Semantic Parsing with Bayesian Tree Transducers

Bevan Keeley Jones[*†] b.k.jones@sms.ed.ac.uk
Mark Johnson[†] Mark.Johnson@mq.edu.au
Sharon Goldwater[*] sgwater@inf.ed.ac.uk

[*] School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK

[†] Department of Computing
Macquarie University
Sydney, NSW 2109, Australia

Abstract

Many semantic parsing models use tree transformations to map between natural language and meaning representation. However, while tree transformations are central to several state-of-the-art approaches, little use has been made of the rich literature on tree automata. This paper makes the connection concrete with a tree transducer based semantic parsing model and suggests that other models can be interpreted in a similar framework, increasing the generality of their contributions. In particular, this paper further introduces a variational Bayesian inference algorithm that is applicable to a wide class of tree transducers, producing state-of-the-art semantic parsing results while remaining applicable to any domain employing probabilistic tree transducers.

1 Introduction

Semantic parsing is the task of mapping natural language sentences to a formal representation of meaning. Typically, a system is trained on pairs of natural language sentences (NLs) and their meaning representation expressions (MRs), as in figure 1(a), and the system must generalize to novel sentences.

Most semantic parsing models rely on an assumption of structural similarity between MR and NL. Since strict isomorphism is overly restrictive, this assumption is often relaxed by applying transformations. Several approaches assume a tree structure to the NL, MR, or both (Ge and Mooney, 2005; Kate and Mooney, 2006; Wong and Mooney, 2006; Lu et al., 2008; Börschinger et al., 2011), and often involve tree transformations either between two trees or a tree and a string.

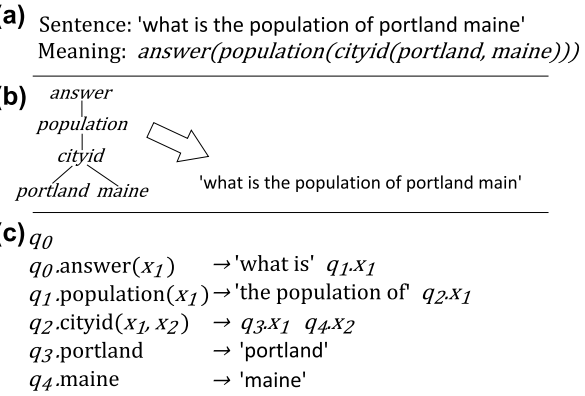

Figure 1: (a) An example sentence/meaning pair, (b) a tree transformation based mapping, and (c) a tree transducer that performs the mapping.

The tree transducer, a formalism from automata theory which has seen interest in machine translation (Yamada and Knight, 2001; Graehl et al., 2008) and has potential applications in many other areas, is well suited to formalizing such tree transformation based models. Yet, while many semantic parsing systems resemble the formalism, each was proposed as an independent model requiring custom algorithms, leaving it unclear how developments in one line of inquiry relate to others. We argue for a unifying theory of tree transformation based semantic parsing by presenting a tree transducer model and drawing connections to other similar systems.

We make a further contribution by bringing to tree transducers the benefits of the Bayesian framework for principled handling of data sparsity and

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 488–496,
Jeju, Republic of Korea, 8-14 July 2012.

prior knowledge. Graehl et al. (2008) present an EM training procedure for top down tree transducers, but while there are Bayesian approaches to string transducers (Chiang et al., 2010) and PCFGs (Kurihara and Sato, 2006), there has yet to be a proposal for Bayesian inference in *tree* transducers. Our variational algorithm produces better semantic parses than EM while remaining general to a broad class of transducers appropriate for other domains.

In short, our contributions are three-fold: we present a new state-of-the-art semantic parsing model, propose a broader theory for tree transformation based semantic parsing, and present a general inference algorithm for the tree transducer framework. We recommend the last of these as just one benefit of working within a general theory: contributions are more broadly applicable.

2 Meaning representations and regular tree grammars

In semantic parsing, an MR is typically an expression from a machine interpretable language (e.g., a database query language or a logical language like Prolog). In this paper we assume MRs can be represented as trees, either by pre-parsing or because they are already trees (often the case for functional languages like LISP).[1] More specifically, we assume the MR language is a regular tree language.

A regular tree grammar (RTG) closely resembles a context free grammar (CFG), and is a way of describing a language of trees. Formally, define T_Σ as the set of trees with symbols from alphabet Σ, and $T_\Sigma(\mathcal{A})$ as the set of all trees in $T_{\Sigma\cup\mathcal{A}}$ where symbols from $\mathcal{A}$ only occur at the leaves. Then an RTG is a tuple $(\mathcal{Q}, \Sigma, q_{start}, \mathcal{R})$, where $\mathcal{Q}$ is a set of states, Σ is an alphabet, $q_{start} \in \mathcal{Q}$ is the initial state, and $\mathcal{R}$ is a set of grammar rules of the form $q \to t$, where q is a state from $\mathcal{Q}$ and t is a tree from $T_\Sigma(\mathcal{Q})$.

A rule typically consists of a parent state (left) and its child states and output symbol (right). We indicate states using all capital letters:

$$\text{NUM} \to \text{population}(\text{PLACE}).$$

Intuitively, an RTG is a CFG where the yield of every parse is itself a tree. In fact, for any CFG G, it is straightforward to produce a corresponding RTG that generates the set of parses of G. Consequently, while we assume we have an RTG for the MR language, there is no loss of generality if the MR language is actually context free.

[1] See Liang et al. (2011) for work in representing lambda calculus expressions with trees.

3 Weighted root-to-frontier, linear, non-deleting tree-to-string transducers

Tree transducers (Rounds, 1970; Thatcher, 1970) are generalizations of finite state machines that operate on trees. Mirroring the branching nature of its input, the transducer may simultaneously transition to several successor states, assigning a separate state to each subtree.

There are many classes of transducer with different formal properties (Knight and Greahl, 2005; Maletti et al., 2009). Figure 1(c) is an example of a root-to-frontier, linear, non-deleting tree-to-string transducer. It is defined using 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 portion of the output string. Variables x_i stand for entire sub-trees, and state-variable pairs $q_j.x_i$ stand for strings produced by applying the transducer starting at state q_j to subtree x_i. Figure 1(b) illustrates an application of the transducer, taking the tree on the left as input and outputting the string on the right.

Formally, a weighted root-to-frontier, tree-to-string transducer is a 5-tuple $(\mathcal{Q}, \Sigma, \Delta, q_{start}, \mathcal{R})$. $\mathcal{Q}$ is a finite set of states, Σ and Δ are the input and output alphabets, q_{start} is the start state, and $\mathcal{R}$ is the set of rules. Denote a pair of symbols, a and b by $a.b$, the cross product of two sets $\mathcal{A}$ and $\mathcal{B}$ by $\mathcal{A}.\mathcal{B}$, and let $\mathcal{X}$ be the set of variables $\{x_0, x_1, ...\}$. Then, each rule $r \in \mathcal{R}$ is of the form $[q.t \to u].v$, where $v \in \Re^{\geq 0}$ is the rule weight, $q \in \mathcal{Q}$, $t \in T_\Sigma(\mathcal{X})$, and u is a string in $(\Delta \cup \mathcal{Q}.\mathcal{X})^*$ such that every $x \in \mathcal{X}$ in u also occurs in t.

We say $q.t$ is the left hand side of rule r and u its 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. In this paper we assume that every tree t on the left hand side is either a single variable x_0 or of the form $\sigma(x_0, ...x_n)$, where $\sigma \in \Sigma$ (i.e., it is a tree of depth ≤ 1).

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 mapping input trees into output strings. A joint distribution allows us to model with a single machine both the input and output languages, which is important during decoding when we want to infer the input given the output.

4 A generative model of semantic parsing

Like the hybrid tree semantic parser (Lu et al., 2008) and the synchronous grammar based WASP (Wong and Mooney, 2006), our model simultaneously generates the input MR tree and the output NL string. The MR tree is built up according to the provided MR grammar, one grammar rule at a time. Coupled with the application of the MR rule, similar CFG-like productions are applied to the NL side, repeated until both the MR and NL are fully generated. In each step, we select an MR rule and then build the NL by first choosing a pattern with which to expand it and then filling out that pattern with words drawn from a unigram distribution.

This kind of coupled generative process can be naturally formalized with tree transducer rules, where the input tree fragment on the left side of each rule describes the derivation of the MR and the right describes the corresponding NL derivation.

For a simple example of a tree-to-string transducer rule consider

$$q.\text{population}(x_1) \rightarrow \text{`population of'}\ \ q.x_1 \qquad (1)$$

which simultaneously generates tree fragment *population(x_1)* on the left and sub-string "population of $q.x_1$" on the right. Variable x_1 stands for an MR subtree under *population*, and, on the right, state-variable pair $q.x_1$ stands for the NL substring generated while processing subtree x_1 starting from q. While this rule can serve as a single step of an MR-to-NL map such as the example transducer shown in Figure 1(c), such rules do not model the grammaticality of the MR and lack flexibility since sub-strings corresponding to a given tree fragment must be completely pre-specified. Instead, we break transductions down into a three stage process of choosing the (i) MR grammar rule, (ii) NL expansion pattern, and (iii) individual words according to a unigram distribution. Such a decomposition incorporates independence assumptions that improve generalizability. See Figure 2 for example rules from our transducer and Figure 3 for a derivation.

To ensure that only grammatical MRs are generated, each state of our transducer encodes the identity of exactly one MR grammar rule. Transitions between q^{MR} and q^{NL} states implicitly select the embedded rule. For instance, rule 2 in Figure 2 selects

$$\begin{aligned}
\text{NUM} &\rightarrow \text{population}(\text{PLACE}) && (m)\\
\text{PLACE} &\rightarrow \text{cityid}(\text{CITY}, \text{STATE}) && (r)\\
\text{CITY} &\rightarrow \text{portland} && (u)\\
\text{STATE} &\rightarrow \text{maine} && (v)
\end{aligned}$$

$$\begin{aligned}
q^{\text{MR}}_{m,1}.x_1 &\rightarrow q^{\text{NL}}_{r}.x_1 && (2)\\
q^{\text{MR}}_{r,1}.x_1 &\rightarrow q^{\text{NL}}_{u}.x_1\\
q^{\text{MR}}_{r,2}.x_1 &\rightarrow q^{\text{NL}}_{v}.x_1
\end{aligned}$$

$$q^{\text{NL}}_{m}.\text{population}(w_1, x_1, w_2) \rightarrow q^{\text{W}}_{m}.w_1\ q^{\text{MR}}_{m,1}.x_1\ q^{END}.w_2 \qquad (3)$$

$$q^{\text{NL}}_{r}.\text{cityid}(w_1, x_1, w_2, x_2, w_3) \rightarrow q^{END}.w_1\ q^{\text{MR}}_{r,2}.x_2\ q^{\text{W}}_{r}.w_2\ q^{\text{MR}}_{r,1}.x_1\ q^{END}.w_3 \qquad (4)$$

$$\begin{aligned}
q^{\text{W}}_{m}.w_1 &\rightarrow \text{`population'}\ \ q^{\text{W}}_{m}.w_1 && (5)\\
q^{\text{W}}_{m}.w_1 &\rightarrow \text{`of'}\ \ q^{\text{W}}_{m}.w_1\\
q^{\text{W}}_{m}.w_1 &\rightarrow \text{...}\ \ q^{\text{W}}_{m}.w_1\\
q^{\text{W}}_{m}.w_1 &\rightarrow \text{`of'}\ \ q^{\text{END}}.w_1 && (6)\\
q^{\text{W}}_{m}.w_1 &\rightarrow \text{...}\ \ q^{\text{END}}.w_1\\
q^{\text{END}}.W &\rightarrow \epsilon && (7)
\end{aligned}$$

Figure 2: Examples of transducer rules (bottom) that generate MR and NL associated with MR rules m-v (top). Transducer rule 2 selects MR rule r from the MR grammar. Rule 3 simultaneously writes the MR associated with rule m and chooses an NL pattern (as does 4 for r). Rules 5-7 generate the words associated with m according to a unigram distribution specific to m.

MR grammar rule r to expand the i^{th} child of the parent produced by rule m. Aside from ensuring the *grammaticality* of the generated MR, rules of this type also model the *probability* of the MR, conditioning the probability of a rule both on the parent rule and the index of the child being expanded. Thus, parent state $q^{\text{MR}}_{m,1}$ encodes not only the identity of rule m, but also the child index, 1 in this case.

Once the MR rule is selected, q^{NL} states are applied to select among rules such as 3 and 4 to generate the MR entity and choose the NL expansion pattern. These rules determine the word order of the language by deciding (i) whether or not to generate words in a given location and (ii) where to insert the result of processing each MR subtree. Decision (i) is made by either transitioning to state q^{W}_r to generate words or to q^{END} to generate the empty string. Decision (ii) is made with the order of x_i's on the right hand side. Rule 4 illustrates the case where *portland* and *maine* in *cityid(portland, maine)* would be realized in reverse order as "maine ... portland".

The particular set of patterns that appear on the right of rules such as 3 embodies the binary word attachment decisions and the particular permutation of x_i in the NL. We allow words to be generated at the beginning and end of each pattern and between the x_is. Thus, rule 4 is just one of 16 such possible patterns (3 binary decisions and 2 permutations), while rule 3 is one of 4. We instantiate all such rules and allow the system to learn weights for them according to the language of the training data.

Finally, the NL is filled out with words chosen according to a unigram distribution, implemented in a PCFG-like fashion, using a different rule for each word which recursively chooses the next word until a string termination rule is reached.[2] Generating word sequence "population of" entails first choosing rule 5 in Figure 2. State q^{W}_r is then recursively applied to choose rule 6, generating "of" at the same time as deciding to terminate the string by transitioning to a new state q^{END} which deterministically concludes by writing the empty string ϵ.

On the MR side, rules 5-7 do very little: the tree on the left side of rules 5 and 6 consists entirely of a subtree variable w_1, indicating that nothing is generated in the MR. Rule 7 subsequently generates these subtrees as W symbols, marking corresponding locations where words might be produced in the NL, which are later removed during post processing.[3]

Figure 3(b) illustrates the coupled generative process. At each step of the derivation, an MR rule is chosen to expand a node of the MR tree, and then a corresponding part of the NL is expanded. Step 1.1 of the example chooses MR rule m, *NUM* $\rightarrow$ *population*(*PLACE*). Transducer rule 3 then generates *population* in the MR (shown in the left column) at the same time as choosing an NL expansion pattern (Step 1.2) which is subsequently filled out with specific words "population" (1.3) and "of" (1.4).

This coupled derivation can be represented by a tree, shown in Figure 3(c), which explicitly represents the dependency structure of the coupled MR and NL (a simplified version is shown in (d) for clarity). In our transducer, which defines a joint distribution over both the MR and NL, the probability of a rule is conditioned on the parent state. Since each state encodes an MR rule, MR rule specific distributions are learned for both the words and their order.

5 Relation to existing models

The tree transducer model can be viewed either as a generative procedure for building up two separate structures or as a transformative machine that takes one as input and produces another as output. Different semantic parsing approaches have taken one or the other view, and both can be captured in this single framework.

WASP (Wong and Mooney, 2006) is an example of the former perspective, coupling the generation of the MR and NL with a synchronous grammar, a formalism closely related to tree transducers. The most significant difference from our approach is that they use machine translation techniques for automatically extracting rules from parallel corpora; similar techniques can be applied to tree transducers (Galley et al., 2004). In fact, synchronous grammars and tree transducers can be seen as instances of the same more general class of automata (Shieber,

[2]There are roughly 25,000 rules in the transducers in our experiments, and the majority of these implement the unigram word distributions since every entity in the MR may potentially produce any of the words it is paired with in training.

[3]The addition of W symbols is a convenience; it is easier to design transducer rules where every substring on the right side corresponds to a subtree on the left.

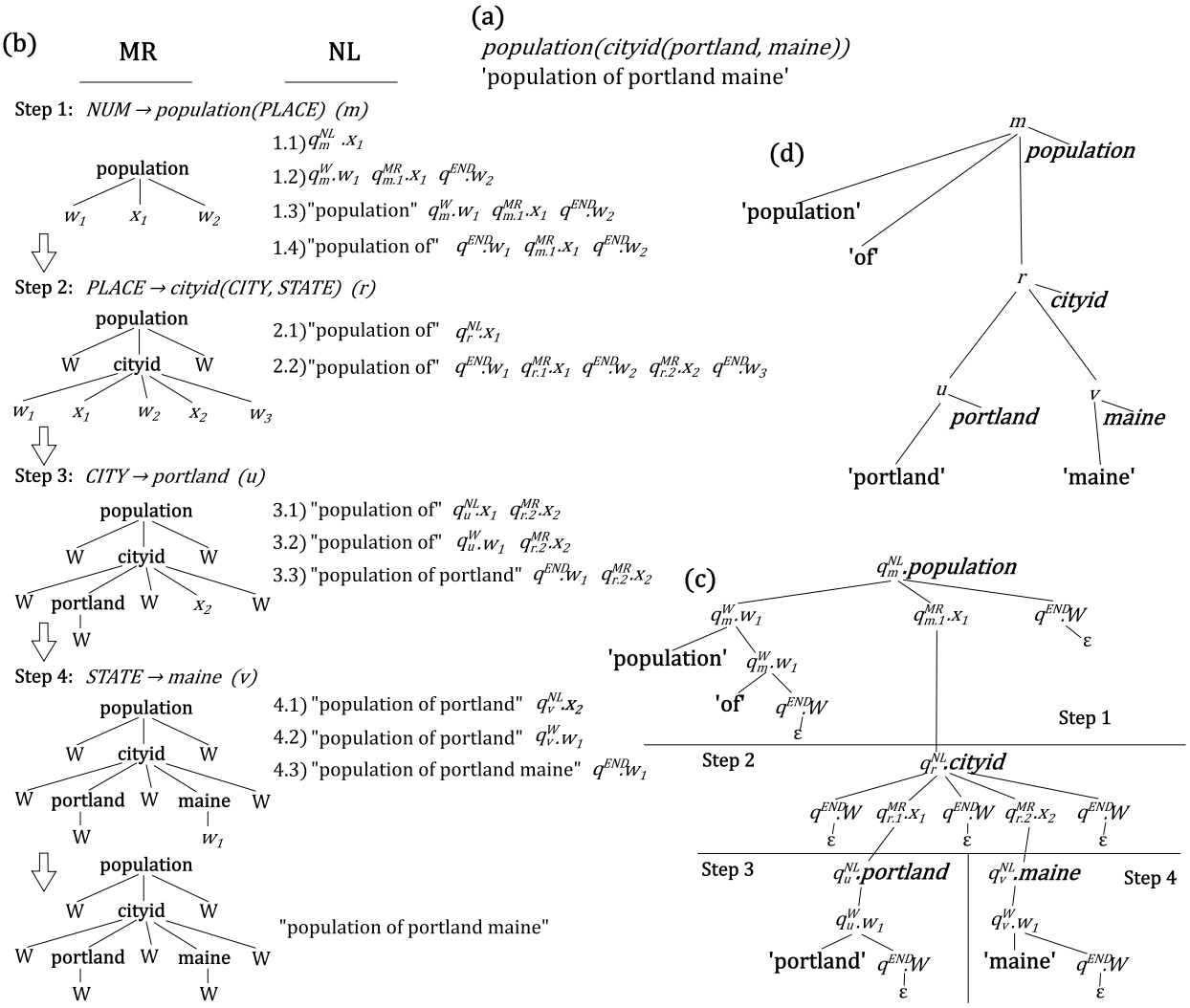

Figure 3: Coupled derivation of an (MR, NL) pair. At each step an MR grammar rule is chosen to expand the MR and the corresponding portion of the NL is then generated. Symbols W stand for locations in the tree corresponding to substrings of the output and are removed in a post-processing step. (a) The (MR, NL) pair. (b) Step by step derivation. (c) The same derivation shown in tree form. (d) The underlying dependency structure of the derivation.

2004). Rather than argue for one or the other, we suggest that other approaches could also be interpreted in terms of general model classes, grounding them in a broader base of theory.

The hybrid tree model (Lu et al., 2008) takes a transformative perspective that is in some ways more similar to our model. In fact, there is a one-to-one relationship between the multinomial parameters of the two models. However, they represent the MR and NL with a single tree and apply tree walking algorithms to extract them. Furthermore, they implement a custom training procedure for searching over the potential MR transformations. The tree transducer, on the other hand, naturally captures the same probabilistic dependencies while maintaining the separation between MR and NL, and further allows us to build upon a larger body of theory.

KRISP (Kate and Mooney, 2006) uses string classifiers to label substrings of the NL with entities from the MR. To focus search, they impose an ordering constraint based on the structure of the MR tree, which they relax by allowing the re-ordering of sibling nodes and devise a procedure for recovering the MR from the permuted tree. This procedure corresponds to backward-application in tree transducers, identifying the most likely input tree given a

particular output string.

SCISSOR (Ge and Mooney, 2005) takes syntactic parses rather than NL strings and attempts to translate them into MR expressions. While few semantic parsers attempt to exploit syntactic information, there are techniques from machine translation for using tree transducers to map between parsed parallel corpora, and these techniques could likely be applied to semantic parsing.

Börschinger et al. (2011) argue for the PCFG as an alternative model class, permitting conventional grammar induction techniques, and tree transducers are similar enough that many techniques are applicable to both. However, the PCFG is less amenable to conceptualizing correspondences between parallel structures, and their model is more restrictive, only applicable to domains with finite MR languages, since their non-terminals encode entire MRs. The tree transducer framework, on the other hand, allows us to condition on individual MR rules.

6 Variational Bayes for tree transducers

As seen in the example in Figure 3(c), tree transducers not only operate on trees, their derivations are themselves trees, making them amenable to dynamic programming and an EM training procedure resembling inside-outside (Graehl et al., 2008). EM assigns zero probability to events not seen in the training data, however, limiting the ability to generalize to novel items. The Bayesian framework offers an elegant solution to this problem, introducing a prior over rule weights which simultaneously ensures that all rules receive non-zero probability and allows the incorporation of prior knowledge and intuitions. Unfortunately, the introduction of a prior makes exact inference intractable, so we use an approximate method, variational Bayesian inference (Bishop, 2006), deriving an algorithm similar to that for PCFGs (Kurihara and Sato, 2006).

The tree transducer defines a joint distribution over the input y, output w, and their derivation x as the product of the weights of the rules appearing in x. That is,

$$p(y, x, w|\theta) = \prod_{r \in \mathcal{R}} \theta(r)^{c_r(x)}$$

where θ is the set of multinomial parameters, r is a transducer rule, $\theta(r)$ is its weight, and $c_r(x)$ is the number of times r appears in x. In EM, we are interested in the point estimate for θ that maximizes $p(\mathcal{Y}, \mathcal{W}|\theta)$, where $\mathcal{Y}$ and $\mathcal{W}$ are the N input-output pairs in the training data. In the Bayesian setting, however, we place a symmetric Dirichlet prior over θ and estimate a posterior distribution over both $\mathcal{X}$ and θ.

$$\begin{aligned} p(\theta, \mathcal{X}|\mathcal{Y}, \mathcal{W}) &= \frac{p(\mathcal{Y}, \mathcal{X}, \mathcal{W}, \theta)}{p(\mathcal{Y}, \mathcal{W})} \\ &= \frac{p(\theta) \prod_{i=1}^{N} p(y_i, x_i, w_i|\theta)}{\int p(\theta) \prod_{i=1}^{N} \sum_{x \in \mathcal{X}_i} p(y_i, x, w_i|\theta) d\theta} \end{aligned}$$

Since the integral in the denominator is intractable, we look for an appropriate approximation $q(\theta, \mathcal{X}) \approx p(\theta, \mathcal{X}|\mathcal{Y}, \mathcal{W})$. In particular, we assume the rule weights and the derivations are independent, i.e., $q(\theta, \mathcal{X}) = q(\theta)q(\mathcal{X})$. The basic idea is then to define a lower bound $\mathcal{F} \leq \ln p(\mathcal{Y}, \mathcal{W})$ in terms of q and then apply the calculus of variations to find a q that maximizes $\mathcal{F}$.

$$\begin{aligned} \ln p(\mathcal{Y}, \mathcal{W}|\alpha) &= \ln E_q[\frac{p(\mathcal{Y}, \mathcal{X}, \mathcal{W}|\theta)}{q(\theta, \mathcal{X})}] \\ &\geq E_q[\ln \frac{p(\mathcal{Y}, \mathcal{X}, \mathcal{W}|\theta)}{q(\theta, \mathcal{X})}] = \mathcal{F}, \end{aligned}$$

Applying our independence assumption, we arrive at the following expression for $\mathcal{F}$, where θ_t is the particular parameter vector corresponding to the rules with parent state t:

$$\begin{aligned} \mathcal{F} = &\sum_{t \in \mathcal{Q}} \left(E_{q(\theta_t)}[\ln p(\theta_t|\alpha_t)] - E_{q(\theta_t)}[\ln q(\theta_t)] \right) \\ &+ \sum_{i=1}^{N} \left(E_q[\ln p(w_i, x_i, y_i|\theta)] - E_{q(x_i)}[\ln q(x_i)] \right). \end{aligned}$$

We find the $q(\theta_t)$ and $q(x_i)$ that maximize $\mathcal{F}$ by taking derivatives of the Lagrangian, setting them to zero, and solving, which yields:

$$\begin{aligned} q(\theta_t) &= \text{Dirichlet}(\theta_t|\hat{\alpha}_t) \\ q(x_i) &= \frac{\prod_{r \in \mathcal{R}} \hat{\theta}(r)^{c_r(x_i)}}{\sum_{x \in \mathcal{X}_i} \prod_{r \in \mathcal{R}} \hat{\theta}(r)^{c_r(x)}} \end{aligned}$$

where

$$\begin{aligned} \hat{\alpha}(r) &= \alpha(r) + \sum_i E_{q(x_i)}[c_r(x_i)] \\ \hat{\theta}(r) &= \exp\left(\Psi(\hat{\alpha}(r)) - \Psi(\sum_{r:s(r)=t} \hat{\alpha}(r)) \right). \end{aligned}$$

The parameters of $q(\theta_t)$ are defined with respect to $q(x_i)$ and the parameters of $q(x_i)$ with respect to the parameters of $q(\theta_t)$. $q(x_i)$ can be computed efficiently using inside-outside. Thus, we can perform an EM-like alternation between calculating $\hat{\alpha}$ and $\hat{\theta}$.[4]

It is also possible to estimate the hyper-parameters α from data, a practice known as *empirical Bayes*, by optimizing $\mathcal{F}$. We explore learning separate hyper-parameters α_t for each θ_t, using a fixed point update described by Minka (2000), where k_t is the number of rules with parent state t:

$$\alpha_t' = \left(\frac{1}{\alpha_t} + \frac{1}{k_t \alpha_t^2} \left(\frac{\partial^2 \mathcal{F}}{\partial \alpha_t^2} \right)^{-1} \left(\frac{\partial \mathcal{F}}{\partial \alpha_t} \right) \right)^{-1}$$

7 Training and decoding

We implement our VB training algorithm inside the tree transducer package Tiburon (May and Knight, 2006), and experiment with both manually set and automatically estimated priors. For our manually set priors, we explore different hyper-parameter settings for three different priors, one for each of the main decision types: MR rule, NL pattern, and word generation. For the automatic priors, we estimate separate hyper-parameters for each multinomial (of which there are hundreds). As is standard, we initialize the word distributions using a variant of IBM model 1, and make use of NP lists (a manually created list of the constants in the MR language paired with the words that refer to them in the corpus).

At test time, since finding the most probable *MR* for a sentence involves summing over all possible derivations, we instead find the MR associated with the most probable *derivation*.

8 Experimental setup and evaluation

We evaluate the system on GeoQuery (Wong and Mooney, 2006), a parallel corpus of 880 English questions and database queries about United States geography, 250 of which were translated into Spanish, Japanese, and Turkish. We present here additional translations of the full 880 sentences into German, Greek, and Thai. For evaluation, following from Kwiatkowski et al. (2010), we reserve 280 sentences for test and train on the remaining 600. During development, we use cross-validation on the 600 sentence training set. At test, we run once on the remaining 280 and perform 10 fold cross-validation on the 250 sentence sets.

To judge correctness, we follow standard practice and submit each parse as a GeoQuery database query, and say the parse is correct only if the answer matches the gold standard. We report raw accuracy (the percentage of sentences with correct answers), as well as F1: the harmonic mean of precision (the proportion of correct answers out of sentences with a parse) and recall (the proportion of correct answers out of all sentences).[5]

We run three other state-of-the-art systems for comparison. *WASP* (Wong and Mooney, 2006) and the *hybrid tree* (Lu et al., 2008) are chosen to represent tree transformation based approaches, and, while this comparison is our primary focus, we also report *UBL-S* (Kwiatkowski et al., 2010) as a non-tree based top-performing system.[6] The hybrid tree is notable as the only other system based on a generative model, and *uni-hybrid*, a version that uses a unigram distribution over words, is very similar to our own model. We also report the best performing version, *re-hybrid*, which incorporates a discriminative re-ranking step.

We report transducer performance under three different training conditions: *tsEM* using EM, *tsVB-auto* using VB with empirical Bayes, and *tsVB-hand* using hyper-parameters manually tuned on the German training data (α of 0.3, 0.8, and 0.25 for MR rule, NL pattern, and word choices, respectively).

Table 1 shows results for 10 fold cross-validation on the training set. The results highlight the benefit of the Dirichlet prior, whether manually or automatically set. VB improves over EM considerably, most likely because (1) the handling of unknown words and MR entities allows it to return an analysis for all sentences, and (2) the sparse Dirichlet prior favors fewer rules, reasonable in this setting where only a few words are likely to share the same meaning.

[4] Because of the resemblance to EM, this procedure has been called VBEM. Unlike EM, however, this procedure alternates between two estimation steps and has no maximization step.

[5] Note that accuracy and f-score reduce to the same formula if there are no parse failures.

[6] UBL-S is based on CCG, which can be viewed as a mapping between graphs more general than trees.

DEV	geo600 - 10 fold cross-val			
	German		Greek	
	Acc	F1	Acc	F1
UBL-S	76.7	76.9	76.2	76.5
WASP	66.3	75.0	71.2	79.7
uni-hybrid	61.7	66.1	71.0	75.4
re-hybrid	62.3	69.5	70.2	76.8
tsEM	61.7	67.9	67.3	73.2
tsVB-auto	74.0	74.0	•**79.8**	•**79.8**
tsVB-hand	•**78.0**	•**78.0**	79.0	79.0
	English		Thai	
UBL-S	**85.3**	**85.4**	74.0	74.1
WASP	73.5	79.4	69.8	73.9
uni-hybrid	76.3	79.0	71.3	73.7
re-hybrid	77.0	82.2	71.7	76.0
tsEM	73.5	78.1	69.8	72.9
tsVB-auto	81.2	81.2	74.7	74.7
tsVB-hand	•83.7	•83.7	•**76.7**	•**76.7**

Table 1: Accuracy and F1 score comparisons on the geo600 training set. Highest scores are in bold, while the highest among the *tree based* models are marked with a bullet. The dotted line separates the tree based from non-tree based models.

TEST	geo880 - 600 train/280 test			
	German		Greek	
	Acc	F1	Acc	F1
UBL-S	**75.0**	**75.0**	73.6	73.7
WASP	65.7	• 74.9	70.7	• **78.6**
re-hybrid	62.1	68.5	69.3	74.6
tsVB-hand	• 74.6	74.6	•**75.4**	75.4
	English		Thai	
UBL-S	**82.1**	**82.1**	66.4	66.4
WASP	71.1	77.7	71.4	75.0
re-hybrid	76.8	• 81.0	73.6	76.7
tsVB-hand	• 79.3	79.3	• **78.2**	• **78.2**
geo250 - 10 fold cross-val				
	English		Spanish	
UBL-S	80.4	80.6	79.7	80.1
WASP	70.0	80.8	72.4	81.0
re-hybrid	74.8	82.6	78.8	• **86.2**
tsVB-hand	• **83.2**	• **83.2**	• **80.0**	80.0
	Japanese		Turkish	
UBL-S	**80.5**	80.6	74.2	74.9
WASP	74.4	• **82.9**	62.4	75.9
re-hybrid	76.8	82.4	66.8	• **77.5**
tsVB-hand	• 78.0	78.0	• **75.6**	75.6

Table 2: Accuracy and F1 score comparisons on the geo880 and geo250 test sets. Highest scores are in bold, while the highest among the *tree based* models are marked with a bullet. The dotted line separates the tree based from non-tree based models.[7]

On the test set (Table 2), we only run the model variants that perform best on the training set. Test set accuracy is consistently higher for the VB trained tree transducer than the other tree transformation based models (and often highest overall), while f-score remains competitive.[7]

9 Conclusion

We have argued that tree transformation based semantic parsing can benefit from the literature on formal language theory and tree automata, and have taken a step in this direction by presenting a tree transducer based semantic parser. Drawing this connection facilitates a greater flow of ideas in the research community, allowing semantic parsing to leverage ideas from other work with tree automata, while making clearer how seemingly isolated efforts might relate to one another. We demonstrate this by both building on previous work in training tree transducers using EM (Graehl et al., 2008), and describing a general purpose variational inference algorithm for adapting tree transducers to the Bayesian framework. The new VB algorithm results in an overall performance improvement for the transducer over EM training, and the general effectiveness of the approach is further demonstrated by the Bayesian transducer achieving highest accuracy among other tree transformation based approaches.

[7] Numbers differ slightly here from previously published results due to the fact that we have standardized the inputs to the different systems.

Acknowledgments

We thank Joel Lang, Michael Auli, Stella Frank, Prachya Boonkwan, Christos Christodoulopoulos, Ioannis Konstas, and Tom Kwiatkowski for providing the new translations of GeoQuery. This research was supported in part under the Australian Research Council's Discovery Projects funding scheme (project number DP110102506).

References

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

Benjamin Börschinger, 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.

David Chiang, Jonathan Graehl, Kevin Knight, Adam Pauls, and Sujith Ravi. Bayesian inference for finite-state transducers. In *Proc. of the annual meeting of the North American Association for Computational Linguistics*, 2010.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proc. of the annual meeting of the North American Association for Computational Linguistics*, 2004.

Ruifang Ge and Raymond J. Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Conference on Computational Natural Language Learning*, 2005.

Jonathon Graehl, Kevin Knight, and Jon May. Training tree transducers. *Computational Linguistics*, 34:391–427, 2008.

Rohit J. Kate and Raymond J. Mooney. Using string-kernels for learning semantic parsers. In *Proc. of the International Conference on Computational Linguistics and the annual meeting of the Association for Computational Linguistics*, 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 *Proc. of the annual meeting of the Association for Computational Linguistics*, 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 the International Conference on Implementation and Application of Automata*, 2006.

Tom Minka. Estimating a Dirichlet distribution. Technical report, M.I.T., 2000.

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

Stuart M. Shieber. Synchronous grammars as tree transducers. In *Proc. of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms*, 2004.

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 and the annual meeting of the North American Chapter of the Association for Computational Linguistics*, 2006.

Kenji Yamada and Kevin Knight. A syntax-based statistical translation model. In *Proc. of the annual meeting of the Association for Computational Linguistics*, 2001.