A Probabilistic Forest-to-String Model for Language Generation from Typed Lambda Calculus Expressions

Wei Lu and **Hwee Tou Ng**
Department of Computer Science
School of Computing
National University of Singapore
{luwei,nght}@comp.nus.edu.sg

Abstract

This paper describes a novel probabilistic approach for generating natural language sentences from their underlying semantics in the form of typed lambda calculus. The approach is built on top of a novel reduction-based weighted synchronous context free grammar formalism, which facilitates the transformation process from typed lambda calculus into natural language sentences. Sentences can then be generated based on such grammar rules with a log-linear model. To acquire such grammar rules automatically in an unsupervised manner, we also propose a novel approach with a generative model, which maps from sub-expressions of logical forms to word sequences in natural language sentences. Experiments on benchmark datasets for both English and Chinese generation tasks yield significant improvements over results obtained by two state-of-the-art machine translation models, in terms of both automatic metrics and human evaluation.

1 Introduction

This work focuses on the task of generating natural language sentences from their underlying meaning representations in the form of formal logical expressions (typed lambda calculus). Many early approaches to generation from logical forms make use of rule-based methods (Wang, 1980; Shieber et al., 1990), which concern surface realization (ordering and inflecting of words) but largely ignore lexical acquisition. Recent approaches start to employ corpus-based probabilistic methods, but many of them assume the underlying meaning representations are of specific forms such as variable-free tree-structured representations (Wong and Mooney, 2007a; Lu et al., 2009) or database entries (Angeli et al., 2010).

While these algorithms usually work well on specific semantic formalisms, it is unclear how well they could be applied to a different semantic formalism. In this work, we propose a general probabilistic model that performs generation from underlying formal semantics in the form of typed lambda calculus expressions (we refer to them as λ-*expressions* throughout this paper), where both lexical acquisition and surface realization are integrated in a single framework.

One natural proposal is to adopt a state-of-the-art statistical machine translation approach. However, unlike text to text translation, which has been extensively studied in the machine translation community, translating from logical forms into text presents additional challenges. Specifically, logical forms such as λ-expressions may have complex internal structures and variable dependencies across sub-expressions. Problems arise when performing automatic acquisition of a translation lexicon, as well as performing lexical selection and surface realization during generation.

In this work, we tackle these challenges by making the following contributions:

- *A novel forest-to-string generation algorithm*: Inspired by the work of Chiang (2007), we introduce a novel reduction-based weighted binary synchronous context-free grammar formalism for generation from logical forms (λ-expressions), which can then be integrated with a probabilistic forest-to-string generation algo-

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1611–1622,
Edinburgh, Scotland, UK, July 27–31, 2011.

rithm.

- *A novel grammar induction algorithm*: To automatically induce such synchronous grammar rules, we propose a novel generative model that establishes phrasal correspondences between logical sub-expressions and natural language word sequences, by extending a previous model proposed for parsing natural language into meaning representations (Lu et al., 2008).

To our best knowledge, this is the first probabilistic model for generating sentences from the lambda calculus encodings of their underlying formal meaning representations, that concerns both surface realization and lexical acquisition. We demonstrate the effectiveness of our model in Section 5.

2 Related Work

The task of language generation from logical forms has a long history. Many early works do not rely on probabilistic approaches. Wang (1980) presented an approach for generation from an extended predicate logic formalism using hand-written rules. Shieber et al. (1990) presented a semantic head-driven approach for generation from logical forms based on rules written in Prolog. Shemtov (1996) presented a system for generation of multiple paraphrases from ambiguous logical forms. Langkilde (2000) presented a probabilistic model for generation from a packed forest meaning representation, without concerning lexical acquisition. Specifically, we are not aware of any prior work that handles both automatic unsupervised lexical acquisition and surface realization for generation from logical forms in a single framework.

Another line of research efforts focused on the task of language generation from other meaning representation formalisms. Wong and Mooney (2007a) as well as Chen and Mooney (2008) made use of synchronous grammars to transform a variable-free tree-structured meaning representation into sentences. Lu et al. (2009) presented a language generation model using the same meaning representation based on tree conditional random fields. Angeli et al. (2010) presented a domain-independent probabilistic approach for generation from database entries. All these models are probabilistic models.

Recently there are also substantial research efforts on the task of mapping natural language to meaning representations in various formalisms – the inverse task of language generation called *semantic parsing*. Examples include Zettlemoyer and Collins (2005; 2007; 2009), Kate and Mooney (2006), Wong and Mooney (2007b), Lu et al. (2008), Ge and Mooney (2009), as well as Kwiatkowski et al. (2010).

Of particular interest is our prior work Lu et al. (2008), in which we presented a joint generative process that produces a hybrid tree structure containing words, syntactic structures, and meaning representations, where the meaning representations are in a variable-free tree-structured form. One important property of the model in our prior work is that it induces a hybrid tree structure automatically in an unsupervised manner, which reveals the correspondences between natural language word sequences and semantic elements. We extend our prior model in the next section, so as to support λ-expressions. The model in turn serves as the basis for inducing the synchronous grammar rules later.

3 λ-Hybrid Tree

In Lu et al. (2008), a generative model was presented to model the process that jointly generates both natural language sentences and their underlying meaning representations of a variable-free tree-structured form. The model was defined over a *hybrid tree*, which consists of meaning representation tokens as internal nodes and natural language words as leaves. One limitation of the hybrid tree model is that it assumes a single fixed tree structure for the meaning representation. However, λ-expressions exhibit complex structures and variable dependencies, and thus it is not obvious how to represent them in a single tree structure.

In this section, we present a novel *λ-hybrid tree* model that provides the following extensions over the model of Lu et al. (2008):

1. The internal nodes of a meaning representation tree involve λ-expressions which are not necessarily of variable-free form;
2. The meaning representation has a packed forest representation, rather than a single deterministic tree structure.

3.1 Packed λ-Meaning Forest

We represent a λ-expression with a packed forest of meaning representation trees (called *λ-meaning for-*

est). Multiple different meaning representation trees (called λ-*meaning trees*) can be extracted from the same λ-meaning forest, but they all convey equivalent semantics via reductions, as discussed next.

Constructing a λ-meaning forest for a given λ-expression requires decomposition of a complete λ-expression into semantically complete and syntactically correct sub-expressions in a principled manner. This can be achieved with a process called higher order unification (Huet, 1975). The process was known to be very complex and was shown to be undecidable in unrestricted form (Huet, 1973). Recently a restricted form of higher order unification was applied to a semantic parsing task (Kwiatkowski et al., 2010). In this work, we employ a similar technique for building the λ-meaning forest.

For a given λ-expression e, our algorithm finds either two expressions h and f such that $(h\ f) \equiv e$, or three expressions h, f, and g such that $((h\ f)\ g) \equiv e$, where the symbol $\equiv$ is interpreted as α-*equivalent* after *reductions*[1] (Barendregt, 1985). We then build the λ-meaning forest based on the expressions h, f, and g. In practice, we develop a BUILDFOREST(e) procedure which recursively builds λ-forests by applying restricted higher-order unification rules on top of the λ-expression e. Each node of the λ-forest is called a λ-production, to which we will give more details in Section 3.2. For example, once a candidate triple (h, f, g) as in $((h\ f)\ g) \equiv e$ has been identified, the procedure creates a λ-forest with the root node being a λ-production involving h, and two sets of child λ-forests given by BUILDFOREST(f) and BUILDFOREST(g) respectively. For restricted higher-order unification, besides the similar assumptions made by Kwiatkowski et al. (2010), we also impose one additional assumption: limited free variable, which states that the expression h must contain no more than one free variable. Note that this process provides a semantically equivalent packed forest representation of the original λ-expression, without altering its semantics in any way.

For better readability, we introduce the symbol $\lhd$ as an alternative notation for functional application. In other words, $h \lhd f$ refers to $(h\ f)$ or $h(f)$, and $h \lhd f \lhd g$ refers to $((h\ f)\ g)$. For example, the expression $\lambda x.state(x) \wedge loc(boston, x)$ can be represented as the functional application form of $[\lambda f.\lambda x.f(x) \wedge loc(boston, x)] \lhd \lambda x.state(x)$.[2]

Such a packed forest representation contains exponentially many tree structures which all convey the same semantics. We believe such a semantic representation is more advantageous than the single fixed tree-structured representation. In fact, one could intuitively regard a different decomposition path as a different way of interpreting the same semantics. Thus, such a representation could potentially accommodate a wider range of natural language expressions, which all share the same semantics but with very different word choices, phrase orderings, and syntactic structures (like paraphrases). It may also alleviate the non-isomorphism issue that was commonly faced by researchers when mapping meaning representations and sentences (Wong and Mooney, 2007b). We will validate our belief later through experiments.

3.2 The Joint Generative Process

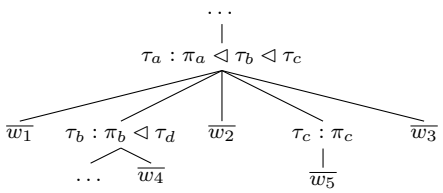

Figure 1: The joint generative process of both λ-meaning tree and its corresponding natural language sentence, which results in a λ-hybrid tree.

The generative process for a sentence together with its corresponding λ-meaning tree is illustrated in Figure 1, which results in a λ-*hybrid tree*. Internal nodes of a λ-hybrid tree are called λ-*productions*, which are building blocks of a λ-forest. Each λ-production in turn has at most two child λ-productions. A λ-production has the form $\tau_a : \pi_a \lhd \overline{\tau_b}$, where τ_a is the expected *type*[3] after type evaluation of the terms to its right, π_a is a λ-expression (serves as the functor), and $\overline{\tau_b}$ are types of the child λ-productions (as the arguments). The leave nodes

[1] In this work, for reductions, we consider α-conversions (changing bound variables) and β-conversions (applying functors to their arguments).

[2] Throughout this paper, we abuse this notation a bit by allowing the arguments to be types rather than actual expressions, such as $\lambda y.\lambda x.loc(y, x)) \lhd e$, which indicates that the functor $\lambda y.\lambda x.loc(y, x)$ expects an expression of type e to serve as its argument.

[3] This work considers basic types: e (entities) and t (truth values). It also allows function types, e.g., $\langle e, t\rangle$ is the type assigned to functions that map from entities to truth values.

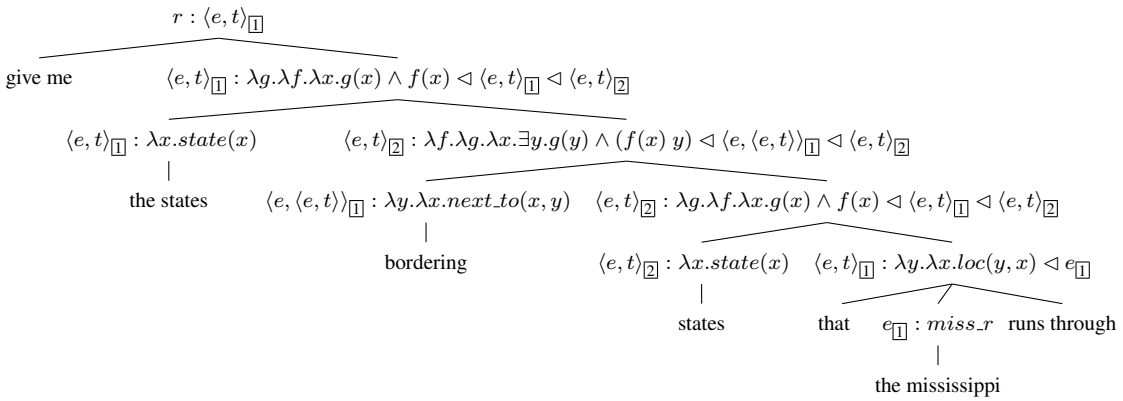

Figure 2: One example λ-hybrid tree for the sentence "give me the states bordering states that the mississippi runs through" together with its logical form "$\lambda x_0.state(x_0) \wedge \exists x_1.[loc(miss\_r, x_1) \wedge state(x_1) \wedge next\_to(x_1, x_0)]$".

$\overline{w}$ are contiguous word sequences. The model repeatedly generates λ-*hybrid sequences*, which consist of words intermixed with λ-productions, from each λ-production at different levels.

Consider part of the example λ-hybrid tree in Figure 2. The probability associated with generation of the subtree that spans the sub-sentence "that the mississippi runs through" can be written as:

$$P\Big(\lambda x.loc(miss\_r, x), \text{that the mississippi runs through}\Big)$$
$$= \phi(m \rightarrow \overline{\mathbf{w}\mathcal{Y}\mathbf{w}}|p_1) \times \psi(\text{that } e_{\boxed{1}} \text{ runs through}|p_1)$$
$$\times \rho(p_2|p_1, \arg_1) \times \phi(m \rightarrow \overline{\mathbf{w}}|p_2) \times \psi(\text{the mississippi}|p_2)$$

where $p_1 = \langle e,t\rangle : \lambda y.\lambda x.loc(y,x) \lhd e_{\boxed{1}}$, and $p_2 = e : miss\_r$.

Following the work of Lu et al. (2008), the generative process involves three types of parameters $\bar{\theta} = \{\phi, \psi, \rho\}$: 1) pattern parameters ϕ, which model in what way the words and child λ-productions are intermixed; 2) emission parameters ψ, which model the generation process of words from λ-productions, where either a unigram or a bigram assumption can be made (Lu et al., 2008); and 3) meaning representation (MR) model parameters ρ, which model the generation process from one λ-production to its child λ-productions. An analogous inside-outside algorithm (Baker, 1979) used there is employed here. Since we allow a packed λ-meaning forest representation rather than a fixed tree structure, the MR model parameters ρ in this work should be estimated with the inside-outside algorithm as well, rather than being estimated directly from the training data by simple counting, as was done in Lu et al. (2008).

4 The Language Generation Algorithm

Now we present the algorithm for language generation. We introduce the grammar first, followed by the features we use. Next, we present the method for grammar induction, and then discuss the decoder.

4.1 The Grammar

We use a weighted synchronous context free grammar (SCFG) (Aho and Ullman, 1969), which was previously used in Chiang (2007) for hierarchical phrase-based machine translation. The grammar is defined as follows:

$$\tau \rightarrow \quad \langle p_\lambda\ , h_w, \sim\rangle \tag{1}$$

where τ is the type associated with the λ-production p_λ[4], and h_w is a sequence consisting of natural language words intermixed with types. The symbol $\sim$ denotes the one-to-one correspondence between nonterminal occurrences (i.e., in this case types of λ-expressions) in both p_λ and h_w.

We allow a maximum of two nonterminal symbols in each synchronous rule, as was also assumed in Chiang (2007), which makes the grammar a binary SCFG. Two example rules are:

$$\langle e,t\rangle \rightarrow \Big\langle \lambda y.\lambda x.loc(y,x) \lhd e_{\boxed{1}}\ ,\ \text{that } e_{\boxed{1}} \text{ runs through} \Big\rangle$$
$$e \rightarrow \Big\langle miss\_r,\ \text{the mississippi} \Big\rangle$$

where the boxed indices give the correspondences between nonterminals.

A derivation with the above two synchronous rules results in the following λ-expression paired with its natural language counterpart:

[4]Since type is already indicated by τ, we avoid redundancy by omitting it when writing p_λ, without loss of information.

Type 1:	$\langle e, \langle e, t\rangle\rangle$	$\rightarrow$	$\left\langle \lambda y.\lambda x.next\_to(x, y)\ ,\ \text{bordering} \right\rangle$
	$\langle e, t\rangle$	$\rightarrow$	$\left\langle \lambda g.\lambda f.\lambda x.g(x) \wedge f(x) \lhd \langle e, t\rangle_{\boxed{1}} \lhd \langle e, t\rangle_{\boxed{2}}\ ,\ \langle e, t\rangle_{\boxed{2}}\ \langle e, t\rangle_{\boxed{1}} \right\rangle$
Type 2:	$\langle e, t\rangle$	$\rightarrow$	$\left\langle \lambda x.loc(miss\_r, x) \wedge state(x)\ ,\ \text{states that the mississippi runs through} \right\rangle$
	$\langle e, t\rangle$	$\rightarrow$	$\left\langle \lambda x.loc(miss\_r, x)\ ,\ \text{that the mississippi runs through} \right\rangle$
Type 3:	$\langle e, t\rangle$	$\rightarrow$	$\left\langle \lambda f.\lambda x.state(x) \wedge \exists y.[f(y) \wedge next\_to(y, x)] \lhd \langle e, t\rangle_{\boxed{1}}\ ,\ \text{the states bordering}\ \langle e, t\rangle_{\boxed{1}} \right\rangle$
	$\langle e, t\rangle$	$\rightarrow$	$\left\langle \lambda y.\lambda x.loc(y, x) \wedge state(x) \lhd e_{\boxed{1}}\ ,\ \text{states that}\ e_{\boxed{1}}\ \text{runs through} \right\rangle$

Figure 3: Example synchronous rules that can be extracted from the λ-hybrid tree of Figure 2.

$$\langle e, t\rangle \rightarrow \left\langle \lambda x.loc(miss\_r, x)\ ,\ \text{that the mississippi runs through} \right\rangle$$

where the source side λ-expression is constructed from the application $\lambda y.\lambda x.loc(y, x) \lhd miss\_r$ followed by a reduction (β-conversion). Assuming the λ-expression to be translated is $\lambda x.loc(miss\_r, x)$, the above rule in fact gives one candidate translation "that the mississippi runs through".

4.2 Features

Following the work of Chiang (2007), we assign scores to derivations with a log-linear model, which are essentially weighted products of feature values.

For generality, we only consider the following four simple features in this work:

1. $\tilde{p}(h_w|p_\lambda)$: the relative frequency estimate of a hybrid sequence h_w given the λ-production p_λ;
2. $\tilde{p}(p_\lambda|h_w, \tau)$: the relative frequency estimate of a λ-production p_λ given the phrase h_w and the type τ;
3. $\exp(-wc(h_w))$: the number of words generated, where $wc(h_w)$ refers to the number of words in h_w (i.e., word penalty); and
4. $p_{LM}(\hat{s})$: the language model score of the generated sentence $\hat{s}$.

The first three features, which are also widely used in state-of-the-art machine translation models (Koehn et al., 2003; Chiang, 2007), are rule-specific and thus can be computed before decoding. The last feature is computed during the decoding phase in combination with the sibling rules used.

We score a derivation D with a log-linear model:

$$\mathbf{w}(D) = \left(\prod_{r \in D} \prod_i f_i(r)^{w_i} \right) \times p_{LM}(\hat{s})^{w_{LM}} \qquad (2)$$

where $r \in D$ refers to a rule r that appears in the derivation D, $\hat{s}$ is the target side (sentence) associated with the derivation D, and f_i is a rule-specific feature (one of features 1–3 above) which is weighted with w_i. The language model feature is weighted with w_{LM}.

Once the feature values are computed, our goal is to find the optimal weight vector $\bar{w}^*$ that maximizes a certain evaluation metric when used for decoding, as we will discuss in Section 4.4.

Following popular approaches to learning feature weights in the machine translation community (Och and Ney, 2004; Chiang, 2005), we use the minimum error rate training (MERT) (Och, 2003) algorithm to learn the feature weights that directly optimize certain automatic evaluation metric. Specifically, the Z-MERT (Zaidan, 2009) implementation of the algorithm is used in this work.

4.3 Grammar Induction

Automatic induction of the grammar rules as described above from training data (which consists of pairs of λ-expressions and natural language sentences) is a challenging task. Current state-of-the-art string-based translation systems (Koehn et al., 2003; Chiang, 2005; Galley and Manning, 2010) typically begin with a word-aligned corpus to construct phrasal correspondences. Word-alignment information can be estimated from alignment models, such as the IBM alignment models (Brown et al., 1993) and HMM-based alignment models (Vogel et al., 1996; Liang et al., 2006). However, unlike texts, logical forms have complex internal structures and variable dependencies across sub-expressions. It is not obvious how to establish alignments between logical terms and texts with such alignment models.

Fortunately, the generative model for λ-hybrid tree introduced in Section 3 explicitly models the mappings from λ-sub-expressions to (possibly discontiguous) word sequences with a joint generative process. This motivates us to extract grammar rules from the λ-hybrid trees. Thus, we first find the Viterbi λ-hybrid trees for all training instances,

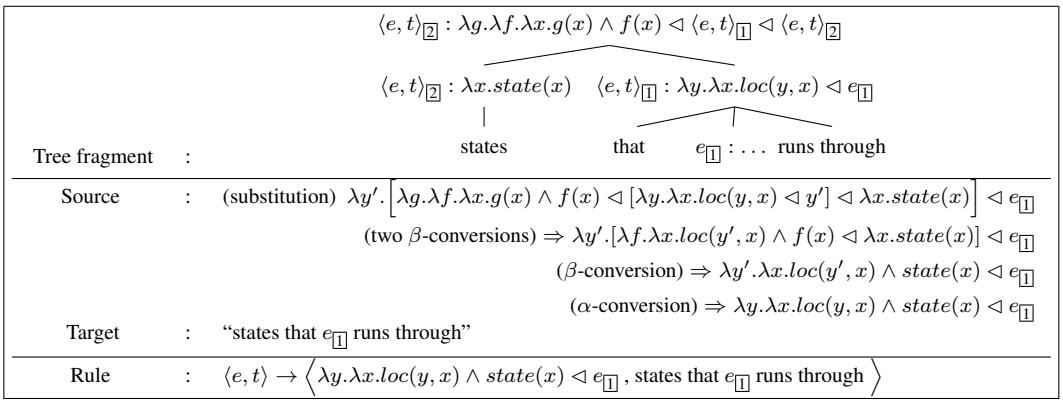

Figure 4: Construction of a two-level λ-hybrid sequence rule via substitution and reductions from a tree fragment. Note that the subtree rooted by $e_{\boxed{1}} : miss\_r$ gets "abstracted" by its type e. The auxiliary variable y' of type e is thus introduced to facilitate the construction process.

based on the learned parameters of the generative λ-hybrid tree model.

Next, we extract grammar rules on top of these λ-hybrid trees. Specifically, we extract the following three types of synchronous grammar rules, with examples given in Figure 3:

1. λ-hybrid sequence rules: They are the conventional rules constructed from one λ-production and its corresponding λ-hybrid sequence.
2. Subtree rules: These rules are constructed from a complete subtree of the λ-hybrid tree. Each rule provides a mapping between a complete sub-expression and a contiguous sub-sentence.
3. Two-level λ-hybrid sequence rules: These rules are constructed from a tree fragment with one of its grandchild subtrees (the subtree rooted by one of its grandchild nodes) being abstracted with its type only. These rules are constructed via substitution and reductions.
   Figure 4 gives an example based on a tree fragment of the λ-hybrid tree in Figure 2. Note that the first step makes use of the auxiliary variable y' of type e to represent the grandchild subtree. $\lambda y'$ is introduced so as to allow any λ-expression of type e serving as this expression's argument to replace y'. In fact, if the semantics conveyed by the grandchild subtree serves as its argument, we will obtain the exact complete semantics of the current subtree. As we can see, the resulting rule is more general, and is able to capture longer structural dependencies. Such rules are thus potentially more useful.

The overall algorithm for learning the grammar rules is sketched in Figure 5.

4.4 Decoding

Our goal in decoding is to find the most probable sentence $\hat{s}$ for a given λ-expression e:

$$\hat{s} = s\left(\underset{D \text{ s.t. } e(D)\equiv e}{\arg\max}\ \mathbf{w}(D)\right) \tag{3}$$

where $e(D)$ refers to the source side (λ-expression) of the derivation D, and $s(D)$ refers to the target side (natural language sentence) of D.

A conventional CKY-style decoder as used by Chiang (2007) is not applicable to this work since the source side does not exhibit a linear structure. As discussed in Section 3.1, λ-expressions are represented as packed λ-meaning forests. Thus, in this work, we make use of a bottom-up dynamic programming chart-parsing algorithm that works directly on translating forest nodes into target natural language words. The algorithm is similar to that of Langkilde (2000) for generation from an underlying packed semantic forest. Language models are incorporated when scoring the n-best candidates at each forest node, where the cube-pruning algorithm of Chiang (2007) is used. In order to accommodate type 2 and type 3 rules as discussed in Section 4.3, whose source side λ-productions are not present in the nodes of the original λ-meaning forest, new λ-productions are created (via substitution and reductions) and attached to the original λ-meaning forest.

Procedures

- $f \leftarrow \text{BUILDFOREST}(e)$
  It takes in a λ-expression e and outputs its λ-meaning forest f. (Sec. 3.1)
- $\bar{\theta} \leftarrow \text{TRAINGENMODEL}(\mathbf{f}, \mathbf{s})$
  It takes in λ-meaning forest-sentence pairs $(\mathbf{f}, \mathbf{s})$, performs EM training of the generative model, and outputs the parameters $\bar{\theta}$. (Sec. 3.2)
- $h \leftarrow \text{FINDHYBRIDTREE}(f, s, \bar{\theta})$
  It finds the most probable λ-hybrid tree h containing the given f-s pair, under the generative model parameters $\bar{\theta}$. (Sec. 4.3)
- $\Gamma_h \leftarrow \text{EXTRACTRULES}(h)$
  It takes in a λ-hybrid tree h, and extracts a set of grammar rules Γ_h out of it. (Sec. 4.3)

Algorithm

1. Inputs and initializations:
   - A training set $(\mathbf{e}, \mathbf{s})$, an empty rule set $\Gamma = \emptyset$
2. Learn the grammar:
   - For each $e_i \in \mathbf{e}$, find its λ-meaning forest: $f_i = \text{BUILDFOREST}(e_i)$. This gives the set $(\mathbf{f}, \mathbf{s})$.
   - Learn the generative model parameter : $\bar{\theta}^* = \text{TRAINGENMODEL}(\mathbf{f}, \mathbf{s})$.
   - For each $(f_i, s_i) \in (\mathbf{f}, \mathbf{s})$, find the most probable λ-hybrid tree h_i, and then extract the grammar rules from it:
     $h_i = \text{FINDHYBRIDTREE}(f_i, s_i, \bar{\theta}^*)$
     $\Gamma = \Gamma \cup \text{EXTRACTRULES}(h_i)$
3. Output the learned grammar rule set Γ.

Figure 5: The algorithm for learning the grammar rules

5 Experiments

For experiments, we evaluated on the GEOQUERY dataset, which consists of 880 queries on U.S. geography. The dataset was manually labeled with λ-expressions as their semantics in Zettlemoyer and Collins (2005). It was used in many previous research efforts on semantic parsing (Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010). The original dataset was annotated with English sentences only. In order to assess the generation performance across different languages, in our work the entire dataset was also manually annotated with Chinese by a native Chinese speaker with linguistics background[5].

For all the experiments we present in this section, we use the same split as that of Kwiatkowski et al. (2010), where 280 instances are used for testing, and the remaining instances are used for learning. We further split the learning set into two portions, where 500 instances are used for training the models, which includes induction of grammar rules, training a language model, and computing feature values, and the remaining 100 instances are used for tuning the feature weights.

As we have mentioned earlier, we are not aware of any previous work that performs generation from formal logical forms that concerns both lexical acquisition and surface realization. The recent work by Angeli et al. (2010) presented a generation system from database records with an additional focus on content selection (selection of records and their subfields for generation). It is not obvious how to adopt their algorithm in our context where content selection is not required but the more complex logical semantic representation is used as input. Other earlier approaches such as the work of Wang (1980) and Shieber et al. (1990) made use of rule-based approaches without automatic lexical acquisition.

We thus compare our system against two state-of-the-art machine translation systems: a phrase-based translation system, implemented in the Moses toolkit (Koehn et al., 2007)[6], and a hierarchical phrase-based translation system, implemented in the Joshua toolkit (Li et al., 2009), which is a reimplementation of the original Hiero system (Chiang, 2005; Chiang, 2007). The state-of-the-art unsupervised Berkeley aligner (Liang et al., 2006) with default setting is used to construct word alignments. We train a trigram language model with modified Kneser-Ney smoothing (Chen and Goodman, 1996) from the training dataset using the SRILM toolkit (Stolcke, 2002), and use the same language model for all three systems. We use an n-best list of size 100 for all three systems when performing MERT.

5.1 Automatic Evaluation

For automatic evaluation, we measure the original IBM BLEU score (Papineni et al., 2002) (4-gram precision with brevity penalty) and the TER score (Snover et al., 2006) (the amount of edits required to change a system output into the reference)[7]. Note that TER measures the translation error rate, thus a

[5]The annotator created annotations with both λ-expressions and corresponding English sentences available as references.

[6]We used the default settings, and enabled the default lexicalized reordering model, which yielded better performance.

[7]We used tercom version 0.7.25 with the default settings.

smaller score indicates a better result. For clarity, we report 1−TER scores. Following the tuning procedure as conducted in Galley and Manning (2010), we perform MERT using BLEU as the metric.

We compare our model against state-of-the-art statistical machine translation systems. As a baseline, we first conduct an experiment with the following naive approach: we treat the λ-expressions as plain texts. All the bound variables (e.g., x in $\lambda x.state(x)$) which do not convey semantics are removed, but free variables (e.g., $state$ in $\lambda x.state(x)$) which might convey semantics are left intact. Quantifiers and logical connectives are also left intact. While this naive approach might not appear very sensible, we merely want to treat it as our simplest baseline.

Alternatively, analogous to the work of Wong and Mooney (2007a), we could first parse the λ-expressions into binary tree structures with a deterministic procedure, and then linearize the tree structure as a sequence. Since there exists different ways to linearize a binary tree, we consider preorder, inorder, and postorder traversal of the trees, and linearize them in these three different ways.

As for our system, during the grammar learning phase, we initialize the generative model parameters with output from the IBM alignment model 1 (Brown et al., 1993)[8], and run the λ-hybrid tree generative model with the unigram emission assumption for 10 iterations, followed by another 10 iterations with the bigram assumption. Grammar rules are then extracted based on the λ-hybrid trees obtained from such learned generative model parameters.

Since MERT is prone to search errors, we run each experiment 5 times with randomly initialized feature weights, and report the averaged scores. Experimental results for both English and Chinese are presented in Table 1. As we can observe, the way that a meaning representation tree is linearized has a significant impact on the translation performance. Interestingly, for both Moses and Joshua, the preorder setting yields the best performance for English, whereas it is inorder that yields the best performance for Chinese. This is perhaps due to the fact that Chinese presents a very different syntactic structure and word ordering from English.

[8]We assume word unigrams are generated from free variables, quantifiers, and logical connectives in IBM model 1.

Our system, on the other hand, employs a packed forest representation for λ-expressions. Therefore, it eliminates the ordering constraint by encompassing exponentially many possible tree structures during both the alignment and decoding stage. As a result, our system obtains significant improvements in both BLEU and 1−TER using the significance test under the paired bootstrap resampling method of Koehn (2004). We obtain $p < 0.01$ for all cases, except when comparing against Joshua-preorder for English, where we obtain $p < 0.05$ for both metrics.

		English		Chinese	
		BLEU	1−TER	BLEU	1−TER
Moses	text	48.93	61.08	43.23	51.71
	preorder	*51.13*	*63.73*	42.08	50.43
	inorder	46.72	57.59	*48.03*	*55.29*
	postorder	44.30	55.05	46.36	54.59
Joshua	text	37.40	48.97	36.60	46.20
	preorder	*51.40*	*64.69*	40.05	49.70
	inorder	40.31	50.47	*48.32*	*54.64*
	postorder	31.10	42.44	41.31	49.71
This work (t)		**54.58**	**67.65**	**55.11**	**63.77**
(t) w/o type 2 rules		53.77	66.43	54.30	62.49
(t) w/o type 3 rules		53.68	66.17	50.96	60.13

Table 1: Performance on generating English and Chinese from λ-expressions with automatic evaluation metrics (we report percentage scores).

5.2 Human Evaluation

We also conducted human evaluation with 5 evaluators each on English and Chinese. We randomly selected about 50% (139) test instances and obtained output sentences from the three systems. Moses and Joshua were run with the top-performing settings in terms of automatic metrics (i.e., preorder for English and inorder for Chinese). Following Angeli et al. (2010), evaluators are instructed to give scores based on language fluency and semantic correctness, on the following scale:

Score	Language Fluency	Semantic Correctness
5	Flawless	Perfect
4	Good	Near Perfect
3	Non-native	Minor Errors
2	Disfluent	Major Errors
1	Gibberish	Completely Wrong

For each test instance, we first randomly shuffled the output sentences of the three systems, and presented them together with the correct reference to the evaluators. The evaluators were then asked to score all the output sentences at once. This evaluation process not only ensures that the annotators have no access to which system generated the out-

English	Judge E1		Judge E2		Judge E3		Judge E4		Judge E5		Average	
	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM
Moses preorder	4.56	4.57	4.58	4.54	4.52	4.52	4.48	4.14	4.28	4.22	4.48 ± 0.12	4.40 ± 0.20
Joshua preorder	4.50	4.43	4.49	4.29	4.44	4.36	4.46	4.04	4.12	4.06	4.40 ± 0.16	4.24 ± 0.18
This work	**4.76**	**4.73**	**4.73**	**4.70**	**4.68**	**4.60**	**4.64**	**4.37**	**4.49**	**4.44**	**4.66** ± 0.10	**4.57** ± 0.16
Chinese	Judge C1		Judge C2		Judge C3		Judge C4		Judge C5		Average	
	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM	FLU	SEM
Moses inorder	4.38	4.22	3.95	3.99	4.01	3.80	4.27	4.19	4.09	4.01	4.14 ± 0.18	4.04 ± 0.17
Joshua inorder	4.32	4.04	3.74	3.91	3.76	3.55	4.21	4.04	3.96	3.97	4.00 ± 0.26	3.90 ± 0.21
This work	**4.61**	**4.47**	**4.53**	**4.43**	**4.50**	**4.31**	**4.71**	**4.55**	**4.57**	**4.32**	**4.59** ± 0.08	**4.42** ± 0.10

Table 2: Human evaluation results on English and Chinese generation. FLU: language fluency; SEM: semantic correctness.

put, but also minimizes bias associated with scoring different outputs for the same input. The detailed and averaged results (with one standard deviation) for human evaluation are presented in Table 2 for English and Chinese respectively. For both languages, our system achieves a significant improvement over Moses and Joshua ($p < 0.01$ with paired t-tests), in terms of both language fluency and semantic correctness. This set of results is important, as it demonstrates that our system produces more fluent texts with more accurate semantics when perceived by real humans.

5.3 Additional Experiments

We also performed the following additional experiments. First, we attempted to increase the number of EM iterations (to 100) when training the model with the bigram assumption, so as to assess the effect of the number of EM iterations on the final generation performance. We observed similar performance. Second, in order to assess the importance of the two types of novel rules – subtree rules (type 2) and two-level λ-hybrid sequence rules (type 3), we also conducted experiments without these rules for generation. Experiments show that these two types of rules are important. Specifically, type 3 rules, which are able to capture longer structural dependencies, are of particular importance for generating Chinese. Detailed results for these additional experiments are presented in Table 1.

5.4 Experiments on Variable-free Meaning Representations

Finally, we also assess the effectiveness of our model on an alternative meaning representation formalism in the form of variable-free tree structures. Specifically, we tested on the ROBOCUP dataset (Kuhlmann et al., 2004), which consists of 300 English instructions for coaching robots for soccer games, and a variable-free version of the GEOQUERY dataset. These are the standard datasets used in the generation tasks of Wong and Mooney (2007a) and Lu et al. (2009). Similar to the technique introduced in Kwiatkowski et al. (2010), our proposed algorithm could still be applied to such datasets by writing the tree-structured representations as function-arguments forms. The higher order unification-based decomposition algorithm could be applied on top of such forms accordingly. For example, $midfield(opp) \equiv \lambda x.midfield(x) \lhd opp$. See Kwiatkowski et al. (2010) for more details. However, since such forms present monotonous structures, and thus give less alternative options in the higher-order unification-based decomposition process, it prevents the algorithm from creating many disjunctive nodes in the packed forest. It is thus hypothesized that the advantages of the packed forest representation could not be fully exploited with such a meaning representation formalism.

Following previous works, we performed 4 runs of 10-fold cross validation based on the same split as that of Wong and Mooney (2007a) and Lu et al. (2009), and measured standard BLEU percentage and NIST (Doddington, 2002) scores. For experimentation on each fold, we trained a trigram language model on the training data of that fold, and randomly selected 70% of the training data for grammar induction, with the remaining 30% for learning of the feature weights using MERT. Next, we performed grammar induction with the complete training data of that fold, and used the learned feature weights for decoding of the test instances. The averaged results are shown in Table 3. Our approach outperforms the previous system WASP^{-1}++ (Wong and Mooney, 2007a) significantly, and achieves comparable or slightly better performance as compared to Lu et al. (2009). This set of results is particularly striking. We note

Variable-present dataset	
λ-expression :	$argmax(x, river(x) \wedge \exists y.[state(y) \wedge next\_to(y, india\_s) \wedge loc(x, y)], len(x))$
Reference :	what is the longest river that flows through a state that borders indiana
Moses :	what is the states that border long indiana
Joshua :	what is the longest river surrounding states border indiana
This work :	what is the longest river in the states that border indiana
λ-expression :	$density(\imath x.loc(argmax(y, loc(y, usa\_co) \wedge river(y), size(y)), x) \wedge state(x))$
Reference :	which is the density of the state that the largest river in the united states runs through
Moses :	what is the population density in lie on the state with the smallest state in the us
Joshua :	what is the population density of states lie on the smallest state in the us
This work :	what is the population density of the state with the largest river in the us
Variable-free datasets	
λ-expression :	$population(largest\_one\_density(state\_all))$
Reference :	what is the population of the state with the highest population density
This work :	how many people live in the state with the largest population density
λ-expression :	$rule(and(bpos(from\_goal\_line(our, jnum(n0.0, n32.0))), not(bpos(left(penalty\_area(our))))),$- $dont(player\_our(n3), intercept))$
Reference :	player 3 should not intercept the ball if the ball is within 32 meters of our goal line and not in our left penalty area
This work :	if the ball is within 32 meters from our goal line and not on the left side of our penalty area then player 3 should not intercept it

Figure 6: Sample English outputs for various datasets. For the variable-present dataset, we also show outputs from Moses and Joshua.

that the algorithm of Lu et al. (2009) is capable of modeling dependencies over phrases, which gives global optimization over the sentence generated, and works by building conditional random fields (Lafferty et al., 2001) over trees. But the algorithm of Lu et al. (2009) is also limited to handling tree-structured meaning representation, and is therefore unable to accept inputs such as the variable version of λ-expressions. Our algorithm works well by introducing additional new types of synchronous rules that are able to capture longer range dependencies. WASP^{-1}++, on the other hand, also makes use of a synchronous parsing-based statistical machine translation approach. Their system, however, requires linearization of the tree structure for both alignment and translation. In contrast, our model directly performs alignment and translation from a packed forest representation to a sentence. As a result, though WASP^{-1}++ made use of additional features (lexical weights), our system yielded better performance. Sample English output sentences are given in Figure 6.

	Robocup		Geoquery	
	BLEU	NIST	BLEU	NIST
WASP^{-1}++	60.22	6.8976	53.70	6.4808
Lu et al. (2009)	62.20	6.9845	57.33	**6.7459**
This work	**62.45**	**7.0011**	**57.62**	6.6867

Table 3: Performance on variable-free representations

6 Conclusions and Future Work

In this work, we presented a novel algorithm for generating natural language sentences from their underlying semantics in the form of typed lambda calculus. We tackled the problem by introducing a novel reduction-based weighted synchronous context-free grammar formalism, which allows sentence generation with a log-linear model. In addition, we proposed a novel generative model that jointly generates lambda calculus expressions and natural language sentences. The model is then used for automatic grammar induction. Empirical results show that our model outperforms state-of-the-art machine translation models, for both English and Chinese, in terms of both automatic and human evaluation. Furthermore, we have demonstrated that the model can also effectively handle inputs with a variable-free version of meaning representation.

We believe the algorithm used for inducing the reduction-based synchronous grammar rules may find applications in other research problems, such as statistical machine translation and phrasal synchronous grammar induction. We are interested in exploring further along such directions in the future.

Acknowledgments

This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore. We would like to thank Tom Kwiatkowski and Luke Zettlemoyer for sharing their dataset, and Omar F. Zaidan for his help with Z-MERT.

References

A.V. Aho and J.D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3(1):37–56.

G. Angeli, P. Liang, and D. Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proc. EMNLP*, pages 502–512.

J. K. Baker. 1979. Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65:S132.

H. P. Barendregt. 1985. *The Lambda Calculus, Its Syntax and Semantics (Studies in Logic and the Foundations of Mathematics, Volume 103). Revised Edition*. North-Holland.

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

S. F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. ACL*, pages 310–318.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proc. ICML*, pages 128–135.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. ACL*, pages 263–270.

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

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proc. HLT*, pages 138–145.

M. Galley and C. D. Manning. 2010. Accurate non-hierarchical phrase-based translation. In *Proc. HLT/NAACL*, pages 966–974.

R. Ge and R. J. Mooney. 2009. Learning a compositional semantic parser using an existing syntactic parser. In *Proc. ACL/IJCNLP*, pages 611–619.

G. P. Huet. 1973. The undecidability of unification in third order logic. *Information and Control*, 22(3):257–267.

G. P. Huet. 1975. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1(1):27–57.

R. J. Kate and R. J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proc. COLING/ACL*, pages 913–920.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. NAACL/HLT*, pages 48–54.

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

P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*, pages 388–395.

G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik. 2004. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *Proc. AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proc. EMNLP*, pages 1223–1233.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, pages 282–289.

I. Langkilde. 2000. Forest-based statistical sentence generation. In *Proc. NAACL*, pages 170–177.

Z. Li, C. Callison-Burch, C. Dyer, J. Ganitkevitch, S. Khudanpur, L. Schwartz, W. N. G. Thornton, J. Weese, and O. F. Zaidan. 2009. Joshua: an open source toolkit for parsing-based machine translation. In *Proc. WMT*, pages 135–139.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proc. HLT/NAACL*, pages 104–111.

W. Lu, H. T. Ng, W. S. Lee, and L. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proc. EMNLP*, pages 783–792.

W. Lu, H. T. Ng, and W. S. Lee. 2009. Natural language generation with tree conditional random fields. In *Proc. EMNLP*, pages 400–409.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160–167.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.

H. Shemtov. 1996. Generation of paraphrases from ambiguous logical forms. In *Proc. COLING*, pages 919–924.

S. M. Shieber, G. van Noord, F. C. N. Pereira, and R. C. Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16(1):30–42.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. AMTA*, pages 223–231.

A. Stolcke. 2002. SRILM-an extensible language modeling toolkit. In *Proc. ICSLP*, pages 901–904.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. COLING*, pages 836–841.

J. Wang. 1980. On computational sentence generation from logical form. In *Proc. COLING*, pages 405–411.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. HLT/NAACL*, pages 439–446.

Y. W. Wong and R. J. Mooney. 2007a. Generation by inverting a semantic parser that uses statistical machine translation. In *Proc. NAACL/HLT*, pages 172–179.

Y. W. Wong and R. J. Mooney. 2007b. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. ACL*, pages 960–967.

O. F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. UAI*, pages 658–666.

L. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proc. EMNLP-CoNLL*, pages 678–687.

L. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proc. ACL/IJCNLP*, pages 976–984.