Semantic Parsing of Ambiguous Input through Paraphrasing and Verification

Philip Arthur, Graham Neubig, Sakriani Sakti, Tomoki Toda, Satoshi Nakamura
Graduate School of Information Science, Nara Institute of Science and Technology, Japan
{philip.arthur.om0, neubig, ssakti, tomoki, s-nakamura}@is.naist.jp

Abstract

We propose a new method for semantic parsing of ambiguous and ungrammatical input, such as search queries. We do so by building on an existing semantic parsing framework that uses synchronous context free grammars (SCFG) to jointly model the input sentence and output meaning representation. We generalize this SCFG framework to allow not one, but multiple outputs. Using this formalism, we construct a grammar that takes an ambiguous input string and jointly maps it into both a meaning representation and a natural language paraphrase that is less ambiguous than the original input. This paraphrase can be used to disambiguate the meaning representation via verification using a language model that calculates the probability of each paraphrase.[1]

1 Introduction

Semantic parsing (SP) is the problem of parsing a given natural language (NL) sentence into a meaning representation (MR) conducive to further processing by applications. One of the major challenges in SP stems from the fact that NL is rife with ambiguities. For example, even the simple sentence "Where can we eat a steak in Kobe?" contains *syntactic ambiguities* ("eat in Kobe" or "steak in Kobe"?), *quantifier scope ambiguities* (do we all eat one steak, or each eat one steak?), and *word sense ambiguities* (is Kobe a city in Japan; or an NBA basketball player?). Previous works using statistical models along with formalisms such as combinatorial categorial grammars, synchronous context free grammars, and dependency based compositional semantics have shown notable success in resolving these ambiguities (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Liang et al., 2011; Kwiatkowski et al., 2013).

Much previous work on SP has focused on the case of answering natural language queries to a database of facts, where the queries generally take the form of full sentences such as "What is the height of Kobe Bryant?" While answering these questions provides an excellent first step to natural language information access, in many cases the input is not a full sentence, but something more underspecified and ungrammatical. For example, this is the case for keyword-based search queries (Sajjad et al., 2012) or short dialogue utterances (Zettlemoyer and Collins, 2007).

Specifically taking the example of search queries, users tend to omit some of the function words and grammatical constructs in the language to make a more concise query. The first column of Table 1 illustrates several search queries of the pattern "Kobe X" where X is another word. From these queries and their MRs in column two, we can see that there are several kinds of ambiguity, including not only the distinction between Kobe as city or a basketball player as in the previous example, but also more pernicious problems unique to the more ambiguous input. Focusing on the queries "Kobe hotels" and "Kobe flight" we can see that it is also necessary to estimate the latent relationship between

[1]Tools to replicate our experiments can be found at http://isw3.naist.jp/~philip-a/tacl2015/index.html.

Search Query	**Meaning Representation**	**Paraphrase**
Kobe Hotel	$\lambda x\,(\text{hotel}(x) \wedge \text{in}(x, \text{kobe\_city}))$	Hotel in Kobe city
Kobe Flight	$\lambda x\,(\text{flight}(x) \wedge \text{to}(x, \text{kobe\_city}))$	Flight to Kobe city
Kobe Height	height(kobe_bryant)	Height of Kobe Bryant

Table 1: Example of a search query, MR, and its paraphrase

words, such as "location" or "destination." However it should be noted that if we take the keyword query and re-express it as a more explicit paraphrase, we can reduce this ambiguity to the point where there is only one reasonable interpretation. For example, in the second line, if we add the preposition "to" the user is likely asking for flights that arriving in Kobe, and if we add "from" the user is asking for departures.

In this paper, we focus on SP of ambiguous input and propose a new method for dealing with the problem of ambiguity. Here we propose a framework where an ambiguous input (Column 1 in Table 1) is simultaneously transformed into both its MR (Column 2) and a more explicit, less ambiguous paraphrase (Column 3). The advantage of this method is that it is then possible to verify that the paraphrase indeed expresses the intended meaning of the underspecified input. This verification can be done either manually by the system user or automatically using a probabilistic model trained to judge the naturalness of the paraphrases.

As a concrete approach, building upon the formalism of synchronous context free grammars (SCFG). Unlike traditional SCFGs, which usually only generate one target string (in semantic parsing, an MR), we introduce a new variety of SCFGs that generate multiple strings on the target side. This allows us to not only generate the MR, but also jointly generate the more explicit paraphrase. We then use a language model over the paraphrases generated by each derivation to help determine which derivations, and consequently which MRs, are more likely.

We perform an evaluation using the standard Geoquery benchmark of 880 query-logic pairs. First we note that baseline SCFG parser achieves reasonable accuracy on regular questions but when the same method is used with underspecified input, the system accuracy decreases significantly. On the other hand, when incorporating the proposed tri-synchronous grammar to generate paraphrases and verify them with a language model, we find that it is possible to recover the loss of accuracy, resulting in a model that is able to parse the ambiguous input with significantly better accuracy.

2 Semantic Parsing using Context Free Grammars

As a baseline SP formalism, we follow Wong and Mooney (2006) in casting SP as a problem of translation from a natural language query into its MR. This translation is done using synchronous context free grammars, which we describe in detail in the following sections.

2.1 Synchronous Context Free Grammars

Synchronous context free grammars are a generalization of context-free grammars (CFGs) that generate pairs of related strings instead of single strings. Slightly modifying the notation of Chiang (2007), we can formalize SCFG rules as:

$$X \rightarrow \langle \gamma_s, \gamma_t \rangle \tag{1}$$

where X is a non-terminal and γ_s and γ_t are strings of terminals and indexed non-terminals on the source and target side of the grammar. SCFGs have recently come into favor as a tool for statistical machine translation (SMT). In SMT, a synchronous rule could, for example, take the form of:

$$X \rightarrow \langle X_0 \text{ eats } X_1,\ X_0 \text{ wa } X_1 \text{ wo taberu} \rangle \tag{2}$$

where γ_s is an English string and γ_t is a Japanese string. Each non-terminal on the right side is indexed, with non-terminals with identical indices corresponding to each-other.

Given the SCFG grammar, we can additionally assign a score to each rule, where higher scored rules are more likely to participate in a derivation. Given the grammar of scored rules, and an input sentence

Grammar

r_0 QUERY $\rightarrow$ $\langle$give me the $\mathbf{CONJ_0}$, answer(x_1, $\mathbf{CONJ_0}$)$\rangle$
r_1 CONJ $\rightarrow$ $\langle\mathbf{FORM_0}$ $\mathbf{FORM_1}$ $\mathbf{STATE_2}$, ($\mathbf{FORM_0}$, $\mathbf{FORM_1}$, const(x_2, stateid($\mathbf{STATE_2}$)))$\rangle$
r_2 FORM $\rightarrow$ $\langle$cities, city(x_1)$\rangle$
r_3 FORM $\rightarrow$ $\langle$in, loc(x_1, x_2)$\rangle$
r_4 STATE $\rightarrow$ $\langle$virginia, virginia$\rangle$

Derivations

$\langle\mathbf{QUERY_0}$, $\mathbf{QUERY_0}\rangle$
$r_0 \Rightarrow$ $\langle$give me the $\mathbf{CONJ_1}$, answer(x_1, $\mathbf{CONJ_1}$))$\rangle$
$r_1 \Rightarrow$ $\langle$give me the $\mathbf{FORM_2}$ $\mathbf{FORM_3}$ $\mathbf{STATE_4}$,
answer(x_1, ($\mathbf{FORM_2}$, $\mathbf{FORM_3}$, const(x_2, stateid($\mathbf{STATE_4}$)))))$\rangle$
$r_2 \Rightarrow$ $\langle$give me the cities $\mathbf{FORM_3}$ $\mathbf{STATE_4}$,
answer(x_1, (city(x_1), $\mathbf{FORM_3}$, const(x_2, stateid($\mathbf{STATE_4}$))))$\rangle$
$r_3 \Rightarrow$ $\langle$give me the cities in $\mathbf{STATE_4}$,
answer(x_1, (city(x_1), loc(x_1,x_2), const(x_2, stateid($\mathbf{STATE_4}$))))$\rangle$
$r_4 \Rightarrow$ $\langle$give me the cities in virginia, answer(x_1, (city(x_1), loc(x_1, x_2), const(x_2, stateid(virginia))))$\rangle$

Figure 1: Example of SP using SCFGs. The left hand and right hand sides are generated simultaneously.

$\mathcal{S}$, the highest scoring parse and output sentence $\mathcal{T}$ can be calculated using the CKY+ algorithm (Chiang, 2007).

2.2 Semantic Parsing with SCFGs

In the simplest form of SP with SCFGs, γ_s is used to construct a natural language string $\mathcal{S}$ and γ_t is used to construct the MR $\mathcal{T}$ (Wong and Mooney, 2006). Figure 1 shows an example of using an SCFG to simultaneously generate a natural language string and its MR. In this picture, the **bold symbols** are non-terminals which can be substituted with other non-terminal productions. Productions end when all the tokens are terminals. The collection of rules used to generate a particular $\langle\mathcal{S},\mathcal{T}\rangle$ pair is a derivation $\mathcal{D}= d_1, d_2, ..., d_{|\mathcal{D}|}$.

Wong and Mooney (2007) further extended this formalism to handle λ-SCFGs, which treat γ_s as the natural language query and γ_t as an MR based on λ calculus. SCFG rules are automatically learned from pairs of sentences with input text and the corresponding MR, where the MR is expressed as a parse tree whose internal nodes are predicates, operators, or quantifiers.

In this paper, we follow Li et al. (2013)'s approach to extract a grammar from this parallel data. In this approach, for each pair, statistical word alignment aligns natural language tokens with the corresponding elements in the MR, then according to the alignment, minimal rules are extracted with the GHKM algorithm (Galley et al., 2004; Li et al., 2013). Then, up to k minimal rules are composed to form longer rules (Galley et al., 2006), while considering the relationship between logical variables. Finally, unaligned NL tokens are aligned by attaching them to the highest node in the tree that does not break the consistencies of alignment, as specified in Galley et al. (2006).

2.3 Additional Rules

While basic rules extracted above are quite effective in parsing the training data,[2] we found several problems when we attempt to parse unseen queries. To make our parser more robust, we add two additional varieties of rules. First, we add a deletion rule which allows us to delete any arbitrary word w with any head symbol X, formally:

$$X \rightarrow \langle w\ X, X\rangle. \tag{3}$$

This rule allows our grammar an option of ignoring words that it does not know what to do with.

[2]We achieve almost 100% F-measure in closed testing.

In addition, to ensure that all of the facts in the database can be accessed by our semantic parser, we provide some additional SCFG rules based on the given database of facts. The Geoquery dataset provides a database of facts represented as logical assertions. For every assertion provided in the database, we produce a single rule using the function name as the label of the non-terminal and one parameter of the assertion as the terminal, depending on the assertion's type. For example, Geoquery provides some details about the state of Michigan with the form `state('michigan',...)`, and thus we add

$$\text{STATE} \rightarrow \langle \text{michigan, michigan} \rangle$$

as an additional rule in the grammar.

3 Semantic Parsing of Keyword Queries

As explained in Section 1, when users input keyword queries, they will often ignore the grammatical structure and omit function words. Based on this, a traditional SP model can be problematic. To give a concrete example, consider the synchronous parse in Figure 1. If we try to parse with only the keywords (e.g. "cities virginia") with a standard grammar, the parser will not be able to recover the latent relationship "loc(x_1, x_2)" between the two words. Unfortunately, we are lacking evidence to recover this relationship, because the token "in" associated with the predicate "loc" will often not occur in a keyword query.

In this work, we perform experiments on this particular variety of ambiguous input, both to examine the effect that it has on parsing accuracy under the baseline model, and to examine whether this sort of ambiguity can be reduced. In order to do so, we need examples of keyword queries. In this work, we simulate the keyword query $\mathcal{K}$ by altering the original question $\mathcal{S}$ to make it more closely match the style of keyword queries. In particular, following the analysis of Leveling (2010), we make two changes to the original queries: stop word deletion, and word order shuffling.

Stop word deletion, as its name implies, simply deletes all stop words from the input sentence. We use a stop word list,[3] making a few subjective changes to make the simulated keyword output more realistic. Specifically, we add "give" and "show," which often occur in statements such as "give me ..." or "show me ..." but are unnatural in keyword queries. We also exclude from the list "us," which often refers to "United States," and function words such as "many," "most," and "much."

[3]http://www.lextek.com/manuals/onix/stopwords2.html

Word order shuffling permutes the order of the keywords remaining after stop word deletion, to simulate the fact that keyword queries often don't have strict order. First we shuffled the tokens randomly, then had a human annotator fix the order of the keywords manually, making the minimal number of changes necessary to ensure that the queries are natural and fluent. This produced a single keyword query $\mathcal{K}$ for a particular question/MR pair in the Geoquery database, which will be used to train and verify our system. At the end we will have a 3-parallel corpus consisting of 880 pairs of keyword, question, and the meaning representation.

We should note that while shortening and reordering are prominent features of search queries (Leveling, 2010), these are not the only phenomenon distinguishing queries from standard text. For example, humans tend to also change content words into an equivalent and easier word of their preference (Gurský et al., 2009). While collecting this data is out of the scope of the present work, if a corpus of real keyword inputs and question paraphrases were available, it is theoretically possible for our proposed method to learn from this data as well.

4 Joint Semantic Parsing and Paraphrasing using Tri-Synchronous Grammars

In this section we describe our proposed method to parse underspecified and ungrammatical input while jointly generating a paraphrase that can be used to disambiguate the meaning of the original query.

4.1 Generalized Synchronous Context Free Grammars

Before defining the actual parsing framework, we first present a generalization of SCFGs, the n-synchronous context free grammar (n-SCFG) (Neubig et al., 2015). In an n-SCFG, the elementary structures are rewrite rules of $n-1$ target sides:

$$X \rightarrow \langle \gamma_1, \gamma_2, ..., \gamma_n \rangle \tag{4}$$

Grammar

r_0 QUERY $\rightarrow$ $\langle$**CONJ$_0$**, give me the **CONJ$_0$**, answer(x_1, **CONJ$_0$**)$\rangle$

r_1 CONJ $\rightarrow$ $\langle$**FORM$_0$ STATE$_1$**, **FORM$_0$** in **STATE$_1$**,
(**FORM$_0$**, loc(x_1, x_2), const(x_2, stateid(**STATE$_1$**)))$\rangle$

r_2 FORM $\rightarrow$ $\langle$cities, cities, city(x_1)$\rangle$

r_3 STATE $\rightarrow$ $\langle$virginia, virginia, virginia$\rangle$

Derivations

$\langle$**QUERY$_0$**, **QUERY$_0$**, **QUERY$_0$** $\rangle$

$r_0 \Rightarrow$ $\langle$**CONJ$_0$**, give me the **CONJ$_0$**, answer(x_1, **CONJ$_0$**)$\rangle$

$r_1 \Rightarrow$ $\langle$**FORM$_2$ STATE$_3$**, give me the **FORM$_2$** in **STATE$_3$**,
answer(x_1, (**FORM$_2$**, loc(x_1, x_2), const(x_2, stateid(**STATE$_3$**))) $\rangle$

$r_2 \Rightarrow$ $\langle$cities **STATE$_3$**, give me the cities in **STATE$_3$**,
answer(x_1, (city(x_1), loc(x_1, x_2), const(x_2, stateid(**STATE$_3$**)))$\rangle$

$r_3 \Rightarrow$ $\langle$cities virginia, give me the cities in virginia,
answer(x_1, (city(x_1), loc(x_1, x_2), const(x_2, stateid(virginia)))$\rangle$

Figure 2: An example of 3-SCFG rules and productions. Here there are 2 target sides, one is the paraphrase and the other is the MR

where X is a non-terminal symbol, γ_1 is the source side string of terminal and non-terminal symbols, and $\gamma_2, ...\gamma_n$ are the target side strings. Therefore, at each derivation step, one non-terminal in γ_1 is chosen and all the corresponding non-terminals with the same index in $\{\gamma_2, ..., \gamma_n\}$ are rewritten using a single rule.

4.2 Tri-Synchronous Grammars for Joint Parsing and Paraphrasing

Based on this framework, we propose a model for joint semantic parsing and paraphrasing using tri-synchronous grammars, or 3-SCFGs. In this framework, input γ_1 corresponds to a keyword query $\mathcal{K}$, and the outputs γ_2 and γ_3 correspond to the paraphrase and MR respectively. An example of jointly generating a keyword query, question, and MR with a 3-SCFG is shown in Figure 2.

In this work, we construct the tri-synchronous grammar by transforming the basic SCFG for semantic parsing $\mathcal{G}$ into a 3-SCFG. Specifically, we first assume that the source question γ_s and target MR γ_t of the original SCFG become the two outputs γ_2 and γ_3 of the new 3-SCFG grammar. γ_1 is the newly added keyword query input.

During the process of model training, we first extract rules consisting of γ_2 and γ_3 using the algorithm in Section 2.2, then generate γ_1 from γ_2 by first deleting the stop-words then rearranging the order of the words based on word alignments between the keyword query and the original question. This is done by assigning each word in $\mathcal{K}$ a range of words in $\mathcal{S}$ to which it is aligned, then sorting words in γ_1 in ascending order of these ranges. It is possible to have cases in which there are some words in $\mathcal{K}$ that have no alignment in $\mathcal{S}$, and these rules are filtered out. Finally, we use the tuple $\langle\gamma_1, \gamma_2, \gamma_3\rangle$ to form rules in our tri-synchronous grammar.

Because of the stop word deletion, we may find that some rules have an empty source side, and consequently cannot be used in an SCFG. For example, in r_3 in Figure 1, “in” is in the stop word list, and thus will be deleted from the source side, leaving it empty. In order to solve this problem, we compose all rules with empty inputs together with their parent rule. It should be noted that this introduces a large amount of ambiguity into the grammar, as the content represented by the deleted content word must now be generated essentially out of thin air, based only on its parent context.

4.3 Integrating Language Models with Tri-SCFGs

When using SCFGs for machine translation, the power of language models (LM) to improve the translation accuracy is widely acknowledged. The LM ensures fluent SMT output by assigning a probability to the target sentence. In case of n-gram language models, this probability is defined as:

$$p_{LM}(W) = \prod_{i=1}^{l} p(w_i|w_{i-1}, w_{i-2}, ...w_{i-n+1}) \quad (5)$$

where the probabilities of sentence W of length l is calculated as the product of the probability of its words, depending on the previous $n-1$ words. Integrating these language models makes the search space larger, precluding the use of the full CKY-style parsing algorithm, but efficient approximate search algorithms such as cube pruning (Chiang, 2007) or incremental search (Heafield et al., 2013) can help ameliorate this problem.

We could also consider constructing a probabilistic LM over MR $\mathcal{T}$ for semantic parsing. However, constructing a language model for the MR is less straightforward for several reasons. First, the order of the words of MR in the same rooted logical tree will not make a difference in the final result (e.g. for a commutative operator node). Second, while language models for natural text benefit from the large amounts of text data available on the web, obtaining correct MRs to train a model is less trivial.

On the other hand, in our tri-synchronous grammar framework, in addition to the MR itself, we are generating a paraphrase that nonetheless holds some disambiguating power over the MR, as described in Section 1. The naturalness of this paraphrase output, like the output of the MT system, can easily be judged by a language model, and might have some correlation with the naturalness of the MR itself. Thus, in this work we add a language model over the paraphrase output as a feature of the scoring model described in the next section.

5 Parse Scoring

Given this SCFG-based parsing model, we must now assign a score to decide which scores are better or worse than others.

5.1 Scoring Function

Our scoring function is a standard log linear model with feature functions defined over $\langle \mathcal{K}, \mathcal{S}, \mathcal{T}, \mathcal{D} \rangle$ tuples:

$$\text{score}(\mathcal{K}, \mathcal{S}, \mathcal{T}, \mathcal{D}) = \boldsymbol{w} \cdot \Phi(\mathcal{K}, \mathcal{S}, \mathcal{T}, \mathcal{D}) \quad (6)$$

where $\Phi(\mathcal{K}, \mathcal{S}, \mathcal{T}, \mathcal{D})$ is a vector of feature functions and $\boldsymbol{w}$ is the weight vector.

5.2 Features

For the baseline model, our feature vector $\Phi(\mathcal{K}, \mathcal{S}, \mathcal{T}, \mathcal{D})$ is simply defined as the element-wise sum of the feature vectors for each rule in the derivation:

$$\Phi\left(\mathcal{K}, \mathcal{S}, \mathcal{T}, \mathcal{D}\right) = \sum_{d \in \mathcal{D}} \Phi\left(d\right) \quad (7)$$

where d takes the form in Equation (4).

We score each basic rule using features widely used in translation as follows:

- Forward Probability: The log probability of source side given all the target sides $p(\gamma_1|\gamma_2, ..., \gamma_n)$, calculated based on rule counts in the training corpus $c(\gamma_1, ..., \gamma n)/c(\gamma_2, ..., \gamma_n)$.
- Backward Probability: Similarly, the log probability of all target sides given the source side $p(\gamma_2, ..., \gamma_n|\gamma_1)$.
- Joint Probability: The log probability of the source and target $p(\gamma_1, \gamma_2, ..., \gamma_n)$.
- Terminal Rule: Equal to one if there is no non-terminal symbol in the rule. This feature is useful to decide whether the model prefers entirely lexicalized rules.
- Deletion: Binary feature for deletion rules.
- Knowledge Base Rule: Binary feature for rules produced from the knowledge base.

For the proposed tri-synchronous grammar with LM verification, we additionally add three features defined over the generated paraphrase.

- Language Model: Counts the log language model probability of the paraphrase.

- Unknown: Counts the number of tokens in the paraphrase that are unknown in the language model.

- Paraphrase Length: Counts the number of words in the paraphrase, and can be calculated for each rule as the number of terminals in the paraphrase. This feature helps compensate for the fact that language models prefer shorter sentences.

5.3 Learning Feature Weights

Now that we have defined the feature space, we need to optimize the weights. For this we use minimum error rate training (MERT) (Och and Ney, 2003), maximizing the number of correct answers over the entire corpus.[4]

6 Experiment and Analysis

We evaluate our system using the Geoquery corpus (Zelle and Mooney, 1996), which contains 880 sentences representing natural language questions about U.S. Geography, and their corresponding MRs.

6.1 Setup

- **Data:** We use the full Geoquery dataset using the same 10 folds of 792 and 88 test data used by Wong and Mooney (2007). We created keyword queries according to the process described in Section 3. We follow standard procedure of removing punctuation for all natural language text, regardless of whether it is a keyword or full question. We also perform stemming on all natural language text, both in the keyword and question queries.

- **Rule Extraction:** Alignment is performed by `pialign` (Neubig et al., 2011) with the setting forcing one-to-many alignments. The algorithm to extract the tri-synchronous grammar is as discussed in Section 4.2 and maximum size of the rules for composition is 4.

- **Decoding:** To query the database, we use `prolog` queries fired against the Geoquery database. The parsing problem can thus be considered the task of decoding from underspecified natural language queries into `prolog` queries. This is done by performing decoding of the SCFG-based parsing model to translate the input query into an MR including λ calculus expressions, performing β-reduction to remove the λ function, then firing the query against the database. Before querying the database, we also apply Wong and Mooney (2007)'s type-checking to ensure that all MRs are logically valid. For parsing, we implemented CKY-based parsing of tri-synchronous grammars on top of the `Travatar` (Neubig, 2013) decoder. Unless otherwise specified, the default settings of the decoder are used.

- **Language Model:** For all 3-SCFG systems we use a 4-gram Kneser-Ney smoothed language model trained using the `KenLM` toolkit (Heafield, 2011). Standard preprocessing such as lowercasing and tokenization is performed before training the models. As it is of interest whether or not the type of data used to train the language model affects the resulting performance, we build language models on several types of data.

  First, we use a corpus of news data from the Workshop on Machine Translation evaluation data (Callison-Burch et al., 2011) (`News`). This data represents standard English text unrelated to questions. Second, we use a part of the question paraphrase data gathered by Fader et al. (2013) (`Questions`).[5] This data consists entirely of questions, and thus is a better representative of the latent questions behind the input queries. Finally, we used the full questions from `Geoquery` sentences to build the language model, building a different language model for each fold, completely separate from the test set. Table 2 gives the details of each dataset.

  In addition, because the `Geoquery` data is useful but small, for all 3-SCFG systems, we perform experiments using an additional 4-

[4]We also tried gradient-based optimization methods and large feature sets as in Wong and Mooney (2007) and Li et al. (2013), but the dense feature set and MERT achieved similar results with shorter training time.

[5]We use only data set 0 from the 30 sets released at http://knowitall.cs.washington.edu/afader/paraphrases.

Data	**Sent.**	**Tok.**	**LM Size**
News	44.0M	891M	5.5G
Questions	20.2M	174M	1.5G
Geoquery	792	$\sim$1.6K	$\sim$96K

Table 2: Details of the data used to build LMs

gram feed-forward neural network language model (NNLM) (Bengio et al., 2003) feature, which is possibly better equipped to handle sparse data than standard n-grams. The NNLM is built on `Geoquery` sentences, excluding the test sentences for each fold. This feature is not produced during parsing, but is separately scored and used to re-rank the n-best list generated by the parser.

Integration with the paraphrase language model is performed using incremental search (Heafield et al., 2013). For the parsing with NNLM, we recalculate the score of the paraphrases by firstly adding the NNLM score as one of the feature in Equation 6 and taking the parse with the best score.

- **Parameter Optimization:** For learning the parameters of the scoring function we use 10-fold cross validation on the training data, i.e. each fold iteration uses model trained on 712 examples and to parse the remaining 79. First we run decoding for all folds and gather the results. Then we run MERT with the combined results to update the parameters. We use the standard evaluation measure of question answering accuracy as our objective function and set the n-best list to be the top 300 derivations.

  To learn the weights for rescoring with the NNLM, we first generate an n-best list with the base model not using the NNLM feature. We then calculate the NNLM feature for each hypothesis in the n-best list, and run one more run of MERT with this feature to obtain the weights used in the rescoring model.

- **Evaluation:** Following the definition from Zettlemoyer and Collins (2005) and Wong and Mooney (2007), we use question answering accuracy as our evaluation measure. We define *recall* as the fraction of correct answers divided by the number of test data, *precision* as the fraction of correct answers divided by the number of parsed queries and F-measure as the harmonic mean of the two. The query is judged correct if and only if the SCFG can generate a valid parse tree, and the resulting query does not produce any syntax errors when accessing the database through a `prolog` query. Note that all 880 questions are used for testing through cross validation, so a recall improvement of 0.001 is approximately equal to answering one more question correctly.

6.2 Parsing Accuracy Results

Input	Method	P	R	F
Question	Direct	.880	.878	.879
Keyword	Direct	.792	.790	.791
	Tri-LM	.804	.790	.797
	Tri+LM	**.830**	**.820**	**.827**

Table 3: Parsing accuracy, where Keyword Direct is the baseline for semantic parsing on keyword queries, and the Tri with the LM for verification is our proposed method. Bold indicates a significant gain over both Direct and Tri-LM for keyword input according to bootstrap resampling (Koehn, 2004) ($p < 0.05$).

First, in this section, we examine the effect of the proposed method on accuracy of parsing ambiguous keyword queries. Specifically, in Table 3 we show the baseline "Direct" method of training a standard SCFG-based semantic parser, the proposed method without language model verification "Tri-LM," and the proposed method using the `Questions` language model with NNLM reranking "Tri+LM."

Looking at the baseline accuracy over full questions (first row), we can see the recall is slightly superior to Wong and Mooney (2007)'s 86.6% and Li et al. (2013)'s 87.6%, demonstrating our baseline is comparable to previous work. When we apply the same method to parse the keyword queries (second row), however, the recall drops almost 9%, showing that the ambiguity included in the keyword query input causes large decreases in accuracy of a semantic parser built according to the baseline method. This ambiguity is also reflected in the number of MRs generatable by the parser for any particular input. In the top 300 list generated by each parser, there were

Ex.	LM	Paraphrase/MR	Correct
1	Direct	answer(A,(capital(A),loc(A,B),largest(C,population(B,C))))	no
	Tri-LM Tri+LM	answer(A,largest(B,(capital(A),population(A,B)))) what capital has the largest population	yes
	Original Question: what capital has the largest population		
	Original MR: answer(A,largest(B,(capital(A),population(A,B))))		
	Keyword: largest population capital		
2	Direct(-)	answer(A,largest(A,(capital(A),city(A),loc(A,B),state(B))))	no
	Tri-LM	answer(A,largest(A,(state(A),loc(A,B),capital(B)))) what is the largest state in capital	no
	Tri+LM	answer(A,(state(A),loc(B,A),largest(B,capital(B)))) what state has the largest capital	yes
	Original Question: what state has the largest capital		
	Original MR: answer(A,(state(A),loc(B,A),largest(B,capital(B))))		
	Keyword: largest capital state		

Table 4: Examples of paraphrase outputs produced by the direct keyword-MR system, and the proposed systems without and with a language model.

a total of 16.54 and 36.77 unique MRs for question and keyword input respectively.

Now we take a look at the 3-SCFG (third row) without the LM verification, we can see the results are similar to the baseline. Then, when adding the language model to the 3-SCFG system (fourth row) we can see a significant of 3-4% gain over the Direct and the Tri-LM systems, demonstrating that the proposed method of paraphrasing and verification is indeed able to resolve some of the ambiguity in the keyword queries.

To illustrate how the language model helps, we provide two examples in Table 4. The first example shows that considering the original question when parsing from keywords can help improve alignment with the MR for more plausible results. The second example shows the effect of adding the language model to disambiguate the keyword query. Here there are several interpretations for the keyword-query "largest capital state," which also can mean "state that has the largest capital," or "largest state in the capital." The system without the language model incorrectly chooses the latter interpretation, but the system with language model correctly disambiguates the sentence as it considers the phrase "state in capital" is unlikely, showing the effectiveness of our method.

6.3 Analysis

We first examine the effect of choice of language model in the first two columns of Table 5. The first column is the full model with NNLM re-ranking, and the second column is without. The rows show the effect of using different data to train the n-gram LM. All the systems using LMs are basically better than the system using neither an n-gram LM nor the NNLM. Looking at the differences between the n-gram LMs, we can see that the `Questions` LM tends to be the most effective. This is particularly encouraging as the `Questions` language model does not contain any domain specific content, but is able to outperform the `Geoquery` domain specific LM. We also found that, as expected, the more sophisticated neural network language model raises the system accuracy by approximately 2%, which also supports our proposed idea that a better LM will better raise system accuracy.

The proposed method aims at reducing nonsensical interpretations, and another trivial baseline that can achieve a similar effect is to filter out the queries that produce empty answers, with the assumption that empty answers are generated from invalid queries. This simple filtering method reduced the number of unique queries to 11.74 for questions and 20.16 for keywords. However, as shown in the "-Empty" results in Table 5, we found that this fil-

Input	Output	LM	Full			-NNLM			-Empty		
			P	R	F	P	R	F	P	R	F
Keyword	Q+MR	-	**.828**	.813	**.821**	.804	.790	.797	.820	.784	.801
		News	**.823**	.814	.819	.806	.797	.802	.817	.786	.802
		Quest	**.830**†	**.820**†	**.827**†	.809	.804	.806	.806	.780	.793
		Geo	**.821**	.812	.817	.804	.794	.799	.824	.794	.808

Table 5: The result of experiment with/without neural network language model for the proposed 3-SCFG framework. Question-LM +NNLM achieved the best accuracy. Bold indicates a significant gain over the baseline Direct Keyword (second row of Table 3) and dagger indicates a significant gain over the 3-SCFG baseline without language model (-NNLM column, first row). The Full and -Empty column use NNLM as language model. The first row of the -NNLM column is the experiment without any language model.

tering method is not effective, causing the system's performance to drop by around 2%. This is caused by the fact that the correct answer is sometimes an empty answer, for example "what states border hawaii?"

6.4 Human Evaluation

While all evaluation up to this point has used language models to disambiguate paraphrases, we can assume that human users will be even better at judging whether or not a paraphrase makes sense. Thus, we perform an additional evaluation in which human annotators evaluate the paraphrases generated from the systems. First, we took the 1-best parse and 7 random parses from the Tri+LM and Tri-LM systems where both systems produced a non-empty n-best. Then we show both the keyword queries and all the paraphrases to human evaluators to annotate: i) a fluency score of 0, 1, or 2 where 0 is completely unnatural English, 1 indicates minor grammatical errors, and 2 indicates flawless English, ii) a letter starting from "A", "B", etc. for the paraphrase that matches their preferred interpretation of the search query.[6] If the input has multiple interpretations, then a different letter is assigned for each possible interpretation in the order that the annotator believes that the interpretation is the correct one, and only paraphrase paraphrase is chosen for each interpretation. If the human annotator does not find the paraphrase that matched his/her pboth features set.igned and annotation starts from "B." 3 annotators were asked to annotate 300 keyword queries and their paraphrases.

[6]Here the letters are just the indicators of ranking with a letter "A" means the most possible interpretation of search queries according to the users.

There are a total of 866 keyword queries (out of 880) that produced a non-empty n-best list in both systems, so we chose random duplications of 34 inputs to make the sum 900.

System	Precision
Tri-LM	.803
Tri+LM	.834
Tri+LM+Human	.846

Table 6: System precision with additional human help.

Table 6 shows the improvement of the system with human help. We take all the answers from the annotators that were annotated with "A" and replaced the answer of Tri+LM system. Overall, there were 35 questions that changed between the 1-best and human choices, with 23 improving and 12 degrading accuracy. This experiment suggests that it is possible to show the generated paraphrases to human users to improve the accuracy of the semantic parser.

System	Fluency	Ratio	Precision
Tri-LM	0	.163	.415
	1	.425	.819
	2	.411	.940
Tri+LM	0	.083	.367
	1	.372	.811
	2	.544	.918

Table 7: Fluency, Ratio, and Precision statistics for the one-best of both systems.

Now we look at the relationship between the fluency of the paraphrase and the accuracy of the semantic parsers in Table 7. The statistics are gathered

Letter	System	Count	Total	Precision
A	Tri-LM	547	721	.919
	Tri+LM	674		.902
B	Tri-LM	308	452	.772
	Tri+LM	340		.752
C	Tri-LM	57	94	.631
	Tri+LM	52		.557
D	Tri-LM	7	13	.428
	Tri+LM	7		.142

Table 8: A result for the letter accuracy from the human evaluation. Note that counts do not sum up to total because it is possible that both systems generate same paraphrases.

from the one best output for both systems. Tri+LM had a significantly larger percentage of fluent paraphrases with score "2" (54% v.s. 41%) compared to the system without the language model. Of the paraphrases that were assigned "2" score, 91% corresponded to correct MRs, indicating that the subjective fluency of the paraphrase is a good indicator of parsing accuracy.

Finally, Table 8 shows the relationship between the rank of the human interpretation and the accuracy of semantic parsing. Out of the 900 problems shown to the annotators, 721 of them were ranked "A." This experiment showed that the interpretation of the paraphrase judged as most likely by the annotators achieves a high precision, confirming our hypotheses that humans are able to use paraphrases to accurately judge whether the interpretation is likely to be correct or not.

6.5 Other Methods for Using Paraphrase Data

In addition to the method describe up until this point, there are several other ways to potentially incorporate paraphrasing into syntactic parsing of underspecified input. In this section we briefly outline two other (unsuccessful) attempts to do so: creation of a pipelined paraphrasing/semantic parsing system, and addition of features from a large paraphrase database.

First, regarding the pipelined system, we build the paraphrasing system using the parallel keyword-question data, with standard settings of hierarchical phrase-based translation (Chiang, 2007), and standard SMT features. We use the `Geoquery` n-gram model for the language model used during decoding and NNLM language model to finally rerank the n-best list. As a result of experiments, even though this system obtained a respectable BLEU score of 57.5, the parsing accuracies were much lower than the direct keyword-MR system at 64.8 F-measure. An analysis showed that, perhaps as expected, this was caused by cascading errors, with unnatural paraphrases also resulting in failed semantic parses.

In addition, we also attempted to use the external `Questions` data to calculate additional features to our Tri+LM system. We do this by first simulating the keyword version for each sentence in the `Questions` data by performing shuffling and stop-word deletion.[7]

Next we train a hierarchical phrase-based system on this data to create a paraphrasing model. Next we intersect this model with our existing model by matching the source side and the target side of the rules and if they match, taking the union of the features sets. Unfortunately, however, this setting also did not allow for a gain in accuracy, likely due to to the low recall (15%) of the matching between paraphrasing grammar and semantic parsing rules. This low recall stemmed from a number of factors including restrictions on the standard Hiero paraphrasing grammars (no more than 2 non-terminals, no consecutive non-terminals on the source side, and no rules without at least one terminal), as well as simple lack of coverage of the words in the paraphrase database. This result does indicate room for improvement by developing algorithms that extract paraphrases that are closely related to the semantic parsing rules, but also suggests potential difficulties in simply applying paraphrasing resources such as PPDB (Ganitkevitch et al., 2013).

7 Related Work

Interpretation of search queries is a major concern in the field of information retrieval as it can affect the choice of retrieved documents. Underspecified queries are commonly entered into search engines, leading to large result sets that are difficult for users

[7]Because the shuffling process is random we could conceivably generate and train with multiple shuffled versions, but because the `Questions` data is relatively large already, we only train the paraphrasing system with the single permutation of keywords generated by the shuffling.

to navigate (Sajjad et al., 2012). Studies have shown that there are several ways to deal with this problem, including query reformulation, which can fall in the categories of query expansion or query substitution (Shokouhi et al., 2014; Xue and Croft, 2013). Leveling (2010) proposed a paraphrasing method that tries to reconstruct original questions given keyword inputs in the IR context, but did not model this reformulation together with semantic parsing. In addition, Wang et al. (2013) showed that doing paraphrasing on the queries for web search is able to reduce the mismatch between queries and documents, resulting in a gain in search accuracy.

Using paraphrasing to resolve ambiguity is not new, as it was used to resolve ambiguity interactively with a user's input (McKeown, 1983). Ge and Mooney (2009) and Miller et al. (1994) have also used the guidance of natural language syntax for semantic parsing. However, the usage of natural language syntax in the semantic parsing on keyword queries are not trivial. For example, the approach using syntax tree of the input side from Ge and Mooney (2009) can not be directly applied to the keyword query as syntax parsing on keyword query itself is not a trivial problem.

There have also been a few methods proposed to combine paraphrasing with semantic parsing. Fader et al. (2013) proposed a method to map from full questions to more canonical forms of these questions, with the canonical NL questions being trivially convertible to an MR. Berant and Liang (2014) extract entities from a full-text question, map these entities into a set of candidate MRs, and generate canonical utterances accordingly. Then the canonical utterance that best paraphrases the input is chosen, thereby outputting the corresponding MR. Our approach is the similar but orthogonal to these works in that we focus on situations where the original user input is underspecified, and try to generate a natural language paraphrase that more explicitly states the user intention for disambiguation purposes. A second difference is that we do not use separate model to do paraphrasing, instead using the same model to do paraphrasing and semantic parsing synchronously. This has the advantage of being able to scale more easily to complicated and highly compositional questions such as the ones found in Geoquery.

In addition to being useful for semantic parsing, SCFGs have also been used for paraphrasing. A variety of research has used SCFG-based paraphrases for text-to-text generation tasks like sentence compression (Cohn and Lapata, 2009; Ganitkevitch et al., 2011), or expanding the set of reference translations for machine translation evaluation (Madnani et al., 2007). In this paper we have introduced a novel use of 3-way SCFGs that allows us to simultaneously do semantic parsing and text-to-text generation.

To our knowledge, this is the first method to parse an underspecified input by trying to reconstruct a more explicit paraphrase of the input and validate the naturalness of the paraphrase to disambiguate the meaning of the original input.

8 Conclusion and Future Work

In this paper we introduced a method for constructing a semantic parser for ambiguous input that paraphrases the ambiguous input into a more explicit form, and verifies the correctness using a language model. We do so through a generalization of synchronous context free grammars that allows for generation of multiple output strings at one time. An evaluation showed that our method is effective in helping compensate for the 9% loss of system accuracies due to the ambiguity of the keyword queries, providing a 3% improvement. Human evaluation also confirmed that manually evaluating the paraphrases generated by our framework can improve the accuracy of the semantic parser further.

There are a number of future directions for this study. First, we plan to scale the proposed method to open domain semantic parsing of search queries over extensive knowledge bases such as FreeBase (Bollacker, 2007). In addition, previous works have tackled semantic parsing directly from question and answer pairs (Liang et al., 2011; Poon and Domingos, 2009; Artzi and Zettlemoyer, 2011). The idea of learning from unannotated data is attractive, and incorporating this learning framework into our model is a promising direction for future work.

Acknowledgments

We thank Professor Yuji Matsumoto and Assistant Professor Kevin Duh for the discussions and insightful ideas for the paper. We also thank Professor Chris Callison-Burch and anonymous reviewers for their suggestions and discussions.

This project is supported by grants from the Ministry of Education, Culture, Sport, Science, and Technology of Japan and from the Microsoft CORE program.

References

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 421–432.

Yoshua Bengio, Ducharme Réjean, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1415–1425.

Kurt Bollacker. 2007. A platform for scalable, collaborative, structured information integration. In *Proceedings of the 22nd Association ofr Advancement of Artificial Intelligence*, pages 22–27.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Omar F Zaidan. 2011. Findings of the 2011 workshop on statistical machine translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 22–64.

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

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research (JAIR)*, 34:637–674.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1608–1618.

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

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 961–968.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1168–1179. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.

Ruifang Ge and Raymond J Mooney. 2009. Learning a compositional semantic parser using an existing syntactic parser. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 611–619.

Peter Gurský, Tomás Horváth, Peter Vojtás, Jozef Jirásek, Stanislav Krajci, Robert Novotny, Jana Pribolová, and Veronika Vaneková. 2009. User preference web search – experiments with a system connecting web and user. *Computing and Informatics*, 28(4):515–553.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2013. Grouping language model boundary words to speed k-best extraction from hypergraphs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 958–968.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 187–197.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 (EMNLP)*, pages 1545–1556.

Johannes Leveling. 2010. A comparative analysis: QA evaluation questions versus real-world queries. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC)*.

Peng Li, Yang Liu, and Maosong Sun. 2013. An extended GHKM algorithm for inducing lambda-SCFG. In *Proceedings of the 27th Association for Advancement of Artificial Intelligence*, pages 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 (ACL)*, pages 590–599.

Nitin Madnani, Necip Fazil Ayan, Philip Resnik, and Bonnie Dorr. 2007. Using paraphrases for parameter tuning in statistical machine translation. In *Proceedings of the Workshop on Statistical Machine Translation (WMT07)*, Prague, Czech Republic.

Kathleen R. McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10.

Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz. 1994. Hidden understanding models of natural language. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 25–32.

Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT-ACL)*, pages 632–641.

Graham Neubig, Philip Arthur, and Kevin Duh. 2015. Multi-target machine translation with multi-synchronous context-free grammars. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, USA, May.

Graham Neubig. 2013. Travatar: A forest-to-string machine translation engine based on tree transducers. In *ACL (Conference System Demonstrations)*, pages 91–96.

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

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

Hassan Sajjad, Patrick Pantel, and Michael Gamon. 2012. Underspecified query refinement via natural language question generation. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 2341–2356.

Milad Shokouhi, Rosie Jones, Umut Ozertem, Karthik Raghunathan, and Fernando Diaz. 2014. Mobile query reformulations. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1011–1014.

Chenguang Wang, Nan Duan, Ming Zhou, and Ming Zhang. 2013. Paraphrasing adaptation for web search ranking. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 41–46.

Yuk Wah Wong and Raymond J Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 439–446.

Yuk Wah Wong and Raymond J Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, number 1, pages 960–967.

Xiaobing Xue and W. Bruce Croft. 2013. Modeling reformulation using query distributions. *ACM Transaction on Information Systems*, (2):6:1–6:34.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 13th National Conference on Artificial Intelligence*, pages 1050–1055.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687.