Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

An Extended GHKM Algorithm for Inducing λ-SCFG

Peng Li, Yang Liu and **Maosong Sun**
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
pengli09@gmail.com, {liuyang2011,sms}@tsinghua.edu.cn

Abstract

Semantic parsing, which aims at mapping a natural language (NL) sentence into its formal meaning representation (e.g., logical form), has received increasing attention in recent years. While synchronous context-free grammar (SCFG) augmented with lambda calculus (λ-SCFG) provides an effective mechanism for semantic parsing, how to learn such λ-SCFG rules still remains a challenge because of the difficulty in determining the correspondence between NL sentences and logical forms. To alleviate this structural divergence problem, we extend the GHKM algorithm, which is a state-of-the-art algorithm for learning synchronous grammars in statistical machine translation, to induce λ-SCFG from pairs of NL sentences and logical forms. By treating logical forms as trees, we reformulate the theory behind GHKM that gives formal semantics to the alignment between NL words and logical form tokens. Experiments on the GEOQUERY dataset show that our semantic parser achieves an F-measure of 90.2%, the best result published to date.

Introduction

Semantic parsing is a task of mapping a natural language (NL) sentence into its formal meaning representation (MR). For example, given an English sentence

Every boy likes a star

its meaning representation could be a logical form

$$\forall x(boy(x) \rightarrow \exists y(human(y) \wedge pop(y) \wedge like(x, y)))$$

where boy, $human$, pop and $like$ are predicates, x and y are variables, $\rightarrow$ denotes implication, $\wedge$ denotes conjunction, and $\forall$ and $\exists$ are universal and existential quantifiers.

As semantic parsing is important for deep understanding of natural languages, it has attracted intensive attention over the past decade (Zelle and Mooney 1996; Tang and Mooney 2001; Ge and Mooney 2005; Zettlemoyer and Collins 2005; Kate, Wong, and Mooney 2005; Wong and Mooney 2006; Kate and Mooney 2006; Wong and Mooney 2007; Zettlemoyer and Collins 2007; Lu et al. 2008; Kwiatkowski et al. 2010; Jones, Johnson, and Goldwater 2011; 2012). While early semantic parsers mainly use inductive logic programming (Zelle and Mooney 1996; Tang and Mooney 2001) and semantically augmented parse tree (Ge and Mooney 2005), probabilistic grammars such as synchronous context-free grammar (SCFG) (Wong and Mooney 2006; 2007) and combinatorial categorial grammar (CCG) (Zettlemoyer and Collins 2005; Kwiatkowski et al. 2010) or tree transducers (Jones, Johnson, and Goldwater 2012) have been successfully applied to semantic parsing recently.

SCFG is a generalization of context-free grammar (CFG) that generates pairs of related strings rather than single strings. Wong and Mooney (2007) firstly introduce SCFG into semantic parsing to model the translation of an NL sentence into its MR. As logical variables play an important role in predicate logic (Blackburn and Bos 2005), Wong and Mooney (2007) augment SCFG with *lambda calculus* (λ-SCFG) to handle variables. More recently, Lu and Ng (2011) have shown that the λ-SCFG based method achieves state-of-the-art performance in language generation as well, the goal of which is to generate NL sentences given MRs. Therefore, λ-SCFG has proven to be a powerful mechanism for modeling bidirectional transformation between NL sentences and MRs.

However, how to learn λ-SCFG from training data still remains a challenge for semantic parsing. As logical forms usually have complex internal structures and variable dependencies across subparts (Lu and Ng 2011), it is much more difficult to identify the correspondence between an NL sentence and its logical form than between two NL sentences. To alleviate this problem, Wong and Mooney (2007) represent a logical form as a linearized parse tree (i.e., a list of CFG productions that generate the logical form) in top-down, left-most order. They use a word alignment tool to find the correspondence between two linear structures. Unfortunately, as the order of CFG productions has an important effect on the number of rules that can be extracted, logical forms must be carefully transformed to ensure that NL sentences and MR parse trees are maximally isomorphic.

On the other hand, inducing synchronous grammars from aligned tree-string pairs is a well-studied problem in the machine translation community. Galley et al. (2004) present a well-founded mathematical theory and a linear algorithm called GHKM for learning syntactically motivated transformation rules from parallel data with various granularities.

r_1	$S \rightarrow \langle X_{\boxed{1}}, X_{\boxed{1}} \rangle$
r_2	$X \rightarrow \langle \textit{Every}\ X_{\boxed{1}}, \lambda f.\forall x(f(x)) \lhd X_{\boxed{1}} \rangle$
r_3	$X \rightarrow \langle X_{\boxed{1}}\ X_{\boxed{2}}, \lambda f.\lambda g.\lambda x.f(x) \rightarrow g(x) \lhd X_{\boxed{1}} \lhd X_{\boxed{2}} \rangle$
r_4	$X \rightarrow \langle \textit{boy}, \lambda x.\textit{boy}(x) \rangle$
r_5	$X \rightarrow \langle X_{\boxed{1}}, \lambda f.\lambda x.\exists y(f(x,y)) \lhd X_{\boxed{1}} \rangle$
r_6	$X \rightarrow \langle X_{\boxed{1}}\ \textit{a star}, \lambda f.\lambda x.\lambda y.\textit{human}(y) \wedge \textit{pop}(y) \wedge f(x,y) \lhd X_{\boxed{1}} \rangle$
r_7	$X \rightarrow \langle \textit{like}, \lambda x.\lambda y.like(x,y) \rangle$

$\langle S_{\boxed{1}}, S_{\boxed{1}} \rangle$
$\overset{r_1}{\Longrightarrow} \langle X_{\boxed{2}}, X_{\boxed{2}} \rangle$
$\overset{r_2}{\Longrightarrow} \langle \textit{Every}\ X_{\boxed{3}}, \lambda f.\forall x(f(x)) \lhd X_{\boxed{3}} \rangle$
$\overset{r_3}{\Longrightarrow} \langle \textit{Every}\ X_{\boxed{4}}\ X_{\boxed{5}}, \lambda f.\lambda g.\forall x(f(x) \rightarrow g(x)) \lhd X_{\boxed{4}} \lhd X_{\boxed{5}} \rangle$
$\overset{r_4}{\Longrightarrow} \langle \textit{Every boy}\ X_{\boxed{5}}, \lambda g.\forall x(\textit{boy}(x) \rightarrow g(x)) \lhd X_{\boxed{5}} \rangle$
$\overset{r_5}{\Longrightarrow} \langle \textit{Every boy}\ X_{\boxed{6}}, \lambda f.\forall x(\textit{boy}(x) \rightarrow \exists y(f(x,y))) \lhd X_{\boxed{6}} \rangle$
$\overset{r_6}{\Longrightarrow} \langle \textit{Every boy}\ X_{\boxed{7}}\ \textit{a star}, \lambda f.\forall x(\textit{boy}(x) \rightarrow \exists y(\textit{human}(y) \wedge \textit{pop}(y) \wedge f(x,y))) \lhd X_{\boxed{7}} \rangle$
$\overset{r_7}{\Longrightarrow} \langle \textit{Every boy likes a star}, \forall x(\textit{boy}(x) \rightarrow \exists y(\textit{human}(y) \wedge \textit{pop}(y) \wedge like(x,y))) \rangle$

Table 1: A derivation for a NL sentence *Every boy likes a star* and its logical form $\forall x(\textit{boy}(x) \rightarrow \exists y(\textit{human}(y) \wedge \textit{pop}(y) \wedge like(x,y)))$. Each step involves the rewriting of a pair of co-indexed non-terminals by the same λ-SCFG rule.

The theory behind GHKM gives formal semantics to word alignments and draws connections among word alignments, derivations and rules. Due to its efficiency and effectiveness, it has been widely used in most syntax-based machine translation systems (Galley et al. 2006; Mi and Huang 2008; Liu, Lü, and Liu 2009; Wu, Matsuzaki, and Tsujii 2010; Zhang et al. 2011; Liu, Liu, and Lü 2011).

In this paper, we extend the GHKM algorithm to induce λ-SCFG from pairs of NL sentences and logical forms for semantic parsing. By treating logical forms as trees, we reformulate the theory behind GHKM to give semantics to the alignment between NL words and logical forms. While the tree nodes in (Wong and Mooney 2007) are CFG productions, we treat atomic logical form tokens as tree nodes to facilitate adapting GHKM. This hopefully alleviates the data sparseness problem especially when training set only contains hundreds of sentences. In addition, λ-SCFG rules extracted by GHKM are robust to the structural divergence between NL sentences and logical forms because GHKM explicitly identifies tree nodes where both sides are isomorphic. Furthermore, our algorithm is capable of extracting λ-SCFG rules with various granularities and therefore generalizes well to unseen text. Experiments on the GEOQUERY dataset show that our method achieves an F-measure of 90.2%, the best result published to date.

λ-SCFG for Semantic Parsing

The formal semantics we use in this paper is lambda calculus expressions (λ-expression). Following Chiang (2007), we define a λ-SCFG rule as follows:

$$X \rightarrow \langle \alpha, \beta, \sim \rangle \tag{1}$$

where X is a non-terminal, α is an NL string of terminals and non-terminals, β is a λ-production (Lu and Ng 2011), $\sim$ is a one-to-one alignment between non-terminals occurrence in α and β.

Consider a λ-SCFG rule without non-terminals:

$$X \rightarrow \langle \textit{boy}, \lambda x.\textit{boy}(x) \rangle \tag{2}$$

The source right-hand side is an NL word *boy*. The target right-hand side is a λ-production $\lambda x.\textit{boy}(x)$. It represents a function named $\textit{boy}$ that takes a variable x as its argument, and the operator λ is said to **bind** x in the function.

The application of $Jack$ to the above λ-production yields

$$\lambda x.\textit{boy}(x) \lhd Jack = \textit{boy}(Jack) \tag{3}$$

Following Lu and Ng (2011), we use the symbol $\lhd$ as the notation for **function application**. Such function application is called $\boldsymbol{\beta}$ **conversion** in lambda calculus. In addition, lambda calculus defines $\boldsymbol{\alpha}$ **conversion** to allow bound variable names to be changed to facilitate name resolution:

$$\lambda x.\textit{boy}(x) = \lambda y.\textit{boy}(y) \tag{4}$$

Now, consider a λ-SCFG rule with one non-terminal:

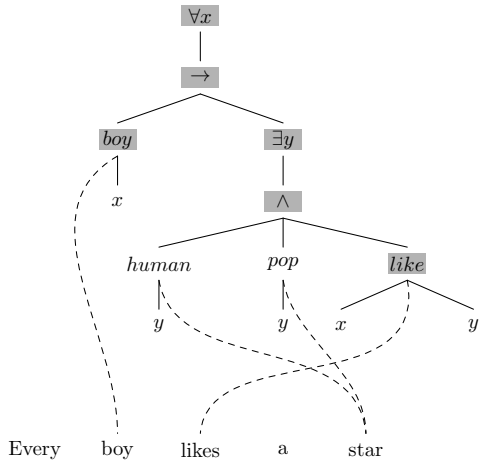
$$X \rightarrow \langle \textit{Every}\ X_{\boxed{1}}, \lambda f.\forall x(f(x)) \lhd X_{\boxed{1}} \rangle \tag{5}$$

where f denotes a function and the subscript indicates the correspondence between non-terminals of the NL string and λ-production. We follow Lu and Ng (2011) to allow the arguments to be non-terminals such as $\lambda f.\forall x(f(x)) \lhd X_{\boxed{1}}$, indicating that the functor $\lambda f.\forall x(f(x))$ expects another λ-production to serve as its argument.

Table 1 shows a derivation composed of seven λ-SCFG rules that produce a NL sentence *Every boy likes a star* and its logical form $\forall x(\textit{boy}(x) \rightarrow \exists y(\textit{human}(y) \wedge \textit{pop}(y) \wedge like(x,y)))$ simultaneously. Each step of a derivation involves the rewriting of a pair of co-indexed non-terminals by the same λ-SCFG rule. The yield is a pair of NL sentence and logical form.

An Extended GHKM Algorithm

Figure 1 shows a training example for our rule extraction algorithm. It consists of a NL sentence *Every boy likes a star*, a logical form $\forall x(\textit{boy}(x) \rightarrow \exists y(\textit{human}(y) \wedge \textit{pop}(y) \wedge like(x,y)))$, and an alignment between NL words and tree nodes. The goal of our algorithm is to extract all λ-SCFG



node	minimal rule
$\forall x$	$X \rightarrow \langle$*Every* $X_{[1]}, \lambda f.\forall x(f(x)) \lhd X_{[1]}\rangle$
$\rightarrow$	$X \rightarrow \langle X_{[1]}\ X_{[2]}, \lambda f.\lambda g.\lambda x.f(x) \rightarrow g(x) \lhd X_{[1]} \lhd X_{[2]}\rangle$
boy	$X \rightarrow \langle$*boy*, $\lambda x.boy(x)\rangle$
$\exists y$	$X \rightarrow \langle X_{[1]}, \lambda f.\lambda x.\exists y(f(x,y)) \lhd X_{[1]}\rangle$
$\wedge$	$X \rightarrow \langle X_{[1]}$ *a star*, $\lambda f.\lambda x.\lambda y.human(y) \wedge pop(y) \wedge f(x,y) \lhd X_{[1]}\rangle$
$like$	$X \rightarrow \langle$likes, $\lambda x.\lambda y.like(x,y)\rangle$

Figure 1: A training example consisting of an NL sentence, a logical form, and the alignment between them. Minimal rules extracted from shaded nodes are listed below.

rules that can explain how the NL sentence and the logical form are synchronously generated, as shown in Table 1.

To do this, we first need to transform a logical form to a tree structure. This can be done by treating atomic elements of logical forms (e.g., variables and predicates) as tree nodes. [1] For example, the logical form $\lambda x.boy(x)$ can be converted to the following λ-expression tree:

$$(\lambda x (boy (x))) \quad (6)$$

Note that our tree representation treats logical form tokens as tree nodes in order to facilitate adapting GHKM. This is different from those in (Wong and Mooney 2007; Lu and Ng 2011), the tree nodes of which are CFG productions.

Given an NL sentence and a logical form tree, there are exponentially many derivations that can explain how they are synchronously generated. Therefore, alignment is usually introduced to constrain the search space. While the original GHKM algorithm uses word alignment, our algorithm introduces an alignment between tree nodes and NL words. We distinguish between **content** and **function** nodes. By content, we mean that the nodes are supposed to be aligned to some NL words. Content nodes are usually predicate names such as boy, $human$, and pop, and constants may also be considered, depending on the meaning representation chosen. In contrast, function nodes are usually logical form tokens that are not supposed to be aligned with any

[1] We define 11 transformation rules, which are omitted here due to the space limit.

node	$\sigma(v)$	$\gamma(v)$	$\delta(v)$	frontier
$\forall x$	$\emptyset$	$\{1,2,3,4,5\}$	$\emptyset$	1
$\rightarrow$	$\emptyset$	$\{2,3,5\}$	$\emptyset$	1
boy	$\{2\}$	$\{2\}$	$\{3,5\}$	1
$\exists y$	$\emptyset$	$\{3,5\}$	$\{2\}$	1
$\wedge$	$\emptyset$	$\{3,5\}$	$\{2\}$	1
$human$	$\{5\}$	$\{5\}$	$\{2,3,5\}$	0
pop	$\{5\}$	$\{5\}$	$\{2,3,5\}$	0
$like$	$\{3\}$	$\{3\}$	$\{2,5\}$	1

Table 2: Node attributes for non-leaf nodes in Figure 1. $\sigma(v)$ denotes node corresponding span, $\gamma(v)$ denotes tree corresponding span, $\delta(v)$ denotes complement span, and "frontier" denotes frontier node.

words (e.g., variables, quantifiers, etc.). To obtain the node-to-word alignment, we extract a list of content nodes from a logical form tree with a pre-order traversal. For example, the content node list for the tree in Figure 1 is [2]

$$boy\ \ human\ \ pop\ \ like$$

Then, word alignment tools such as GIZA++ (Och and Ney 2003) can be used to produce the alignment between the content node list and its corresponding NL sentence.

Given an aligned tree-string pair as input, our algorithm generally involves three steps:

1. Identifying frontier nodes: to find tree nodes in the logical form tree that subsume tree-string pairs consistent with alignment;
2. Extracting minimal rules: to obtain atomic λ-SCFG rules that forms a unique derivation of the training example;
3. Rule composition: to obtain composed rules by combining minimal rules.

Identifying Frontier Nodes

A tree node is said to be **frontier** if and only if the tree-string pair it subsumes is *consistent* with alignment. Alignment consistency is firstly introduced by Och and Ney (2002) to extract phrase pairs. As an effective heuristic for identifying translation equivalents, it has become the fundamental criterion in GHKM. For example, consider the node boy in Figure 1, it subsumes a subtree (boy (x)) and a substring *boy*. As the subtree is exclusively aligned to the substring and vice versa, they are likely to be translation equivalents. Therefore, boy is a frontier node. On the contrary, as the subtree subsumed by the node pop is aligned to *star* but *star* is also aligned to $human$, they are said to be inconsistent with alignment. Therefore, pop is not a frontier node.

More formally, our algorithm calculates a number of node attributes to identify frontier nodes. [3]

Definition 1. Given a node v, its **node corresponding span** $\sigma(v)$ is an index set of NL words to which it is aligned.

[2] In our experiments, we treat all predicate names and constants as content nodes, the same definition is applied to this example.

[3] Our definitions of corresponding spans and complement spans are significantly different from the original GHKM because node-word alignment rather than word-word alignment is used in the semantic scenario.

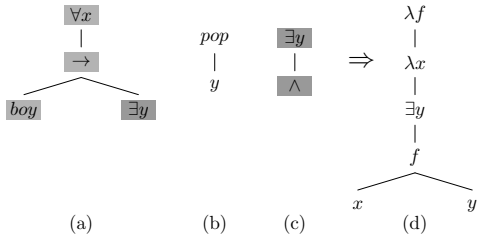


Figure 2: (a) A frontier tree, (b) a non-frontier tree, (c) a minimal frontier tree, and (d) λ-expression tree corresponding to the minimal frontier tree. The λ-expression tree can be easily converted to a λ-expression: $\lambda f.\lambda x.\exists y(f(x,y))$.

For example, as node *boy* is aligned to the second English word *boy*, its node corresponding span is $\{2\}$.

Definition 2. Given a node v, its **tree corresponding span** $\gamma(v)$ is an index set of NL words to which the subtree it subsumes is aligned.

For example, although node $\rightarrow$ is not aligned to any NL words, the subtree it subsumes is aligned to *boy*, *likes*, and *star*. Therefore, its tree corresponding span is $\{2,3,5\}$.

We define that the tree corresponding span of a root node is always the index set of all NL words.

Definition 3. Given a node v, its **complement span** $\delta(v)$ is defined as

$$\begin{aligned}\delta(v) \quad = \quad & \big(\cup_{v' \in a(v)}\, \sigma(v')\big) \cup \\ & \big(\cup_{v' \text{ s.t. } v' \notin a(v) \wedge v' \notin d(v)}\, \gamma(v')\big) \end{aligned} \tag{7}$$

where $a(v)$ denotes the set of antecedents of v, $d(v)$ denotes the set of descendants of v.

For example, consider node *boy*, nodes $\forall x$ and $\rightarrow$ are antecedents of *boy*, and the union of their node corresponding span is $\emptyset$, node $\exists y$ and its descendants are neither antecedents nor descendants of *boy*, and the union of their tree corresponding span is $\{3,5\}$. So the complement span of *boy* is $\emptyset \cup \{3,5\} = \{3,5\}$.

Definition 4. A node v is said to be a **frontier node** if and only if $closure(\gamma(v)) \cap \delta(v) = \emptyset$, where $closure(\gamma(v))$ is the smallest set of consecutive integers such that $\gamma(v) \subset closure(\gamma(v))$.

For example, consider node $\wedge$, the closure of its tree corresponding span $\{3,5\}$ is $\{3,4,5\}$, its complement span is $\{2\}$, the intersection of $\{3,4,5\}$ and $\{2\}$ is an empty set. Therefore, node $\wedge$ is a frontier node. Similarly, it is easy to verify that $human$ and pop are not frontier nodes.

In Figure 1, frontier nodes are highlighted by shading, from which λ-SCFG rules are ready to extract.

Extracting Minimal Rules

Given frontier nodes, the next step is to identify consistent tree-string pairs, from which λ-SCFG rules derive. Following Galley et al. (2006), we distinguish between **minimal** and **composed** rules. While minimal rules are atomic and cannot be decomposed, composed rules can be decomposed into a number of minimal rules. Also, we introduce a number of notions to help identify minimal λ-SCFG rules.

r_3	$X \rightarrow \langle X_{\boxed{1}}\ X_{\boxed{2}}, \lambda f.\lambda g.\lambda x.f(x) \rightarrow g(x) \lhd X_{\boxed{1}} \lhd X_{\boxed{2}} \rangle$
r_4	$X \rightarrow \langle boy, \lambda x.boy(x) \rangle$
r_5	$X \rightarrow \langle X_{\boxed{1}}, \lambda f.\lambda x.\exists y(f(x,y)) \lhd X_{\boxed{1}} \rangle$
$+$	$X \rightarrow \langle boy\ X_{\boxed{1}}, \lambda f.\lambda x.boy(x) \rightarrow \exists y(f(x,y)) \lhd X_{\boxed{1}} \rangle$

Table 3: Rule composition.

Definition 5. A **frontier tree** is a subtree satisfying:

1. Its root is a frontier node;
2. Its leaves are either frontier nodes or the leaves of the input tree.

For example, while Figure 2(a) is a frontier tree, Figure 2(b) is not because its root is not a frontier node.

Definition 6. A **minimal frontier tree** is a frontier tree such that all nodes other than the root and leaves are non-frontier nodes.

For example, Figure 2(a) is not a minimal frontier tree because the internal node $\rightarrow$ is a frontier node. On the contrary, Figure 2(c) is a minimal frontier tree because both the root and leaves are frontier nodes and there are no internal nodes.

Given a minimal frontier tree, it is trivial to identify its corresponding NL string using alignment information. Consider Figure 2(c), as its leaf node is a non-terminal, its corresponding NL string is simply $X_{\boxed{1}}$.

However, unlike syntactic GHKM, a minimal frontier tree does not directly correspond to a λ-expression because the associated lambda operators and dominated variables are not explicitly attached to the tree structure. Therefore, to recover a minimal λ-SCFG rule from a minimal frontier tree, we need first to transform the minimal frontier tree to a λ-expression.

Consider Figure 2(c) again, as node $\wedge$ is a frontier leaf, it should be a non-terminal. As the variables dominated by node $\wedge$ are x and y, node $\wedge$ should correspond to $f(x,y)$ in the λ-expression.

More formally, we introduce inside variable set, outside variable set, and variable set to determine arguments of functions for frontier nodes.

Definition 7. Given a node v, its **inside variable set** $in(v)$ is a set of logical variables that appear in the subtree it subsumes.

For example, as x and y appear in the subtree rooted at node $\wedge$, its inside variable set is $\{x,y\}$. The inside variable sets can be computed in a bottom-up way.

Definition 8. Given a node v, its **outside variable set** $out(v)$ is a set of logical variables such that v falls in their domains.

For example, as node $\wedge$ falls in the domains of x and y, which are introduced by $\forall x$ and $\exists y$ respectively, its outside variable set is $\{x,y\}$. Similarly, as node *boy* falls in the domains of x, its outside variable set is $\{x\}$ [4]. The outside variable sets can be computed in a top-down way.

[4]The inside and outside variable sets of a node are not always the same, e.g., $in(human) = \{x\}$ and $out(human) = \{x,y\}$.

```
1: procedure EXTRACT(T, S, A)
2:     R ← ∅
3:     V_f ← FRONTIER(T, S, A)                ▷ Frontier nodes
4:     for each v ∈ V_f do                    ▷ Extract minimal rules
5:         r ← MINIMAL(T, S, A, v)
6:         R ← R ∪ {r}
7:     end for
8:     COMPOSE(R)                             ▷ Get composed rules
9:     return R
10: end procedure
```

Figure 3: The extended GHKM algorithm for extracting λ-SCFG rules from a logical form tree T, a natural language sentence S, and the alignment A between them.

Definition 9. Given a node v, its **variable set** $var(v)$ is the intersection of inside variable set and outside variable set: $var(v) = in(v) \cap out(v)$.

Finally, λf and λx are added in the beginning of the λ-expression to ensure all functions and variables are bound. Therefore, the minimal rule extracted from node $\exists y$ is

$$X \rightarrow \langle X_{[1]}, \lambda f.\lambda x.\exists y(f(x,y)) \lhd X_{[1]} \rangle \quad (8)$$

Figure 1 lists the minimal rules extracted from frontier nodes. It is easy to verify that minimal rules extracted in this way form a valid derivation of the training example.

Rule Composition

Although minimal rules are capable of explaining the training data, they hardly take surrounding context into consideration to resolve parsing ambiguity. Generated by combining minimal rules, composed rules are able to memorize local context. Galley et al. (2006) report that using composed rules significantly improves rule coverage and finally improves translation quality. It still holds for semantic parsing.

Table 3 gives an example of rule composition. As there are usually exponentially many composed rules, we follow Liu et al. (2006) to use tree height h to maintain a reasonable grammar size. This can be done by composing rules for frontier nodes using the *cube pruning* algorithm (Chiang 2007) in a bottom-up way.

In practice, we handle unaligned words in a similar way with Galley et al. (2006) to further improve rule coverage. [5] For example, the following rule can also be extracted from the frontier node $like$ in Figure 1:

$$X \rightarrow \langle \textit{likes } a, \lambda x.\lambda y.like(x,y) \rangle \quad (9)$$

Therefore, our algorithm is capable of extracting all granularities of λ-SCFG rules, which potentially improve the generalization ability (i.e., the ability to parse correctly on unseen text) of our semantic parser.

To estimate rule probability $p(\beta|\alpha)$, we use the relative frequency estimate of a logical form given an NL sentence.[6]

[5] We also attach unaligned words in multiple ways but do not estimate rule probabilities based on forest for simplicity.

[6] We find that the reverse probability $p(\alpha|\beta)$ fails to result in improvements. Therefore, we exclude it in our experiments.

System	P	R	F
Independent Test Set			
Zettlemoyer and Collins (2005)	**96.3**	79.3	87.0
Zettlemoyer and Collins (2007)	95.5	83.2	88.9
Kwiatkowski *et al.* (2010)	94.1	85.0	89.3
Cross Validation Results			
Kate *et al. (2005)*	89.0	54.1	67.3
Wong and Mooney (2006)	87.2	74.8	80.5
Kate and Mooney (2006)	93.3	71.7	81.1
Lu *et al.* (2008)	89.3	81.5	85.2
Ge and Mooney (2005)	95.5	77.2	85.4
Wong and Mooney (2007)	92.0	86.6	89.2
this work	93.0	**87.6**	**90.2**

Table 4: Comparison with existing English semantic parsers on GEOQUERY.

Figure 3 shows the entire extended GHKM algorithm. While identifying frontier nodes and extracting minimal rules generally runs in linear time, the complexity of obtaining composed rules can be controlled using the tree height limit parameter h.

Experiments

Our semantic parser resembles the CKY parser used in statistical machine translation (Chiang 2007), which is based on a discriminative model (Och and Ney 2002) with rule probabilities, rule count, and language models as features. The model parameters are optimized with minimum error rate training (Och and Ney 2003). Given an NL sentence S, our semantic parser finds the target yield of the single best derivation D that has source yield of S: [7]

$$\hat{T} = \beta \left(\underset{D \ s.t. \ \alpha(D)=S}{\arg\max} P(D) \right) \quad (10)$$

We evaluated our semantic parser on the GEOQUERY corpus, a corpus consists of 880 English questions on USA geography with Prolog logical forms (Wong and Mooney 2006). The following is an example from this corpus:

How big is Alaska ?

$answer(A, (size(B, A), const(B, stateid(alaska))))$

The metrics for evaluating performance include *precision* (percentage of logical forms that are correct), *recall* (percentage of test sentences that are correctly parsed), and *F-measure* (harmonic mean of precision and recall). Note that a logical form is considered correct only if it retrieves the same answer as the correct logical form.

We compared our method with 9 semantic parsers on the GEOQUERY dataset. Standard 10-fold cross validation with the same data splits as (Wong and Mooney 2007) was used.

As shown in Table 4, our approach achieves the best results with the tree height limit h is set to 7. We find that semantic parsers based on lambda calculus (e.g., Zettlemoyer

[7] We use an efficient cubic-time CKY-style parsing algorithm proposed by DeNero *et al.* (2009). In addition, we follow Wong and Mooney (2007) to use *type checking* to prune impossible derivations as early as possible during parsing.

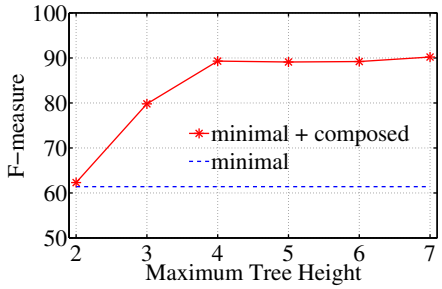


Figure 4: Effect of tree height limit.

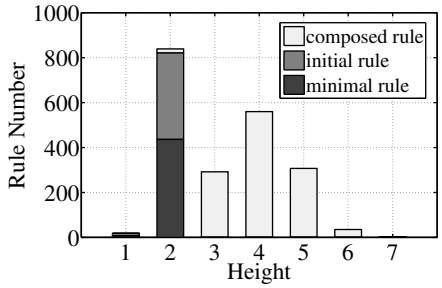


Figure 5: Distribution of rules in terms of tree height.

and Collins (2005), Wong and Mooney (2007), Kwiatkowski *et al.* (2010), and this work) generally outperform those using variable free MR, suggesting that lambda calculus is a powerful mechanism for semantic parsing.

We investigated the effect of multi-level rules on parsing performance. On one hand, as shown in Figure 4, the F-measures of using both minimal and composed rules rise with the increase of tree height limit h, suggesting that bigger rules improve accuracy because they memorize local context. However, the gains shrink when h is greater than 4 due to the sparse data. Furthermore, it is clear to see that using composed rules brings significant improvements. Note that minimal rules are not limited by h. On the other hand, We find that more than 60% of rules used in 10-fold cross validation are composed rules, as shown in Figure 5 [8]. Therefore, multi-level rules did play an important role.

Table 5 shows the performance on the German, Greek and Thai version of the GEOQUERY corpus (Jones, Johnson, and Goldwater 2012). Following Jones et al. (2012), we used 600 sentences for training and the remaining 280 for test. Our method outperforms the baseline systems on the English and Greek dataset and achieves comparable result on German and Thai. We find that Thai sentences are significantly longer than other languages. In addition, Thai has quite different word order and distinct characteristics (e.g., using duplicated adverbs to express intensity), which makes it quite difficult to find the alignment between NL words and logical form tree nodes [9].

[8]The same set of initial rules as (Wong and Mooney 2006) was used in our experiments.

[9]We merge each of the top 16 bigrams measured by frequency into one symbol in Thai sentences to improve alignment quality.

System	en	ge	el	th
Wong and Mooney (2006)	77.7	74.9	78.6	75.0
Lu *et al.* (2008)	81.0	68.5	74.6	76.7
Kwiatkowski *et al.* (2010)	82.1	**75.0**	73.7	66.4
Jones *et al.* (2012)	79.3	74.6	75.4	**78.2**
this work	**84.2**	74.6	**79.4**	76.7

Table 5: F-measures on the multilingual version of GEOQUERY. "en", "ge", "el" and "th" stand for English, German, Greek and Thai, respectively.

Related Work

Our work follows Wong and Mooney (2007) to couple the generation of NL sentences and logical forms with λ-SCFG. The most significant difference lies in the way to represent a logical form as a tree. While the tree nodes in (Wong and Mooney 2007) are CFG productions, we treat atomic logical form tokens as tree nodes to facilitate adapting GHKM. This hopefully alleviates the data sparseness problem as GEOQUERY only contains hundreds of sentences. In addition, tree-to-string rules extracted by GHKM are robust to the nonisomorphism between NL sentences and logical forms because it explicitly identifies frontier nodes where both sides are isomorphic. Furthermore, our algorithm is able to extract rules with varying granularities in a principled way. Another interesting direction is to directly learn λ-SCFG without explicit tree-string alignment. Lu and Ng (2011) propose a hybrid tree model from a transformative perspective and apply tree transversal algorithms to extract λ-SCFG.

Different from the original GHKM algorithm and its variants used in syntactic machine translation (Galley et al. 2004; 2006; Mi and Huang 2008; Liu, Lü, and Liu 2009; Wu, Matsuzaki, and Tsujii 2010; Liu, Liu, and Lü 2011), our extended GHKM relies on node-to-word alignment rather than alignment between surface strings. More importantly, we need to reformulate the semantics of alignment to transform frontier tree fragments to λ-expressions.

Conclusion

In this paper, we have presented a semantic version of GHKM for inducing λ-SCFG. This is done by transforming logical forms to tree structures and finding correspondence between logical form trees and natural language sentences. Our semantic parser achieves promising results on the multilingual GEOQUERY corpus.

It is necessary to develop alignment models that consider characteristics of logical forms to link logical forms and natural language sentences. We also plan to investigate tree binarization to further improve rule coverage (Zhang et al. 2011). Finally, it is interesting to use EM or Monte Carlo methods to better estimate λ-SCFG rule probabilities.

Acknowledgments

This research is supported by the 863 Program under the grant No 2012AA011102 and No. 2011AA01A207 and by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

Blackburn, P., and Bos, J. 2005. *Representation and inference for natural language: A first course in computational semantics*.

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

DeNero, J.; Bansal, M.; Pauls, A.; and Klein, D. 2009. Efficient parsing for transducer grammars. In *Proceedings of NAACL HLT 2009*, 227–235.

Galley, M.; Hopkins, M.; Knight, K.; and Marcu, D. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL-04*, 273–280.

Galley, M.; Graehl, J.; Knight, K.; Marcu, D.; DeNeefe, S.; Wang, W.; and Thayer, I. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING·ACL 2006*, 961–968.

Ge, R., and Mooney, R. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of CoNLL-2005*, 9–16.

Jones, B.; Johnson, M.; and Goldwater, S. 2011. Formalizing semantic parsing with tree transducers. In *Proceedings of ALTA Workshop 2011*, 19–28.

Jones, B.; Johnson, M.; and Goldwater, S. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of ACL 2012*, 488–496.

Kate, R. J., and Mooney, R. J. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of COLING·ACL 2006*, 913–920.

Kate, R. J.; Wong, Y. W.; and Mooney, R. J. 2005. Learning to transform natural to formal languages. In *Proceedings of AAAI-05*, 1062–1068.

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

Liu, Y.; Liu, Q.; and Lin, S. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING·ACL 2006*, 609–616.

Liu, Y.; Liu, Q.; and Lü, Y. 2011. Adjoining tree-to-string translation. In *Proceedings of ACL HLT 2011*, 1278–1287.

Liu, Y.; Lü, Y.; and Liu, Q. 2009. Improving tree-to-tree translation with packed forests. In *Proceedings of ACL-IJCNLP 2009*, 558–566.

Lu, W., and Ng, H. T. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of EMNLP 2011*, 1611–1622.

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

Mi, H., and Huang, L. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP 2008*, 206–214.

Och, F. J., and Ney, H. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL 2002*, 295–302.

Och, F. J., and Ney, H. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29(1):19–51.

Tang, L., and Mooney, R. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. 466–477.

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

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

Wu, X.; Matsuzaki, T.; and Tsujii, J. 2010. Fine-grained tree-to-string translation rule extraction. In *Proceedings of ACL 2010*, 325–334.

Zelle, J. M., and Mooney, R. J. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of AAAI-96*, 1050–1055.

Zettlemoyer, L. S., and Collins, M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI-05*, 658–666.

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

Zhang, H.; Fang, L.; Xu, P.; and Wu, X. 2011. Binarized forest to string translation. In *Proceedings of ACL HLT 2011*, 835–845.