# Algorithms for Syntax-Aware Statistical Machine Translation

I. Dan Melamed
Computer Science Department
New York University
New York, NY
10003-6806 U.S.A.
{lastname}@cs.nyu.edu

**Abstract**

All of the non-trivial algorithms that are necessary for building and applying a rudimentary syntax-aware statistical machine translation system are generalized parsers. This paper extends the "translation by parsing" architecture by adding two components that are invariably used by state-of-the-art statistical machine translation systems. First, the paper shows how a generic syntax-aware translation algorithm can accommodate independently estimated $n$-gram language models. Second, the paper introduces an automatic MT evaluation method that accounts for meaning-preserving syntactic alternations. The core algorithms in both extensions are, again, generalized parsers.

## 1 Introduction

A generalized parser is an algorithm whose input is a tuple of strings and whose output is a multi-dimensional tree covering the strings. All of the non-trivial algorithms that are necessary for building and applying a rudimentary syntax-aware statistical machine translation system are generalized parsers (Melamed and Wang, 2004). The benefit of this insight is that optimizations and other improvements invented for one of these algorithms can often be applied to all of them. Moreover, all of these generalized parsers are special cases of one particular generalized parser. Therefore, with proper software design, an entire MT system can be trained and run using the implementation of just one algorithm.

This paper extends the "translation by parsing" architecture by adding two components that are invariably used by state-of-the-art statistical machine translation (SMT) systems. First, in Section 4, the paper shows how a generic syntax-aware translation algorithm can accommodate independently estimated $n$-gram language models. Such an extension combines the explanatory power of structured translation models with the robustness afforded by abundant monolingual training data (Brown et al., 1993). Second, in Section 5, the paper introduces an automatic MT evaluation method that accounts for meaning-preserving syntactic alternations. The method involves a monolingual synchronous grammar. The algorithms for inducing the synchronous grammar and for applying it to MT evaluation are generalized parsers. Before describing these innovations, the

paper briefly introduces a particular grammar formalism that will be used as a running example, and reviews the translation algorithm of Melamed and Wang (2004).

## 2 Multitext Grammars and Multitrees

Many syntactic parsers are based on a grammar formalism, such as CFG. Similarly, many syntax-aware translation algorithms are based on a synchronous grammar formalism, although the grammar is often implicit. The algorithms in this paper can be adapted to any synchronous grammar formalism. The running example in this paper shall be multitext grammar (MTG), which is a generalization of context-free grammar to the synchronous case. For simplicity, we shall focus our attention on 2-dimensional MTGs (2-MTGs) in Generalized Chomsky Normal Form (GCNF) (Melamed et al., 2004). This normal form allows simpler algorithm descriptions than the normal forms used by Wu (1997) and Melamed (2003).

In GCNF, every production is either a terminal production or a nonterminal production. A nonterminal production in a 2-MTG might look like this:

$$\begin{array}{lll} X & [1,2] & \begin{pmatrix} \text{A} & \text{B} \\ () & C \\ \text{D(2)} & \text{E} \end{pmatrix} \\ Y \Rightarrow\bowtie & [1] & \\ Z & [1,2,1] & \end{array} \tag{1}$$

There are nonterminals on the left-hand side (LHS) and in parentheses on the right-hand side (RHS). The nonterminals on the RHS are written in columns called **links**. Links express translational equivalence. Some nonterminals might have no translation in some components, indicated by (), as in the 2nd row. Each row of the production describes rewriting in a different component text of a multitext. In each row, a **role template** describes the relative order and contiguity of the RHS nonterminals. E.g., in the top row, [1,2] indicates that the first nonterminal (A) precedes the second (B). In the bottom row, [1,2,1] indicates that the first nonterminal both precedes and follows the second, i.e. D is discontinuous. All of the role templates in a production rule constitute the **role template vector (RTV)**. Discontinuous nonterminals are annotated with the number of their contiguous segments, as in $D(2)$. The $\bowtie$ ("join") operator rearranges the nonterminals in each component according to their role template. Terminal productions have exactly one "active" component, in which there is exactly one terminal on the RHS. The other components are inactive, indicated by empty parentheses. E.g.,

$$\begin{array}{l} () \\ Y \end{array} \Rightarrow \begin{array}{l} () \\ a \end{array} \tag{2}$$

The semantics of $\Rightarrow$ are the usual semantics of rewriting systems, i.e., that the expression on the LHS can be rewritten as the expression on the RHS. However, all the nonterminals in the same link must be rewritten simultaneously. In this manner, MTGs generate tuples of parse trees that are isomorphic up to reordering of sibling nodes and deletion. Figure 1 shows two representations of a tree that might be generated by an MTG in GCNF for the imperative sentence pair (*Drink some water / Vodi vipiy* ). The tree exhibits both deletion and inversion in translation. We shall refer to such multidimensional trees as **multitrees**.

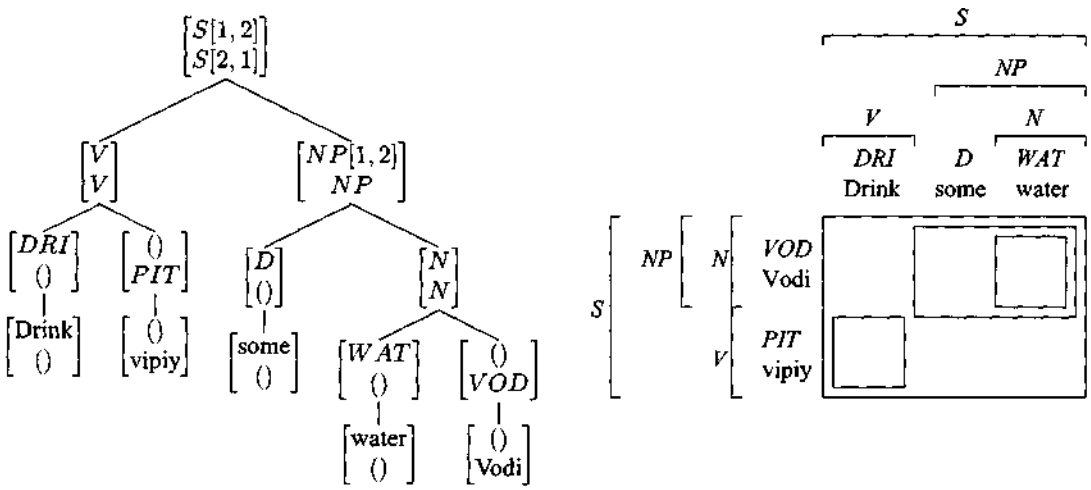


Figure 1: Two representations of a tree generated by a 2-MTG in English and (transliterated) Russian. Left: Every internal node is annotated with the linear order of its children, in every component where there are two children. Right: Rectangles are 2D constituents.

## 3 Translation Based on Synchronous Grammars

We shall describe translation algorithms in terms of their underlying logic. This descriptive device has been used extensively in research on parsing algorithms (Goodman, 1999, e.g.). Logical descriptions of parsing algorithms are nondeterministic: they do not indicate the order in which a parser should add items to its chart, and that is precisely their advantage. Design and analysis of the parser actions that can be made is easier when it is abstracted away from design and analysis of the *order* in which they should be made. A deterministic parsing strategy can always be chosen later, to suit the application. We shall apply the same kind of abstraction to the design and analysis of translation algorithms. Parsing and translation logics also abstract away from semiring specifications, though we shall focus on probabilistic semirings in this paper.

Our translators deal in items that are stored in charts, just like chart parsers. The translation process begins with an empty chart, and ends when a certain pre-specified goal item appears in the chart. We shall write declarative inference rules to indicate what items can be placed in the chart under what conditions. Inference rules describe the translator's possible actions. Each action is a transition from one state to another. We shall express inference rules as sequents: $\frac{Y_1,\ldots,Y_k}{X}$ means that the **consequent** $X$ can be deduced from the **antecedents** $Y_1,\ldots,Y_k$. In other words, if a translator's state includes $Y_1,\ldots,Y_k$ then it can transition to a state that also includes $X$. Each antecedent term refers either to a grammatical constraint, such as a production rule, or to an item. Consequents are always items. An item that appears in an inference rule stands for the proposition that the item is part of the parser's state. In less abstract terms, this means that the item is in the chart. Under probabilistic semirings, inference rule terms carry probabilities. The probability

associated with grammar terms is the probability assigned to their arguments by the grammar. The probability an item is the probability associated with that item in the chart.

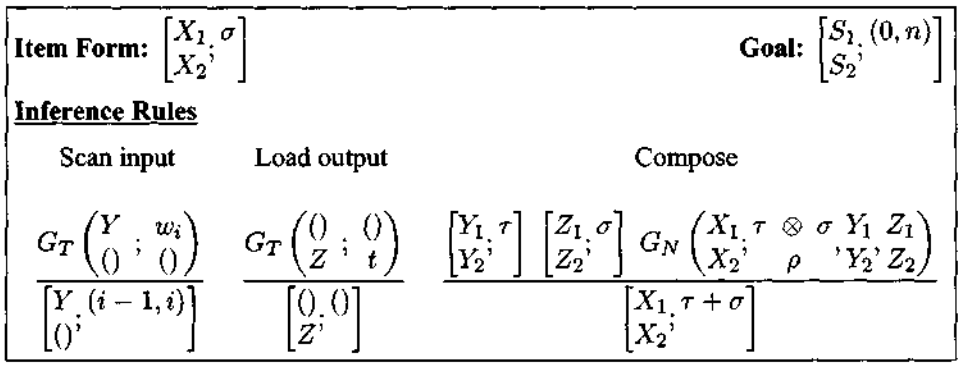


Figure 2: Logic for Translator D2C ("D2" for two-dimensional; "C" for CKY)

Figure 2 shows the logic for Translator D2C. Under a probabilistic semiring, the input to the translator is a Probabilistic MTG (PMTG)[1] and a text of length $n$. The output is a 2D multitree that has the input text on one dimension and its translation on the other, along with their syntactic structures. We shall refer to these as the **input component/dimension** and the **output component/dimension**. In practice, translations are usually desired as strings, rather than as trees. The intended ordering of the terminals in the output dimension can be assembled by a trivial post-process involving a single walk around the finished multitree.

The item form of the translator includes two linked nonterminals and a d-span[2] for the input component. It must be a d-span, rather than an ordinary span, because Translator D2C needs to know all the boundaries of each item, not just the outermost boundaries. No d-span is necessary for the output component, because it is not necessary to constrain absolute word positions in the output (see below regarding relative positions). One (but not both) of an item's components can be inactive, denoted by empty parentheses (). The d-span in an inactive component is always empty, denoted by ().

Translator D2C begins with an empty chart. The only inferences that can fire in this state are those with no antecedent items, though they can have antecedent grammar terms. In Translator D2C, under a probabilistic semiring, $G_T(\chi;\nu)$ is the probability that the grammar assigns to the terminal production $\chi \Rightarrow \nu$. The probability of the consequent of a Scan or a Load inference is set to this value. A Scan inference can fire for the $i$th input word $w_i$ if that word appears in the input component on the RHS of a terminal production in the grammar. If a word appears on the RHS of the input component of multiple productions (with different LHSs), then multiple Scan inferences can fire for that word. The d-span of each item inferred by a Scan inference has the form $(i-1,i)$ because such items always span one word, so the distance between the item's boundaries is always one. The labels and d-spans of Scan consequents are always empty in the output component. In

[1] A PMTG is an MTG with probabilities over production rules, such that the probabilities of all rules with the same LHS sum to 1. This paper considers only the probabilistic case.

[2] A discontinuous span (d-span) is a list of non-overlapping contiguous spans (Melamed and Wang, 2004).

Load inferences, the words come from the grammar rather than from the input, and the labels and d-spans are empty in the input component.

Translator D2C spends most of its time composing pairs of items into larger items. The parser can compose two items whenever they satisfy both of the following constraints:

- *Immediate Dominance (ID).* Their nonterminal links must appear on the RHS of a production rule in the grammar.
- *Linear Precedence (LP).* The relative order and contiguity of the two items must be consistent with the RTV in that production rule. This constraint is expressed by the role templates with the help of the d-span operators: $+$ is the d-span concatenation operator; $\otimes$ computes the role template that represents the contiguity and relative order of intervals in two d-spans (Melamed and Wang, 2004). Both operators apply componentwise to vectors of d-spans.

Both constraints are enforced by the $G_N$ term of the Compose inference rule. In Translator D2C, under a probabilistic semiring, the function $G_N(\chi; \rho, \nu, \kappa)$ represents the probability that the grammar assigns to the nonterminal production $\chi \Rightarrow \bowtie \rho(\nu\kappa)$.

Goodman (1999) showed how a nondeterministic parser can be combined with various semirings to compute different kinds of information about the input. Depending on the chosen semiring, a parser can compute the most probable derivation and/or its probability, the $k$ most probable derivations and/or their total probability, all possible derivations and/or their total probability, the number of possible derivations, etc. All the parsing semirings catalogued by Goodman apply the same way to translation logics. For example, under the Viterbi semiring, Translator D2C can compute the probability of the most probable multitree that covers the input. The computation proceeds bottom-up, computing the probabilities of larger chart items from their smaller antecedents. The probability of each item $\alpha$, including the Goal item, is computed as follows:

$$\Pr(\alpha) = \max_{\alpha_1,\ldots,\alpha_k,\ s.t.\ \frac{\alpha_1,\ \ldots,\ \alpha_k}{\alpha}} \prod_{i=1}^{k} \Pr(\alpha_i) \tag{3}$$

The $\alpha_i$ above can be any term — either an item or a grammar term ($G$). The equation says that the value of any chart item is the maximum, over all inferences that derive that item, of the product of the probabilities of the antecedents of the inference.

The translator terminates if it infers the Goal item, i.e. an item whose labels are the grammar's start symbol, and whose d-span consists of one interval spanning the entire input. Under the Viterbi semiring, the probability of the Goal item is the probability of the multitree.

The Viterbi-derivation semiring would be used most often in practice. In addition to the probabilities of chart items, this semiring keeps track of the antecedents of each item, so that the multitree can be later recovered from its root (the Goal item). Given a PMTG, Translator D2C can use this semiring to find the single most probable two-dimensional multitree that covers the input (rather than just its probability). The multitree inferred by the translator will have the words of both the input and the output components in its leaves. For example, given a suitable grammar and the input *Drink some water*, Translator D2C can make the 9 inferences in Figure 3, one per internal node, to

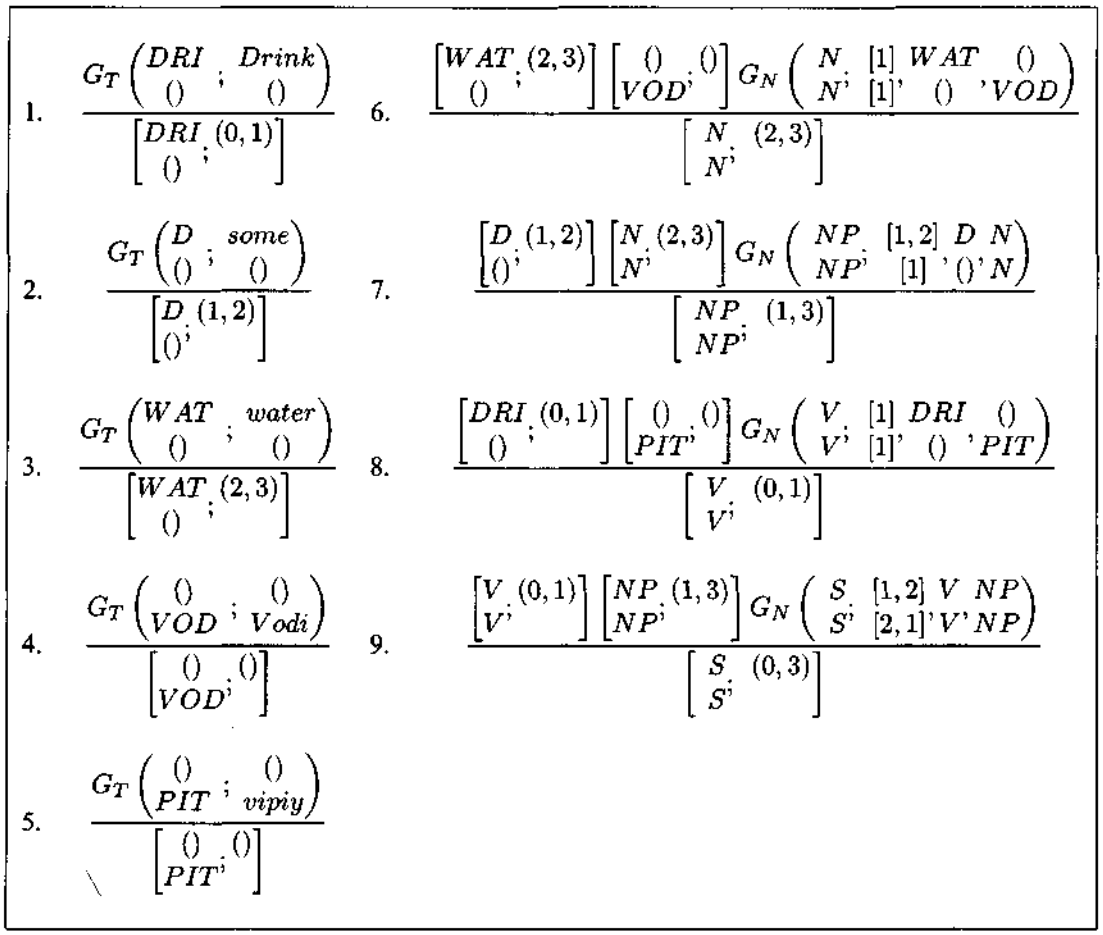


Figure 3: Set of inferences for the tree in Figure 1.

infer the multitree in Figure 1. The grammar would have to assign non-zero probability to all the production rules represented by the $G_T$ and $G_N$ terms in the inferences.

At least some of the translator's inferences compose Scan consequents with Load consequents, i.e., items that cover only input words and only output words, respectively. Such an inference is, de facto, a translation. Translation can happen the same way for multi-word items, which are themselves consequents of other Compose inferences. The possible translations are determined by consulting the grammar. Thus, in addition to its usual function of evaluating syntactic structures, the grammar simultaneously functions as a translation model.

Translator D2C is conceptually simpler than the translators of Wu (1997) and Alshawi et al. (2000) because it uses only one kind of item, and never composes terminals. The Scan and Compose inference rules of Translator D2C are generalizations of the same inference rules in ordinary CKY

parsers. The Load inference rule can be viewed as a generalization of the ordinary CKY Scan inference, where the antecedent terminal is not constrained by the input.

# 4 Grammar Mixtures for Translation

The usual way that a parser evaluates grammar terms is by looking up their arguments in its grammar. However, any complete function over immediate derivations can be used for this purpose, as long as its range is compatible with the chosen semiring. In particular, if the semiring is probabilistic, then the $G$ terms can be evaluated by any probability distribution function, such as a mixture of stochastic grammars.

An important application of this idea is in combining different kinds of training data to improve a statistical model. For example, an SMT system can benefit from mixing the predictions of its translation model with those of a more reliable monolingual language model. The classic way to mix a translation model with a language model is the so-called noisy-channel framework, based on Bayes's Rule. Bayes's Rule applies to conditional models, but probabilistic synchronous grammars are joint models. So, instead of using Bayes's Rule, we use the chain rule:

$$\begin{aligned} G_N \begin{pmatrix} X_1, \rho_1 & Y_1 & Z_1 \\ X_2, \rho_2, & Y_2, & Z_2 \end{pmatrix} &= \Pr(\rho_1, \rho_2, Y_1, Y_2, Z_1, Z_2 \,|\, X_1, X_2) \\ &= \Pr(\rho_2, Y_2, Z_2, |X_1, X_2) \times \Pr(\rho_1, Y_1, Z_1, |\rho_2, Y_2, Z_2, X_1, X_2) \end{aligned} \quad (4)$$

In Equation 4, the first factor can be computed by marginalization of a PMTG. The second factor can be computed by conditionalization of a PMTG, or by direct discriminative estimation (Och and Ney, 2002).

The first factor in Equation 4 is analogous to the language model in the traditional noisy channel decomposition (Brown et al., 1993, Equation 2). We can replace this factor by a language model in the chain-rule based decomposition too. The replacement is straightforward when the language model is a grammar such as a PCFG, which can evaluate the marginalized factors by direct reference to its production rule probabilities (Wu and Wong, 1998; Charniak et al., 2003). Unfortunately, such models are not available for most languages. It is much easier to induce an $n$-gram language model from unannotated text.

The difficulty with using $n$-gram language models for this purpose is that their standard definition does not refer to nonterminals, let alone to discontinuous multidimensional constituents. In order to incorporate $n$-gram language models into multitree-based translators in a principled way, we shall encapsulate them in $n$-gram grammars. **$n$-gram grammars** range over ordinary word strings and also over discontinuous word strings. Before developing a way to evaluate $n$-gram grammar probabilities, we introduce some machinery to describe and manipulate discontinuous strings.

## 4.1 Discontinuous Strings

A **discontinuous string** (or **d-string**) is a sequence of zero or more strings $[s_1; \ldots; s_c]$. We will find it convenient to represent each d-string by the list of its string extremum pairs. The **string**

**extremum pair (SEP)** of a string is the string's first and last elements. The **SEP list** of a d-string $\phi$ is a list of $c$ symbol pairs $\langle l_1, r_1; \ldots; l_c, r_c \rangle$, where

- $l_i, r_i$ is the SEP of the $i$th string in $\phi$, and
- $c$ is the number of strings in $\phi$.

For example, the SEP list of $[New; York]$ is $\langle N, w; Y, k \rangle$.

To compose pairs of SEP lists, we introduce the % operator.[3] This operator takes a role template as a third operand. Wherever the role template lists a contiguous pair of roles (i.e. separated by a comma rather than a semicolon), two SEPs are fused into one. E.g.,

$$\%([1,2;2,1], \langle a, d; e, h \rangle, \langle i, l; m, p \rangle) = \langle a, l; m, h \rangle \tag{5}$$

In the above example, the role template begins with the contiguous pair of roles [1,2]. So, the SEP $(a, d)$ from the first d-string is fused with the SEP $(i, l)$ from the second d-string to create the SEP $(a, l)$ in the result.

At every point of fusion, the right extremum of one SEP and the left extremum of another SEP disappear. We call such pairs of disappearing extrema **fused pairs**. A set of fused pairs participates in every d-string composition. The $\nabla$ operator[4] computes the fused pairs. E.g.,

$$\nabla([1,2;2,1], \langle a, d; e, h \rangle, \langle i, l; m, p \rangle) = \langle d, i; p, e \rangle \tag{6}$$

For any two SEP lists $X$ and $Y$, and for any role template $\rho$, every extremum that appears in $X$ or in $Y$ will appear either in the result of $\%(\rho, X, Y)$ or in the result of $\nabla(\rho, X, Y)$.

## 4.2 A Bigram Grammar (BG)

For simplicity of exposition, we limit our attention to bigram language models and bigram grammars (BGs). Models over longer $n$-grams can be employed analogously. A bigram language model can be viewed as a bilexical regular grammar. This grammar's terminal set consists of the vocabulary, as usual, plus the special terminal $\Diamond$, which acts as a start-of-string marker. The nonterminal set is based on only one (dummy) nonterminal label $\delta$, which is lexicalized by each of the terminals, including $\Diamond$. Thus, the number of nonterminal symbols in the grammar is the same as the number of terminal symbols. $\delta[\Diamond]$ is the special start symbol. Each production of the grammar is in one of the following two forms, where $t$ and $u$ range over terminals:

$$\delta[t] \Rightarrow t\delta[u] \tag{7}$$

$$\delta[t] \Rightarrow t \tag{8}$$

$\Rightarrow$ has the usual semantics of rewriting here. For example, the bigram grammar would derive the string "abc" as $\delta[\Diamond] \Rightarrow \Diamond\delta[a] \Rightarrow \Diamond a\delta[b] \Rightarrow \Diamond ab\delta[c] \Rightarrow \Diamond abc$. We posit that the derivation process deterministically removes the initial $\Diamond$ just before termination.

[3] Mnemonic: what remains?

[4] Mnemonic: what changed?

As usual, we can assign probabilities to the productions of the grammar, so that the probabilities of all productions with the same LHS sum to 1.0. Let $BG()$ denote the BG's probability function, and let $\stackrel{*}{\Rightarrow}$ be the reflexive and transitive closure of $\Rightarrow$, as usual. Then the probability of generating a unigram $t$ is

$$BG(t) = \Pr(\delta[\Diamond] \stackrel{*}{\Rightarrow} \Diamond t) = \Pr(\Diamond\delta[t]|\delta[\Diamond]) \times \Pr(t|\delta[t]). \tag{9}$$

The probability of an $n$-gram $t_1 \ldots t_n$ is

$$BG(t_1 \ldots t_n) = \Pr(\delta[\Diamond] \stackrel{*}{\Rightarrow} \Diamond t_1 \ldots t_n) = \tag{10}$$

$$= \Pr(\Diamond\delta[t_1]|\delta[\Diamond]) \times \Pr(t_n|\delta[t_n]) \times \prod_{i=2}^{n} \Pr(t_{i-1}\delta[t_i]|\delta[t_{i-1}]). \tag{11}$$

To support a stochastic CKY parser, a BG must be able to evaluate the probability of a string given the probabilities of its substrings. A PCFG can directly evaluate the probability of composing any two constituents, but a BG carries no explicit information about the probability of string composition, let alone d-string composition. At first glance, the BG might seem unable to evaluate such probabilities efficiently. However, algebraic manipulation of the terms of Equation 11 shows that

$$BG(y_1 \ldots y_n z_1 \ldots z_m) = BG(y_1 \ldots y_n) \times BG(z_1 \ldots z_m) \times \frac{\Pr(y_n\delta[z_1]|\delta[y_n])}{\Pr(y_n|\delta[y_n]) \times \Pr(\Diamond\delta[z_1]|\delta[\Diamond])} \tag{12}$$

Given the probabilities of two strings, a BG can compute the probability of their concatenation based solely on the last word of the first string and the first word of the second string. The remaining words in both strings do not affect the result. Thus, Equation 12 can be evaluated in constant time, regardless of the lengths of the strings being composed. This property enables the unambiguous expression of Equation 12 in terms of SEP lists and the % operator:

$$BG(\%([1,2], \langle y_1, y_n\rangle, \langle z_1, z_m\rangle)) =$$

$$= BG(\langle y_1, y_n\rangle) \times BG(\langle z_1, z_m\rangle) \times \frac{\Pr(y_n\delta[z_1]|\delta[y_n])}{\Pr(y_n|\delta[y_n]) \times \Pr(\Diamond\delta[z_1]|\delta[\Diamond])} \tag{13}$$

Equation 13 admits only one fused pair. To compose arbitrary d-strings, represented by SEP lists $\lambda$ and $\mu$, we generalize it for arbitrary role templates $\rho$ using the $\nabla$ operator:

$$BG(\%(\rho, \lambda, \mu)) = BG(\lambda) \times BG(\mu) \times \prod_{(y,z)\in\nabla(\rho,\lambda,\mu)} \frac{\Pr(y\delta[z]|\delta[y])}{\Pr(y|\delta[y]) \times \Pr(\Diamond\delta[z]|\delta[\Diamond])} \tag{14}$$

Equation 14 accounts for all the fused pairs in the SEP lists $\lambda$ and $\mu$.[5] If the result of the fusion is a continuous string, then its probability will have the same value as if it were computed all at once by Equation 11. The above machinery enables an inference algorithm that can compute the same

[5]We have ignored the BG's inability to generate discontinuous strings. To justify this oversight, we can view a BG as part of a larger class of models that allocate some constant probability $c$ to discontinuities. Since the parser's goal item has no discontinuities, these factors always cancel out by the time the parser reaches the goal. The exact value of $c$ is therefore immaterial. A BG is just the special case where $c = 0$.

quantity by composing d-strings. The computational complexity of evaluating Equation 14 is linear in the number of fused pairs, so the BG can compute the probability of any string in time that is linear in the string's length.

### 4.3 A Mixed-Grammar Translator

In order for a parser to be effectively guided by a grammar, the parser's items must store all the information that the grammar needs to evaluate derivations. A parser guided by a mixture of grammars needs to store all the information needed by every grammar in the mixture. The items of Translator D2C store nonterminal labels and d-spans. Nothing more is required to look up MTG production rules. In order for a translator to be guided by a mixture of an MTG and a BG, its items must also store SEP lists.

**Item Form:** $\left[\begin{matrix} X_1 \\ X_2[\lambda] \end{matrix}; \sigma\right]$ **Goal:** $\left[\begin{matrix} S_1 \\ S_2[\langle w_1, w_n \rangle] \end{matrix}; (0,n)\right]$

**Inference Rules**

Scan input

$$\frac{G_T\begin{pmatrix} Y & ; & w_i \\ () & ; & () \end{pmatrix}}{\left[\begin{matrix} Y \\ () \end{matrix}; (i-1,i)\right]}$$

Load output

$$\frac{G_T\begin{pmatrix} () & ; & () \\ Z & ; & t \end{pmatrix}}{\left[\begin{matrix} () \\ Z[\langle t,t \rangle] \end{matrix}; ()\right]}$$

Compose

$$\frac{\left[\begin{matrix} Y_1 \\ Y_2[\lambda] \end{matrix}; \tau\right] \left[\begin{matrix} Z_1 \\ Z_2[\mu] \end{matrix}; \sigma\right] G_N\begin{pmatrix} X_1 & \tau \otimes \sigma & Y_1 & Z_1 \\ X_2 & \rho & Y_2[\lambda] & Z_2[\mu] \end{pmatrix}}{\left[\begin{matrix} X_1 \\ X_2[\%(\rho,\lambda,\mu)] \end{matrix}; \tau+\sigma\right]}$$

Figure 4: Translator D2CM

Figure 4 contains Translator D2CM, which generalizes Translator D2C in two ways. First, in addition to nonterminal labels and d-spans, its items also store SEP lists.[6] We write SEP lists as subscripts of nonterminal labels, because SEP lists are a kind of lexicalization. The SEP lists are relevant and recorded only in the output dimensions. Load consequents start with a degenerate SEP list containing only one SEP, in which both extrema refer to the same word. The Compose inference constructs a new SEP for its consequent by fusing the SEPs in the antecedent items. The SEP list of the Goal item is constrained to contain at most one SEP, because the output must always be contiguous. Second, Translator D2CM's $G_N$ term includes the antecedent SEP lists. In this manner, the BG can evaluate derivations in tandem with the translation model.[7] The value of each Load and Compose consequent depends on both grammars in the mixture.

We are now ready to define the translator's grammar terms for this purpose. For Scan inferences, $G_T$ is computed as before, using only the PMTG. For Load inferences, we make the simplifying assumption that a bigram grammar cannot distinguish nonterminal labels. Certainly this is the case

[6] It seems that the less general translator of Wu (1996) does something similar, but he does not give full details.

[7] SEP lists are not necessary in $G_T$ terms because the degenerate SEP is fully specified by the terminal on the RHS.

for a standard $n$-gram language model, and so we let

$$G_T \begin{pmatrix} () \\ Z \end{pmatrix} ; \begin{pmatrix} () \\ t \end{pmatrix} = BG(t). \quad (15)$$

For Compose inferences, we refine Equation 4 as follows:

$$G_N \begin{pmatrix} X_1, \rho_1 & Y_1 & Z_1 \\ X_2, \rho_2, & Y_2[\lambda], & Z_2[\mu] \end{pmatrix} = BG(\%(\rho_2, \lambda, \mu)) \times TM(\rho_1, Y_1, Z_1, |\rho_2, Y_2, Z_2, X_1, X_2) \quad (16)$$

where $TM$ is the conditional translation model probability and $BG$ is the bigram grammar probability. Again, the BG ignores nonterminals. The mixture of the BG and the TM need not be uniform. We can weight the TM more or less heavily than the BG, to reflect the relative reliability of these two knowledge sources (Och and Ney, 2002).

Using the BG to help evaluate every inference can greatly accelerate the translation process, in comparison to algorithms that apply a language model only after a complete multitree has been inferred. For example, the decoder of Yamada and Knight (2002) first builds a forest of multitrees, and then searches for the single most probable output in the forest using a language model. If only a single translation is desired, then there is no need to compute a parse forest. Moreover, if only the single most probable translation is desired, then A* search algorithms can be used to prune unlikely constituents (Klein and Manning, 2003). A BG can provide sharper probability estimates for chart items, making the A* heuristics more efficient. A BG can be integrated the same way with monolingual structured language models for speech recognition (Charniak, 2001).

Translator D2CM is a parser guided by an MTG and a bigram grammar. The literature describes several less general translators that employ language models (Wu, 1996; Alshawi et al., 2000; Yamada and Knight, 2002). To the best of our knowledge, Translator D2CM is the first translator to mix an independent $n$-gram language model with a synchronous grammar formalism that admits discontinuities. Melamed (2003) has shown that discontinuities are practically unavoidable in broad-coverage syntax-aware machine translation.

## 5 Translation Evaluation by Parsing

More sophisticated MT systems will require more sophisticated MT evaluation methods. For example, suppose that the reference translation is (R) below, and that two MT systems output the translations (T1) and (T2). All of the currently popular automatic evaluation methods would incorrectly assign a higher score to (T2) than to (T1), because (T2) has a longer matching $n$-gram with (R).

(R) Pat asked Sandy on Friday about the man from Oslo.

(T1) On Friday, Pat asked Sandy about the man from Oslo.

(T2) Pat from Oslo asked Sandy on Friday about the man.

The problem is that the similarity functions used by string-based evaluation methods offer only a crude approximation to syntactic similarity. Methods that measure the grammaticality of translations independently of a reference translation (e.g., (Rajman and Hartley, 2001)) are also incapable of making the desired distinctions — (T1) and (T2) are equally grammatical.

In order to correctly evaluate examples like the one above, an evaluation method needs a catalogue of the syntactic alternations that preserve the meaning of an utterance. Synchronous grammars offer a perspicuous way to describe such alternations. For example, the production

$$\begin{matrix}\text{NP} \\ \text{NP}\end{matrix} \Rightarrow \begin{matrix}[1,2,3] \\ [1,3,2]\end{matrix} \begin{pmatrix}\text{NN PP}_1\ \text{PP}_2 \\ \text{NN PP}_1\ \text{PP}_2\end{pmatrix} \tag{17}$$

could be included, to allow prepositional phrases to switch places. However, the relative order of determiners and adjectives in English noun phrases is strict, so the production

$$\begin{matrix}\text{NP} \\ \text{NP}\end{matrix} \Rightarrow \begin{matrix}[1,2,3] \\ [2,1,3]\end{matrix} \begin{pmatrix}\text{Det Adj NN} \\ \text{Det Adj NN}\end{pmatrix} \tag{18}$$

would not be included. The grammar would also include productions that have identical components in both dimensions, such as terminal productions that have the same word in both parts of the RHS.

Given such a grammar, a reference translation $R$, and an MT system output $T$, a synchronous parser can attempt to find a multitree covering the bitext $(R, T)$ under $G$. If the parser succeeds, then, according to the grammar, $T$ is a valid translation (actually, paraphrase) with respect to $R$. If the parser fails, then $T$ is not an acceptable paraphrase of $R$, either because it does not mean the same thing or because it is simply ungrammatical.

There are two practical problems with this approach. First, it is usually desirable to obtain a numerical grade of translation quality, rather than just a boolean indicator of acceptability. Second, it would probably be infeasible, or at least unreliable, to compile all the valid syntactic alternations manually. Both of these problems can be solved empirically. The Linguistic Data Consortium has recently published several "multiple-translation" corpora. These are corpora containing multiple independent translations of a set of source documents, aligned at the sentence level. Each set of independent translations can be viewed as mutual paraphrases (Pang et al., 2003). We can estimate a monolingual PMTG[8] from these sets of parallel sentences (Melamed and Wang, 2004). Given such a PMTG, a *probabilistic* synchronous parser can return the *probability* that a translation is valid with respect to a reference. Different translations, and the different MT systems that output them, can be compared on these scores.

As with ordinary probabilistic grammars, different estimation strategies are possible for PMTGs. The quality of a PMTG can be measured in terms of the probability that it assigns to previously unseen multitext. Thus, the problem of PMTG parameter estimation can be viewed as synchronous language modeling, which we shall also call **multitext modeling**, for brevity. To solve this problem, we can adopt all the methods of ordinary language modeling. In particular, to avoid overfitting, the PMTG parameters should be smoothed. For example, a non-zero probability should be allocated to all terminal productions that have the same word in both components of the RHS.

[8] I.e., a PMTG that generates multitexts that have the same language in all components.

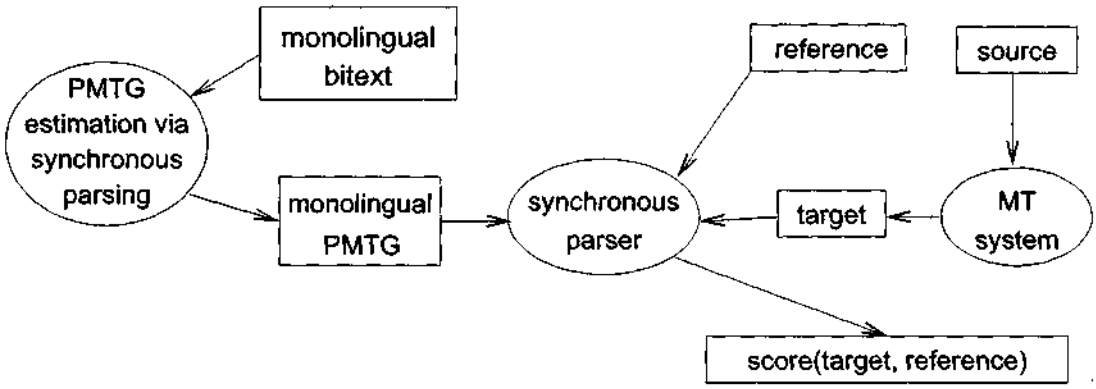


Figure 5: Data-flow diagram for a rudimentary MT evaluation system based on synchronous parsing.

Figure 5 illustrates how synchronous parsing can drive the creation and application of MT evaluation systems. MT evaluation by means of a monolingual PMTG has two advantages over most of the currently popular methods. First, this method can be sensitive to meaning-preserving syntactic alternations. Second, the method itself can be objectively evaluated in terms of perplexity on held-out data, without expensive and unreliable human judgments. A temporary disadvantage of this approach, however, is that accurate multitext modeling is still an open research problem.

## 6 Conclusion

This paper has extended the "translation by parsing" architecture by adding two components that are invariably used by state-of-the-art statistical machine translation (SMT) systems. First, the paper showed how a generic syntax-aware translation algorithm can accommodate independently estimated $n$-gram language models. Such an extension combines the explanatory power of structured translation models with the robustness afforded by abundant monolingual training data. Second, the paper introduced an automatic MT evaluation method that accounts for meaning-preserving syntactic alternations. The method itself can be objectively evaluated without expensive and unreliable human judgments. With these two extensions, implementations of the "rudimentary" architecture described by Melamed and Wang (2004) might become less rudimentary and more competitive with the state of the art.

## Acknowledgments

This research was supported by an NSF CAREER Award and by an equipment grant from Sun Microsystems.

## References


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

Peter F. Brown, Stephen A. Della Pietra, Vincent Della J. Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In Elliott Macklovitch, editor, *Proceedings of MT Summit IX*, New Orleans, September. International Association for Machine Translation.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, Toulouse, July.

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

Dan Klein and Christopher D. Manning. 2003. A* parsing: Fast exact viterbi parse selection. In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada.

I. Dan Melamed and Wei Wang. 2004. Statistical machine translation by parsing. Technical Report 04-024, Proteus Project, New York University. `http://nlp.cs.nyu.edu/pubs/`.

I. Dan Melamed, Giorgio Satta, and Benjamin Wellington. 2004. Generalized multitext grammars. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Barcelona, Spain.

I. Dan Melamed. 2003. Multitext grammars and synchronous parsers. In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 158–165, Edmonton, Canada.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, July.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada.

Martin Rajman and Tony Hartley. 2001. Automatically predicting MT systems rankings compatible with fluency, adequacy or informativeness scores. In *Proceedings of the Workshop on Machine Translation Evaluation: "Who Did What To Whom"*, pages 29–34, Santiago de Compostela, Spain.

Dekai Wu and Hongsing Wong. 1998. Machine translation with a stochastic grammatical channel. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, Montreal, Canada, July.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 152–158, Santa Cruz, USA.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, September.

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 303–310, Philadelphia, July.