# Cube Summing, Approximate Inference with Non-Local Features, and Dynamic Programming without Semirings

**Kevin Gimpel** and **Noah A. Smith**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{kgimpel,nasmith}@cs.cmu.edu

## Abstract

We introduce *cube summing*, a technique that permits dynamic programming algorithms for summing over structures (like the forward and inside algorithms) to be extended with non-local features that violate the classical structural independence assumptions. It is inspired by cube pruning (Chiang, 2007; Huang and Chiang, 2007) in its computation of non-local features dynamically using scored $k$-best lists, but also maintains additional residual quantities used in calculating approximate marginals. When restricted to local features, cube summing reduces to a novel semiring (*$k$-best+residual*) that generalizes many of the semirings of Goodman (1999). When non-local features are included, cube summing does not reduce to any semiring, but is compatible with generic techniques for solving dynamic programming equations.

## 1 Introduction

Probabilistic NLP researchers frequently make independence assumptions to keep inference algorithms tractable. Doing so limits the *features* that are available to our models, requiring features to be structurally local. Yet many problems in NLP—machine translation, parsing, named-entity recognition, and others—have benefited from the addition of *non-local* features that break classical independence assumptions. Doing so has required algorithms for *approximate* inference.

Recently *cube pruning* (Chiang, 2007; Huang and Chiang, 2007) was proposed as a way to leverage existing dynamic programming algorithms that find optimal-scoring derivations or structures when only local features are involved. Cube pruning permits approximate decoding with non-local features, but leaves open the question of how the feature weights or probabilities are learned. Meanwhile, some learning algorithms, like maximum likelihood for conditional log-linear models (Lafferty et al., 2001), unsupervised models (Pereira and Schabes, 1992), and models with hidden variables (Koo and Collins, 2005; Wang et al., 2007; Blunsom et al., 2008), require *summing* over the scores of many structures to calculate marginals.

We first review the semiring-weighted logic programming view of dynamic programming algorithms (Shieber et al., 1995) and identify an intuitive property of a program called *proof locality* that follows from feature locality in the underlying probability model (§2). We then provide an analysis of cube pruning as an approximation to the intractable problem of exact optimization over structures with non-local features and show how the use of non-local features with $k$-best lists breaks certain semiring properties (§3). The primary contribution of this paper is a novel technique—*cube summing*—for approximate summing over discrete structures with non-local features, which we relate to cube pruning (§4). We discuss implementation (§5) and show that cube summing becomes exact and expressible as a semiring when restricted to local features; this semiring generalizes many commonly-used semirings in dynamic programming (§6).

## 2 Background

In this section, we discuss dynamic programming algorithms as semiring-weighted logic programs. We then review the definition of semirings and important examples. We discuss the relationship between locally-factored structure scores and proofs in logic programs.

### 2.1 Dynamic Programming

Many algorithms in NLP involve dynamic programming (e.g., the Viterbi, forward-backward,

*Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 318–326,
Athens, Greece, 30 March – 3 April 2009. 

probabilistic Earley's, and minimum edit distance algorithms). Dynamic programming (DP) involves solving certain kinds of recursive equations with shared substructure and a topological ordering of the variables.

Shieber et al. (1995) showed a connection between DP (specifically, as used in parsing) and *logic programming*, and Goodman (1999) augmented such logic programs with semiring weights, giving an algebraic explanation for the intuitive connections among classes of algorithms with the same logical structure. For example, in Goodman's framework, the forward algorithm and the Viterbi algorithm are comprised of the same logic program with different semirings. Goodman defined other semirings, including ones we will use here. This formal framework was the basis for the Dyna programming language, which permits a declarative specification of the logic program and compiles it into an efficient, agenda-based, bottom-up procedure (Eisner et al., 2005).

For our purposes, a DP consists of a set of recursive equations over a set of indexed variables. For example, the probabilistic CKY algorithm (run on sentence $w_1 w_2 ... w_n$) is written as

$$\begin{aligned} C_{X,i-1,i} &= p_{X \to w_i} \qquad (1) \\ C_{X,i,k} &= \max_{Y,Z \in \mathcal{N}; j \in \{i+1,\ldots,k-1\}} p_{X \to YZ} \times C_{Y,i,j} \times C_{Z,j,k} \\ goal &= C_{\mathbf{S},0,n} \end{aligned}$$

where $\mathcal{N}$ is the nonterminal set and $\mathbf{S} \in \mathcal{N}$ is the start symbol. Each $C_{X,i,j}$ variable corresponds to the chart value (probability of the most likely subtree) of an $X$-constituent spanning the substring $w_{i+1} ... w_j$. *goal* is a special variable of greatest interest, though solving for *goal* correctly may (in general, but not in this example) require solving for all the other values. We will use the term "index" to refer to the subscript values on variables ($X, i, j$ on $C_{X,i,j}$).

Where convenient, we will make use of Shieber et al.'s logic programming view of dynamic programming. In this view, each variable (e.g., $C_{X,i,j}$ in Eq. 1) corresponds to the value of a "theorem," the constants in the equations (e.g., $p_{X \to YZ}$ in Eq. 1) correspond to the values of "axioms," and the DP defines quantities corresponding to weighted "proofs" of the *goal* theorem (e.g., finding the maximum-valued proof, or aggregating proof values). The value of a proof is a combination of the values of the axioms it starts with. Semirings define these values and define two operators over them, called "aggregation" ($\max$ in Eq. 1) and "combination" ($\times$ in Eq. 1).

Goodman and Eisner et al. assumed that the values of the variables are in a semiring, and that the equations are defined solely in terms of the two semiring operations. We will often refer to the "probability" of a proof, by which we mean a non-negative $\mathbb{R}$-valued score defined by the semantics of the dynamic program variables; it may not be a normalized probability.

### 2.2 Semirings

A *semiring* is a tuple $\langle A, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$, in which $A$ is a set, $\oplus : A \times A \to A$ is the aggregation operation, $\otimes : A \times A \to A$ is the combination operation, $\mathbf{0}$ is the additive identity element ($\forall a \in A, a \oplus \mathbf{0} = a$), and $\mathbf{1}$ is the multiplicative identity element ($\forall a \in A, a \otimes \mathbf{1} = a$). A semiring requires $\oplus$ to be associative and commutative, and $\otimes$ to be associative and to distribute over $\oplus$. Finally, we require $a \otimes \mathbf{0} = \mathbf{0} \otimes a = \mathbf{0}$ for all $a \in A$.[1] Examples include the inside semiring, $\langle \mathbb{R}_{\geq 0}, +, \times, 0, 1 \rangle$, and the Viterbi semiring, $\langle \mathbb{R}_{\geq 0}, \max, \times, 0, 1 \rangle$. The former sums the probabilities of all proofs of each theorem. The latter (used in Eq. 1) calculates the probability of the most probable proof of each theorem. Two more examples follow.

**Viterbi proof semiring.** We typically need to recover the steps in the most probable proof in addition to its probability. This is often done using backpointers, but can also be accomplished by representing the most probable proof for each theorem in its entirety as part of the semiring value (Goodman, 1999). For generality, we define a proof as a string that is constructed from strings associated with axioms, but the particular form of a proof is problem-dependent. The "Viterbi proof" semiring includes the probability of the most probable proof and the proof itself. Letting $\mathcal{L} \subseteq \Sigma^*$ be the proof language on some symbol set $\Sigma$, this semiring is defined on the set $\mathbb{R}_{\geq 0} \times \mathcal{L}$ with $\mathbf{0}$ element $\langle 0, \epsilon \rangle$ and $\mathbf{1}$ element $\langle 1, \epsilon \rangle$. For two values $\langle u_1, U_1 \rangle$ and $\langle u_2, U_2 \rangle$, the aggregation operator returns $\langle \max(u_1, u_2), U_{\text{argmax}_{i \in \{1,2\}} u_i} \rangle$.

[1] When cycles are permitted, i.e., where the value of one variable depends on itself, infinite sums can be involved. We must ensure that these infinite sums are well defined under the semiring. So-called *complete semirings* satisfy additional conditions to handle infinite sums, but for simplicity we will restrict our attention to DPs that do not involve cycles.

| Semiring | $A$ | Aggregation ($\oplus$) | Combination ($\otimes$) | $\mathbf{0}$ | $\mathbf{1}$ |
|---|---|---|---|---|---|
| inside | $\mathbb{R}_{\geq 0}$ | $u_1 + u_2$ | $u_1 u_2$ | 0 | 1 |
| Viterbi | $\mathbb{R}_{\geq 0}$ | $\max(u_1, u_2)$ | $u_1 u_2$ | 0 | 1 |
| Viterbi proof | $\mathbb{R}_{\geq 0} \times \mathcal{L}$ | $\langle \max(u_1, u_2), U_{\text{argmax}_{i \in \{1,2\}} u_i} \rangle$ | $\langle u_1 u_2, U_1.U_2 \rangle$ | $\langle 0, \epsilon \rangle$ | $\langle 1, \epsilon \rangle$ |
| $k$-best proof | $(\mathbb{R}_{\geq 0} \times \mathcal{L})^{\leq k}$ | max-$k(\mathbf{u}_1 \cup \mathbf{u}_2)$ | max-$k(\mathbf{u}_1 \star \mathbf{u}_2)$ | $\emptyset$ | $\{\langle 1, \epsilon \rangle\}$ |

Table 1: Commonly used semirings. An element in the Viterbi proof semiring is denoted $\langle u_1, U_1 \rangle$, where $u_1$ is the probability of proof $U_1$. The max-$k$ function returns a sorted list of the top-$k$ proofs from a set. The $\star$ function performs a cross-product on two $k$-best proof lists (Eq. 2).

The combination operator returns $\langle u_1 u_2, U_1.U_2 \rangle$, where $U_1.U_2$ denotes the string concatenation of $U_1$ and $U_2$.[2]

**$k$-best proof semiring.** The "$k$-best proof" semiring computes the values and proof strings of the $k$ most-probable proofs for each theorem. The set is $(\mathbb{R}_{\geq 0} \times \mathcal{L})^{\leq k}$, i.e., sequences (up to length $k$) of sorted probability/proof pairs. The aggregation operator $\oplus$ uses max-$k$, which chooses the $k$ highest-scoring proofs from its argument (a set of scored proofs) and sorts them in decreasing order. To define the combination operator $\otimes$, we require a cross-product that pairs probabilities and proofs from two $k$-best lists. We call this $\star$, defined on two semiring values $\mathbf{u} = \langle \langle u_1, U_1 \rangle, ..., \langle u_k, U_k \rangle \rangle$ and $\mathbf{v} = \langle \langle v_1, V_1 \rangle, ..., \langle v_k, V_k \rangle \rangle$ by:

$$\mathbf{u} \star \mathbf{v} = \{ \langle u_i v_j, U_i.V_j \rangle \mid i, j \in \{1, ..., k\} \} \quad (2)$$

Then, $\mathbf{u} \otimes \mathbf{v} = \text{max-}k(\mathbf{u} \star \mathbf{v})$. This is similar to the $k$-best semiring defined by Goodman (1999).

These semirings are summarized in Table 1.

### 2.3 Features and Inference

Let $\mathcal{X}$ be the space of inputs to our logic program, i.e., $x \in \mathcal{X}$ is a set of axioms. Let $\mathcal{L}$ denote the proof language and let $\mathcal{Y} \subseteq \mathcal{L}$ denote the set of proof strings that constitute full proofs, i.e., proofs of the special *goal* theorem. We assume an exponential probabilistic model such that

$$p(y \mid x) \propto \prod_{m=1}^{M} \lambda_m^{h_m(x,y)} \quad (3)$$

where each $\lambda_m \geq 0$ is a parameter of the model and each $h_m$ is a *feature function*. There is a bijection between $\mathcal{Y}$ and the space of discrete structures that our model predicts.

Given such a model, DP is helpful for solving two kinds of inference problems. The first problem, *decoding*, is to find the highest scoring proof $\hat{y} \in \mathcal{Y}$ for a given input $x \in \mathcal{X}$:

$$\hat{y}(x) = \text{argmax}_{y \in \mathcal{Y}} \prod_{m=1}^{M} \lambda_m^{h_m(x,y)} \quad (4)$$

The second is the *summing* problem, which marginalizes the proof probabilities (without normalization):

$$s(x) = \sum_{y \in \mathcal{Y}} \prod_{m=1}^{M} \lambda_m^{h_m(x,y)} \quad (5)$$

As defined, the feature functions $h_m$ can depend on arbitrary parts of the input axiom set $x$ and the entire output proof $y$.

### 2.4 Proof and Feature Locality

An important characteristic of problems suited for DP is that the *global* calculation (i.e., the value of *goal*) depend only on *local* factored parts. In DP equations, this means that each equation connects a relatively small number of indexed variables related through a relatively small number of indices. In the logic programming formulation, it means that each step of the proof depends only on the theorems being used at that step, *not* the full proofs of those theorems. We call this property *proof locality*. In the statistical modeling view of Eq. 3, classical DP requires that the probability model make strong Markovian conditional independence assumptions (e.g., in HMMs, $S_{t-1} \perp S_{t+1} \mid S_t$); in exponential families over discrete structures, this corresponds to *feature locality*.

For a particular proof $y$ of *goal* consisting of $t$ intermediate theorems, we define a set of proof strings $\ell_i \in \mathcal{L}$ for $i \in \{1, ..., t\}$, where $\ell_i$ corresponds to the proof of the $i$th theorem.[3] We can break the computation of feature function $h_m$ into a summation over terms corresponding to each $\ell_i$:

$$h_m(x, y) = \sum_{i=1}^{t} f_m(x, \ell_i) \quad (6)$$

This is simply a way of noting that feature functions "fire" incrementally at specific points in the

[2] We assume for simplicity that the best proof will never be a tie among more than one proof. Goodman (1999) handles this situation more carefully, though our version is more likely to be used in practice for both the Viterbi proof and $k$-best proof semirings.

[3] The theorem indexing scheme might be based on a topological ordering given by the proof structure, but is not important for our purposes.

proof, normally at the first opportunity. *Any* feature function can be expressed this way. For local features, we can go farther; we define a function $top(\ell)$ that returns the proof string corresponding to the antecedents and consequent of the *last* inference step in $\ell$. Local features have the property:

$$h_m^{loc}(x, y) = \sum_{i=1}^{t} f_m(x, top(\ell_i)) \quad (7)$$

Local features only have access to the most recent deductive proof step (though they may "fire" repeatedly in the proof), while non-local features have access to the entire proof up to a given theorem. For both kinds of features, the "$f$" terms are used within the DP formulation. When taking an inference step to prove theorem $i$, the value $\prod_{m=1}^{M} \lambda_m^{f_m(x,\ell_i)}$ is combined into the calculation of that theorem's value, along with the values of the antecedents. Note that typically only a small number of $f_m$ are nonzero for theorem $i$.

When non-local $h_m$/$f_m$ that depend on arbitrary parts of the proof are involved, the decoding and summing inference problems are NP-hard (they instantiate probabilistic inference in a fully connected graphical model). Sometimes, it is possible to achieve proof locality by adding more indices to the DP variables (for example, consider modifying the bigram HMM Viterbi algorithm for trigram HMMs). This increases the number of variables and hence computational cost. In general, it leads to exponential-time inference in the worst case.

There have been many algorithms proposed for approximately solving instances of these decoding and summing problems with non-local features. Some stem from work on graphical models, including loopy belief propagation (Sutton and McCallum, 2004; Smith and Eisner, 2008), Gibbs sampling (Finkel et al., 2005), sequential Monte Carlo methods such as particle filtering (Levy et al., 2008), and variational inference (Jordan et al., 1999; MacKay, 1997; Kurihara and Sato, 2006). Also relevant are stacked learning (Cohen and Carvalho, 2005), interpretable as approximation of non-local feature values (Martins et al., 2008), and M-estimation (Smith et al., 2007), which allows training without inference. Several other approaches used frequently in NLP are approximate methods for decoding only. These include beam search (Lowerre, 1976), cube pruning, which we discuss in §3, integer linear programming (Roth and Yih, 2004), in which arbitrary features can act as constraints on $y$, and approximate solutions like McDonald and Pereira (2006), in which an exact solution to a related decoding problem is found and then modified to fit the problem of interest.

## 3 Approximate Decoding

Cube pruning (Chiang, 2007; Huang and Chiang, 2007) is an approximate technique for decoding (Eq. 4); it is used widely in machine translation. Given proof locality, it is essentially an efficient implementation of the $k$-best proof semiring. Cube pruning goes farther in that it permits non-local features to weigh in on the proof probabilities, at the expense of making the $k$-best operation approximate. We describe the two approximations cube pruning makes, then propose *cube decoding*, which removes the second approximation. Cube decoding cannot be represented as a semiring; we propose a more general algebraic structure that accommodates it.

### 3.1 Approximations in Cube Pruning

Cube pruning is an approximate solution to the decoding problem (Eq. 4) in two ways.

**Approximation 1: $k < \infty$.** Cube pruning uses a finite $k$ for the $k$-best lists stored in each value. If $k = \infty$, the algorithm performs exact decoding with non-local features (at obviously formidable expense in combinatorial problems).

**Approximation 2: lazy computation.** Cube pruning exploits the fact that $k < \infty$ to use lazy computation. When combining the $k$-best proof lists of $d$ theorems' values, cube pruning does not enumerate all $k^d$ proofs, apply non-local features to all of them, and then return the top $k$. Instead, cube pruning uses a more efficient but approximate solution that only calculates the non-local factors on $O(k)$ proofs to obtain the approximate top $k$. This trick is only approximate if non-local features are involved.

Approximation 2 makes it impossible to formulate cube pruning using separate aggregation and combination operations, as the use of lazy computation causes these two operations to effectively be performed simultaneously. To more directly relate our summing algorithm (§4) to cube pruning, we suggest a modified version of cube pruning that does not use lazy computation. We call this algorithm *cube decoding*. This algorithm can be written down in terms of separate aggregation

and combination operations, though we will show it is not a semiring.

### 3.2 Cube Decoding

We formally describe cube decoding, show that it does not instantiate a semiring, then describe a more general algebraic structure that it does instantiate.

Consider the set $\mathcal{G}$ of non-local feature functions that map $\mathcal{X} \times \mathcal{L} \to \mathbb{R}_{\geq 0}$.[4] Our definitions in §2.2 for the $k$-best proof semiring can be expanded to accommodate these functions within the semiring value. Recall that values in the $k$-best proof semiring fall in $A_k = (\mathbb{R}_{\geq 0} \times \mathcal{L})^{\leq k}$. For cube decoding, we use a different set $A_{cd}$ defined as

$$A_{cd} = \underbrace{(\mathbb{R}_{\geq 0} \times \mathcal{L})^{\leq k}}_{A_k} \times \mathcal{G} \times \{0, 1\}$$

where the binary variable indicates whether the value contains a $k$-best list (0, which we call an "ordinary" value) or a non-local feature function in $\mathcal{G}$ (1, which we call a "function" value). We denote a value $\mathbf{u} \in A_{cd}$ by

$$\mathbf{u} = \langle \underbrace{\langle \langle u_1, U_1 \rangle, \langle u_2, U_2 \rangle, ..., \langle u_k, U_k \rangle \rangle}_{\bar{\mathbf{u}}}, g_u, u_s \rangle$$

where each $u_i \in \mathbb{R}_{\geq 0}$ is a probability and each $U_i \in \mathcal{L}$ is a proof string.

We use $\oplus_k$ and $\otimes_k$ to denote the $k$-best proof semiring's operators, defined in §2.2. We let $g_0$ be such that $g_0(\ell)$ is undefined for all $\ell \in \mathcal{L}$. For two values $\mathbf{u} = \langle \bar{\mathbf{u}}, g_u, u_s \rangle, \mathbf{v} = \langle \bar{\mathbf{v}}, g_v, v_s \rangle \in A_{cd}$, cube decoding's aggregation operator is:

$$\mathbf{u} \oplus_{cd} \mathbf{v} = \langle \bar{\mathbf{u}} \oplus_k \bar{\mathbf{v}}, g_0, 0 \rangle \text{ if } \neg u_s \wedge \neg v_s \quad (8)$$

Under standard models, only ordinary values will be operands of $\oplus_{cd}$, so $\oplus_{cd}$ is undefined when $u_s \vee v_s$. We define the combination operator $\otimes_{cd}$:

$$\mathbf{u} \otimes_{cd} \mathbf{v} = \quad (9)$$
$$\begin{cases} \langle \bar{\mathbf{u}} \otimes_k \bar{\mathbf{v}}, g_0, 0 \rangle & \text{if } \neg u_s \wedge \neg v_s, \\ \langle \max\text{-}k(exec(g_v, \bar{\mathbf{u}})), g_0, 0 \rangle & \text{if } \neg u_s \wedge v_s, \\ \langle \max\text{-}k(exec(g_u, \bar{\mathbf{v}})), g_0, 0 \rangle & \text{if } u_s \wedge \neg v_s, \\ \langle \langle \rangle, \lambda z.(g_u(z) \times g_v(z)), 1 \rangle & \text{if } u_s \wedge v_s. \end{cases}$$

where $exec(g, \bar{\mathbf{u}})$ executes the function $g$ upon each proof in the proof list $\bar{\mathbf{u}}$, modifies the scores in place by multiplying in the function result, and returns the modified proof list:

$$\begin{aligned} g' &= \lambda \ell . g(x, \ell) \\ exec(g, \bar{\mathbf{u}}) &= \langle \langle u_1 g'(U_1), U_1 \rangle, \langle u_2 g'(U_2), U_2 \rangle, \\ &\quad ..., \langle u_k g'(U_k), U_k \rangle \rangle \end{aligned}$$

Here, $\max$-$k$ is simply used to re-sort the $k$-best proof list following function evaluation.

The semiring properties fail to hold when introducing non-local features in this way. In particular, $\otimes_{cd}$ is not associative when $1 < k < \infty$. For example, consider the probabilistic CKY algorithm as above, but using the cube decoding semiring with the non-local feature functions collectively known as "$NGramTree$" features (Huang, 2008) that score the string of terminals and nonterminals along the path from word $j$ to word $j+1$ when two constituents $C_{Y,i,j}$ and $C_{Z,j,k}$ are combined. The semiring value associated with such a feature is $\mathbf{u} = \langle \langle \rangle, NGramTree_\pi(), 1 \rangle$ (for a specific path $\pi$), and we rewrite Eq. 1 as follows (where ranges for summation are omitted for space):

$$C_{X,i,k} = \bigoplus_{cd} p_{X \to YZ} \otimes_{cd} C_{Y,i,j} \otimes_{cd} C_{Z,j,k} \otimes_{cd} \mathbf{u}$$

The combination operator is not associative since the following will give different answers:[5]

$$(p_{X \to YZ} \otimes_{cd} C_{Y,i,j}) \otimes_{cd} (C_{Z,j,k} \otimes_{cd} \mathbf{u}) \quad (10)$$
$$((p_{X \to YZ} \otimes_{cd} C_{Y,i,j}) \otimes_{cd} C_{Z,j,k}) \otimes_{cd} \mathbf{u} \quad (11)$$

In Eq. 10, the non-local feature function is executed on the $k$-best proof list for $Z$, while in Eq. 11, $NGramTree_\pi$ is called on the $k$-best proof list for the $X$ constructed from $Y$ and $Z$. Furthermore, neither of the above gives the desired result, since we actually wish to expand the full set of $k^2$ proofs of $X$ and then apply $NGramTree_\pi$ to each of them (or a higher-dimensional "cube" if more operands are present) before selecting the $k$-best. The binary operations above retain only the top $k$ proofs of $X$ in Eq. 11 before applying $NGramTree_\pi$ to each of them. We actually would like to redefine combination so that it can operate on *arbitrarily-sized sets* of values.

We can understand cube decoding through an algebraic structure with two operations $\oplus$ and $\otimes$, where $\otimes$ need not be associative and need not distribute over $\oplus$, and furthermore where $\oplus$ and $\otimes$ are

[4] In our setting, $g_m(x, \ell)$ will most commonly be defined as $\lambda_m^{f_m(x,\ell)}$ in the notation of §2.3. But functions in $\mathcal{G}$ could also be used to implement, e.g., hard constraints or other non-local score factors.

[5] Distributivity of combination over aggregation fails for related reasons. We omit a full discussion due to space.

defined on arbitrarily many operands. We will refer here to such a structure as a *generalized semiring*.[6] To define $\otimes_{cd}$ on a set of operands with $N'$ ordinary operands and $N$ function operands, we first compute the full $O(k^{N'})$ cross-product of the ordinary operands, then apply each of the $N$ functions from the remaining operands in turn upon the full $N'$-dimensional "cube," finally calling $\max$-$k$ on the result.

## 4 Cube Summing

We present an approximate solution to the summing problem when non-local features are involved, which we call *cube summing*. It is an extension of cube decoding, and so we will describe it as a generalized semiring. The key addition is to maintain in each value, in addition to the $k$-best list of proofs from $A_k$, a scalar corresponding to the *residual* probability (possibly unnormalized) of all proofs not among the $k$-best.[7] The $k$-best proofs are still used for dynamically computing non-local features but the aggregation and combination operations are redefined to update the residual as appropriate.

We define the set $A_{cs}$ for cube summing as

$$A_{cs} = \mathbb{R}_{\geq 0} \times (\mathbb{R}_{\geq 0} \times \mathcal{L})^{\leq k} \times \mathcal{G} \times \{0, 1\}$$

A value $\mathbf{u} \in A_{cs}$ is defined as

$$\mathbf{u} = \langle u_0, \underbrace{\langle \langle u_1, U_1 \rangle, \langle u_2, U_2 \rangle, ..., \langle u_k, U_k \rangle \rangle}_{\bar{\mathbf{u}}}, g_u, u_s \rangle$$

For a proof list $\bar{\mathbf{u}}$, we use $\|\bar{\mathbf{u}}\|$ to denote the sum of all proof scores, $\sum_{i:\langle u_i, U_i \rangle \in \bar{\mathbf{u}}} u_i$.

The aggregation operator over operands $\{\mathbf{u}_i\}_{i=1}^N$, all such that $u_{is} = 0$,[8] is defined by:

$$\bigoplus_{i=1}^N \mathbf{u}_i = \left\langle \sum_{i=1}^N u_{i0} + \left\| Res\left(\bigcup_{i=1}^N \bar{\mathbf{u}}_i\right)\right\|, \max\text{-}k\left(\bigcup_{i=1}^N \bar{\mathbf{u}}_i\right), g_0, 0 \right\rangle \quad (12)$$

where $Res$ returns the "residual" set of scored proofs *not* in the $k$-best among its arguments, possibly the empty set.

For a set of $N+N'$ operands $\{\mathbf{v}_i\}_{i=1}^N \cup \{\mathbf{w}_j\}_{j=1}^{N'}$ such that $v_{is} = 1$ (non-local feature functions) and $w_{js} = 1$ (ordinary values), the combination operator $\otimes$ is shown in Eq. 13 Fig. 1. Note that the case where $N' = 0$ is not needed in this application; an ordinary value will always be included in combination.

In the special case of two ordinary operands (where $u_s = v_s = 0$), Eq. 13 reduces to

$$\mathbf{u} \otimes \mathbf{v} = \langle u_0 v_0 + u_0 \|\bar{\mathbf{v}}\| + v_0 \|\bar{\mathbf{u}}\| + \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\|, \max\text{-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}), g_0, 0 \rangle \quad (14)$$

We define $\mathbf{0}$ as $\langle 0, \langle\rangle, g_0, 0\rangle$; an appropriate definition for the combination identity element is less straightforward and of little practical importance; we leave it to future work.

If we use this generalized semiring to solve a DP and achieve *goal* value of $\mathbf{u}$, the approximate sum of all proof probabilities is given by $u_0 + \|\bar{\mathbf{u}}\|$. If all features are local, the approach is exact. With non-local features, the $k$-best list may not contain the $k$-best proofs, and the residual score, while including all possible proofs, may not include all of the non-local features in all of those proofs' probabilities.

## 5 Implementation

We have so far viewed dynamic programming algorithms in terms of their declarative specifications as semiring-weighted logic programs. Solvers have been proposed by Goodman (1999), by Klein and Manning (2001) using a hypergraph representation, and by Eisner et al. (2005). Because Goodman's and Eisner et al.'s algorithms assume semirings, adapting them for cube summing is non-trivial.[9]

To generalize Goodman's algorithm, we suggest using the directed-graph data structure known variously as an *arithmetic circuit* or *computation graph*.[10] Arithmetic circuits have recently drawn interest in the graphical model community as a

[6] Algebraic structures are typically defined with binary operators only, so we were unable to find a suitable term for this structure in the literature.

[7] Blunsom and Osborne (2008) described a related approach to approximate summing using the chart computed during cube pruning, but did not keep track of the residual terms as we do here.

[8] We assume that operands $\mathbf{u}_i$ to $\oplus_{cs}$ will never be such that $u_{is} = 1$ (non-local feature functions). This is reasonable in the widely used log-linear model setting we have adopted, where weights $\lambda_m$ are factors in a proof's product score.

[9] The bottom-up agenda algorithm in Eisner et al. (2005) might possibly be generalized so that associativity, distributivity, and binary operators are not required (John Blatz, p.c.).

[10] This data structure is not specific to any particular set of operations. We have also used it successfully with the inside semiring.

$$\bigotimes_{i=1}^{N} \mathbf{v}_i \otimes \bigotimes_{j=1}^{N'} \mathbf{w}_j = \Bigg\langle \left( \sum_{B \in \mathcal{P}(S)} \prod_{b \in B} w_{b0} \prod_{c \in S \setminus B} \|\bar{\mathbf{w}}_c\| \right) \tag{13}$$
$$+ \|Res(exec(g_{v_1}, \ldots exec(g_{v_N}, \bar{\mathbf{w}}_1 \star \cdots \star \bar{\mathbf{w}}_{N'}) \ldots))\|,$$
$$\max\text{-}k(exec(g_{v_1}, \ldots exec(g_{v_N}, \bar{\mathbf{w}}_1 \star \cdots \star \bar{\mathbf{w}}_{N'}) \ldots)), g_0, 0 \Bigg\rangle$$

Figure 1: Combination operation for cube summing, where $S = \{1, 2, \ldots, N'\}$ and $\mathcal{P}(S)$ is the power set of $S$ excluding $\emptyset$.

tool for performing probabilistic inference (Darwiche, 2003). In the directed graph, there are vertices corresponding to axioms (these are sinks in the graph), $\oplus$ vertices corresponding to theorems, and $\otimes$ vertices corresponding to summands in the dynamic programming equations. Directed edges point from each node to the nodes it depends on; $\oplus$ vertices depend on $\otimes$ vertices, which depend on $\oplus$ and axiom vertices.

Arithmetic circuits are amenable to automatic differentiation in the reverse mode (Griewank and Corliss, 1991), commonly used in backpropagation algorithms. Importantly, this permits us to calculate the *exact* gradient of the *approximate* summation with respect to axiom values, following Eisner et al. (2005). This is desirable when carrying out the optimization problems involved in parameter estimation. Another differentiation technique, implemented within the semiring, is given by Eisner (2002).

Cube pruning is based on the $k$-best algorithms of Huang and Chiang (2005), which save time over generic semiring implementations through lazy computation in both the aggregation and combination operations. Their techniques are not as clearly applicable here, because our goal is to sum over *all proofs* instead of only finding a small subset of them. If computing non-local features is a computational bottleneck, they can be computed only for the $O(k)$ proofs considered when choosing the best $k$ as in cube pruning. Then, the computational requirements for approximate summing are nearly equivalent to cube pruning, but the approximation is less accurate.

## 6 Semirings Old and New

We now consider interesting special cases and variations of cube summing.

### 6.1 The $k$-best+residual Semiring

When restricted to local features, cube pruning and cube summing can be seen as proper semirings. Cube pruning reduces to an implementation of the $k$-best semiring (Goodman, 1998), and cube summing reduces to a novel semiring we call the *$k$-best+residual* semiring. Binary instantiations of $\otimes$ and $\oplus$ can be iteratively reapplied to give the equivalent formulations in Eqs. 12 and 13. We define $\mathbf{0}$ as $\langle 0, \langle\rangle\rangle$ and $\mathbf{1}$ as $\langle 1, \langle 1, \epsilon\rangle\rangle$. The $\oplus$ operator is easily shown to be commutative. That $\oplus$ is associative follows from associativity of $\max$-$k$, shown by Goodman (1998). Showing that $\otimes$ is associative and that $\otimes$ distributes over $\oplus$ are less straightforward; proof sketches are provided in Appendix A. The $k$-best+residual semiring generalizes many semirings previously introduced in the literature; see Fig. 2.

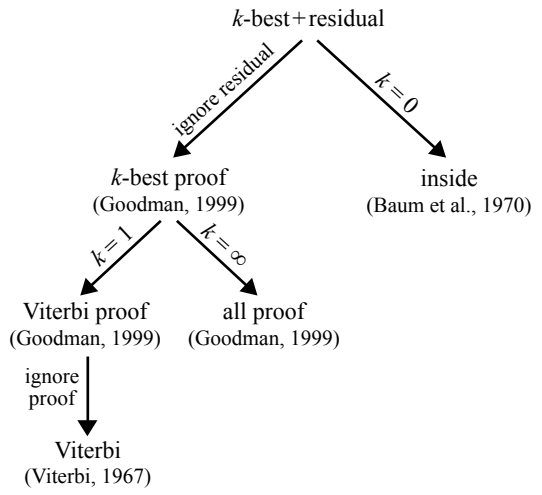


Figure 2: Semirings generalized by $k$-best+residual.

### 6.2 Variations

Once we relax requirements about associativity and distributivity and permit aggregation and combination operators to operate on sets, several extensions to cube summing become possible. First, when computing approximate summations with non-local features, we may not always be interested in the *best* proofs for each item. Since the purpose of summing is often to calculate statistics

under a model distribution, we may wish instead to *sample* from that distribution. We can replace the max-$k$ function with a sample-$k$ function that samples $k$ proofs from the scored list in its argument, possibly using the scores or possibly uniformly at random. This breaks associativity of $\oplus$. We conjecture that this approach can be used to simulate particle filtering for structured models.

Another variation is to vary $k$ for different theorems. This might be used to simulate beam search, or to reserve computation for theorems closer to $goal$, which have more proofs.

## 7 Conclusion

This paper has drawn a connection between cube pruning, a popular technique for approximately solving decoding problems, and the semiring-weighted logic programming view of dynamic programming. We have introduced a generalization called cube summing, to be used for solving summing problems, and have argued that cube pruning and cube summing are both semirings that can be used generically, as long as the underlying probability models only include local features. *With* non-local features, cube pruning and cube summing can be used for approximate decoding and summing, respectively, and although they no longer correspond to semirings, generic algorithms can still be used.


## Acknowledgments

We thank three anonymous EACL reviewers, John Blatz, Pedro Domingos, Jason Eisner, Joshua Goodman, and members of the ARK group for helpful comments and feedback that improved this paper. This research was supported by NSF IIS-0836431 and an IBM faculty award.


## A $k$-best+residual is a Semiring

In showing that $k$-best+residual is a semiring, we will restrict our attention to the computation of the residuals. The computation over proof lists is identical to that performed in the $k$-best proof semiring, which was shown to be a semiring by Goodman (1998). We sketch the proofs that $\otimes$ is associative and that $\otimes$ distributes over $\oplus$; associativity of $\oplus$ is straightforward.

For a proof list $\bar{\mathbf{a}}$, $\|\bar{\mathbf{a}}\|$ denotes the sum of proof scores, $\sum_{i:\langle a_i, A_i\rangle \in \bar{\mathbf{a}}} a_i$. Note that:

$$\|Res(\bar{\mathbf{a}})\| + \|\text{max-}k(\bar{\mathbf{a}})\| = \|\bar{\mathbf{a}}\| \quad (15)$$

$$\left\|\bar{\mathbf{a}} \star \bar{\mathbf{b}}\right\| = \|\bar{\mathbf{a}}\| \left\|\bar{\mathbf{b}}\right\| \quad (16)$$

**Associativity.** Given three semiring values $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$, we need to show that $(\mathbf{u}\otimes\mathbf{v})\otimes\mathbf{w} = \mathbf{u}\otimes(\mathbf{v}\otimes\mathbf{w})$. After expanding the expressions for the residuals using Eq. 14, there are 10 terms on each side, five of which are identical and cancel out immediately. Three more cancel using Eq. 15, leaving:

$$\begin{aligned}\text{LHS} &= \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\| \, \|\bar{\mathbf{w}}\| + \|Res(\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \star \bar{\mathbf{w}})\| \\ \text{RHS} &= \|\bar{\mathbf{u}}\| \, \|Res(\bar{\mathbf{v}} \star \bar{\mathbf{w}})\| + \|Res(\bar{\mathbf{u}} \star \text{max-}k(\bar{\mathbf{v}} \star \bar{\mathbf{w}}))\|\end{aligned}$$

If LHS = RHS, associativity holds. Using Eq. 15 again, we can rewrite the second term in LHS to obtain

$$\begin{aligned}\text{LHS} = \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\| \, \|\bar{\mathbf{w}}\| &+ \|\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \star \bar{\mathbf{w}}\| \\ &- \|\text{max-}k(\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \star \bar{\mathbf{w}})\|\end{aligned}$$

Using Eq. 16 and pulling out the common term $\|\bar{\mathbf{w}}\|$, we have

$$\begin{aligned}\text{LHS} &= (\|Res(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\| + \|\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\|) \, \|\bar{\mathbf{w}}\| \\ &\quad - \|\text{max-}k(\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \star \bar{\mathbf{w}})\| \\ &= \|(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \star \bar{\mathbf{w}}\| - \|\text{max-}k(\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \star \bar{\mathbf{w}})\| \\ &= \|(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \star \bar{\mathbf{w}}\| - \|\text{max-}k((\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \star \bar{\mathbf{w}})\|\end{aligned}$$

The resulting expression is intuitive: the residual of $(\mathbf{u}\otimes\mathbf{v})\otimes\mathbf{w}$ is the difference between the sum of all proof scores and the sum of the $k$-best. RHS can be transformed into this same expression with a similar line of reasoning (and using associativity of $\star$). Therefore, LHS = RHS and $\otimes$ is associative.

**Distributivity.** To prove that $\otimes$ distributes over $\oplus$, we must show left-distributivity, i.e., that $\mathbf{u}\otimes(\mathbf{v}\oplus\mathbf{w}) = (\mathbf{u}\otimes\mathbf{v})\oplus(\mathbf{u}\otimes\mathbf{w})$, and right-distributivity. We show left-distributivity here. As above, we expand the expressions, finding 8 terms on the LHS and 9 on the RHS. Six on each side cancel, leaving:

$$\begin{aligned}\text{LHS} &= \|Res(\bar{\mathbf{v}} \cup \bar{\mathbf{w}})\| \, \|\bar{\mathbf{u}}\| + \|Res(\bar{\mathbf{u}} \star \text{max-}k(\bar{\mathbf{v}} \cup \bar{\mathbf{w}}))\| \\ \text{RHS} &= \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\| + \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{w}})\| \\ &\quad + \|Res(\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \cup \text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{w}}))\|\end{aligned}$$

We can rewrite LHS as:

$$\begin{aligned}\text{LHS} &= \|Res(\bar{\mathbf{v}} \cup \bar{\mathbf{w}})\| \, \|\bar{\mathbf{u}}\| + \|\bar{\mathbf{u}} \star \text{max-}k(\bar{\mathbf{v}} \cup \bar{\mathbf{w}})\| \\ &\quad - \|\text{max-}k(\bar{\mathbf{u}} \star \text{max-}k(\bar{\mathbf{v}} \cup \bar{\mathbf{w}}))\| \\ &= \|\bar{\mathbf{u}}\| \, (\|Res(\bar{\mathbf{v}} \cup \bar{\mathbf{w}})\| + \|\text{max-}k(\bar{\mathbf{v}} \cup \bar{\mathbf{w}})\|) \\ &\quad - \|\text{max-}k(\bar{\mathbf{u}} \star \text{max-}k(\bar{\mathbf{v}} \cup \bar{\mathbf{w}}))\| \\ &= \|\bar{\mathbf{u}}\| \, \|\bar{\mathbf{v}} \cup \bar{\mathbf{w}}\| - \|\text{max-}k(\bar{\mathbf{u}} \star (\bar{\mathbf{v}} \cup \bar{\mathbf{w}}))\| \\ &= \|\bar{\mathbf{u}}\| \, \|\bar{\mathbf{v}} \cup \bar{\mathbf{w}}\| - \|\text{max-}k((\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \cup (\bar{\mathbf{u}} \star \bar{\mathbf{w}}))\|\end{aligned}$$

where the last line follows because $\star$ distributes over $\cup$ (Goodman, 1998). We now work with the RHS:

$$\begin{aligned}\text{RHS} &= \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\| + \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{w}})\| \\ &\quad + \|Res(\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \cup \text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{w}}))\| \\ &= \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\| + \|Res(\bar{\mathbf{u}} \star \bar{\mathbf{w}})\| \\ &\quad + \|\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \cup \text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{w}})\| \\ &\quad - \|\text{max-}k(\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \cup \text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{w}}))\|\end{aligned}$$

Since max-$k(\bar{\mathbf{u}} \star \bar{\mathbf{v}})$ and max-$k(\bar{\mathbf{u}} \star \bar{\mathbf{w}})$ are disjoint (we assume no duplicates; i.e., two different theorems cannot have exactly the same proof), the third term becomes $\|\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}})\| + \|\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{w}})\|$ and we have

$$\begin{aligned}&= \|\bar{\mathbf{u}} \star \bar{\mathbf{v}}\| + \|\bar{\mathbf{u}} \star \bar{\mathbf{w}}\| \\ &\quad - \|\text{max-}k(\text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \cup \text{max-}k(\bar{\mathbf{u}} \star \bar{\mathbf{w}}))\| \\ &= \|\bar{\mathbf{u}}\| \, \|\bar{\mathbf{v}}\| + \|\bar{\mathbf{u}}\| \, \|\bar{\mathbf{w}}\| \\ &\quad - \|\text{max-}k((\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \cup (\bar{\mathbf{u}} \star \bar{\mathbf{w}}))\| \\ &= \|\bar{\mathbf{u}}\| \, \|\bar{\mathbf{v}} \cup \bar{\mathbf{w}}\| - \|\text{max-}k((\bar{\mathbf{u}} \star \bar{\mathbf{v}}) \cup (\bar{\mathbf{u}} \star \bar{\mathbf{w}}))\| \, .\end{aligned}$$

## References


L. E. Baum, T. Petrie, G. Soules, and N. Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41(1).

P. Blunsom and M. Osborne. 2008. Probabilistic inference for machine translation. In *Proc. of EMNLP*.

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

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

W. W. Cohen and V. Carvalho. 2005. Stacked sequential learning. In *Proc. of IJCAI*.

A. Darwiche. 2003. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3).

J. Eisner, E. Goldlust, and N. A. Smith. 2005. Compiling Comp Ling: Practical weighted dynamic programming and the Dyna language. In *Proc. of HLT-EMNLP*.

J. Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proc. of ACL*.

J. R. Finkel, T. Grenager, and C. D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of ACL*.

J. Goodman. 1998. *Parsing inside-out*. Ph.D. thesis, Harvard University.

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

A. Griewank and G. Corliss. 1991. *Automatic Differentiation of Algorithms*. SIAM.

L. Huang and D. Chiang. 2005. Better $k$-best parsing. In *Proc. of IWPT*.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL*.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL*.

M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2).

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

T. Koo and M. Collins. 2005. Hidden-variable models for discriminative reranking. In *Proc. of EMNLP*.

K. Kurihara and T. Sato. 2006. Variational Bayesian grammar induction for natural language. In *Proc. of ICGI*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*.

R. Levy, F. Reali, and T. Griffiths. 2008. Modeling the effects of memory on human online sentence processing with particle filters. In *Advances in NIPS*.

B. T. Lowerre. 1976. *The Harpy Speech Recognition System*. Ph.D. thesis, Carnegie Mellon University.

D. J. C. MacKay. 1997. Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, Cambridge.

A. F. T. Martins, D. Das, N. A. Smith, and E. P. Xing. 2008. Stacking dependency parsers. In *Proc. of EMNLP*.

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*.

F. C. N. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. of ACL*, pages 128–135.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of CoNLL*.

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

D. A. Smith and J. Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

N. A. Smith, D. L. Vail, and J. D. Lafferty. 2007. Computationally efficient M-estimation of log-linear structure models. In *Proc. of ACL*.

C. Sutton and A. McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *Proc. of ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*.

A. J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Processing*, 13(2).

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.