-best Viterbi derivations, derivation forest, and number of paths.[9] Eisner (2002) describes the *expectation semiring* for parameter learning. Gimpel and Smith (2009) describe *approximation semirings* for approximate summing in (usually intractable) models. In parsing, the boolean semiring $\langle \{\top, \bot\}, \cap, \cup \rangle$ is used to determine grammaticality of an input string. In translation it is relevant for alignment (§6.1).

5 Adding Non-Local Parameterizations with the PRODUCT Transform

A problem arises with the semiring-weighted deductive formalism when we add non-local parameterizations such as an n-gram model (Equation 2). Suppose we have a derivation $D = (d_1, ..., d_M)$, where each d_m is a rule application. We can view the language model as a function on D.

$$P(D) = f(d_1, ..., d_m, ..., d_M) \tag{6}$$

The problem is that replacing d_m with a lower-scoring rule d'_m may actually improve f due to the language model dependency. This means that f is *nonmonotonic*—it does not display the *optimal substructure* property on partial derivations, which is required for dynamic programming (Cormen et al., 2001). The logics still work for some semirings (e.g. boolean), but not others. Therefore, non-local parameterizations break semiring-weighted deduction, because we can no longer use the same logic under all semirings. We need new logics; for this we will use a logic programming transform called the PRODUCT transform (Cohen et al., 2008).

We first define a logic for the non-local parameterization. The logic for an n-gram language model generates sequence $e_1...e_Q$ by generating each new word given the past $n-1$ words.[10]

item form: $[e_q, ..., e_{q+n-2}]$ **goal:** $[e_{Q-n+2}, ..., e_Q]$

$$\textbf{rule: } \frac{[e_q, ..., e_{q+n-2}] R(e_q, ..., e_{q+n-1})}{[e_{q+1}, ..., e_{q+n-1}]}$$

(Logic NGRAM)

Now we want to combine NGRAM and MONOTONE. To make things easier, we modify MONOTONE to encode the idea that once a source phrase has been recognized, its target words are generated one at a time. We will use u_e and v_e to denote (possibly empty) sequences in $e_j...e'_j$. Borrowing the notation of Earley (1970), we encode progress using a dotted phrase $u_e \bullet v_e$.

item form: $[i, u_e \bullet v_e]$ **goal:** $[I, u_e \bullet v_e]$

rules:

$$\frac{[i, u_e \bullet] \;\; R(f_{i+1}...f_{i'}/e_j v_e)}{[i', e_j \bullet v_e]} \qquad \frac{[i, u_e \bullet e_j v_e]}{[i, u_e e_j \bullet v_e]}$$

(Logic MONOTONE-GENERATE)

We combine NGRAM and MONOTONE-GENERATE using the PRODUCT transform, which takes two logics as input and essentially does the following.[11]

1. Create a new item type from the cross-product of item types in the input logics.
2. Create inference rules for the new item type from the cross-product of all inference rules in the input logics.
3. Constrain the new logic as needed. This is done by hand, but quite simple, as we will show by example.

The first two steps give us logic MONOTONE-GENERATE $\circ$ NGRAM (Figure 2). This is close to what we want, but not quite done. The constraint we want to apply is that each word written by logic MONOTONE-GENERATE is equal to the word generated by logic NGRAM. We accomplish this by unifying variables e_q and e_{n-i} in the inference rules, giving us logic MONOTONE-GENERATE + NGRAM (Figure 2).

[7] General weighted deduction subsumes semiring-weighted deduction (Eisner et al., 2005; Eisner and Blatz, 2006; Nederhof, 2003), but semiring-weighted deduction covers all translation models we are aware of, so it is a good first step in applying weighted deduction to translation.

[8] See Goodman (1999) for additional conditions on semirings used in this framework.

[9] Eisner and Blatz (2006) give an alternate strategy for the best derivation.

[10] We ignore start and stop probabilities for simplicity.

[11] The description of Cohen et al. (2008) is much more complete and includes several examples.

(1)

item form: $[i, u_e \bullet v_e, e_q, ..., e_{q+n-2}]$

goal: $[I, u_e\bullet, e_{Q-n+2}, ..., e_Q]$

rules:

$$\frac{[i, u_e\bullet, e_q, ..., e_{q+n-2}] \;\; R(f_i...f_{i'}/e_j u_e) \;\; R(e_q, ..., e_{q+n-1})}{[i', e_j \bullet u_e, e_{q+1}, ..., e_{q+n-1}]}$$

$$\frac{[i, u_e \bullet e_j v_e, e_q, ..., e_{q+n-2}] \;\; R(e_q, ..., e_{q+n-1})}{[i, u_e e_j \bullet v_e, e_{q+1}, ..., e_{q+n-1}]}$$

---

(2)

item form: $[i, u_e \bullet v_e, e_j, ..., e_{j+n-2}]$

goal: $[I, u_e\bullet, e_{J-n+2}, ..., e_J]$

rules:

$$\frac{[i, u_e\bullet, e_{j-n+1}, ..., e_{j-1}] \;\; R(f_i...f_{i'}/e_j v_e) \;\; R(e_{j-n+2}...e_j)}{[i', e_j \bullet v_e, e_{j-n+2}, ..., e_j]}$$

$$\frac{[i, u_e \bullet e_{i+n-1} v_e, e_i, ..., e_{i+n-2}] \;\; R(e_{j-n+2}...e_j)}{[i+1, u_e e_j \bullet v_e, e_{j-n+2}, ..., e_j]}$$

---

(3) **item form:** $[i, u_e \bullet v_e, e_i, ..., e_{n-i-2}]$ **goal:** $[I, u_e\bullet, e_{I-n+2}, ..., e_I]$

rule: $$\frac{[i, e_{j-n+1}, ..., e_{j-1}] \;\; R(f_i...f_{i'}/e_j...e_{j'}) R(e_{j-n+1}, ..., e_j)...R(e_{j'-n+1}...e_{j'})}{[i', e_{j'-n+2}...e_{j'}]}$$

Figure 2: Logics (1) MONOTONE-GENERATE $\circ$ NGRAM, (2) MONOTONE-GENERATE $+$ NGRAM and (3) MONOTONE-GENERATE $+$ NGRAM SINGLE-SHOT.

This logic restores the optimal subproblem property and we can apply semiring-weighted deduction. Efficient algorithms are given in §7, but a brief comment is in order about the new logic. In most descriptions of phrase-based decoding, the n-gram language model is applied all at once. MONOTONE-GENERATE+NGRAM applies the n-gram language model one word at a time. This illuminates a space of search strategies that are to our knowledge unexplored. If a four-word phrase were proposed as an extension of a partial hypothesis in a typical decoder implementation using a five-word language model, all four n-grams will be applied even though the first n-gram might have a very low score. Viewing each n-gram application as producing a new state may yield new strategies for approximate search.

We can derive the more familiar logic by applying a different transform: *unfolding* (Eisner and Blatz, 2006). The idea is to replace an item with the sequence of antecedents used to produce it (similar to function inlining). This gives us MONOTONE-GENERATE+NGRAM SINGLE-SHOT (Figure 2).

We call the ruleset-based logic the **minimal logic** and the logic enhanced with non-local parameterization the **complete logic**. Note that the set of variables in the complete logic is a superset of the set of variables in the minimal logic. We can view the minimal logic as a projection of the complete logic into a smaller dimensional space. It is important to note that complete logic is substantially more complex than the minimal logic, by a factor of $O(|V_E|^n)$ for a target vocabulary of V_E. Thus, the complexity of non-local parameterizations often makes search spaces large regardless of the complexity of the minimal logic.

6 Other Uses of the PRODUCT Transform

The PRODUCT transform can also implement alignment and help derive new models.

6.1 Alignment

In the alignment problem (sometimes called *constrained decoding* or *forced decoding*), we are given a reference target sentence $r_1, ..., r_J$, and we require the translation model to generate only derivations that produce that sentence. Alignment is often used in training both generative and discriminative models (Brown et al., 1993; Blunsom et al., 2008; Liang et al., 2006). Our approach to alignment is similar to the one for language modeling. First, we implement a logic requiring an

input to be identical to the reference.

$$\textbf{item form: } [j] \quad \textbf{goal: } [J] \qquad \textbf{rule: } \frac{[j]}{[j+1]}\ e_{j+1} = r_{j+1}$$

(Logic RECOGNIZE)

The logic only reaches its goal if the input is identical to the reference. In fact, partial derivations must produce a prefix of the reference. When we combine this logic with MONOTONE-GENERATE, we obtain a logic that only succeeds if the translation logic generates the reference.

$$\textbf{item form: } [i, j, u_e \bullet v_e] \quad \textbf{goal: } [I, j, u_e \bullet]$$

$$\textbf{rules: } \begin{cases} \dfrac{[i, j, u_e \bullet]\ \ R(f_i ... f_{i'} / e_j ... e_{j'})}{[i', j, \bullet e_j ... e_{j'}]} \\ \dfrac{[i, j, u_e \bullet e_j v_e]}{[i, j+1, u_e e_j \bullet v_e]}\ e_{j+1} = r_{j+1} \end{cases}$$

(Logic MONOTONE-ALIGN)

Under the boolean semiring, this (minimal) logic decides if a training example is reachable by the model, which is required by some discriminative training regimens (Liang et al., 2006; Blunsom et al., 2008). We can also compute the Viterbi derivation or the sum over all derivations of a training example, needed for some parameter estimation methods. Cohen et al. (2008) derive an alignment logic for ITG from the product of two CKY logics.

6.2 Translation Model Design

A motivation for many syntax-based translation models is to use target-side syntax as a language model (Charniak et al., 2003). Och et al. (2004) showed that simply parsing the N-best outputs of a phrase-based model did not work; to obtain the full power of a language model, we need to integrate it into the search process. Most approaches to this problem focus on synchronous grammars, but it is possible to integrate the target-side language model with a phrase-based translation model. As an exercise, we integrate CKY with the output of logic MONOTONE-GENERATE. The constraint is that the indices of the CKY items unify with the *items* of the translation logic, which form a word lattice. Note that this logic retains the rules in the basic MONOTONE logic, which are not depicted (Figure 3).

The result is a lattice parser on the output of the translation model. Lattice parsing is not new to translation (Dyer et al., 2008), but to our knowledge it has not been used in this way. Viewing translation as deduction is helpful for the design and construction of novel models.

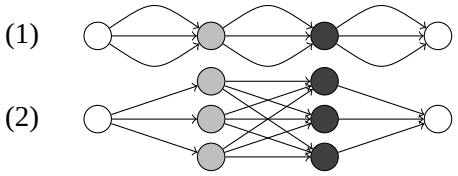

Figure 4: Example graphs corresponding to a simple minimal (1) and complete (2) logic, with corresponding nodes in the same color. The state-splitting induced by non-local features produces in a large number of arcs which must be evaluated, which can be reduced by cube pruning.

7 Algorithms

Most translation logics are too expensive to exhaustively search. However, the logics conveniently specify the full search space, which forms a hypergraph (Klein and Manning, 2001).[12] The equivalence is useful for complexity analysis: items correspond to nodes and deductions correspond to hyperarcs. These equivalences make it easy to compute algorithmic bounds.

Cube pruning (Chiang, 2007) is an approximate search technique for syntax-based translation models with integrated language models. It operates on two objects: a $-LM$ graph containing no language model state, and a $+LM$ hypergraph containing state. The idea is to generate a fixed number of nodes in the $+LM$ for each node in the $-LM$ graph, using a clever enumeration strategy. We can view cube pruning as arising from the interaction between a minimal logic and the state splits induced by non-local features. Figure 4 shows how the added state information can dramatically increase the number of deductions that must be evaluated. Cube pruning works by considering the most promising states paired with the most promising extensions. In this way, it easily fits any search space constructed using the technique of §5. Note that the efficiency of cube pruning is limited by the minimal logic.

Stack decoding is a search heuristic that simplifies the complexity of searching a minimal logic. Each item is associated with a stack whose signa-

[12] Specifically a B-hypergraph, equivalent to an and-or graph (Gallo et al., 1993) or context-free grammar (Nederhof, 2003). In the degenerate case, this is simply a graph, as is the case with most phrase-based models.

item forms: $[i, u_e \bullet v_e], [A, i, u_e \bullet v_e, i', u'_e \bullet v'_e]$ **goal:** $[S, 0, \bullet, I, u_e \bullet]$

rules:

$$\frac{[i, u_e \bullet] \;\; R(f_{i+1} ... f_{i'} / e_j v_e) \;\; R(A \to e_j)}{[A, i, u_e \bullet, i', e_j \bullet v_e]} \qquad \frac{[i, u_e \bullet e_j v_e] \;\; R(A \to e_j)}{[A, i, u_e \bullet e_j v_e, i, u_e e_j \bullet v_e]}$$

$$\frac{[B, i, u_e \bullet v_e, i'', u''_e \bullet v''_e] \;\; [C, i'', u''_e \bullet v''_e, i', u'_e \bullet v'_e] \;\; R(A \to BC)}{[A, i, u_e \bullet v_e, i', u'_e \bullet v'_e]}$$

Figure 3: Logic MONOTONE-GENERATE + CKY

ture is a projection of the item signature (or a predicate on the item signatures)—multiple items are associated to the same stack. The strength of the pruning (and resulting complexity improvements) depending on how much the projection reduces the search space. In many phrase-based implementations the stack signature is just the number of words translated, but other strategies are possible (Tillman and Ney, 2003).

It is worth noting that logic FdUW (§3.2), depends on stack pruning for speed. Because the number of stacks is linear in the length of the input, so is the number of unpruned nodes in the search graph. In contrast, the complexity of logic WLd is naturally linear in input length. As mentioned in §3.2, this implies a wide divergence in the model and search errors of these logics, which to our knowledge has not been investigated.

8 Related Work

We are not the first to observe that phrase-based models can be represented as logic programs (Eisner et al., 2005; Eisner and Blatz, 2006), but to our knowledge we are the first to provide explicit logics for them.[13] We also showed that deductive logic is a useful analytical tool to tackle a variety of problems in translation algorithm design.

Our work is strongly influenced by Goodman (1999) and Eisner et al. (2005). They describe many issues not mentioned here, including conditions on semirings, termination conditions, and strategies for cyclic search graphs. However, while their weighted deductive formalism is general, they focus on concerns relevant to parsing, such as boolean semirings and cyclicity. Our work focuses on concerns common for translation, including a general view of non-local parameterizations and cube pruning.

[13] Huang and Chiang (2007) give an informal example, but do not elaborate on it.

9 Conclusions and Future Work

We have described a general framework that synthesizes and extends deductive parsing and semiring parsing, and adapts it to translation. Our goal has been to show that logics make an attractive shorthand for description, analysis, and construction of translation models. For instance, we have shown that it is quite easy to mechanically construct search spaces using non-local features, and to create exotic models. We showed that different flavors of phrase-based models should suffer from quite different types of error, a problem that to our knowledge was heretofore unknown. However, we have only scratched the surface, and we believe it is possibly to unify a wide variety of translation algorithms. For example, we believe that cube pruning can be described as an agenda discipline in chart parsing (Kay, 1986).

Although the work presented here is abstract, our motivation is practical. Isolating the errors in translation systems is a difficult task which can be made easier by describing and analyzing models in a modular way (Auli et al., 2009). Furthermore, building large-scale translation systems from scratch should be unnecessary if existing systems were built using modular logics and algorithms. We aim to build such systems.

Acknowledgments

This work developed from discussions with Phil Blunsom, Chris Callison-Burch, Chris Dyer, Hieu Hoang, Martin Kay, Philipp Koehn, Josh Schroeder, and Lane Schwartz. Many thanks go to Chris Dyer, Josh Schroeder, the three anonymous EACL reviewers, and one anonymous NAACL reviewer for very helpful comments on earlier drafts. This research was supported by the Euromatrix Project funded by the European Commission (6th Framework Programme).

References

M. Auli, A. Lopez, P. Koehn, and H. Hoang. 2009. A systematic analysis of translation model search spaces. In *Proc. of WMT*, Mar.

P. Blunsom, T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. of ACL:HLT*.

P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, Jun.

E. Charniak, K. Knight, and K. Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit*, Sept.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

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

S. B. Cohen, R. J. Simmons, and N. A. Smith. 2008. Dynamic programming algorithms as products of weighted logic programs. In *Proc. of ICLP*.

T. H. Cormen, C. D. Leiserson, R. L. Rivest, and C. Stein. 2001. *Introduction to Algorithms*. MIT Press, 2nd edition.

M. R. Costa-jussà and J. A. R. Fonollosa. 2006. Statistical machine reordering. In *Proc. of EMNLP*, pages 70–76, Jul.

C. J. Dyer, S. Muresan, and P. Resnik. 2008. Generalizing word lattice translation. In *Proc. of ACL:HLT*, pages 1012–1020.

J. Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, Feb.

J. Eisner and J. Blatz. 2006. Program transformations for optimization of parsing algorithms and other weighted logic programs. In *Proc. of Formal Grammar*, pages 45–85.

J. Eisner, E. Goldlust, and N. A. Smith. 2005. Compiling comp ling: Weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*, pages 281–290.

J. Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*, pages 1–8, Jul.

G. Gallo, G. Longo, and S. Pallottino. 1993. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2), Apr.

K. Gimpel and N. A. Smith. 2009. Approximation semirings: Dynamic programming with non-local features. In *Proc. of EACL*, Mar.

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

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL*, pages 144–151, Jun.

M. Kay. 1986. Algorithm schemata and data structures in syntactic processing. In B. J. Grosz, K. S. Jones, and B. L. Webber, editors, *Readings in Natural Language Processing*, pages 35–70. Morgan Kaufmann.

D. Klein and C. Manning. 2001. Parsing and hypergraphs. In *Proc. of IWPT*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL Demo and Poster Sessions*, pages 177–180, Jun.

S. Kumar and W. Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proc. of HLT-EMNLP*, pages 161–168.

P. Liang, A. Bouchard-Côté, B. Taskar, and D. Klein. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of ACL-COLING*, pages 761–768, Jul.

A. Lopez. 2008. Statistical machine translation. *ACM Computing Surveys*, 40(3), Aug.

J. B. Mariño, R. E. Banchs, J. M. Crego, A. de Gispert, P. Lambert, J. A. R. Fonollosa, and M. R. Costa-jussà. 2006. N-gram based statistical machine translation. *Computational Linguistics*, 32(4):527–549, Dec.

D. McAllester. 1999. On the complexity analysis of static analyses. In *Proc. of Static Analysis Symposium*, volume 1694/1999 of *LNCS*. Springer Verlag.

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

R. C. Moore and C. Quirk. 2007. Faster beam-search decoding for phrasal statistical machine translation. In *Proc. of MT Summit*.

M.-J. Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143, Mar.

F. J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proc. of HLT-NAACL*, pages 161–168, May.

F. C. N. Pereira and D. H. D. Warren. 1983. Parsing as deduction. In *Proc. of ACL*, pages 137–144.

S. M. Shieber, Y. Schabes, and F. C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, Jul.

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

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

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

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