Formalizing Semantic Parsing with Tree Transducers

Bevan Keeley Jones & Mark Johnson
Department of Computing
Macquarie University
Sydney, NSW 2109, Australia
Bevan.Jones@students.mq.edu.au
Mark.Johnson@mq.edu.au

Sharon Goldwater
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
sgwater@inf.ed.ac.uk

Abstract

This paper introduces tree transducers as a unifying theory for semantic parsing models based on tree transformations. Many existing models use tree transformations, but implement specialized training and smoothing methods, which makes it difficult to modify or extend the models. By connecting to the rich literature on tree automata, we show how semantic parsing models can be developed using completely general estimation methods. We demonstrate the approach by reframing and extending one state-of-the-art model as a tree automaton. Using a variant of the inside-outside algorithm with variational Bayesian estimation, our generative model achieves higher raw accuracy than existing generative and discriminative approaches on a standard data set.

1 Introduction

Automatically interpreting language is an important challenge for computational linguistics. *Semantic parsing* addresses the specific task of learning to map natural language sentences to formal representations of their meaning, a problem that arises in developing natural language interfaces, for example. Given a set of (sentence, meaning representation) pairs like the example below, we want to to learn a map that generalizes to previously unseen sentences.

1. Sentence: what is the capital of texas ?
   Meaning: answer(capital_1(stateid(texas)))

Researchers have formalized the learning problem in various ways, with approaches including string classifiers (Kate and Mooney, 2006), synchronous grammar (Wong and Mooney, 2006), combinatory categorial grammar (Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010), and PCFG-based approaches (Lu et al., 2008; Borschinger et al., 2011). Each approach has required its own custom algorithms, which has made model development and innovation slow. Nevertheless, there are many similarities between the approaches, which all exploit parallels between the structure of the meaning representation and that of the natural language. The meaning representation, as a context-free formal language, has an obvious tree structure. Trees are also widely used to describe natural language structure. Consequently, the semantic parsing problem can be generally defined as learning a mapping between trees, one of which may be latent. This mapping can be expressed as a *tree transducer*, a formalism from automata theory that maps input trees to output trees or strings. Tree transducers have well understood properties and algorithms, and a rich literature, making them a particularly appealing model class.

Although some previous approaches strongly resemble tree transducers, to our knowledge, we are the first to explicitly formulate the problem in this way. We argue that connecting semantic parsing to the tree automata literature will free researchers from devising custom solutions and allow them to focus on studying and improving their models and developing more general learning algorithms.

To demonstrate the effectiveness of the approach, we choose one state-of-the-art model, the hybrid tree (Lu et al., 2008), translate it into the tree transducer

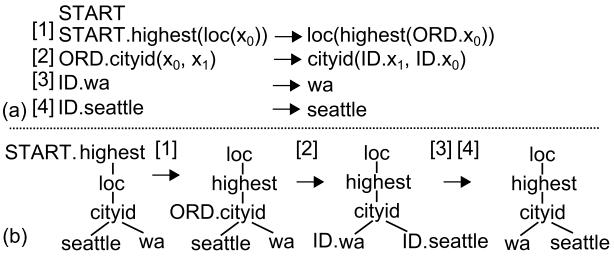

Figure 1: An extended left hand side, root-to-frontier, linear, non-deleting, tree-to-tree transducer (a) and an example derivation (b). Numbered arrows in the derivation indicate which rules apply during that step. Rule [1] is the only rule with an extended left hand side.

framework, and add a small extension, made easy by the framework. We also update a standard tree transducer training algorithm to incorporate a Variational Bayes approximation. The result is the first purely generative model to achieve state-of-the-art results on a standard data set.

2 Extended, root-to-frontier, linear, non-deleting tree transducers

Tree transducers (Rounds, 1970; Thatcher, 1970) are generalizations of finite state machines that take trees as inputs and either output a string or another tree. Mirroring the branching nature of its input, the tree transducer may simultaneously transition to any number of successor states, assigning a separate state to process each sub-tree. Although they were originally conceived of by Rounds (1970) as a way to formalize tree transformations in linguistic theory, they have since received far more interest in theoretical computer science. Recently, however, they have also been used for syntax-based statistical machine translation (Graehl et al., 2008; Knight and Greahl, 2005).

Figure 1 presents an example of a tree-to-tree transducer. It is defined using tree transformation rules, where the left hand side identifies a state of the transducer and a fragment of the input tree, and the right hand side describes a fragment of the output tree. Variables x_i stand for entire sub-trees. There are many classes of transducer, each with its own selection of algorithms (Knight and Greahl, 2005). In this paper we restrict consideration primarily to the extended left hand side, root-to-frontier, linear, non-deleting tree transducers (Maletti et al., 2009), and we particularly make use of tree-to-string transducers.

Formally, an extended left hand side, root-to-frontier, tree-to-tree transducer is a 5-tuple $(Q, \Sigma, \Delta, q_{start}, \mathcal{R})$. Q is a finite set of states, Σ and Δ are the input and output tree alphabets, q_{start} is the start state, and $\mathcal{R}$ is the set of rules. We denote a pair of symbols, a and b by $a.b$, and the cross product of two sets A and B by $A.B$. Let X be the set of variables $\{x_0, x_1, ...\}$. Finally, let $T_\Sigma(A)$ be the set of trees with non-terminals from alphabet Σ and leaf symbols from alphabet A. Then, each rule $r \in \mathcal{R}$ is of the form $[q.t \rightarrow u].v$, where $q \in Q$, $t \in T_\Sigma(X)$, $u \in T_\Delta(Q.X)$ such that every $x \in X$ in u also occurs in t, and $v \in \Re^{\geq 0}$ is the weight of the rule.

We say $q.t$ is the left hand side of the rule and u is the right hand side. The transducer is *linear* iff no variable appears more than once on the right hand side. It is *non-deleting* iff all variables on the left hand side also occur on the right hand side. Iff every tree t on the left hand side is of the form $\sigma(x_0, ...x_n)$, where $\sigma \in \Sigma$ (i.e., it is a tree of depth ≤ 1), then the transducer is simply root-to-frontier, otherwise we say it has an *extended left hand side* with the added power to look a bounded depth into the tree at each step. Finally, for a *tree-to-string* transducer, Δ is an alphabet, and the right hand sides of the rules consist of finite tuples of elements taken from $\Delta \cup Q.X$.

A weighted tree transducer may define a probability distribution, either a joint distribution over input and output pairs or a conditional distribution of the output given the input. Here, we will use joint distributions, which can be defined by ensuring that the weights of all rules with the same state on the left-hand side sum to one. In this case, it can be helpful to view the transducer as simultaneously generating both the input and output, rather than the usual view of reading inputs and writing outputs.

3 Semantic parsing and meaning representation languages

The goal of semantic parsing is to assign formal meanings to natural language (NL) sentences, requiring a formal meaning language. Some systems use lambda expressions; others use variable free logical languages or functional languages (such as that

of example 1 in the introduction). Here we deal with meaning representations (MRs) of the latter form where the bracketing makes the tree structure obvious [1] We refer to functions and predicates in the MR as either symbols or entities. Since MRs are trees, the language can be defined by a Regular Tree Grammar (a kind of CFG that generates trees). We refer to this grammar as the *meaning representation grammar* or MR grammar. Figure 3 shows a fragment of such a grammar and an MR parse. The parse is just the MR with each symbol labeled with its grammar rule. Like most systems, the MR grammar is one of our inputs.

4 The hybrid tree model

The idea of the hybrid tree model (Lu et al., 2008) is to start with the MR and apply a series of transformations to create a kind of parse tree for the NL. There are two types of transformation. The first determines word order by simultaneously choosing where to attach words (but not the particular words) and whether or not to swap the order of siblings (Figure 2a). Once the order is determined, word generating transformations are then applied to insert specific words in the determined locations (Figure 2b).

The hybrid tree includes parameters for the MR as well as the transformations in Figure 2 that relate words to meaning representations. The probability of each symbol in the MR is conditioned on the MR grammar rules that derive its parent symbol. Defining symbol probabilities in terms of their parents' grammar rules (as opposed to parent symbols as in a standard PCFG) distinguishes between functions and predicates with the same name but different semantics (Wong and Mooney, 2006).

To formally define the probability of the MR, let $paths$ be the set of paths from the root to every node in the MR where paths are represented using a variety of Gorn's notation (Gorn, 1962)[2]. Let $args_i$ be the set of indices of the children of the node at path i; and R_i be the grammar rule that derives the symbol at i according to the MR parse. Then, the following equation defines $P(MR)$.

$$P(MR) = P(R_\epsilon) \prod_{i \in paths} \prod_{j \in args_i} P(R_{i \cdot j} | j, R_i) \quad (1)$$

In other words, each node in the tree is generated according to the probability of the MR rule that derives it conditioned on (1) the MR rule R_i that derives its parent symbol and (2) its position j beneath that parent.

The hybrid tree model then re-orders and extends this basic skeleton to include the NL. The probability of this hybrid tree can be formally defined as follows if we let pat_i be the word order pattern used to generate the children of the node at path i, and $words_i$ be the indices of the words attached under the node at i.

$$P(\textit{NL-MR hybrid}) = P(R_\epsilon) \prod_{i \in paths} P(pat_i | R_i) \cdot \prod_{j \in args_i} P(R_{i \cdot j} | j, R_i) \prod_{k \in words_i} P(w_{i \cdot k} | R_i) \quad (2)$$

Note that $P(pat|R)$ and $P(w|R)$ correspond respectively to the weights on the word order and word generation transformations. In fact, equation 2 is a joint probability over not only the NL and MR pair but also the actual set of transformations chosen to produce the particular hybrid tree relating them.

5 Reframing the hybrid tree as a tree transducer

We now define a tree transducer that simultaneously generates an MR tree and NL string according to the joint probability defined by equation 2. We create separate states for each of the two transformation types (*order* states for word order selection and *word* states for word generation). In order to model the properties of the MR grammar (necessary for modeling equation 1), we create one additional state type for selecting MR children (*arg* states) and embed the MR grammar rules into the states so that each state is identified with exactly one grammar rule. Transitions between transducer states then simulate the action of the MR grammar as it generates a new MR tree. Notationally, we employ subscripts to indicate each state's basic type (*arg*, *order*, or *word*) and superscripts to indicate the associated MR grammar

[1] With a pre-parsing step, it may also be possible to represent lambda expressions with trees (see Liang et al. (2011)).

[2] I.e., paths are represented by strings where the empty string ϵ is the path to the root, and if i is a path and j is the index of a child of the node at i, $i \cdot j$ is the path to that child.

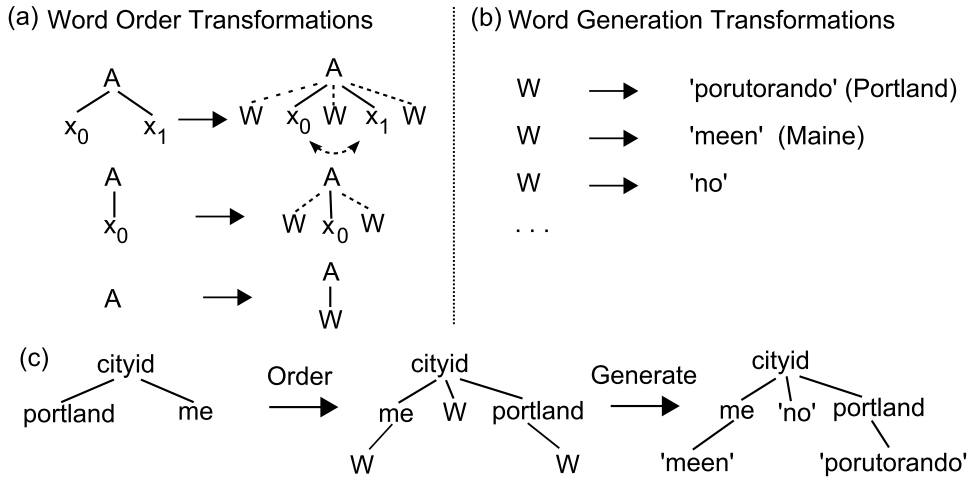

Figure 2: The two transformation types of the hybrid tree model and an example of their application. (a) Word order transformations simultaneously permute arguments and add W symbols where words should be attached. The dotted lines indicate that W symbols may or may not be attached in each of the possible locations, and siblings may or may not be swapped. Each possible configuration of sibling orderings and W attachments corresponds to a single transformation. Thus there are 4 different transformations for the case where A has one child, and 16 for when it has 2. In the case where A has no children, word attachment is not optional. (b) Word generation replaces each W symbol with actual words. (c) The series of transformations from example MR *cityid(portland,me)* to produce a parse for the Japanese equivalent of 'portland, maine'.

rule, so that, for instance, state $q^R_{\mathbf{order}}$ is an *order* state associated with MRL grammar rule R.

Figure 3 presents a graphical representation of the basic state transitions of the transducer, where the states for each grammar rule are clustered inside dotted lines beneath its associated grammar rule label. The transducer begins in an *arg* state and proceeds as follows. First, the *arg* state selects the next child by transitioning to an *order* state corresponding to the MR rule that generates the appropriate child. The *order* state then chooses the appropriate word order pattern and transitions to the *word* and *arg* states associated with that same grammar rule[3]. The *word* states proceed to generate words one at a time in a loop and finally terminate the string. Then the *arg* state begins the cycle over again by transitioning to the *order* of the next child in the MR tree.

Table 1 lists the actual transducer rule types. Rule probabilities are conditioned on the state on the left hand side. Thus, since states identify both their function and the grammar rule of the current MR node, rule weights correspond directly to the terms in equation 2: $P(R_{i\cdot j}|j, R_i)$, $P(pat|R)$, and $P(w|R)$.

[3]Note that only *arg* states are permitted to transition to states for different grammar rules.

5.1 Source tree language model: $P(R_{i\cdot j}|j, R_i)$

Rule type 1 in Table 1 begins the process by transitioning from start state $q_{\mathbf{start}}$ to $q^R_{\mathbf{order}}$, where the grammar rule R ranges over those rules with the start symbol S on the left hand side. Choosing exactly which $q^R_{\mathbf{order}}$ to transition to corresponds to the decision of choosing the root symbol of the MR tree (the symbol generated by R), and these transducer rules define the $P(R_\epsilon)$ term in equation 1, i.e., the probability of the grammar rule corresponding to the root symbol of the MR tree.

For each pair of MR grammar rules R^p and R^c, we add a transducer rule of the form of rule type 2 that transitions from the states associated with R^p to those for R^c if R^c generates a valid child of the symbol generated by R^p. Thus, the choice of state transition here corresponds to choosing the child of the last generated symbol of the input tree. State $q^{R^p}_{\mathbf{arg},i}$ selects the i^{th} argument of the current function in the MR without generating anything in the input tree. With rules described in the next section, state $q^{R^c}_{\mathbf{order}}$ then writes the symbol to the input tree specified by MR grammar rule R^c.

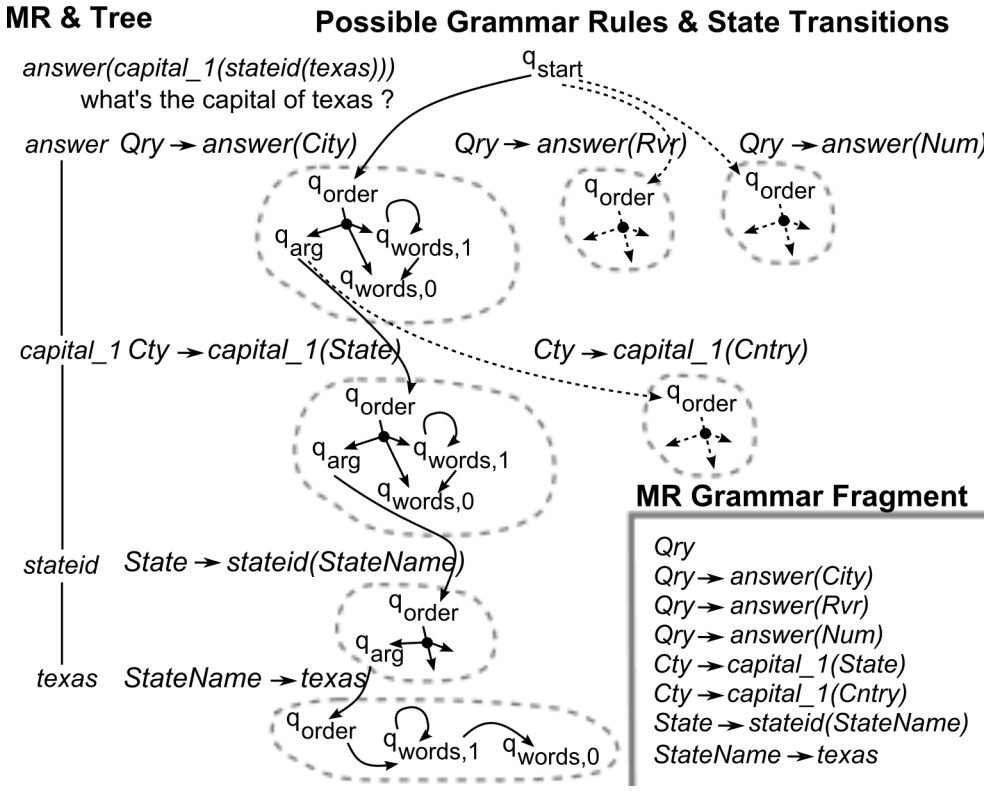

Figure 3: State transitions selecting appropriate grammar rules for generating an MR. Solid arcs indicate a state transition was taken; dotted lines are alternatives. States are divided up into disjoint sets and associated with a specific MR rule. Transitioning between state sets implicitly chooses an MR rule. The rules lined up with the MR tree to the left constitute an MR parse. The bottom right shows the grammar fragment corresponding to this portion of the transducer.

Since the state on the left encodes the rule of the parent and the argument number, and the state on the right the child rule, the weights for transducer rules of type 2 define $P(R_{i\cdot j}|j, R_i)$ in equation 1.

5.2 Order decisions: $P(pat|R)$

Word ordering decisions are made with the aid of preprocessing step that adds W symbols to the input tree wherever words can be attached. These symbols are just a convenience: it is easier to design rules where every output structure has a counterpart in the input. The symbols are removed later in a postprocessing step (also using a tree transducer). Attachment decisions are then made by deciding which of these W symbols to replace with the empty string (no attachment) or a string of words.

We add transducer rules of the form of rule type 3 in Table 1 for each MR grammar rule R^f, to define the selection of one of the word order patterns of the hybrid tree. These rules simply enumerate the conjunction of all the possible word attachment patterns and argument order decisions. Binary sequence i indicates the word attachment portion of the hybrid tree pattern, where each bit is either 1 indicating an attachment, or 0 for a decision not to attach. For an n argument function, there are $n+1$ such choices, requiring an $n+1$ bit sequence, where i_k is the decision for the k^{th} position. Argument order is indicated by j, a permutation of the numbers $0, 1, ...n-1$, and j_k is the k^{th} number in the permutation, indicating which argument appears at position k. State $q^{R^f}_{\mathbf{words},1}$ generates the words for f, state $q^{R^f}_{\mathbf{words},0}$ replaces the symbol W with the empty string, and the states $q^{R^f}_{\mathbf{arg},k}$ select the grammar rule with which to generate the k^{th} child. When there is only a single child W, no decisions about argument order or child attachment are needed; rule type 4 always generates words for these constants.

$$q_{\mathbf{start}}.x_0 \rightarrow q^{R}_{\mathbf{order}}.x_0 \quad (1)$$

$$q^{R^p}_{\mathbf{arg},i}.x_0 \rightarrow q^{R^c}_{\mathbf{order}}.x_0 \quad (2)$$

$$q^{R^f}_{\mathbf{order}}.f(w_0, x_0, w_1, x_1, w_2, ...x_{n-1}, w_n) \rightarrow q^{R^f}_{\mathbf{words},i_0}.w_0 \; q^{R^f}_{\mathbf{arg},j_0}.x_{j_0} \; q^{R^f}_{\mathbf{words},i_1}.w_1 \; q^{R^f}_{\mathbf{arg},j_1}.x_{j_1} \; q^{R^f}_{\mathbf{words},i_2}.w_2 \; ... \; q^{R^f}_{\mathbf{arg},j_{n-1}}.x_{j_{n-1}} \; q^{R^f}_{\mathbf{words},i_n}.w_n \quad (3)$$

$$q^{R^f}_{\mathbf{order}}.f(w_0) \rightarrow q^{R^f}_{\mathbf{words},1}.w_0 \quad (4)$$

$$q^{R}_{\mathbf{words},1}.x_0 \rightarrow word_k \; q^{R}_{\mathbf{words},1}.x_0 \quad (5)$$

$$q^{R}_{\mathbf{words},1}.x_0 \rightarrow word_k \; q^{R}_{\mathbf{words},0}.x_0 \quad (6)$$

$$q^{R}_{\mathbf{words},0}.W \rightarrow \epsilon \quad (7)$$

Table 1: Seven transducer rule types for three classes of transformation. (1)-(2) define $P(R_{i \cdot j}|j, R_i)$, (3)-(4) define $P(pat|R_i)$, and (5)-(7) define $P(w|R_i)$.

The following input tree and output string pair is illustrates an intermediate computation produced by interleaving these two kinds of ordering rules with the argument selection rules of the previous section, and applying them to the example in Figure 2:

$$q^{R^{cityid}}_{\mathbf{order}}.cityid(W, portland(W), W, me(W), W) \stackrel{*}{\Rightarrow} q^{R^{cityid}}_{\mathbf{words},0}.W \; q^{R^{me}}_{\mathbf{words},1}.W \; q^{R^{cityid}}_{\mathbf{words},1}.W \; q^{R^{portland}}_{\mathbf{words},1}.W \; q^{R^{cityid}}_{\mathbf{words},0}.W$$

The weights on these rules define the conditional probability $P(pat|R)$, where pat is one of the patterns of the word transformations illustrated in Figure 2.

5.3 Word generation: $P(w|R)$

Rule types 5 and 6 in Table 1 define the conditional probability of a word $word_k$ given an MR grammar rule, and rule type 7 terminates generation by generating W in the input and ϵ in the output. Using the same example as in the previous section, this yields 5 W symbols in the input tree and the string 'meen no porutorando' in the output.

$$q^{R^{cityid}}_{\mathbf{words},0}.W \stackrel{*}{\Rightarrow} \epsilon$$
$$q^{R^{me}}_{\mathbf{words},1}.W \stackrel{*}{\Rightarrow} \text{'meen'} \; \epsilon$$
$$q^{R^{cityid}}_{\mathbf{words},1}.W \stackrel{*}{\Rightarrow} \text{'no'} \; \epsilon$$
$$q^{R^{portland}}_{\mathbf{words},1}.W \stackrel{*}{\Rightarrow} \text{'porutorando'} \; \epsilon$$
$$q^{R^{cityid}}_{\mathbf{words},0}.W \stackrel{*}{\Rightarrow} \epsilon$$

5.4 Derivation weights and the joint probability distribution

The transducer applies the rules from the three classes of transformation in Table 1 to ultimately produce an MR-NL pair. The probability of this derivation is essentially the same quantity as that of the hybrid tree of the original model (shown in equation 2).

6 An extension: head-switching

Reordering siblings allows the hybrid tree to capture a large number of word orders, but it is still constrained by the hierarchy of the tree. This constraint reduces the search space but also prevents the model from learning some word orders. Figure 4 illustrates with trees from the following Japanese sentence meaning *what's the highest point in the USA?* (the third line gives the correct alignment of words to components of the gold MR, which cannot be learned by the hybrid tree):

beikoku no	*mottomo takai*	*chiten*	*wa nan desu ka*
america's	most high	point	what is
loc(america)	*highest()*	*place()*	*answer()*

To address this problem, we modify the transducer to allow it to rotate parents with their children in addition to re-ordering siblings. This change is easy within the transducer framework but would be difficult in the original implementation, requiring a complete reworking of the training and decoding algorithms. In the original transducer, rules oper-

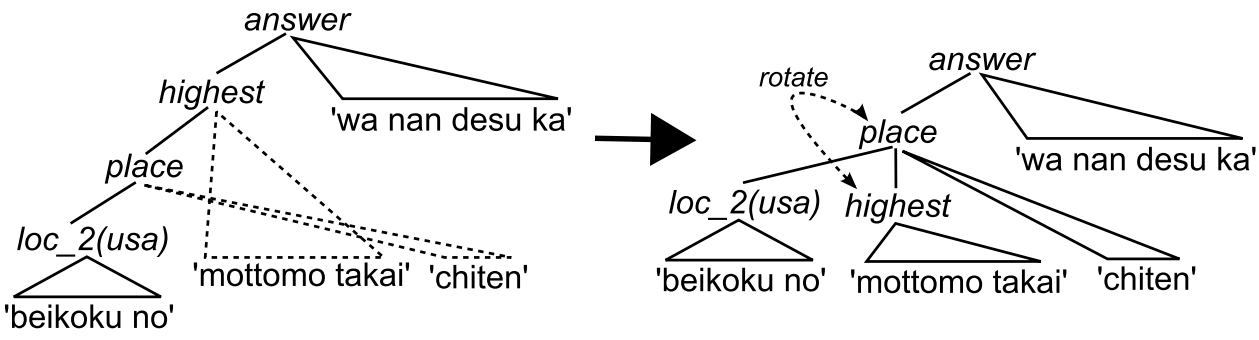

Figure 4: An example from Japanese illustrating head-switching. The tree on the left attempts (and fails) to generate the target sentence from the gold meaning representation. Switching *highest* and *place* allows the correct MR-NL map.

ate on tree fragments of depth ≤ 1. We implement the change using extended left-hand-side transducers, which can operate on larger fragments as long as the depth is bounded (Maletti et al., 2009). In particular, we introduce rules like the following:

$$q^{R^p}_{\mathbf{order}}.p(w^p_0, c(w^c_0, x^c_0, w^c_1), w^p_1) \rightarrow q^{R^c}_{\mathbf{words},i_0}.w^c_0$$
$$q^{R^p}_{\mathbf{words},i_1}.w^p_0\ q^{R^c}_{\mathbf{arg},0}.x^c_0\ q^{R^p}_{\mathbf{words},i_2}.w^p_1\ q^{R^c}_{\mathbf{words},i_3}.w^c_1$$

This rule begins the word generation process simultaneously for both the parent and child, reordering the words to simulate the new nesting structure, and then proceeds to choose the child function's argument. We add similar rules for the various cases where the child and parent have multiple arguments.

7 Varitional Bayes parameter estimation

Tree transducer derivations are themselves trees, allowing for the computation of inside and outside probabilities much as for the derivation trees of PCFGs. EM can then be applied in much the same way as for PCFGs, substituting the tree-to-string derivation algorithm for standard PCFG parsing (Graehl et al., 2008). Note that while EM maximizes the likelihood of the training data, items not observed during training receive zero probability, limiting the ability of models to generalize to new data sets. Furthermore, many items that are actually present in the training data are only seen a very few times, which can lead to a poor estimate of their distribution in the target data set. Bayesian estimation techniques such as Variational Bayes (VB) address these problems by allowing us to place a prior probability over the parameters, which particularly influence parameter estimates for sparse items and, depending on the choice of prior, may also assign some non-zero probability to unseen items.

We give a high-level outline of how a Dirichlet prior can be incorporated into tree transducer training using Variational Bayes, drawing heavily on the essential similarity of inside-outside for PCFGs and training for tree transducers. We direct the reader to Kurihara and Sato (2006) for the details of PCFG training using VB, and to Graehl et al. (2008) for the full treatment of the basic EM algorithm for tree transducers, on which our VB training algorithm is closely based. See Bishop (2006) for a general introduction to VB and Beal (2003) for a derivation of VB as applied to Dirichlet-multinomials.

The objective of training is to find an estimate for the weights θ of the transducer rules given some symmetric Dirichlet prior with hyperparameter α and observed pairs of natural language sentences W and meaning representation trees Y.

$$p(\theta|\alpha, W, Y) = \frac{p(W, Y, \theta|\alpha)}{p(W, Y|\alpha)} \tag{8}$$

The tree transducer defines the probability $p(W, X, Y|\theta)$, where X is a vector of derivations such that $x_i \in X$ is the derivation from MRL tree $y_i \in Y$ to NL string $w_i \in W$. We put a symmetric Dirichlet prior over θ so that the probability $p(\theta|\alpha)$ follows directly from the definition of the Dirichlet distribution. Thus, computing the denominator of equation 8 involves integrating out θ and X.

$$p(W, Y|\alpha) = \int p(W, X, Y|\theta)p(\theta|\alpha)dXd\theta$$

However, this integral is intractable, so instead, following from Variational Bayes, we make an approximation $q(X, \theta)$ for the posterior probability $p(X, \theta|W, Y, \alpha)$.

$$\begin{aligned}\log p(W, Y|\alpha) &= \log \int p(W, X, Y, \theta|\alpha) dX d\theta \\ &= \log \int q(X, \theta) \frac{p(W, X, Y, \theta|\alpha)}{q(X, \theta)} dX d\theta \\ &\geq \int q(X, \theta) \log \frac{p(W, X, Y, \theta|\alpha)}{q(X, \theta)} dX d\theta \\ &= \mathcal{F}\end{aligned}$$

We can minimize the KL divergence between $q(X, \theta)$ and $p(W, Y|\alpha)$ by maximizing the lower bound $\mathcal{F}$, called the variational free energy. Since $\mathcal{F}$ is a function of q, this amounts to maximizing q.

Following from Kurihara and Sato (2006)'s treatment of PCFGs, we employ the mean field approximation that assumes the posterior is well approximated by a factorized function $q(X, \theta) = q_1(X)q_2(\theta)$, which treats the derivations X and the rule weights θ as independent. This allows us to maximizing q by alternately updating parameters for q_1 with q_2 fixed, and then updating parameters for q_2 with q_1 fixed, essentially in the same manner that E and M steps alternate in EM. The mathematical derivation of the modified inside-outside algorithm then follow directly from Kurihara and Sato (2006).

In practice, VB requires only a slight modification to the basic EM algorithm, and we refer the reader to Graehl et al. (2008) for the details of EM for tree transducers. As in inside-outside for PCFGs, the E-step involves computing estimated rule counts, weighted using inside and outside probabilities. The M-step resolves to calculating the vector parameters of the multinomial distributions over transducer rules using these count estimates. That is, if $\theta_{\mathbf{s}}$ is a multinomial parameter vector for transducer rules with state s on the left hand side, $\theta_{s,k}$ is its k^{th} component (i.e., the weight of the k^{th} rule with s on the left hand side), and $c_{s,k}$ is the corresponding expected count, we have the following equation for straight EM.

$$\theta_{s,k} = \frac{c_{s,k}}{\sum_{k'} c_{s,k'}}$$

Incorporating a Dirichlet prior with parameter α using our VB approximation simply requires replacing this ratio with the following alternative quantity τ, where Ψ is the digamma function.

$$\tau_{s,k} = \exp\left(\Psi(c_{s,k} + \alpha) - \Psi\left(\sum_{k'} c_{s,k'} + \alpha\right)\right)$$

For each step of EM, the updated τ vectors from the previous M-step are then used to compute the expected counts c during the current E-step.

8 Experimental setup

We use Tiburon (May and Knight, 2006), a tree transducer toolkit, to train our transducer using 40 iterations of its inside-outside-like EM training procedure, and modify it slightly to include the mean field VB approximation for a symmetric Dirichlet prior over the multinomial parameters as just described.

Decoding is handled the same by Tiburon for both training procedures, producing the MR input tree with the tree transducer derivation that maximizes the probability over derivations of equation 2.

In keeping with the original hybrid tree, we run 100 iterations of IBM alignment model 1 (Brown et al., 1993) to initialize the word distribution parameters. Also in keeping with Lu et al. (2008), we use the standard noun phrase list from the given language to help initialize the word distributions for their counterparts in the meaning representation language.

9 Results

To evaluate our models, we use the the GeoQuery corpus, a standard benchmark data set. The corpus contains English sentences (questions about U.S. geography) paired with an MR in a database query language, 250 of which were translated into Japanese (among other languages) yielding two training sets using the same MRs. For testing we run 10-fold cross validation, using the standard train and test splits of Wong and Mooney (2006), and micro-average our performance metrics across folds.

We measure performance using precision, recall, and f-score (the harmonic mean of precision and recall) as standardly defined in the semantic parsing literature. Recall is simply the raw accuracy: the percentage of correct parses found out of all test sentences (where a parse is considered correct if it retrieves the same results from the GeoQuery database

System	English			Japanese		
	Pre.	Rec.	F1	Pre.	Rec.	F1
UBL-s	80.8	80.4	80.6	80.6	80.5	80.6
WASP	**95.4**	70.0	80.8	**92.0**	74.4	**82.9**
Lu-uni	80.2	71.2	75.4	79.7	73.6	76.5
Lu-dis	91.5	72.8	81.1	87.6	76.0	81.4
trs	88.3	69.6	77.9	82.4	67.2	74.0
trsVB	82.0	82.0	82.0	78.0	78.0	78.0
hs	89.5	71.6	79.6	84.3	68.8	75.8
hsVB	82.8	**82.8**	**82.8**	80.8	**80.8**	80.8

Table 2: Performance of the various models on the multilingual section of GeoQuery.

as the gold MR). Precision is the percentage of correct parses out of all sentences for which we find any parse at all.

Table 2 compares our models' performance to previously published results. We list two versions of our model: the direct adaptation of the hybrid tree and the transducer with parent-child swapping rules. We train each version with both standard EM and the VB approximation (hyperparameter 0.1). The other state-of-the-art systems shown are: 1) two versions of the original hybrid tree (Lu et al., 2008): *Lu-uni*, which uses a unigram distribution over words, and is therefore the most similar to our transducer implementation, and *Lu-dis*, the best-performing version, which uses a mixture of unigram and bigram model with discriminative re-ranking; 2) WASP (Wong and Mooney, 2006), which uses a synchronous grammar approach; and 3) UBL-s (Kwiatkowski et al., 2010), the model with the highest published raw accuracy (recall).

The transducers are competitive with the state-of-the-art, especially when using VB. VB smooths the parameter estimates, so there are no parse failures in the test set due to unseen words or functions; precision, recall, and f-score all reduce to raw accuracy. The basic transducer with VB has higher accuracy (recall) than all other models except for UBL-s, which does better on Japanese. The head-switch transducer is better still, with the highest recall on both languages. Although the improvement over the basic transducer is small, we anticipate that using the transducer framework will allow us to easily explore many other possible extensions that could increase performance further.

As expected, the basic EM-trained transducer gets numbers that are similar, though not identical, to Lu-uni. The main reason for the discrepancy is that Lu et al. (2008) use custom smoothing methods for the source tree language model and word probabilities. While these could be emulated in a transducer, we instead use a more general approach, VB, with better pay-off. Lu-uni was the simplest model presented by Lu et al. (2008), yet applying VB to our transducer implementation yields a fully generative model whose performance rivals their best-performing system that uses discriminative reranking.

10 Conclusion

In this paper, we have shown how to formulate semantic parsing as tree transduction. This formulation is more general than previous approaches and allows us to exploit the rich literature on transducers, including theoretical results as well as standard algorithms and toolkits. We focused here on extended left hand side, root-to-frontier, linear, nondeleting, tree-to-string transducers (Maletti et al., 2009), using them to reformulate and extend an existing model (Lu et al., 2008). Although we tried only one simple extension, our purely generative model already outperforms all previous models on raw accuracy, with comparable f-score. Since the transducer framework makes modifications easy, we anticipate further gains in future, especially if we add a discriminative reranking step as in Lu et al. (2008). We also hope to investigate other transducer classes. Finally, we note that working with a general framework encourages the development of algorithms that are widely applicable, even if developed for a particular application. The VB training algorithm presented here is just one example of such a contribution.

Acknowledgments

We would like to thank Wei Lu and Jon May for generously providing source code and support for the hybrid tree parser and Tiburon, respectively. Also, this work was supported under the Australian Research Council's Discovery Projects funding scheme (project number DP110102506).

References

Matthew J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience unit, University College London, 2003.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Benjamin Borschinger, Bevan K. Jones, and Mark Johnson. Reducing grounded learning tasks to grammatical inference. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2011.

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

Saul Gorn. Processors for infinite codes of shannon-fano type. In *Symp. Math. Theory of Automata*, 1962.

Jonathon Graehl, Kevin Knight, and Jon May. Training tree transducers. *Computational Linguistics*, 34, 2008.

Rohit J. Kate and Raymond J. Mooney. Using string-kernels for learning semantic parsers. In *Proc. of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 913–920, 2006.

Kevin Knight and Jonathon Greahl. An overview of probabilistic tree transducers for natural language processing. In *Proc. of the 6th International Conference on Intelligent Text Processing and Computational Linguistics*, 2005.

Kenichi Kurihara and Taisuke Sato. Variational bayesian grammar induction for natural language. In *Proc. of the 8th International Colloquium on Grammatical Inference*, 2006.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2010.

Percy Liang, Michael I. Jordan, and Dan Klein. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, 2011.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. A generative model for parsing natural language to meaning representations. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2008.

Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. The power of extended top-down tree transducers. *SIAM J. Comput.*, 39:410–430, June 2009.

Jon May and Kevin Knight. Tiburon: A weighted tree automata toolkit. In *Proc. of International Conference on Implementation and Application of Automata*, 2006.

W.C. Rounds. Mappings and grammars on trees. *Mathematical Systems Theory 4*, pages 257–287, 1970.

J.W. Thatcher. Generalized sequential machine maps. *J. Comput. System Sci. 4*, pages 339–367, 1970.

Yuk Wah Wong and Raymond J. Mooney. Learning for semantic parsing with statistical machine translation. In *Proc. of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting*, pages 439–446, New York City, NY, 2006.

Luke. S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.