Improving Semantic Parsing with Enriched Synchronous Context-Free Grammars in Statistical Machine Translation

JUNHUI LI, Soochow University, China
MUHUA ZHU, Alibaba Inc., China
WEI LU, Singapore University of Technology and Design, Singapore
GUODONG ZHOU, Soochow University, China

Semantic parsing maps a sentence in natural language into a structured meaning representation. Previous studies show that semantic parsing with synchronous context-free grammars (SCFGs) achieves favorable performance over most other alternatives. Motivated by the observation that the performance of semantic parsing with SCFGs is closely tied to the translation rules, this article explores to extend translation rules with high quality and increased coverage in three ways. First, we examine the difference between word alignments for semantic parsing and statistical machine translation (SMT) to better adapt word alignment in SMT to semantic parsing. Second, we introduce both structure and syntax informed nonterminals, better guiding the parsing in favor of well-formed structure, instead of using a uninformed nonterminal in SCFGs. Third, we address the unknown word translation issue via synthetic translation rules. Last but not least, we use a filtering approach to improve performance via predicting answer type. Evaluation on the standard GeoQuery benchmark dataset shows that our approach greatly outperforms the state of the art across various languages, including English, Chinese, Thai, German, and Greek.

CCS Concepts: • **Computing methodologies → Natural language processing;**

Additional Key Words and Phrases: Semantic parsing, statistical machine translation, enriched synchronous context-free grammars, word alignment

ACM Reference Format:
Junhui Li, Muhua Zhu, Wei Lu, and Guodong Zhou. 2016. Improving semantic parsing with enriched synchronous context-free grammars in statistical machine translation. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 16, 1, Article 6 (November 2016), 24 pages.
DOI: http://dx.doi.org/10.1145/2963099

1. INTRODUCTION

Semantic parsing, the task of mapping natural language (NL) sentences into logical forms in a meaning representation language (MRL), has recently received a significant amount of attention, with various models proposed over the past few years. Consider the example of an NL sentence paired with its corresponding form in MRL as shown in Figure 1(a) and (b). Our goal is to learn a semantic parser that can effectively parse unseen NL sentences into their representations in MRL. The MRL shown in the

J. Li and G. Zhou were supported by Key Project 61331011 and Projects 61305088 and 61401295 under the National Natural Science Foundation of China. W. Lu was supported by SUTD grant SRG ISTD 2013 064.
Authors' addresses: J. Li and G. Zhou (corresponding author), School of Computer Science and Technology, Soochow University, 1 Shizi St., Suzhou, 215006, China; emails: {lijunhui, gdzhou}@suda.edu.cn; M. Zhu, Alibaba Inc., Hangzhou, 311100, China; email: muhua.zmh@alibaba-inc.com; W. Lu, Information Systems Technology and Design, Singapore University of Technology and Design, 8 Somapah Rd., 487372, Singapore; email: luwei@sutd.edu.sg.

DOI: http://dx.doi.org/10.1145/2963099

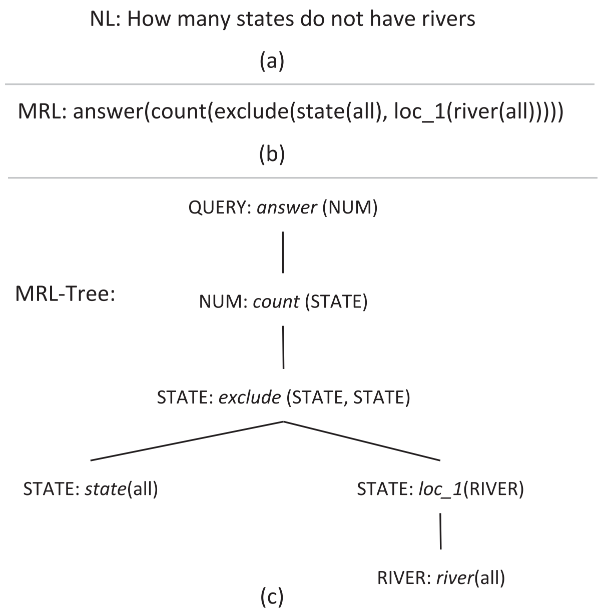

NL: How many states do not have rivers
(a)
MRL: answer(count(exclude(state(all), loc_1(river(all)))))
(b)
MRL-Tree:
QUERY: *answer* (NUM)
NUM: *count* (STATE)
STATE: *exclude* (STATE, STATE)
STATE: *state*(all)
STATE: *loc_1*(RIVER)
RIVER: *river*(all)
(c)

Fig. 1. Example of a natural sentence (a) paired with its MRL (b) and its tree-structured semantic representation (c).

example can be further represented as certain recursive structures such as trees, which consist of atomic semantic units as tree nodes. For example, the MRL in Figure 1(b) can be presented in an equivalent tree structure as illustrated in Figure 1(c).

To parse sentences in NL into representations in MRL, semantic parsing can be naturally viewed as a statistical machine translation (SMT) task, which translates a sentence in NL (i.e., the source language in SMT) into its meaning representation (MR) in MRL (i.e., the target language in SMT). Indeed, many attempts have been made to directly apply SMT systems (or methodologies) to semantic parsing [Papineni et al. 1997; Macherey et al. 2001; Wong and Mooney 2006; Andreas et al. 2013]. However, although recent studies [Wong and Mooney 2006; Andreas et al. 2013] show that semantic parsing with synchronous context-free grammars (SCFGs), which form the basis of most existing statistical syntax-based translation models [Yamada and Knight 2001; Liu et al. 2006; Chiang 2007], achieves favorable results, this approach is still behind the most recent state of the art. For details, please see a performance comparison in Andreas et al. [2013] and Lu [2014].

The key issues behind the limited success of applying SMT systems directly to semantic parsing lie in the difference between semantic parsing and SMT: MRL is not a real NL and has different properties from NLs. First, MRL is machine interpretable and thus strictly structured with the MR in a nested structure of functors and arguments. Second, the two languages are intrinsically asymmetric since each token in MRL carries specific meaning,[1] whereas this does not hold in NL since auxiliary words and some function words usually have no counterparts in MRL. Third and finally, the expressions in NL are more flexible with respect to lexicon selection and token ordering. For example, since sentences in NL such as *could you tell me the states that utah*

[1]As seen in Section 3, delimiters, including parentheses and commas that do not carry any meaning, will be removed in preprocessing and be recovered in postprocessing.

borders, *what states does utah border*, and *utah borders what states* convey the same meaning, they should share the same expression in MRL.

Motivated by the preceding observations, we believe that semantic parsing with off-the-shelf SMT components, either phrase based, hierarchical phrase based (HPB), or even syntax based, is not an ideal approach. Alternatively, for better semantic parsing, this article proposes an effective yet simple way to enrich SCFG in HPB SMT, which achieves higher semantic parsing performance than either phrase-based or syntax-based SMT. Specifically, since the translation rules play a critical role in SMT, we explore to increase its coverage and improve translation rule quality in three ways. First, we examine the difference between word alignments for semantic parsing and SMT to better adapt word alignment in SMT to semantic parsing. Second, we enrich nonterminal symbols to capture contextual and structured information. The enrichment of nonterminal symbols not only guides the translation in favor of well-formed structures but also is beneficial to translation quality. Third, unlike most existing SMT systems that keep unknown words untranslated and intact in translation, we exploit the translation of unknown words via synthetic translation rules. Moreover, we further improve semantic parsing performance by filtering the *n*-best translation list with the motivation that a sentence in NL and its representation in MRL share the same answer type. Evaluation on the GeoQuery benchmark dataset shows that our approach obtains consistent improvement and greatly outperforms the state of the art across various languages, including English, Chinese, Thai, German, and Greek.

This article is an extension of our previous work in Li et al. [2015], where we did not make use of MRL-Trees as shown in Figure 1(c). In this work, we have improved on those earlier results in two specific ways by taking advantages of such MRL-Trees. First, we transform MR trees into WSJ format, allowing us to propose two more enriched SCFGs that employ an MR tree to refine nonterminal symbols (Sections 5.1.2 and 5.1.3). Second, we propose a simple yet effective approach of filtering the *n*-best list to improve semantic parsing performance (Section 6). Moreover, we discuss performance comparison to the phrase-based and tree-based SMT approach, as well as result analysis of semantic parsing.

The article is organized as follows. Section 2 provides an overview of related studies in semantic parsing. In Section 3, we present an overview of the framework of semantic parsing as SMT, as well as a brief of the HPB translation model. Section 4 discusses word alignment for semantic parsing, whereas Section 5 describes the details of our advanced SCFGs. In Section 6, we present our *n*-best list filtering approach to improve semantic parsing. Finally, we report on experiments that show the robustness of our approach in Section 7, then conclude the article in Section 8.

2. RELATED WORK

Various semantic formalisms have been considered for semantic parsing. Examples include the variable-free semantic representation (i.e., the MR for each utterance is tree shaped [Wong and Mooney 2006]), the lambda calculus expression [Zettlemoyer and Collins 2005; Wong and Mooney 2007], dependency-based compositional semantic (DCS) representation [Liang et al. 2011], and abstract MR [Banarescu et al. 2013]. Although there has been substantial work on semantic parsing per each semantic formalism, we limit our discussions primarily to several approaches (e.g., the SCFG approach, hybrid tree approach, and other approaches) that focus on the variable-free, functional query language, which is also the semantic formalism in this article. The functional query language is designed to allow for direct translation of the functional forms into first-order logical forms, which are used to query the target database. For more details about this semantic representation, please refer to http://www.cs.utexas.edu/∼ml/wasp/geo-funql.html.

Word Alignment-Based Semantic Parsing (WASP) [Wong and Mooney 2006] pioneers the exploration of casting the semantic parsing as a string-to-string translation problem by employing the SMT approach with SCFGs [Chiang 2007]. Different from Chiang [2007], WASP extracts nonterminal symbols from MRL tree structures and thus consequently results in hard constrains on translation phrase boundaries. Recent SMT-SemParse [Andreas et al. 2013] further casts the semantic parsing as a standard SMT problem by adapting off-the-shelf string-to-string SMT systems for semantic parsing. Although string-to-string SMT translation models usually rarely respect linguistically motivated syntax, SMT-SemParse does not exploit the structural information tree as conveyed by the semantics. Experiments of SMT-SemParse show that semantic parsing with the popular translation system Moses [Koehn et al. 2007] achieves favorable results compared to those of purpose-built semantic parsers.

HYBRIDTREE+ [Lu et al. 2008] learns a synchronous generative model that outputs hybrid tree structures. In a hybrid tree, it jointly presents the sentence and the semantic tree in a top-down recursive manner: words in the sentence as leaves and semantic units in the semantic tree as nodes. Relaxed hybrid trees (RHT) [Lu 2014] further defines distributions over RHT structures that jointly represent both sentences and semantics. Most recently, f-RHT (RHT with constrained semantic forests) [Lu 2015] introduces constrained semantic forests to improve the RHT model. tsVB Tree transducers with variational Bayes (tsVB) [Jones et al. 2012] uses tree transducers, which are similar to the hybrid tree structures, to learn a generative process under a Bayesian framework.

Semantic Composition That Integrates Syntax and Semantics to Get Optimal Representations (SCISSOR) [Ge and Mooney 2005] uses a two-step strategy for semantic parsing. It first parses a sentence in NL into a semantically augmented parse tree, in which each nonterminal node has both a syntactic and a semantic label. Then a compositional semantics procedure is used to map the augmented parse tree into a final MR. Such a semantically augmented parse tree conveys more information but requires language-specific syntactic analysis. Kernel-Based Robust Interpretation by Semantic Parsing (KRISP) [Kate and Mooney 2006] uses string classifiers to label substrings of an NL sentence with entities from the MR. The unification-based learning (UBL) algorithm [Kwiatkowski et al. 2010] performs semantic parsing with an automatically induced CCG lexicon, in which a lexical item or a word sequence is associated with a semantic unit. The model can parse a, NL sentence into both variable-free semantic representation and the lambda calculus expression.

Moveover, Goldwasser et al. [2011] explored an unsupervised approach for semantic parsing based on self-training driven by confidence estimation. Jie and Lu [2014] considered multilingual semantic parsing that simultaneously parses semantically equivalent sentences from multiple different NLs into their corresponding variable-free semantic representations.

In addition to the preceding approaches that focus on the variable-free semantic representation, there has also been substantial work on alternative semantic representations. For instance, Poon and Domingos [2009] proposed an unsupervised approach to learn a semantic parser that transforms dependency trees into semantic representations using Markov logic [Richardson and Domingos 2006]. Liang et al. [2011] proposed a model for learning the DCS, which can be customized for optimizing the end-task performance. Artzi and Zettlemoyer [2013] proposed a grounded CCG semantic parsing approach to learn a model for mapping NL instructions to actions with weak supervision. Li et al. [2013] extended the GHKM algorithm [Galley et al. 2004], a state-of-the-art algorithm for learning synchronous grammars in SMT, for parsing NL sentences into their lambda calculus expressions.

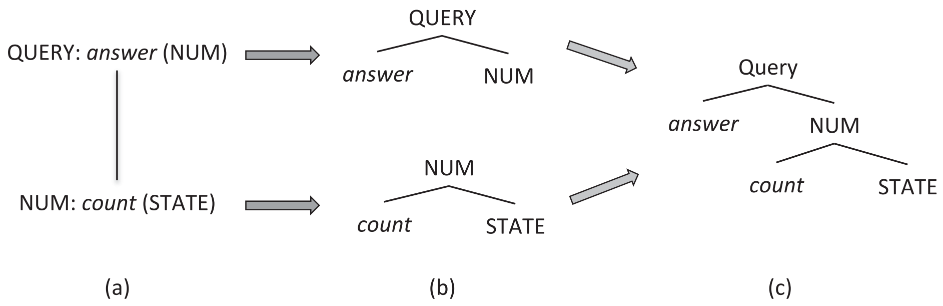


Fig. 2. Illustration of tree transformation from MRL-Tree to MRL-Tree′.

3. BACKGROUND

In this section, we first present the framework of semantic parsing as SMT, which was proposed by Andreas et al. [2013]. Then we briefly review the HPB translation model, which is one of the most popular and effective SMT models.

3.1. Semantic Parsing as SMT

In Figure 1, the MRL (e.g., Figure 1(b)) is represented as a series of nested functions, whereas Figure 1(c) illustrates its equivalent tree structure. Each node in Figure 1(c) represents a semantic unit in the form $m_a \equiv \tau_a : p_a(\tau_b*)$, where m_a denotes a complete semantic unit consisting of its semantic type τ_a, its functor p_a, as well as an argument list τ_b*. Alternatively, we can view a semantic unit as a function that takes in other semantics of specific types as arguments and returns new semantics of a particular type. The node "QUERY: *answer* (NUM)," for example, indicates that the function *answer* takes a single argument with type NUM and returns a semantic unit with type QUERY.

Preprocessing. To learn the semantic parser using off-the-shelf SMT components, we need to transform these MRs to forms more similar to NL. To do so, we simply transform the MR tree structures, like Figure 1(c), into more popular WSJ-style tree structures, in which terminals represent its corresponding MR. This is done by the following three steps:

(1) For each semantic unit in an MR structure with a form of $\tau_a : p_a(\tau_{b1}, \tau_{b2}, \ldots, \tau_{bn})$, we create a node with name of τ_a, then create its $n+1$ children from left to right: $p_a, \tau_{b1}, \tau_{b2}, \ldots, \tau_{bn}$. Figure 2(b) shows the subtree construction for units "QUERY: *answer* (NUM)" and "NUM: *count* (STATE)."
(2) For two linked semantic units in an MR structure, we merge their corresponding subtrees to form a new one, as shown in Figure 2(c).
(3) Finally, we merge all created subtrees into one: the terminals are either the functor labels or the argument value labels. For functor labels, we augment it with the number of arguments it takes. And for argument values, we argument it with *"s"* if the value is a string or *"0"* otherwise. Figure 3(c) shows the transformed output of MRL-Tree in Figure 1(c).

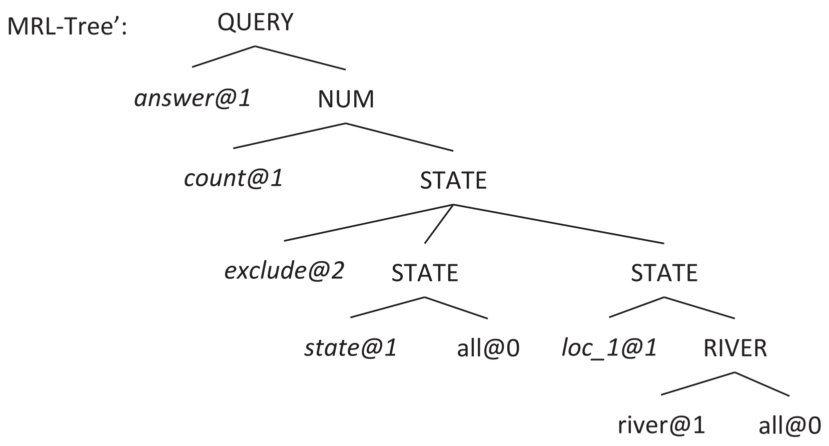
It is worth noting that the terminals in the transformed tree are the same as the words in the preprocessed MRL in Andreas et al. [2013]. On the source side, we perform stemming (for English only) and lowercasing to overcome data sparseness. Hereafter, we refer to the preprocessed NL, MRL, and MRL-Tree as NL′, MRL′, and MRL-Tree′, respectively.

NL′: how many state do not have river

(a)

MRL′: answer@1 count@1 exclude@2 state@1 all@0 loc_1@1 river@1 all@0

(b)



(c)

Fig. 3. Example of preprocessed NL, MRL and MRL-Tree.

Translation. Given a corpus of NL′ sentences paired with MRL′ or MRL-Tree′, we learn a semantic parser by adopting off-the-shelf HPB translation systems.[2] In the rest of this article, we refer to the source language (side) as NL′ and the target language (side) as MRL′.

Postprocessing. We convert MRL′ back into MRL by recovering parentheses and commas to reconstruct the corresponding tree structure in MRL. This can easily be done by examining each symbol's arity. It eliminates any possible ambiguity from the tree reconstruction: given any sequence of tokens in MRL′, we can always reconstruct the tree structure (if one exists). For those translations that cannot be successfully converted, we call them *ill-formed* translations.

3.2. HPB Translation Model

The HPB translation model is one type of string-to-string models. In HPB models, synchronous rules take the form $X \rightarrow \langle \gamma, \alpha, \sim \rangle$, where X is the nonterminal symbol; γ and α are strings of lexical items and nonterminals in the source and target side, respectively; and $\sim$ indicates the one-to-one correspondence between nonterminals in γ and α. Each such rule is associated with a set of translation model features $\{\phi_i\}$, including phrase translation probability $p(\alpha \mid \gamma)$ and its inverse $p(\gamma \mid \alpha)$, the lexical translation probability $p_{lex}(\alpha \mid \gamma)$ and its inverse $p_{lex}(\gamma \mid \alpha)$, and a rule penalty that

[2]Actually, we can also learn a semantic parser with either phrase-based or string-to-tree translation systems. However, the semantic parsing performance with phrase-based and tree-based systems is much lower than that with HPB systems, as illustrated in Section 7.2.

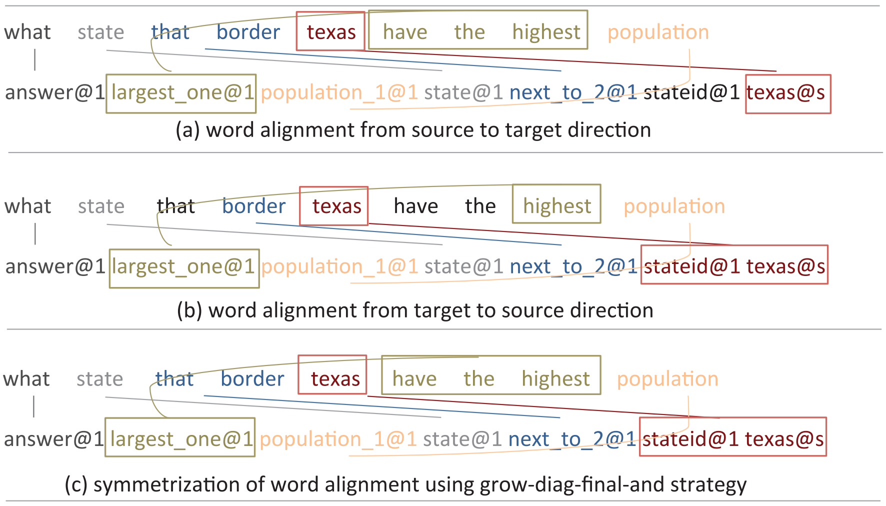

what state that border texas have the highest population
answer@1 largest_one@1 population_1@1 state@1 next_to_2@1 stateid@1 texas@s
(a) word alignment from source to target direction
what state that border texas have the highest population
answer@1 largest_one@1 population_1@1 state@1 next_to_2@1 stateid@1 texas@s
(b) word alignment from target to source direction
what state that border texas have the highest population
answer@1 largest_one@1 population_1@1 state@1 next_to_2@1 stateid@1 texas@s
(c) symmetrization of word alignment using grow-diag-final-and strategy

Fig. 4. Example of a sentence pair with different alignments.

affects a preference for longer or shorter derivations. Two other widely used features are a target language model feature and a target word penalty.

Given a derivation d, its translation probability is estimated as

$$P(d) \propto \prod_i \phi_i (d)^{\lambda_i}, \tag{1}$$

where λ_i is the corresponding weight of feature ϕ_i. See Chiang [2007] for more details.

4. WORD ALIGNMENT FOR SEMANTIC PARSING

Word alignment is an essential step for rule extraction in SMT, where recognizing that *wo shi* in Chinese is a good translation for *I am* in English requires establishing a correspondence between *wo* and *I*, and between *shi* and *am*. In the SMT community, researchers have developed standard, proven alignment tools such as GIZA++ [Och and Ney 2003], which can be used to train IBM models 1 through 5. However, there is one fundamental problem with the IBM models [Brown et al. 1993]: each word on one side can be traced back to exactly one particular word on the other side (or the *null* token, which indicates that the word aligns to no word on the other side). Figure 4(a) shows an example of GIZA++ alignment output from the source side to the target side, from which we can see that each source word aligns to exactly one target word. Although alignment of multiple target words to one source word is common in SMT, a trick is then to run IBM model training in both directions. Then two resulting word alignments can be symmetrized—for instance, taking the intersection or the union of alignment points of each alignment. For example, Figure 4(b) shows GIZA++ alignment output from the target side to the source side, whereas Figure 4(c) shows the symmetrization result with the widely used *grow-diag-final-and* strategy.

Although symmetrization of word alignments works for SMT, can it be applied to semantic parsing? There are reasons to be doubtful. Word alignment for semantic parsing differs from alignment for SMT in several important aspects, at least including the following:

(1) *It is intrinsically asymmetric*: Within the semantic formalism used in this article, NL′ is often longer than MRL′ and commonly contains words that have no counterpart in MRL′.

(2) *Little training data is available*: SMT alignment models are typically trained in an unsupervised fashion, inducing lexical correspondences from massive quantities of sentence-aligned bitexts.

Consequently, the symmetrization of word alignments may not work perfectly for semantic parsing. According to word alignment in Figure 4(c), a phrase extractor will generate a phrase pair ⟨*have the highest, largest_one@1*⟩, which is nonintuitive. By contrast, a more useful and general phrase pair ⟨*highest, largest_one@1*⟩ is typically excluded because *largest_one@1* aligns to three source words: *have*, *the*, and *highest*. Similarly, another useful phrase pair ⟨*texas, texas@s*⟩ is prohibited, since *texas* aligns to both *stateid@1* and *texas@s*.

Ideally, a new semantic parsing aligner should be able to capture the semantic equivalence. Unfortunately, we are not aware of any research on alignment for semantic parsing, possibly due to lack of a paucity of high-quality, publicly available data from which to learn. Instead of developing a new alignment algorithm for semantic parsing, we make use of all alignments as shown in Figure 4. In other words, we triple the training data with each sentence pair having three alignments: two alignments in both directions and the symmetrization alignment.[3] The advantages include the following. First, considering more possible alignments would increase the phrase coverage, especially when the training data is little. Second, including the alignment from both directions would alleviate the error propagation caused by mis-aligned stop words (e.g., *be*, *the* in NL′ and *stateid@1* in MRL′). As a result, the phrase extractor will include phrase pairs of both ⟨*highest, largest_one@1*⟩ and ⟨*texas, texas@s*⟩. Our experiment shows that using the combination of all three alignments achieves better performance than using any one alone. Moreover, we found that we could achieve comparable performance even with manual alignment on English.

5. SEMANTIC PARSING WITH ENRICHED SCFG

In this section, we present the details of our enriched SCFGs for semantic parsing.

5.1. Enriched SCFGs

From an aligned phrase pair <*state that border, state@1 next_to_2@1*> in Figure 4(a), for example, we can extract a synchronous rule $X \rightarrow \langle \text{state } X_{\boxed{1}},\ \text{state@1 } X_{\boxed{1}} \rangle$, where we use boxed indices to indicate which nonterminal occurrences are linked by $\sim$. The fact that SCFGs in HPB models contain only one type of nonterminal symbol[4] is responsible for ill-formed translation (e.g., *answer@1 state@1*, of which the functor *state* has no argument on the right side) or invalid translation (e.g., *answer@1 highest@1 state@1 all@0*, which violates the functor *highest*'s argument type requirement). To this end, we propose three different schemes with either structural or syntactic or both information to relabel nonterminal symbol X into refined ones. This is also inspired by recent studies in SMT that have shown that introducing multiple nonterminal symbols in SCFGs benefits translation [Zollmann and Venugopal 2006; Li et al. 2012]. For easy notation, we refer to the traditional SCFG with one type of nonterminal symbol X as a nonenriched SCFG, whereas we refer to the enriched SCFGs with the following three schemes as a structure-enriched SCFG (Section 5.1.1), syntax-enriched SCFG (Section 5.1.2), and structure-syntax–enriched SCFG (Section 5.1.3), respectively.

[3]Combining multiple alignments from different alignment models usually improves translation performance in SMT [Tu et al. 2012]. However, our preliminary experiments showed that this did not yield higher improvement in semantic parsing, which in turn also demonstrates the difference in alignments for semantic parsing and SMT.
[4]In practice, nonterminal symbol S is used in glue rules. However, this is not relevant in the present discussion.

Table I. Examples of Nonterminal Symbols over Different SCFG Schemes

target phrase	Structure Enriched	Syntax Enriched	Structure-syntax Enriched
river@1 all@0	C	RIVER	RIVER
exclude@2 state@1 all@0	C/A1	exclude@2+STATE	STATE/A1
answer@1 count@1	C/A1	QUERY/STATE	QUERY/A1
state@1 all@0 loc_1@1	(C/A1)\F2	X	(STATE/A1)\F2

Similar to the tree-to-string translation model, the nonenriched SCFGs use syntactic labels to refine nonterminal symbols in HPB, capturing the syntactic information of translation phrases. In contrast, the nonenriched SCFGs allow one to include phrases corresponding to not only the well-constructed subtrees but also broken subtrees.

5.1.1. Enriched Nonterminals with Structural Information. To handle MRL with a nested structure, we enrich the nonterminals to capture the structure information, guiding the translation in favor of well-formed translations. Given a word sequence e_j^i from position i to position j in MRL′, we enrich the nonterminal symbol X to reflect the internal structure of the word sequence of e_j^i. A correct translation rule selection therefore not only maps source terminals into target terminals but is both constrained and guided by structure information in the nonterminals. As mentioned earlier, we regard the nested structure in MRL′ as a function-argument structure where each function takes one or more arguments as input, whereas its return serves as an argument to the outside function. As in Figure 1, the function *exclude* holds two arguments and returns as an argument to the function *count*. For a word sequence e_j^i, we examine its completeness, which is defined as follows.

Definition 1. For word sequence e_j^i, it is regarded as complete if it satisfies the following: (1) every functor (if it exists) meets its argument requirement, and (2) it can serve as one argument to another functor.

We use symbol C to label word sequences that are complete. For an incomplete word sequence, we examine (1) the number of arguments that it requires on the right to be complete and 2) the arity of a functor that it requires on the left to be complete. Then the sequence is labeled as $(C/An)\backslash Fm$, indicating that the completeness requires n arguments on the right and a functor with m arities on the left.[5] Specifically, we omit $\backslash Fm$ and $/An$ if $m = 0$ (or $m = 1$) and $n = 0$, respectively.[6] Note that in this scheme, we do not use the tree structure information to relabel nonterminal X.

The second column in Table I demonstrates a few examples of nonterminal symbols in our structure-enriched SCFG. For instance, the target side word sequence *river@1 all@0* is complete and thus labeled as C. Similarly, to be complete, word sequence *state@1 all@0 loc_1@1* requires one additional argument on the right side as well as a functor with two arities on the left side, labeled as $(C/A1)\backslash F2$ accordingly.

5.1.2. Enriched Nonterminals with Syntactic Information. Given parse trees (i.e., MRL-Tree′) of MRL′, we can relabel nonterminal symbols with syntax information to reflect the internal syntax of the word sequence.[7] Inspired by Zollmann and Venugopal [2006], we refine nonterminal symbol X by using target-side parse trees. If the target-side word sequence e_j^i corresponds to a syntactic category S of the target-side parse tree, we label

[5]This is similar to the naming convention in combinatory categorial grammar (CCG) [Steedman 2000].
[6]If $m = 0$ or $m = 1$, it indicates that no functor on the left side is needed.
[7]It may be not correct to refer to MRL-Tree′ as a syntactic structure of MRL′. However, this is not relevant in the present discussion either.

the phrase pair with that syntactic category (e.g., *loc_1@1 river@1 all@0: STATE*).[8] This label corresponds to the left-hand side of our synchronous grammar. If e_j^i is not spanned by a single constituent in the parse tree, we use the labels of subsuming, subsumed, and neighboring constituents in MRL-Tree′ to assign an extended label of the form S_1+S_2, S_1/S_2, or $S_2\backslash S_1$, indicating that the phrase pair's target side spans two adjacent syntactic categories (e.g., *state@1 all@0 loc_@1 river@1 all@0: STATE+STATE*), a partial syntactic category S_1 missing a S_2 at the right (e.g., *exclude@2 state@1 all@0: STATE/STATE*) or a partial S_1 missing a S_2 at the left, respectively. If no label is assignable by either of these three methods, a default label X is assigned (e.g., *all@0 loc_1@1: X*).

The third column in Table I demonstrates a few examples of nonterminal symbols in our syntax-enriched SCFG. For instance, target-side word sequence *river@1 all@0* in Figure 3(c) corresponds to syntactic category *RIVER* and thus is labeled as *RIVER*. Similarly, word sequence *exclude@2 state@1 all@0* maps to two neighboring constituents *exclude@2* and *STATE*, labeled as *exclude@2+STATE*.

5.1.3. Enriched Nonterminals with Both Structural and Syntactic Information. Although structure-enriched SCFG guides the translation in favor of those satisfying the arity requirement (i.e., a functor is with a correct number of arguments) and syntax-enriched SCFG is in favor of those satisfying the argument type requirement (i.e., a functor is with correct argument type), we propose structure-syntax–enriched SCFG, which takes both the arity and argument type into account. In structure-syntax-enriched SCFG, if the target-side word sequence e_j^i corresponds to a syntactic category S of the target-side parse tree, we label the phrase pair with that syntactic category (e.g., *loc_1@1 river@1 all@0: STATE* as in syntax-enriched SCFG). Otherwise, we relabel it as $(S/An)\backslash Fm$, where S is the syntactic category of the lowest ancestor node that fully covers e_j^i, similarly $/An$ and $\backslash Fm$ indicate that it requires n arguments on the right and a functor with m arities on the left for the sequence to be complete. This is similar to structure-enriched SCFG, except instead we use the syntactic category to imply the type of the word sequence.

The fourth column in Table I demonstrates a few examples of nonterminal symbols in our structure-syntax–enriched SCFG. For instance, target-side word sequence *river@1 all@0* in Figure 3(c) maps to the syntactic category *RIVER* and thus is labeled as *RIVER*. Similarly, to be complete as its lowest ancestor node *STATE*, word sequence *state@1 all@0 loc_1@1* requires one additional argument on the right side as well as a functor with two arities on the left side, labeled as *(STATE/A1)\F2* accordingly.

5.2. Glue Rules

In SMT decoding [Chiang 2007], if no rule (e.g., a rule whose left-hand side is X) can be applied or the length of the potential source span is larger than a predefined length (e.g., 10 as in Chiang [2007]), a glue rule (either $S \rightarrow \langle X_{\boxed{1}}, X_{\boxed{1}}\rangle$ or $S \rightarrow \langle S_{\boxed{1}}\, X_{\boxed{2}}, S_{\boxed{1}}\, X_{\boxed{2}}\rangle$) will be used to simply stitch two consequent translated phrases together in a straight way.However, such two glue rules are not applicable to our models, as we include more refined nonterminals in our SCFGs. Moreover, the strategies of using straight glue rules only and of applying glue rules on top of derivations in SMT decoding lie in the fact that violating these strategies will fiercely increase computational and modeling challenges, considering that the training data size for SMT is usually at a scale of millions. Alternatively, in this work, we additionally use an inverted glue rule that

[8]If e_j^i exactly corresponds to two or more constituents in the parse tree due to unary rules, we always use the syntactic category of the topmost one.

Table II. Examples of Glue Translation Rules in a Structure-Enriched SCFG

Source Side	Target Side	Glue Rule
state that border	state@1 next_to_2@1	$C/A1 \rightarrow \langle C/A1_{\boxed{1}}\, C/A1_{\boxed{2}},\ C/A1_{\boxed{1}}\, C/A1_{\boxed{2}} \rangle$
state that border texas have the highest population	l.._one@1 p.._1@1 state@1 n..@1 s..@1 t..@s	$C \rightarrow \langle C_{\boxed{1}}\, C/A1_{\boxed{2}},\ C/A1_{\boxed{2}}\, C_{\boxed{1}} \rangle$

Note: The underline and underwave indicate the first and second phrases, respectively.

ALGORITHM 1: Generating Synthetic Translation Rules for Unknown Words

```
Input: Unknown word $w_u$ in the source language
       Source-side vocabulary from the training data: $W$
       Lexical translation tables T1 and T2 (two directions)
Output: Synthetic translation rule set $R$ for $w_u$
1. foreach word $w_i$ in $W$
2.     $s_i = sim(w_u, w_i)$
3. get the top $n$ words $WB = \{wb_1 \dots wb_n\}$ with the highest $\{s_i\}$
4. $R = \phi$
5. foreach $wb_i$ in $WB$
6.     foreach $t_j$ such $\langle wb_i, t_j \rangle$ in T1 and T2
7.         $R \cup = generate\_rule(w_u, wb_i, t_j, T_1, T_2)$
8. return $R$
```

sim: returns the similarity between w_u and w_i.

generate_rule: returns rule $\langle w_u, t_j \rangle$ with a feature indicating the similarity between w_u and wb_i, and two features indicating the lexical translation probabilities from wb_i to t_j and the way around.

combines two nonterminals in an inverted way. Each glue rule, either straight or inverted, contains only two nonterminal symbols and is associated with two features, including phrase translation probability $p(\alpha \mid \gamma)$ and a glue rule penalty. Table II shows examples of straight and inverted glue rules in a structure-enriched SCFG (and likewise for syntax-enriched and structure-syntax–enriched SCFGs). Moreover, these glue rules can be applied to any two neighboring translation nodes if the nonterminal symbols are matched.

5.3. Synthetic Translation Rules for Unknown Word Translation

Most NLP tasks face the problem of unknown words, especially if only little training data is available. For example, it is estimated that 5.7% of sentences in the (English) test data in our experiments have unknown words. Unknown words usually remain intact in the translation in most machine translation systems [Koehn et al. 2007; Dyer et al. 2010], resulting in the fact that certain translations cannot be converted back to tree structures. This indicates that in semantic parsing, the translation of a word can be from two categories: (1) a token in MRL or (2) *null* (i.e., not translated at all); we generate synthetic translation rules for unknown word translation.

As a baseline, we simply skip unknown words like Kwiatkowski et al. [2010] by adding translation rules that translate them to *null* in MRL′. Each such rule is accompanied with one feature indicating that it is a translation rule for an unknown word.

Alternatively, taking advantage of publicly available resources, we generate synthetic translation rules for unknown words pivoted by their semantically close words. Algorithm 1 illustrates the process to generate synthetic translation rules for unknown word translation. Given an unknown word w_u, it generates its synthetic rules in two steps: (1) finding top n (e.g., 5 as in our experiments) close words via Word2Vec[9] and (2) generating synthetic translation rules based on the close words. Note that it may generate a synthetic rule with *null* at the target side since the lexical translation table derived from aligned training data contains translation to *null*. Each synthetic translation rule for unknown words is associated with three features returned from the function *generate_rule*.

6. FILTERING: IMPROVE SEMANTIC PARSING WITH PREDICTED ANSWER TYPE

An MRL-Tree′ illustrates the step-by-step process of parsing MRL′ to generate the answer for its corresponding input question. From an MRL-Tree′, we use the following two rules to retrieve the answer type by examining the top production in MRL′ (i.e., $\tau_a \rightarrow p_a + \tau_{b1} + \tau_{b2} + \cdots + \tau_{bn}$):

—if p_a is *answer@1*, then the answer type is τ_{b1};
—otherwise, the answer type is τ_a.

For example, the answer type of Figure 3(c) is *NUM*, indicating that the answer for the input is a number. In the preceding string-to-string translation approach with enriched SCFGs, translations are optimized to achieve the highest score of Equation (1) while toward well-formed MRs. However, this ignores the fact that the translations, even well formed, may not lead to the correct answer type. Therefore, we need to restrict ourselves to translations corresponding to not only well-formed MRs but also to MRs with the correct answer type. In principle, this could be done by rewriting the beam search algorithm used in decoding to encourage translations that lead to preferred answer types. Alternatively in this section, we simply filter the regular n-best translation list until we find a translation that presumedly has the same answer type as the input source sentence.

Predict answer type for NL′. Given an input question in NL′, we treat the problem of predicting its answer type (*NUM* for number, *CITY* for city, etc.) as a multiple classification. Predicting the answer type is relatively straightforward, as the input question itself has strong signs to imply its answer type. For example, *how many* usually implies an answer type *NUM*, whereas *give me the lake* implies an answer type *PLACE*. To this end, we simply use 1∼5-gram as features to predict its answer type.

Predict answer type for MRL′. Given an input in MRL′, we can also treat the problem of predicting its answer type as a multiple classification, as the leftmost function labels also strongly imply a certain answer type. For example, *answer@1 city@1* implies answer type *CITY*. However, casting it as a pure classification problem is not enough to know whether the MRL′ is valid or not. Alternatively, we train a syntactic parser to parse an input in MRL′ into its parse tree in MRL-Tree′. Then we retrieve the answer type from its MRL-Tree′. If an input in MRL′ cannot be successfully parsed into a tree in MRL-Tree′ with existing grammars, it is reasonable to believe that the MRL′ is invalid, although it might be well informed.

Use predicted answer types to filter the translation list. Given an input nl in NL′, we first find the n-best list $\{mrl_1, \ldots, mrl_n\}$ of translations in MRL′. Then we predict the answer type for input nl while we get the answer type for each entry mrl_i in the n-best list (if the answer type exists). Next, we filter the n-best translation list until we find a

[9]It is available at http://code.google.com/p/word2vec/. We use Word2Vec rather than other linguistic resources like WordNet because the approach can be easily adopted to other languages only if there exists large monolingual data to train Word2Vec models.

Table III. Size of the Preprocessed GeoQuery

	English		Chinese		Thai	
	NL′	MRL′	NL′	MRL′	NL′	MRL′
Size of vocabulary	252	165	260	165	274	165
Number of words	6,660	5,286	6,209	5,286	7,606	5,286

translation whose answer type is the same as the predicted answer type of input *nl*. If no such translation is found, we keep the top best translation as our output.

7. EXPERIMENTATION

In this section, we test our approach on the GeoQuery dataset, which is publicly available.

7.1. Experimental Settings

Data. The GeoQuery dataset consists of 880 questions paired with their MRLs in variable-free format, as well as their corresponding tree-structured semantic representations.[10] The original dataset was fully annotated in English, and recently Jones et al. [2012] released a new version of this dataset with four additional language annotations (German, Greek, Chinese, and Thai). Due to space considerations, we mainly report the performance on English and the other two Asian languages, Chinese and Thai, and the averaged performance on English, Chinese, and Thai. Table III presents the vocabulary size and number of words in the three languages. Following the experimental setup in Jones et al. [2012], we use the 600 question pairs to train and tune our SMT decoder and evaluate on the remaining 280. Note that there is another version of GeoQuery dataset where the semantic representation is annotated with lambda calculus expressions and which is extensively studied [Zettlemoyer and Collins 2005; Wong and Mooney 2007; Liang et al. 2011; Kwiatkowski et al. 2013; Li et al. 2013]. Performance on the version of lambda calculus is usually higher than that on the tree-structured version; however, the results obtained over the two versions are not directly comparable.

SMT setting. We use cdec [Dyer et al. 2010] as our HPB decoder. As mentioned earlier, 600 instances are used to train and tune our decoder. To get fair results, we split the 600 instances into 10 folds, each having 60 instances. Then for each fold, we use it as the tuning data, whereas the other 540 instances and the NP list are used as the training data.[11] We use the IRSTLM toolkit [Federico et al. 2008] to train a 5-gram LM on the MRL′ side of the training data, using modified Kneser-Ney smoothing. We use MIRA [Chiang et al. 2008] to tune the parameters of the system to maximize BLEU [Papineni et al. 2002]. When extracting translation rules from aligned training data, we include both *tight* phrases, which are bounded by word alignments, and *loose* phrases, which allow unaligned words at phrase edges.

Evaluation. We use the standard evaluation criteria for evaluation by executing both the predicted MRL and the gold standard against the database and obtaining their respective answer.[12] Specifically, we convert a translation from MRL′ into MRL (if it exists). The translation then is considered correct if and only if its MRL retrieves

[10]The GeoQuery dataset is available at http://www.cs.utexas.edu/∼ml/wasp/geo-funql/corpus.xml.
[11]The NP list is from the GeoQuery dataset in Jones et al. [2012], which contains MRs for every noun phrase that appears in the NL utterances of each language. The NP list has about 260 entries in every language. As in Andreas et al. [2013], the NP list is included by appending all entries as extra training sentences with 50 times the weight of regular training examples to ensure that they are learned as translation rules.
[12]The WASP 1.0 toolkit provides API to obtain answers against the database. WASP 1.0 is available at http://www.cs.utexas.edu/users/ml/wasp/.

Table IV. Performance over Phrase-Based, HPB, and String-to-Tree Translation Models

Model	English		Chinese		Thai		Avg.	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Phrase based (NiuTrans)	67.3	75.4	72.7	79.6	60.2	67.6	66.7	74.2
HPB (NiuTrans)	74.6	81.8	**74.9**	**81.0**	63.8	70.6	71.1	77.8
String-to-tree (NiuTrans)	63.5	70.1	60.3	66.0	47.7	55.0	57.2	63.7
HPB (cdec)	**77.5**	**83.5**	**74.9**	**81.0**	**65.4**	**72.4**	**72.6**	**79.0**

the same answers as the gold standard MRL [Jones et al. 2012], allowing for a fair comparison between our systems and previous works. As in Jones et al. [2012], we report accuracy, which is the percentage of translations with correct answers, and F1, which is the harmonic mean of precision (the proportion of correct answers out of translations with an answer) and recall (the proportion of correct answers out of all translations). In this section, we report our performance scores and analysis numbers averaged on our 10 SMT models.

7.2. Phrase-Based Versus HPB Versus the String-to-Tree Translation Model

In addition to HPB translation models, we also noticed a recent advance in phrase-based and tree-based translation models. Specifically, applying tree-based models, such as string-to-tree or tree-to-tree translation models [Yamada and Knight 2001; Shen et al. 2008], to semantic parsing will naturally resolve the inconsistent semantic structure issue since the translation is of tree structure. However, due to hard constraints that each target phrase needs to strictly map to a syntactic constituent, phrase tables in tree-based translation models usually suffer from the low coverage issue, especially if the training data size is small. By taking advantage of MRL-Tree′, we use NiuTrans [Xiao et al. 2012] as our string-to-tree decoder to test the string-to-tree model on the task of semantic parsing. For better comparison, we also report the performance of the phrase-based decoder and the HPB decoder of NiuTrans. For all translation models, we use their default parameter settings, except we additionally include loose phrases in both cdec and NiuTrans HPB models.

Table IV shows the results of the four translation models. According to the performance of NiuTrans, we clearly observe that the string-to-string translation model, either phrase based or HPB, substantially outperforms the string-to-tree model across all languages, indicating that the former is more appropriate for semantic parsing than the latter. To better know the reasons why the string-to-tree model is far from ideal, a closer examination finds that 58/48/75 (20%/17%/27%) sentences in the English/Chinese/Thai test set even fail to yield a well-formed translation in the string-to-tree translation model, respectively. This is partially due to the low coverage of translation rules. Statistics shows that the string-to-tree model extracts only 4.5K translation rules in total from English training sentence pairs. To avoid problems with the strictness that requires a phrase to map to a syntactic constituent, recent research in tree-based SMT increases the translation rule coverage by leveraging syntactic constraints yet still allows cross-constituent translations. However, applying the improved tree-based SMT to semantic parsing is beyond the scope of this article. Table IV also indicates that the HPB translation model outperforms the phrase-based model, which agrees with results in Andreas et al. [2013]. Finally, although the two HPB decoders obtain similar performance on Chinese, the HPB of cdec outperforms that of NiuTrans on the other two languages. We blame the performance difference of the two HPB decoders on the implementation details and default parameters. In the rest of this article, we stick to the HPB decoder of cdec.

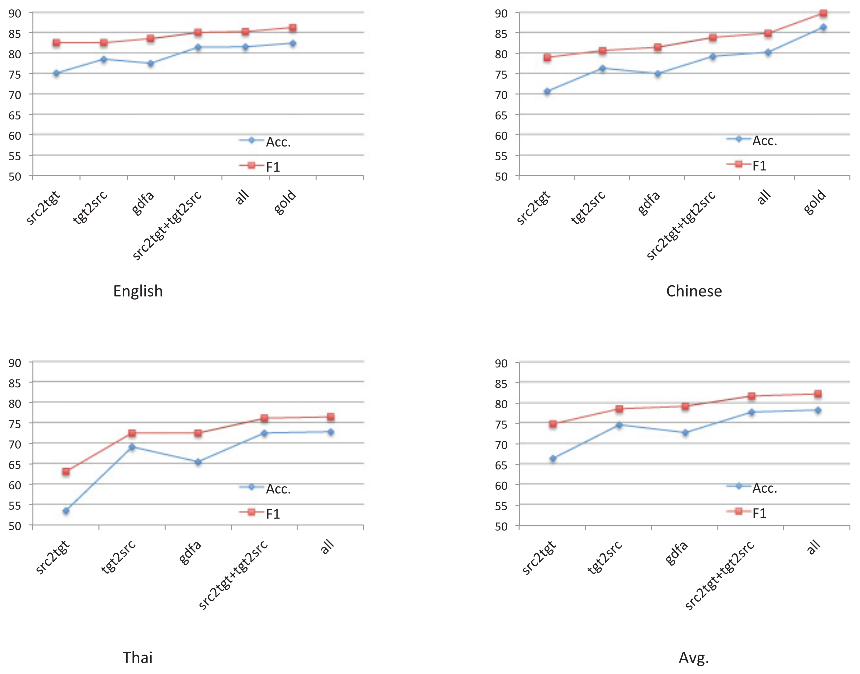


Fig. 5. Semantic parsing performance over different word alignment strategies.

7.3. Effect of Word Alignment

We compare six different word alignment strategies over the performance of semantic parsing:

—*src2tgt* indicates alignment of the source to target direction;
—*tgt2src* indicates alignment of the target to source direction;
—*gdfa* indicates symmetrization over src2tgt and tgt2src alignments with the *grow-diag-final-and* strategy;
—*src2tgt+tgt2src* indicates doubling the training data with each sentence pair having two alignments (i.e., *src2tgt* and *tgt2src*);
—*all* indicates tripling the training data with each sentence pair having three alignments (i.e., *src2tgt*, *tgt2src*, and *gdfa*); and
—*gold* indicates using gold alignment, and we manually aligned sentence pairs between NL′ and MRL′ in English and Chinese.

Figure 5 shows semantic parsing performance over different word alignment strategies among various languages. From it, we observe quite consistent patterns across the three languages and make the following observations:

—Semantic parsing is substantially sensitive to alignment. Surprisingly, *gdfa* alignment, which is widely adopted in SMT, is obviously inferior to *tgt2src* alignment in accuracy. As expected, *src2tgt* alignment achieves the worst performance. Examining the gold alignment reveals that the gold alignment is more apt to the assumption of *tgt2src* alignment: each semantic word tends to align to exactly one particular source word (or the *null* token).
—Thanks to the increased coverage, doubling the training data (e.g., *src2tgt+tgt2src*) usually outperforms its corresponding single alignment. Moreover, tripling the training data (e.g., *all*) achieves slightly better performance than doubling the training

Table V. Alignment Performance on English and Chinese

Alignment	English			Chinese		
	Rec.	Pre.	F1	Rec.	Pre.	F1
src2tgt	**96.3**	68.8	80.3	69.0	50.6	58.4
tgt2src	90.0	**77.0**	**83.0**	60.0	**51.2**	55.1
gdfa	95.1	62.7	75.6	**74.7**	49.7	**59.7**

Table VI. Performance over Different SCFGs

SCFG	English		Chinese		Thai		Avg.	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Nonenriched	81.5	85.2	80.2	84.8	72.7	76.4	78.1	82.1
Structure enriched	82.9	86.1	84.9	**88.1**	75.2	77.9	81.0	84.0
Syntax enriched	81.0	85.5	81.9	85.9	77.2	**80.9**	80.0	84.1
Structure-syntax enriched	**83.8**	**86.9**	**85.6**	87.8	**78.6**	80.4	**82.7**	**85.0**

data. This is expected since the *gdfa* alignment actually originates from the alignments of *src2tgt* and *tgt2src*, thus doubling the training data with *src2tgt* and *tgt2src* has already included most aligns in *gdfa* alignment.

—On English, our approach of tripling the training data achieves comparable performance to the one with gold alignment, suggesting that instead of developing a brand new algorithm for semantic parsing alignment, we can simply make use of GIZA++ alignment output. However, although we obtain solid improvement on Chinese, there is still some room for improvement by building a better semantic parsing aligner.

We examine the alignment performance against the gold alignment on English and Chinese training data (except the NP list part). As shown in Table V, the alignment performance of English is much better than that of Chinese. The reasons for lower performance of alignment for Chinese include the following. First, Chinese sentence structure is more flexible, resulting in frequent inverted reorderings. This is illustrated by the statistics on the two gold alignments that 83% aligns of Chinese cross with at least one other align, whereas only 16% aligns of English are crossed. Second, the Chinese functional word *DE* is ubiquitous and is a major source of alignment error. The alignment performance for *DE* is 30.0 in F1, which is much lower than the overall performance.

Due to better semantic parsing performance achieved by using *all* alignment, in the rest of this section, we take *all* alignment as our default alignment strategy.

7.4. Effect of Enriched SCFG

Table VI shows the results of our HPB translation model over different SCFGs. From the table, we observe that all of our three enriched SCFG systems outperform the nonenriched SCFG system over all languages, indicating the effect of enriching nonterminals. Our structure-enriched SCFG has extracted about 57, 52, and 61 glue rules from the training data on English, Chinese, and Thai, respectively. However, the size of glue rules soars sharply when moving to the other two enriched SCFGs. For example, our syntax-enriched SCFG (structure-syntax–enriched SCFG) has extracted about 3.2K (1.2K), 2.0K (1.0K), and 2.3K (1.0K) glue rules on the three languages.The increase is due to the use of more refined nonterminals in the other two enriched SCFGs, as shown in Table I. Moreover, compared to structure-syntax–enriched SCFG, the syntax-enriched one is more likely to include MRL′ words as a nonterminal label (e.g., *exclude@2* + *STATE* in Table I), resulting in more glue rules.

As mentioned earlier, the nonenriched SCFG system may result in ill-formed translations, which cannot be converted back to the tree structure. One natural way to

Table VII. Performance with Unknown Word Translation

	English		Chinese		Thai		Avg.	
System	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
No unknown word translation	83.8	86.9	85.6	87.8	78.6	80.4	82.7	85.0
Null: baseline	86.8	87.6	86.9	87.3	**80.1**	**80.4**	**84.6**	**85.1**
Semantic: ours	**87.5**	**88.2**	**87.6**	**88.0**	—	—	—	—

overcome this issue, as in Andreas et al. [2013], would be to simply filter the *n*-best translation until a well-formed one is found. However, we see very limited performance changes in accuracy and F1, suggesting that the effect of using the *n*-best translation is very limited. For example, after using the *n*-best translation, the nonenriched SCFG system obtains 82.0 in accuracy (increased from 81.5) and 84.5 in F1 (reduced from 85.2).

Although any system of our three enriched SCFGs outperforms the nonenriched SCFG system, it is interesting to find that the three enriched systems perform asymmetrically, resulting in that we cannot draw a conclusion as to which one outperforms best across different languages. For instance, structure-syntax–enriched SCFG achieved the best performance on English, whereas structure-enriched SCFG and syntax-enriched SCFG yielded the best F1 scores on Chinese and Thai, respectively. Nevertheless, we stick to structure-syntax–enriched SCFG in the rest of this section, as it achieved the highest average performance.

7.5. Effect of Unknown Word Translation

Since each of our SMT models is actually trained on 540 instances (plus the NP list), the rate of unknown words in the test data tends to be higher than that in a system trained with the whole 600 instances. Based on the system of enriched SCFG with *all* alignment, Table VII shows the results of applying unknown word translation. It shows that translating all unknown words into *null* obtains 1.9 points in accuracy over the system without it (e.g., 84.6 vs. 82.7).

To find semantically close words in English, we use the off-the-shelf word vectors trained on part of the Google News dataset (about 100 billion words) from the Word2Vec toolkit Web site. For Chinese, we obtain the word vectors by training the Word2Vec model on Chinese Gigaword with 10M sentences. Unfortunately, we are not able to find large monolingual data in Thai. However, the slight improvement or even decrease in F1 (e.g., 87.6 vs. 86.9 in English and 87.3 vs. 87.8 in Chinese) suggests that there are many scenarios that translating unknown words into *null* is incorrect. Fortunately, our semantic approach is partially able to generate correct translation rules for those unknown words that have translation in MRL′. Actually, the effect of our approach is highly dependent on the quality of the semantically close words found via Word2Vec. With a manual examination on the English test data, we found that 11 out of all 17 unknown words should be translated into a corresponding token in MRL. For 8 of them, the synthetic translation rule set returned by Algorithm 1 contains correct translation rules.

7.6. Effect of Filtering Approach

To predict the answer type for a source input, we use SVMLight [Joachims 1999] as our classifier. To handle the multiclassification problem, we apply the one vs. others strategy, which builds K ($K = 6$ in our experiments since we have six answer types, i.e., *CITY*, *COUNTRY*, *NUM*, *PLACE*, *RIVER*, *STATE*) classifiers so as to separate one class from all others. We train our classifiers on the 600 sentences with default parameter settings. Table VIII presents the answer type prediction performance on the

Table VIII. Accuracy of Answer Type Prediction on Source Side of Test Data

	English	Chinese	Thai	Avg.
Acc.	96.8	97.9	93.9	96.2

Table IX. Performance with or without Filtering the *n*-Best Translation List

	English		Chinese		Thai		Avg.	
System	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Without filtering	87.5	88.2	87.6	88.0	80.1	80.4	85.1	85.5
With filtering	**88.3**	**88.9**	**88.8**	**89.2**	**84.0**	**84.1**	**87.0**	**87.4**

test data in accuracy (i.e., the proportion correctly predicted out of all test sentences). It shows that our classifiers with *n*-gram features achieve average performance of 96.2 in accuracy. For those sentences incorrectly predicted, we conjecture that features such as the first noun phrase from language-specific syntactic analysis will be helpful. On the target side, the accuracy of answer type prediction is 97.1.

For the *n*-best translations, we use a CKY parsing algorithm to get their corresponding parse tree.[13] Unlike syntactic parsing for NLs, the context-free grammar (CFG) here is relatively simple and less ambiguous. For most translations, they have none or only one parse tree within CFGs extracted from the training data (i.e., the 600 sentences). Due to the one-to-one mapping between a translation and its potential parse tree (if it exists), we can simply filter the *n*-best translations to select the most appropriate translation as output rather than developing a more complex reranking algorithm. Table IX compares the semantic parsing performance with or without filtering the *n*-best translation list. It shows that filtering the *n*-best list obtains improvement of 2.1 in accuracy and 2.0 in F1 on average. Compared to the moderate improvement on English and Chinese, it is interesting to point out that the filtering approach achieves more significant improvement on Thai, even though the accuracy of answer type prediction on the source side of Thai is lower than that of any the other two. Analyzing the final output reveals that the filtering approach successfully helps 15 (4 and 7) sentences to be correctly parsed yet causes 4 (2 and 2) sentences to be incorrectly parsed for Thai (likewise for English and Chinese).

7.7. Comparison with Other Systems

Table X shows the evaluation results of our system as well as those of several other comparable related works that share the same experimental setup as ours.[14] We can observe from Table X that our final best semantic parser with SMT components greatly outperforms the state of the art across various languages, including English, Chinese, Thai, German, and Greek. In particular, compared to our previous work in Li et al. [2015], the improvement comes from upgrading SCFG from structure enriched to structure-syntax enriched, and the approach of filtering the *n*-best list.

Although by far we report the performance of extrinsic evaluation by comparing the answers retrieved by the predicted MRL to those retrieved by the gold MRL, the last row of Table X shows the performance of intrinsic evaluation by comparing the predicted MRL directly to the gold MRL (i.e., the predicted MRL is correct only if it is exactly the same as the gold MRL). Except on English, the performance on other languages substantially decreases when moving from extrinsic evaluation to intrinsic evaluation. A closer examination reveals that most of the decrease is caused by the

[13]We set *n* as 100. However, the SMT decoder returns 6 (13 and 10) translations per sentence on average on English (likewise on Chinese and Thai).
[14]Due to recent availability of annotation on Chinese, previous work did not report performance on Chinese.

Table X. Performance Comparison for the Multilingual GeoQuery Test Set

System	English		German		Greek		Thai		Chinese	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
WASP	71.1	77.7	65.7	74.9	70.7	78.6	71.4	75.0	—	—
SMT-SemParse	80.5	81.8	68.9	71.8	69.1	72.3	70.4	70.7	—	—
HYBRIDTREE+	76.8	81.0	62.1	68.5	69.3	74.6	73.6	76.7	—	—
tsVB	79.3	79.3	74.6	74.6	75.4	75.4	78.2	78.2	—	—
RHT	83.6	83.6	74.3	74.3	78.2	78.2	79.3	79.3	—	—
f-RHT	86.8	86.8	75.7	75.7	79.3	79.3	80.7	80.7	—	—
UBL	82.1	82.1	73.6	73.7	75.0	75.0	66.4	66.4	—	—
Our work in Li et al. [2015]	86.3	87.1	79.1	80.3	80.5	81.6	76.3	77.9	85.8	87.8
This work	**88.3**	**88.9**	**80.8**	**81.1**	**83.9**	**84.2**	**84.0**	**84.1**	**88.8**	**89.2**
This work (intrinsic evaluation)	85.8	86.4	64.3	64.5	75.9	76.1	67.4	67.5	76.4	76.6

Note: The performance of WASP, HYBRIDTREE+, tsVB, and UBL is taken from Jones et al. [2012].

first two following types of mismatch, whereas the rest is mainly caused by the third type of mismatch (gold MRL ||| predicted MRL). Fortunately, these mismatches do not have an impact on the answer retrieved against the GeoQuery database. In the first type, for instance, segment *river(riverid('delaware'))* returns the same value as *riverid('delaware')*:

(1) answer(len(**river**(riverid('delaware')))) ||| answer(len(riverid('delaware')))
(2) answer(area_1(**state**(stateid('texas')))) ||| answer(area_1(stateid('texas')))
(3) answer(count(state(**next_to_1**(stateid('tennessee'))))) |||
answer(count(state(**next_to_2**(stateid('tennessee'))))).

7.8. Analysis

Contribution analysis. Starting from a baseline system with standard HPB SMT components (i.e., a nonenriched system with the *gdfa* alignment strategy and without unknown word translation nor filtering the *n*-best list), we incrementally augment the baseline system with our approach proposed from Section 4 through Section 6. Figure 6 summarizes the individual contributions coming from the *all* alignment strategy, structure-syntax–enriched SCFG, unknown word translation, and filtering the *n*-best list, respectively. It shows that major improvement of our final system comes from adopting the *all* alignment strategy, followed by improvement from structure-syntax–enriched SCFG, filtering the *n*-best list, and finally unknown word translation.

Error analysis. To find the errors in semantic parsing, we manually compare our translation output to the reference translation. Inspired by the error analysis used in machine translation [Vilar et al. 2006], we split the errors in semantic parsing into four classes: missing words, word order, incorrect words, and unknown words. A *missing words* error is produced when some words in the translation are missing. The missing words are essential for expressing the meaning of the sentence. The next category concerns the *word order* of the translation, for which we can generate a correct translation by moving individual words, independently of each other. The widest category of error in SMT is the *incorrect words* error, which occurs when the system is unable to find the correct translation of a given word or just finds a wrong translation. *Unknown words* are also sources of errors. Table XI lists a few examples on English for the four error types. Next, we use a set of translation output on English and Chinese to analyze in more detail the most prominent source of errors. To find errors in the translation process, we use our structure-syntax–enriched system with the *all* alignment strategy, skipping unknown words, but without filtering the *n*-best translation list. Figure 7 presents the complete error statistics on English and

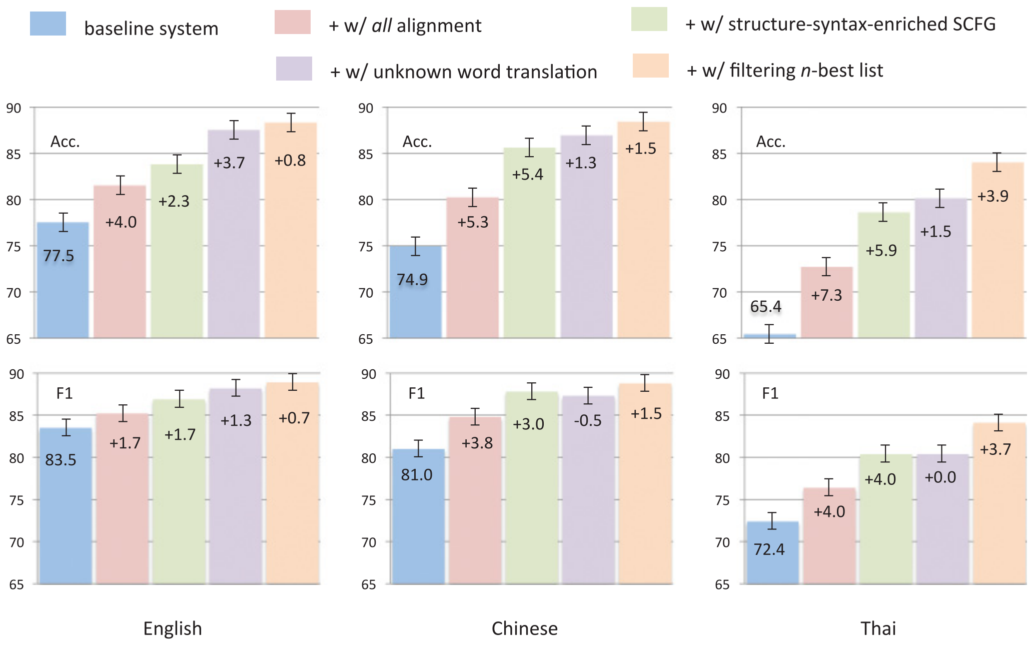


Fig. 6. Individual contribution of our approach over English, Chinese, and Thai.

Table XI. Examples of Error Translation on English. The Sentences of Input are Pre-Processed

Error Type	Example
Missing words	**Input**: how <u>*many*</u> capital do rhode island have **Output**: answer@1 capital@1 loc_2@1 stateid@1 rhode_island@s **Reference**: answer@1 *count@1* capital@1 loc_2@1 stateid@1 rhode_island@s **Description**: *count@1* is missing in the translation. This error is caused by the language model feature, which returns a relatively low score on bigram *count@1 capital@1* since it never occurs in the language model training data.
Word order	**Input**: what <u>*river run through*</u> the state with the <u>*most*</u> city **Output**: answer@1 *most@1 river@1 traverse_2@1* state@1 loc_1@1 city@1 all@0 **Reference**: answer@1 *river@1 traverse_2@1 most@1* state@1 loc_1@1 city@1 all@0 **Description**: <u>*most*</u> has a word order error with the phrase <u>*river run through*</u>.
Incorrect words	**Input**: how much <u>*population*</u> do texas have **Output**: answer@1 *high_point_1@1 loc_2@1* stateid@1 texas@s **Reference**: answer@1 *population_1@1* stateid@1 texas@s **Description**: <u>*population*</u> is mistakenly translated as *high_point_1@1* rather than *population_1@1*.
Unknown words	**Input**: how <u>*tall*</u> be the highest point in montana **Output**: answer@1 highest@1 place@1 loc_2@1 stateid@1 montana@s **Reference**: answer@1 *elevation_1@1* highest@1 place@1 loc_2@1 stateid@1 montana@s **Description**: <u>*tall*</u> is an unknown word.

Note: The <u>*underline*</u> indicates source words activating the corresponding error type, and the *underdash* and underwave indicate the automatic and reference translation, respectively.

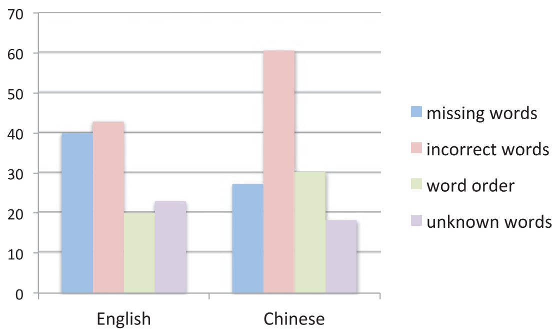


Fig. 7. Error distribution (%) on English and Chinese.

Chinese. Note that an incorrect translation may contain two or more error types, and thus there exist overlaps among the numbers in Figure 7.

Decoding time analysis. We analyze the effect on the decoding time of our approach, which is closely related to the size of phrase tables.[15] First, using triple alignments leads to higher coverage but larger phrase tables. This is illustrated by the increase in the average number of phrases per sentence (in English test data, hereafter) from 453 to 911, whereas the decoding time (on a machine with an i7-4770 CPU at 3.40GHz, hereafter) moves from 0.11 seconds to 0.14 seconds per sentence on average. Second, splitting nonterminal X into enriched ones also increases the size of phrase tables. This is not surprising, as a phrase with nonterminal X (e.g., the X on the source side) may be further specified as multiple phrases with various nonterminals (*the STATE*, *the STATE/A1*, etc., in our structure-syntax–enriched SCFG). As a result, the average number of phrases per sentence fiercely increases from 911 to 6,207, whereas the decoding time of the SMT decoder increases from 0.14 seconds to 0.31 seconds per sentence on average. Finally, finding similar words via Word2Vec, however, is quite fast, as this is bounded by the vocabulary size of our training set. Thanks to the small size of unknown words, adding unknown word translation rules has a very limited impact on the size of the phrase table and, consequently, negligible changes on the decoding time.

8. CONCLUSION AND FUTURE WORK

In this article, we have presented an enriched SCFG approach for semantic parsing that realizes the potential of the SMT approach. The performance improvement is derived from the extension of translation rules with informative symbols and increased coverage. Such an extension shares a similar spirit with the generalization of a CCG lexicon for a CCG-based semantic parser [Kwiatkowski et al. 2011; Wang et al. 2014]. Experiments on benchmark data have shown that our model obtained significant and sometimes substantial gains over a baseline system with standard SMT components. Compared to previous work, our approach greatly outperforms the state-of-the-art performance across various languages.

Recently, research on semantic parsing in the open domain with weakly (or un-) supervised setups [Cai and Yates 2013], under different settings where the goal was to optimize the performance of certain downstream NLP tasks such as answering questions, has received a significant amount of attention [Clarke et al. 2010; Berant et al. 2013; Berant and Liang 2014]. One direction of our future work is to extend the current framework to support the generation of synthetic translation rules from

[15]In cdec, we generate a phrase table for each sentence.

weaker signals (e.g., from question-answer pairs) rather than from aligned parallel data.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their insightful comments. We also thank Dr. Tong Xiao for his help in running the NiuTrans decoder.

REFERENCES

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association of Computational Linguistics*. 47–52.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics* 1, 49–62.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. 178–186.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1415–1425.

Peter E. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics* 19, 2, 263–313.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 423–433.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics* 33, 2, 201–228.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 224–233.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the worlds response. In *Proceedings of the 14th Conference on Computational Natural Language Learning*. 18–27.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*. 7–12.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: An open source toolkit for handling large scale language models. In *Proceedings of the 9th Annual Conference of the International Speech Communication Association (Interspeech'08)*. 1618–1621.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the Natural Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'04)*. 273–280.

Ruifang Ge and Raymond Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL'05)*. 9–16.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 1486–1495.

Zhanming Jie and Wei Lu. 2014. Multilingual semantic parsing: Parsing multiple languages into semantic representations. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers (COLING'14)*. 1291–1301.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola (Eds.). MIT Press, Cambridge, MA, 169–184.

Bevan Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with Bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 488–496.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. 913–920.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics on Interactive Poster and Demonstration Sessions (ACL'07)*. 177–180.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 1545–1556.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. 1223–1233.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. 1512–1523.

Junhui Li, Zhaopeng Tu, Guodong Zhou, and Josef van Genabith. 2012. Using syntactic head information in hierarchical phrase-based translation. In *Proceedings of the 7th Workshop on Statistical Machine Translation*. 232–242.

Junhui Li, Muhua Zhu, Wei Lu, and Guodong Zhou. 2015. Improving semantic parsing with enriched synchronous context-free grammar. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1455–1465.

Peng Li, Yang Liu, and Maosong Sun. 2013. An extended GHKM algorithm for inducing lambda-SCFG. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. 605–611.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 590–599.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. 609–616.

Wei Lu. 2014. Semantic parsing with relaxed hybrid trees. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1308–1318.

Wei Lu. 2015. Constrained semantic forests for improved discriminative semantic parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 737–742.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 783–792.

Klaus Macherey, Franz Josef Och, and Hermann Ney. 2001. Natural language understanding using statistical machine translation. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EuroSpeech'01)*. 2205–2208.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* 29, 1, 19–51.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 311–318.

Kishore A. Papineni, Salim Roukos, and Todd Ward. 1997. Feature-based language understanding. In *Proceedings of the 5th European Conference on Speech Communication and Technology (EuroSpeech 1997).* 1435–1438.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*. 1–10.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning* 62, 1–2, 107–136.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL'08)*. 577–585.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Zhaopeng Tu, Yang Liu, Yifan He, Josef van Genabith, Qun Liu, and Shouxun Lin. 2012. Combining multiple alignments to improve machine translation. In *Proceedings of the 24th International Conference on Computational Linguistics: Posters (COLING'12)*. 1249–1260.

David Vilar, Jia Xu, Luis Fernando D'Haro, and Hermann Ney. 2006. Error analysis of statistical machine translation output. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*. 697–702.

Adrienne Wang, Tom Kwiatkowski, and Luke Zettlemoyer. 2014. Morpho-syntactic lexical generalization for CCG semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1284–1295.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*. 439–446.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. 960–967.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. NiuTrans: An open source toolkit for phrase-based and syntax-based machine translation. In *Proceedings of the Association for Computational Linguistics 2012 System Demonstrations (ACL'12)*. 19–24.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. 523–530.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI'05)*. 658–666.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*. 138–141.

Received February 2016; revised May 2016; accepted June 2016