Learning compositional semantics for open domain semantic parsing

Le, P.; Zuidema, W.

Publication date
2012
Document Version
Author accepted manuscript
Published in
24th International Conference on Computational Linguistics: proceedings of COLING 2012: technical papers: 8-15 December 2012, Mumbai, India

Learning Compositional Semantics for Open Domain Semantic Parsing

Phong LE and Willem ZUIDEMA
Institute for Logic, Language and Computation
University of Amsterdam, the Netherlands
{p.le,w.h.zuidema}@uva.nl

Abstract
This paper introduces a new approach to learning compositional semantics for open domain semantic parsing. Our approach is called Dependency-based Semantic Composition using Graphs (DeSCoG) and deviates from existing approaches in several ways. First, we remove the need of the lambda calculus by using a graph-based variant of Discourse Representation Structures to represent semantic building blocks and defining new combinatory operations for our graph structures. Second, we propose a probability model to approximate probability distributions over possible semantic compositions. And third, we use a variant of alignment algorithms from machine translation to learn a lexicon. On the Groningen Meaning Bank (a recently released, large-scale, domain-general, semantically annotated corpus; Basile et al. (2012)), where we preprocess sentences with an existing dependency parser, we achieve results significantly better than the baseline. On Geoquery we obtain performance comparable to semantic parsers that were developed specifically for that domain.

Title and Abstract in Vietnamese

Học Tổng hợp Ngữ nghĩa cho Phân tích Ngữ nghĩa Miền Mở

Bài báo này giới thiệu một phương pháp mới cho học tổng hợp ngữ nghĩa trong phân tích ngữ nghĩa miền mở. Phương pháp của chúng tôi được gọi là Dependency-based Semantic Composition using Graphs (DeSCoG) và khác biệt với các phương pháp có sẵn trên nhiều phương diện. Trước tiên, chúng tôi loại bỏ sự cần thiết của phép tính lambda bằng cách dùng một biến thể dựa trên đồ thị cho Discourse Representation Structure để miêu tả các khối ngữ nghĩa và định nghĩa các phép kết hợp cho những đồ thị này. Thứ hai, chúng tôi đề xuất một mô hình xác suất để xấp xỉ những phân bố xác suất trên các ngữ nghĩa tổng hợp có thể có. Và thứ ba, chúng tôi dùng một biến thể của các thuật toán gióng hàng từ dịch máy để học tập từ vựng. Thực nghiệm trên Groningen Meaning Bank (một ngữ liệu vừa được công bố, lớn, tổng quát, và đã được gán nhãn ngữ nghĩa; Basile et al. (2012)), với các câu được tiền xử lý bằng một bộ phân tích phụ thuộc có sẵn, chúng tôi đạt được kết quả tốt hơn rất nhiều so với phương pháp cơ sở. Đối với Geoquery, chúng tôi có được kết quả tương đương với các bộ phân tích ngữ nghĩa được phát triển cho chính lĩnh vực đó.

KEYWORDS: semantics, parsing, dependency structure, graph.

KEYWORDS IN VIETNAMESE: ngữ nghĩa, phân tích, cấu trúc phụ thuộc, đồ thị .

1 Introduction

Semantic parsing is the task of translating natural language sentences to formal meaning representations. The dominant approach for solving this task has been, since Montague (1970), to handcraft a semantic lexicon, using a logic to represent meanings and the lambda calculus to regulate meaning combination. Semantic parsing has until recently thus remained relatively immune to the wave of corpus-based, probabilistic approaches that have revolutionized neighboring fields, such as syntactic parsing, speech recognition and machine translation.

However, an increasingly popular line of work has emerged in the last few years on using probabilistic and corpus-based methods for semantics as well. Much of this work makes use of the semantically annotated Geoquery corpus, that provides representations of queries to a geographic information system in both a natural language and a meaning representation (MR) language. For instance, the query *Which states border Arizona?* is manually translated into `answer(A,(state(A),const(B,stateid(arizona)),next_to(A,B)))`. Although the corpus is small and the domain restricted, some interesting results have been obtained in trying to learn the mapping from natural to formal language. For instance, Zettlemoyer and Collins (2005); Kwiatkowski et al. (2010, 2011) approached the problem with structural learning, Wong (2007) with machine translation, Kate and Mooney (2006) with a kernel method, and so on. Their approaches are supervised in the sense that a set of (sentence, MR) pairs is needed for training. Other work has also addressed learning with less supervision: Clarke et al. (2010); Liang et al. (2011) use the "world's response" (the answer to the query), while Goldwasser et al. (2011) even need no supervision at all.

These studies (and related ones using the CLang domain, where the input comes from natural language instruction) have started filling a glaring gap in statistical natural language processing. However, surveying this literature, we find some major difficulties in applying these techniques to real NLP applications in less restricted domains. First, many of the used MR languages lack expressiveness. For example, FUNQL, used in the Geoquery domain, and CLang are not designed to express tense (i.e., they make no distinction between *is*, *will be* and *has been*) or to handle possibility (e.g. *can*) nor necessity (e.g. *must*). Moreover, scalability is a big problem because of the explosion of the number of concepts in open-domain texts and also of the number of syntactic structures. For instance, in Geoquery, learning in many of the current approaches succeed because a small number of constructions like *What is the...*, *How many states have...* and *A borders B* dominate the corpus.

Hence, the challenge of developing corpus-based, probabilistic techniques for *open-domain* semantic parsing remains unsolved. Approaches to this challenge face two major obstacles. First, sentences in open-domain texts are more complicated than those in domain-dependent texts, i.e. they are longer and contain more linguistic phenomena as well as syntactic structures. Second, large, standardized semantic corpora are not available (Bos, 2011). Wide coverage semantic parsers therefore continue to rely on hand-written rules (Bos, 2008; Allen et al., 2008), or try to use unsupervised techniques (Poon and Domingos, 2009). Alshawi et al. (2011) is based on so-called "natural logical forms" (derived directly from dependency structures) with limited applicability.

In this paper, we present an ambitious attempt to bridge the gap between hand-built approaches for open-domain semantic parsing and learning methods for domain-dependent semantic parsing. Crucial to the success of our approach, named DeSCoG (Dependency-based Semantic Composition using Graphs), is that we make maximum use of the information provided by

the syntactic structure of a sentence, and that we abandon the lambda calculus in favor of a more flexible graph-based representation. The syntactic structure comes in because we use a state-of-the-art syntactic dependency parser to drive semantic composition. There are several reasons to do that. First, dependency structures encode predicate-argument relations which are strongly related to semantics (Covington, 2001). Second, the total complexity is reduced significantly compared with parsing syntax and semantic simultaneously (Poon and Domingos, 2009). Finally, according to Ge (2010), prior knowledge of syntax is particularly helpful when sentences are long and complex.

We abandon the lambda calculus, like Liang et al. (2011), but unlike many approaches (Bos, 2008; Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2010; Wong, 2007; Ge and Mooney, 2009; Baral et al., 2011). Although the lambda calculus is undeniably an elegant tool for semantic composition in a bottom-up manner, it makes the learning of a lexicon difficult. This is because of the notorious, λ-inverse problem, which is to find h and g such that $f = F(h, g)$ where f and F are given (Kwiatkowski et al., 2010). Zettlemoyer and Collins (2005, 2007) solved the λ-inverse problem via generating all candidates in predefined templates which correspond to categories. However, because there are more than 300 frequent categories (in the WSJ corpus) (Bos, 2005), this method needs a huge effort to build a set of templates. Kwiatkowski et al. (2010) used restricted higher-order unification to carry out the λ-inverse. To adapt this method to open-domain semantic parsing, we need loosened restriction, which leads to higher complexity in learning a lexicon. Deviating from those approaches, DeSCoG represents semantics by graphs, which provide a way to flexibly break and combine components. At same time, the formalism we use is very expressive: the graphs we obtain for complete sentences are equivalent to Discourse Representation Structure (DRS) introduced by Kamp (1981) and the formalism of choice in the hand-built Boxer system (Bos, 2008).

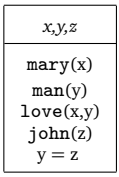
The paper is structured as follows. Section 2 gives a brief introduction to DRS and how to use graphs to replace the traditional, box-like representations. Section 3 presents our semantic composition method, which is based on dependency structures and a probabilistic parsing model. We will describe the learning in Section 4, and finally, give experimental results on both the Groningen Meaning Bank (GMB) and Geoquery in Section 5.

2 Meaning Representation with Semantic Graphs

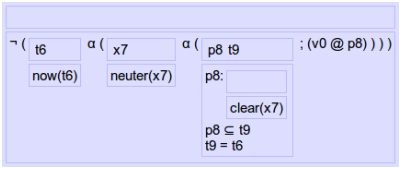
2.1 Discourse Representation Structure

Discourse Representation Theory (DRT), which was introduced by Kamp (1981), is a theoretical framework that can handle a wide range of linguistic phenomena such as tense and anaphora, within and across sentences. It uses Discourse Representation Structures (DRS) to represent a mental representation of the hearer as the discourse unfolds (Geurts and Beaver, 2011). The traditional representation for DRS is a box-like structure which contains two components: (i) *discourse referents* (e.g. x, y, z) representing entities in the discourse, and (ii) *discourse conditions* (e.g. $\texttt{man}(x), \texttt{love}(y, x)$) representing the information about the discourse referents which is encoded in the discourse. For instance, Figure 1a shows a DRS representing the meaning of the discourse *Mary loves a man. He is John.*

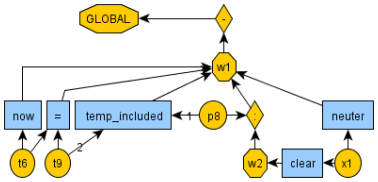
Because the pronoun *he* can only link back to the indefinite *a man*, y and z must represent the same entity. This is called *pronoun resolution*. In DRT, pronoun resolution has to satisfy the *accessibility constraint*, which says that "if y is the discourse referent introduced by a pronoun, and x is a previously introduced discourse referent, then we are only allowed to add the



(a) A sample DRS.



(b) DRS for the sentence *It is not clear* .



(c) Semantic graph for the DRS shown in Figure 1b.

Figure 1: Sample DRSs and semantic graph.

condition $y = x$ if x is in the universe of a DRS that is *accessible* from the DRS whose universe contains y" (Blackburn and Bos, 2000). Here, a DRS B_1 is immediately accessible from another DRS B_2 if (i) B_1 contains a condition of one of the forms: $\neg B_2$, $B_2 \vee B$, $B_2 \Rightarrow B$, for some DRS B; or (ii) $B_1 \Rightarrow B_2$ is a condition in some DRS B. B_1 is accessible from B_2 if B_1 is immediately accessible from B_2 or there is some DRS B such that B_1 is accessible from B which is immediately accessible from B_2.

Bos (2009) then developed Partial DRS which is a combination of the classic DRS, modal logic, type theory, and lambda calculus. It is used by Boxer (Bos, 2008), a state-of-the-art wide-coverage hand-built open-domain semantic parser.

2.2 Semantic Graph

Although the box-like representation captures the DRS structure very well, it does not provide us with a way to freely break and combine components. Therefore, we represent a DRS via a *semantic graph*, which is simple and directed (see Figure 1). There are four node types corresponding to basic elements of a DRS: referent (ellipse) (e.g. x_1), predicate (rectangle) (e.g. $thing(.)$), wrapper/box (hexagon) (e.g. w_1), and operator (rhombus) (e.g. $\neg$). A global wrapper node, named `GLOBAL`, corresponds to the outermost box of a DRS. An edge is a link

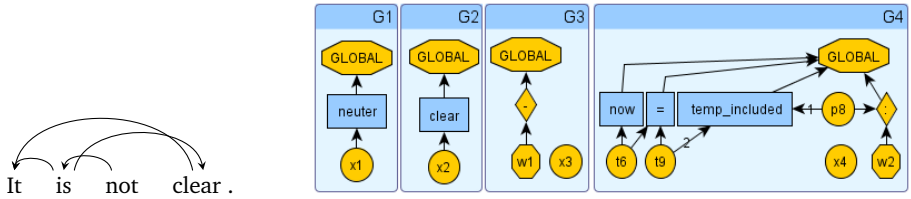
- from a referent node x to a predicate node p (i.e. p has an argument x), or
- from a predicate node p to a wrapper node w (i.e. p belongs to the DRS labelled w), or
- from a wrapper node w to an operator node o (i.e. the DRS labelled w is an argument of the operator o), or
- from a referent node x to an operator node o (i.e. referent x is an argument of the operator o).

Thanks to this representation, we can break or combine components simply by removing or adding links/nodes. From this point, we will call graphs which represent meanings of words/constituents, and need to be combined with other graphs, *partial graphs*[1].

3 Semantic Composition

In this section we describe the components of the probabilistic, semantic parser that we will use in our model. This parser computes the best semantic graph (or DRS) for a natural language sentence, given a dependency parse of that sentence and a lexicon (which might be handcrafted, or learned, as explored in section 4).

[1]A partial semantic graph should be seen as a *unsaturated formula* or a *schema*.



(a) Dependency structure. (b) Set of partial graphs.

Figure 2: Dependency structure and set of partial graphs for the example in Section 3.2

3.1 Combinatory Operators

By abandoning the lambda calculus, we also loose its method for meaning combination. To combine partial graphs anyway, we introduce two new combinatory operators. The first (dealing with argument identity) is *binding*[2], which is to bind a referent node x with another referent node v, denoted by $x \bowtie v$. The second (dealing with the box structure in DRS) is *wrapping*, which is to link a predicate/operator node p to a wrapper node w, denoted by $p \odot w$.

Now that the lambda calculus plays no role in our new semantic representation, λ-inverse is no longer a problem. Indeed, semantic graphs provide us with maximal freedom in breaking components. Therefore, learning a semantic lexicon becomes easier. However, because it is very flexible in combining partial semantic graphs, the search space explodes. As a consequence, parsing becomes more difficult. In order to efficiently search in such a large space, we need a mechanism to bias the search. We will, in the following, give details about two key ideas: (i) partial semantic graphs are combined following the guide of given dependency structures; and (ii) some combinations are more preferable than others thanks to a probability model.

3.2 Partial Graph Combination with Dependency Structures

In the following, we describe how to combine partial graphs following a dependency structure, which is given by the C&C parser (Clark and Curran, 2007) (however, the model is general in the sense that it does not depend on which dependency grammar is used). Given a sentence $S = s_1 ... s_n$, a set of partial graphs $\{G^i\}$, and a dependency structure $D = \{s_i \curvearrowright s_j\}$ where $u \curvearrowright v$ means word u is a head of word v, a constructive process to create a graph G consists of three steps:

1. for each word $s_i \in S$, select a partial graph G^i to represent its meaning;
2. for each $s_i \curvearrowright s_j \in D$, select a pair of referents (u, v) such that u in G^j and v in G^i to apply a binding operation to;
3. for each predicate/operator node f, select a wrapper node w to apply a wrapping operation to.

An example is given in the following. Given four partial graphs shown in Figure 2b, and a sentence with its dependency structure in Figure 2a, firstly, we assign a partial graph to each word: *It* :- $G_c^1 = G_1$, *is* :- $G_c^2 = G_4$, *not* :- $G_c^3 = G_3$, *clear* :- $G_c^4 = G_2$. Then, we bind x_1 with x_2, x_1 with x_4, x_2 with x_4, and p_8 with x_3. Finally, using the wrapper operation, we link `neuter`, `temp_included`, `=`, `now`, `:` to w_1, and `clear` to w_2. The graph in Figure 1c is what we should achieve.

[2]It works similarly to *unification*

It is important to note that wrapping operations should not be carried out arbitrarily. For instance, the node `clear` (Figure 1c) must not be allowed to link to the wrapper node `GLOBAL`. To stipulate that, we introduce a *wrapping constraint*: for all dependencies $s_i \curvearrowright s_j \in D$, if a referent node v in G^j binds with a referent node u in G^i then all the predicate/operator nodes in G^i linked from u must link to wrapper nodes which have access[3] to v. The constraint is based on the observation that in a C&C predicate-argument relation, the argument is always the one introducing referents.

Even with the dependency structure given, there are typically many choices to make among the possible partial graphs for each word, and possible binding or wrapping operations for each dependency. Computing the probabilities of all of these choices, and the most probable resulting semantic graph, is the task of the probability model.

3.3 Probability Model

The parsing task is to find the most probable semantic graph G^* such that

$$G^* = \arg\max_G Pr(G|S) = \arg\max_G \sum_D Pr(G|D,S)Pr(D|S) \quad (1)$$

where the sum is taken over all dependency derivations D for a sentence S. Similar to the model given by Ge and Mooney (2009), we assume that the dependency parse we obtain from a third party parser (here C&C) is correct. That is, we assume that the probability distribution over dependency structure derivations given a sentence S densely locates at the derivation D_S yielded by the used syntactic dependency parser, i.e. $Pr(D|S) = \delta(D = D_S)$ where $\delta(.)$ is the Kronecker delta function. As a consequence, the parsing problem becomes: given a sentence S and a dependency structure D, find the best semantic graph G^* such that $G^* = \arg\max_G Pr(G|S,D)$.

Let $G_c = \{G_c^1, ..., G_c^n\}$ be a set of assigned partial graphs, $B = \{u \bowtie v\}$ be a set of binding operations, and $W = \{f \hat{\odot}_k o\}$ be a set of in-wrapper relations, then we denote $G = (G_c, B, W)_{S,D}$, which means G is yielded by applying B, W on G_c following the dependency-structure D. We can now factorize $Pr(G|S,D)$ as follows:

$$Pr(G|S,D) = Pr(G_c, B, W|S,D) = Pr(G_c|S,D)Pr(B|G_c,S,D)Pr(W|G_c,B,S,D) \quad (2)$$

Under some independence assumptions,

$$Pr(G_c|S,D) = \prod_{i=1}^{n} Pr_l(G_c^i|s_i, POS(s_i), POS(Dep(s_i))) \quad (3)$$

$$Pr(B|G_c,S,D) = \prod_{u \bowtie v \in B} Pr_b\big(u \bowtie v|G_c(u), G_c(v), POS(s(u)), POS(s(v))\big) \quad (4)$$

where $G_c(x)$ is the partial graph containing node x and $s(x)$ is the word whose the partial graph is $G_c(x)$; $POS(.)$ is the function to get part-of-speech tags; and $Dep(s)$ is the function to get the list of the dependents of s. For the third component of Equation 2,

$$Pr(W|G_c,B,S,D) = Z \times \psi(W) \times \prod_{f \hat{\odot}_k o \in W} Pr_w\big(f \hat{\odot}_k o|G_c(f), G_c(o), POS(path(s(f), s(o)))\big) \quad (5)$$

[3]Because a wrapper node corresponds to a box of a DRS, the definition of *accessibility* is also applicable in this case.

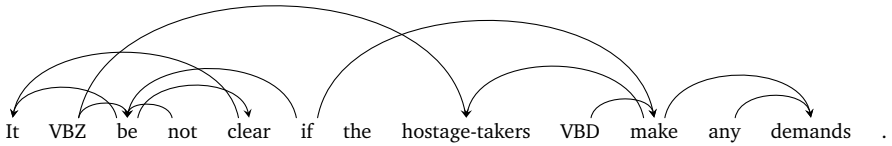


Figure 3: Dependency structure after extracting tenses out of verbs.

where Z is the normalizing factor. $\psi(W)$ is the indicator function that returns 1 if W satisfies the wrapping constraint and 0 otherwise. We use the notation $f\,\hat{\odot}_k o$ for the in-wrapper relation[4] where there is a directed path from f to the k-th wrapper of the operator node o (i.e., f links to the k-th wrapper of an operator node o, or there is an operator node o' such that f links to one of o's wrappers and $o'\hat{\odot}_k o$). $f\,\hat{\odot}_0 o$ then means that there are no directed paths from f to o (e.g. in Figure 1c, `clear` $\hat{\odot}_1\langle\neg\rangle$ and `neuter` $\hat{\odot}_0\langle:\rangle$). Finally, $path(s_i, s_j)$ is the shortest path from s_i to s_j on the graph representing the dependency structure D. For instance, $path(It, not) = It \uparrow is \uparrow not$ (Fig. 2a).

Given this probability model, we can use standard techniques for finding the best semantic graph[5] G^*, given a sentence S and its dependency structure D:

$$G^* = \underset{G=(G_c,B,W)_{S,D}}{\arg\max}\ Pr(G_c|S,D)Pr(B|G_c,S,D)Pr(W|G_c,B,S,D) \tag{6}$$

Solving this optimization problem required using beam search, integer linear programming and some additional heuristics, as detailed in the appendix.

4 Learning

In this section we describe how to obtain a lexicon of partial graphs. The learning algorithm is presented with combinations of sentences, their dependency parses and their complete semantic graphs, i.e., with a set of triples (S, D, G).

4.1 Splitting Semantic Graphs

Given a sentence S and its semantic graph G, we need to split G into partial graphs corresponding to individual words. It turns out that the task is similar to the word-to-word alignment problem in machine translation. Here, English is the source language, the set of all semantic graphs is the target language, and a "word" in the target language is a connected partial graph. Hence, our problem is *word-to-graph alignment*. In Figure 3 we give a more complicated example sentence and its dependency graph (where verbs are split up in stems and tags with tense/aspect information, as explained below). In Figure 4 we give an example of an alignment with a semantic graph for this sentence .

[4] In the box representation of a DRS, $f\,\hat{\odot}_k o$ means the predicate/operator f locates in the k-th box of the operator o (directly or indirectly), and $f\,\hat{\odot}_0 o$ means f does not locate in any box of o.

[5] Are the resulting DRSs valid? Although we have not provided a formal proof, we are confident they are well-formed because of the following two reasons. First, the definition of 'link' ensures that elements of a DRS (referents, predicates, boxes, etc.) are in proper relationships (e.g. it is impossible that a predicate is an argument for another predicate). Second, the wrapping constraint guarantees that the accessibility constraint is not violated. A rigorous proof of this statement is part of our future work.

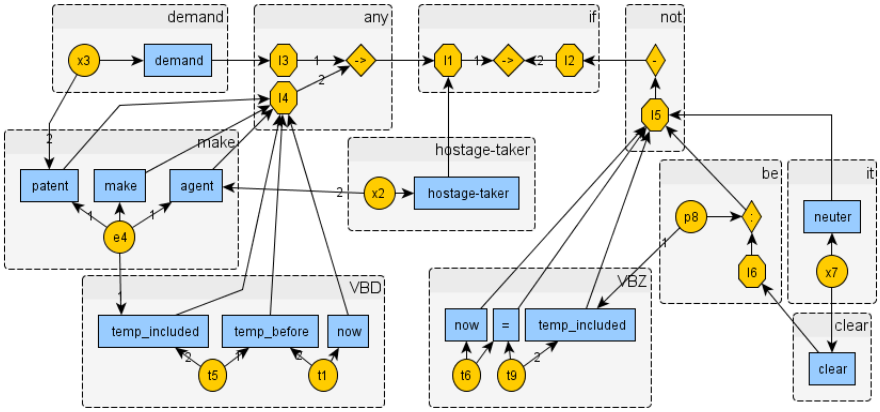


Figure 4: Semantic graph with the correct alignment for the sentence given in Figure 3.

For each triple (S, D, G), let V be the node set of G, V_F be the predicate/operator node set of G, an alignment A be a set of pairs (u, w) where $u \in V$, and $s \in S$. An alignment is feasible if all of the following constraints are satisfied

1. for each $u \in V$, if u is a predicate or operator node, there is one and only one $s \in S$ such that $(u, s) \in A$; otherwise, for each predicate/operator node f that u links to, if $(f, s) \in A$, then $(u, s) \in A$,
2. for each $s \in S$, let $V_s^A = \{u \in V | (u, s) \in A\}$, V_s^A is undirectedly connected on G.
3. for each $s \curvearrowright s' \in D$, there are $u' \in V_{s'}$ and $u \in V_s$ such that u' is a predicate node. If u is a predicate node, there is a referent node x linking to both u, u'; else if u is an operator node, there is a wrapper node w linking to u such that u' links to w.

Let $\Lambda(S, D, G)$ be the set of all feasible alignments. The alignment phase is given as follows. First, a word-to-word alignment application[6] is used to estimate the probabilities $Pr(\text{node}|\text{word})$. Then, for each triple (S, D, G), the best alignment A^* is defined as

$$A^* = \underset{A \in \Lambda(S,D,G)}{\arg\max}\ Pr(V_F | S = s_1 ... s_n, A) \tag{7}$$

Because this kind of alignment is one-to-many, the number of targets for one source is considered. Hence,

$$Pr(V_F | S = s_1 ... s_n, A) \propto \prod_{s \in S} Pr(V_F \cap V_s^A | s) \propto \prod_{s \in S} \left(Pr_n(|V_F \cap V_s^A| | s) \prod_{f \in V_F \cap V_s^A} Pr_f(f | s) \right) \tag{8}$$

Considering solving Equation 7 as searching an A in the alignment space to minimize the objective function $g(A) = -\log\left(Pr(V_F^A | S^A, A)\right)$, we use the **A**-star algorithm with the future cost function $h(A) = \min_{A'}\{-\log\left(Pr(V_F \setminus V_F^A | S \setminus S^A, A')\right)\}$, where $V_F^A = \{u | \exists s, (u, s) \in A\} \cap V_F$ and $S^A = \{s | \exists u, (u, s) \in A\}$. It is worth noting that, because the future cost is smaller than the real cost, the optimal solution (if it exists) is guaranteed to be found.

[6] We used GIZA++ (Och and Ney, 2003) with the default parameters.

Level	$Pr_l(G\|...)$
1	$Pr_l(G\|s, POS(s), POS(Dep(s)))$
2	$Pr_l(G\|s, POS(s))$
Level	$Pr_b(u \bowtie v\|...)$
1	$Pr_b(u \bowtie v\|G(u), G(v), POS(s(u)), POS(s(v)))$
2	$Pr_b(u \bowtie v\|G(u), G(v))$
3	$U_{u,v}(u \bowtie v\|G(u), G(v))$
Level	$Pr_w(f \hat{\odot}_k o\|...)$
1	$Pr_w(f \hat{\odot}_k o\|G(f), G(o), POS(path(s(o), s(v))))$
2	$Pr_w(f \hat{\odot}_k o\|G(f), G(o), POS(s(u)), POS(s(v)))$
3	$U_k(f \hat{\odot}_k o\|G(f), G(o), POS(s(u)), POS(s(v)))$

Table 1: Multilevel back-off for the parameter estimation. $U_{u,v}(u \bowtie v|G(u), G(v))$ and $U_k(f \hat{\odot}_k o|G(f), G(o), POS(s(u)), POS(s(v)))$ are respectively the uniform distributions over u, v and over k.

Implementation The word-to-graph alignment method has a drawback: large partial graphs are not preferred because the probabilities of some nodes in a large partial graph given the best matched word tend to be small. Unfortunately, verbs with tenses have complex DRSs, which lead to large partial graphs (e.g. G_4 in Figure 2b represents the meaning of the verb *is*). To avoid this phenomenon, tense/aspect information is extracted from verbs (e.g. Figure 3).

We apply some heuristics to make the word-to-graph alignment more efficient. Operator nodes are only allowed to align to functional words (e.g. *be, will, could*), and predicate nodes related to tenses (e.g. `temp_included, temp_before`) are only allowed to align to tense words (e.g. *VBZ, VBN*). Because the **A**-star algorithm will search the whole solution space in the worst case, the alignment process spends a lot of time on difficult sentences. Therefore, sentences with lengths larger than MAX_LENGTH are ignored and the time for processing a sentence is not allowed to exceed MAX_TIME[7].

4.2 Parameter Estimation

The parameters which are used in the parsing model are simply estimated via relative frequencies. To avoid the problem of sparse data, two techniques are used (see Jurafsky and Martin, 2009, Section 4.5 and Section 4.7): (i) Good-Turing technique, and (ii) multilevel back-off which is shown in Table 1.

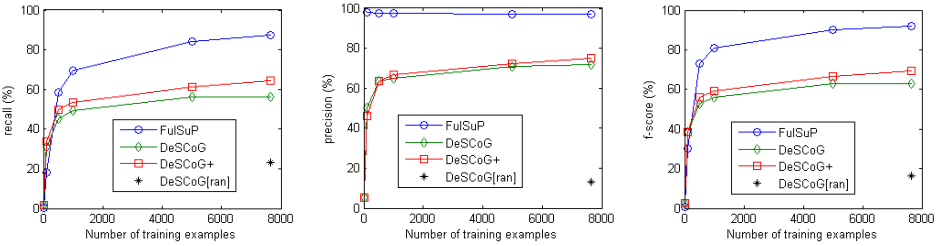
5 Experimental Results

In this section, we will show experimental results on the two corpora: Groningen Meaning Bank (GMB), an open-domain corpus, and Geoquery, a popular domain-dependent corpus.

5.1 Groningen Meaning Bank

The Groningen Meaning Bank (GMB) corpus (Basile et al., 2012) has been developed at the University of Groningen for the tasks related to deep semantic representation. The current version 1.1 contains 2000 documents with 9418 sentences from many public sources: Voice of America, fables, CIA World Factbook, and MASC Full. Each sentence is annotated with a CCG parse tree with a Partial DRS at each tree node representing the meaning of the corresponding constituent. Although GMB covers many linguistic phenomena, such as anaphora

[7]In our experiments, we set $MAX\_LENGTH = 25, MAX\_TIME = 5sec$.



(a) Recall (b) Precision (c) F-score

Figure 5: Recall, precision, and f-score curves comparing the four parsers on GMB.

and presupposition, we leave them aside. For instance, DeSCoG does not resolve anaphora whereas it resolves presupposition simply with local resolution.

Evaluation Method Our evaluation method is similar to the one introduced by Allen et al. (2008), which is based on partial credit assignment. Let G and T be semantic graphs for a gold-standard DRS and a test DRS respectively. Given an one-to-one alignment between G's nodes and T's nodes such that it results their maximum common subgraph, then

- a predicate node p in T is counted if it and all referent nodes linking to it are aligned,
- an operator node o in T is counted if it and all wrapper and referent nodes linking to it are aligned,
- count one for a predicate/operator node f in T if the wrapper node linked from f is aligned and the operator node that it links to is counted and f itself is also counted.

Intuitively, the last counting assigns credit to a component in the test DRS if its position is correct w.r.t the gold-standard DRS. Let $\Omega(G,T)$ be the number of counts, then recall $= \frac{\Omega(G,T)}{\Omega(G,G)}$, precision $= \frac{\Omega(G,T)}{\Omega(T,T)}$, and f-score $= \frac{2\Omega(G,T)}{\Omega(G,G)+\Omega(T,T)}$.

Settings We split GMB into two parts: 7642 examples in the sections from 0 to 79 for training (GMB.0-79), and the rest, 1776 examples, for testing (GMB.80-99). Because DeSCoG, to our knowledge, is the first working on this corpus, we created the following three parsers for the comparison purpose. **FulSuP** (Fully Supervised Parser) is a probabilistic semantic parser using the supervised semantic lexicon directly extracted from the GMB and some manually created semantic combinatory rules. This parser is actually a probabilistic interpretation of Boxer, the core builder of GMB. **DeSCoG+** is DeSCoG with the help from an "oracle" for the alignment process: the information about which predicate/operator nodes' labels are aligned with which words is known beforehand thanks to the semantic lexicon given by GMB. **DeSCoG[ran]** (baseline) is DeSCoG with random parameters. It is important to point out that the four parsers use different forms of supervision. FulSuP, as its name recalls, uses most supervision form; DeSCoG+ needs a set of (sentence, MRs) pairs and an oracle for training. DeSCoG and DeSCoG[ran] learn from only (S, D, G) triples.

Results

Figure 5 shows the performance of DeSCoG compared to the other three parsers. The results clearly reflect the advantage of using the given semantic lexicon for training in FulSuP and DeSCoG+. The fact that FulSuP achieves the best performance (f-score 92.1% with the full

training dataset) is not surprising at all because its model is a probabilistic interpretation of the Boxer's model, which is the core builder of GMB. At the other end, DeSCoG[ran] performs remarkably poorly with an f-score of 16.3%. The result discloses the effectiveness of the parameter estimation, which helps DeSCoG gain the f-score 63.1%, which is significantly better than DeSCoG[ran].

DeSCoG+ performs better than DeSCoG. An interesting point here is that the distance between their performances is proportional to the training data size. This suggests that the alignment process of DeSCoG insufficiently exploited new structures in the added data. This suspicion is supported by the recall curve of DeSCoG: the recall is nearly unchanged after the training data size reaches 5000. DeSCoG+, on the other hand, overcomes this obstacle thanks to the help of the oracle.

Analysing the Alignment Phase

As discussed above, the quality of the alignment process strongly affects the final performance of DeSCoG. Therefore, in this section, we will look into it in some more details. First of all, because sentences longer than $MAX\_LENGTH = 25$ are ignored and the time for processing one sentence is not allowed to exceed $MAX\_TIME = 5sec$, it is clear that not all the sentences and their semantic graphs in the training dataset are successfully aligned. We found that, for the GMB.0-79, the alignment phase succeeded 5725 times, which is 74.9%.

To see when the alignment phase is wrong, let's consider an alignment result. Figure 3, 4 show the dependency structure and the semantic graph with the correct alignment for the sentence *It is not clear if the hostage-takers made any demands*. The alignment algorithm correctly aligned all of the words except *any* and *if* (Figure 6). Because $Pr(\rightarrow|any) = 0.69 > Pr(\rightarrow|if) = 0.48$, both the operator node →'s were aligned with the word *any*, and, consequently, no nodes were with the word *if*.

Theoretically, incorrect alignments could be tolerated thanks to the probability model presented in Section 3.3. For instance, if *any* and *if* do not appear together so frequently, the above fault will not have a strong effect. That is why DeSCoG+ does not perform significantly better than DeSCoG. However, the fact that the recall of DeSCoG is nearly unchanged when the training data size is over 5000 suggests that there are some structures that the alignment phase missed even when more training data were used.

5.2 Geoquery

Geoquery is a corpus which is used in the Geoquery domain, a natural language interface to a U.S geography database (Zelle and Mooney, 1996). The database contains 800 facts (e.g. names, areas, and populations), which are represented as Prolog assertions. The corpus is a collection of 880 English queries and their manually annotated MRs in a first-order Prolog language augmented with meta-predicates and Functional Query Language (FUNQL).

Although our purpose is learning open-domain semantic parsing, we found two reasons to test DeSCoG on Geoquery. First, because there are many existing semantic parsers working on this corpus, the comparison is more reliable than the previous. Second, because of many differences between Geoquery and GMB, we want to investigate the flexibility of DeSCoG. In our experiments, 10-fold cross validation was used. A test MR is correct if it and the gold-standard MR receive the same answer. Let n_c, n_D, n_P be respectively the number of correct MRs, the test

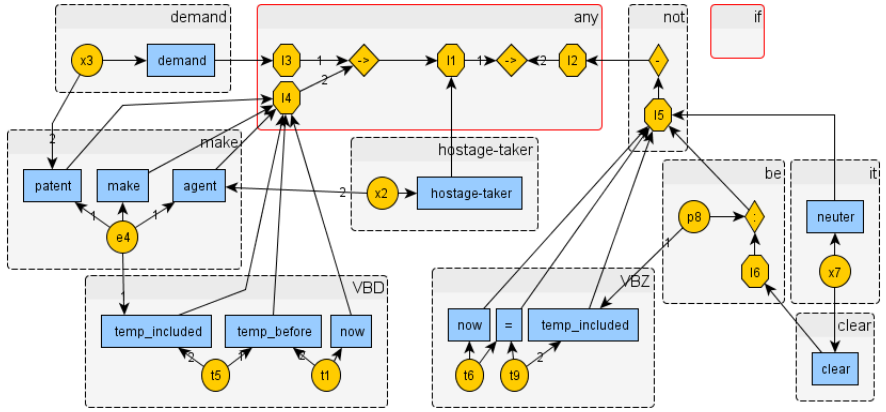
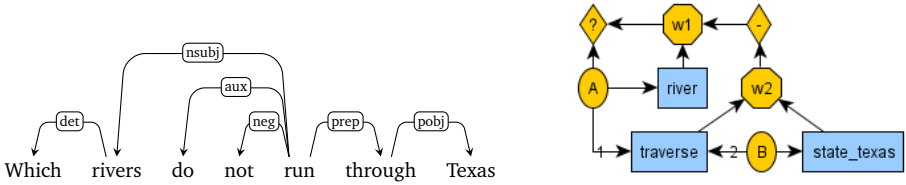


Figure 6: The alignment phase correctly aligned all the words except *any* and *if*. The correct one is given in Figure 4.



(a) Dependency structure.

(b) Semantic graph.

Figure 7: Dependency structure and semantic graph for the query *which rivers do not run through Texas ?*, which has the MR
`answer(A,(river(A),not((traverse(A,B),const(B,stateid(Texas))))))`.

data size, and the number of completed MRs, then $recall = n_c/n_D$, $precision = n_c/n_P$, and $fscore = 2n_c/(n_D + n_P)$.

We made the following changes in DeSCoG. Firstly, semantic graphs are now for representing Prolog-based first-order forms. This is easily done by using operator nodes to represent metapredicates such as `answer` and `largest` (see Figure 7). Secondly, the Stanford parser[8] (De Marneffe et al., 2006) was used to syntactically parse English sentences, because it better deals with questions. Finally, the wrapping constraint was removed.

DeSCoG was compared with the following alternatives: SCISSOR (Ge and Mooney, 2005), an integrated syntactic-semantic parser; KRISP (Kate and Mooney, 2006), a SVM-based parser using string kernels; WASP (Wong and Mooney, 2006) and λ-WASP (Wong, 2007), two parsers based on synchronous grammars; Z&C05[9] (Zettlemoyer and Collins, 2005), a parser using structural learning with CCG grammars; and SYN0 (Ge and Mooney, 2009), a parser using an existing syntactic parser. Among those systems, SYN0 is the closest to DeSCoG: both of them use existing syntactic parsers and a Prolog-based first-order language as their MR languages. Table 2 shows the experimental results.

[8] http://nlp.stanford.edu/software/corenlp.shtml

[9] The results were reported on 600 training examples and 280 testing examples.

	Recall	Precision	Fscore
DeSCoG	74.89	87.40	80.66
SYN0	78.98	81.76	80.35
λWASP	86.59	91.95	89.19
Z&C05	79.29	96.25	86.95
SCISSOR	72.3	91.5	80.77
WASP	74.8	87.2	80.5
KRISP	71.7	93.3	81.1

Table 2: The results for DeSCoG and the alternatives on the Geoquery corpus.

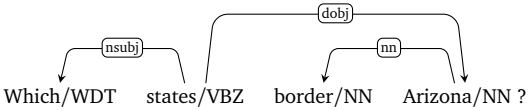


Figure 8: A sentence with its incorrect syntactic parse.

The results show that DeSCoG preforms equivalently to SYN0, SCISSOR, WASP, and KRISP, but worse than λWASP and Z&C05. According to Wong (2007), SCISSOR, WASP, and KRISP perform worse than λWASP and Z&C05 because they use a different MR language, FUNQL. However, because FUNQL is variable-free, it is clear that those parsers cannot be widely used (e.g. they are not able to work on GMB). DeSCoG, on the other hand, is flexible. Although it was mainly designed for GMB (i.e. using a DRS language, the C&C parser), it achieved equal performance with many other parsers.

As mentioned above, DeSCoG and SYN0 use existing syntactic parsers and the same MR language with λWASP. So why do they perform worse than λWASP and Z&C05? Firstly, syntactic parsers are not really helpful when sentences are short. This is because sentence structures could be easily learned just from (sentence, MR) pairs in this case (Ge, 2010). Secondly, incorrect syntactic parses could lead to incorrect (or uncompleted) MRs. For example, with the incorrect syntactic parse shown in Figure 8, it is very difficult to build the correct MR `answer(A,(state(B),next_to(A,B),const(B,stateid(arizona))` because one of the two words *state* and *Arizona* has to receive the meaning `next_to(A,B)`, which is very unlikely. On the other hand, parsers which learn parsing syntax and semantics simultaneously could easily overcome this problem when they recognize that the word *border* should be a head of two dependents. This is thanks to the high frequency of the appearance of the structure *A border B* in the Geoquery corpus, and that the word *border* likely matches the meaning *next_to(A,B)*.

Conclusion

This paper introduced a new learning approach, DeSCoG, mainly for open-domain semantic parsing. Our approach is different from others which use lambda calculus. In those approaches, learning semantic lexicon requires huge effort because they face the λ-inverse problem. In our approach, this problem is avoided thanks to the usage of semantic graphs, which provides maximal freedom for breaking and combining MRs. In order to bias the search in a very large parse space, we used the prior knowledge of syntax encoded in dependency structures and a probability model. The former provides a skeleton for semantic composition whereas the latter tells the parser which combinations are more preferable.

DeSCoG has common points with some approaches. First, the idea of flexible semantic com-

position is also proposed by Clarke et al. (2010); Liang et al. (2011) where lambda forms and semantic composition rules are removed. Second, the use of existing syntactic parsers in DeSCoG is inspired by Ge and Mooney (2009). Finally, UPS (Poon and Domingos, 2009), Alshawi et al. (2011), and DeSCoG use dependency structures as the prior knowledge of syntax for semantic composition.

We tested DeSCoG on the two corpora, GMB for open-domain semantic parsing and Geoquery for domain-dependent semantic parsing. On the one hand, DeSCoG achieved the f-score 63.1% on GMB. The fact that this f-score is significantly higher than the f-score of the random parser (16.3%) reflects that our proposed parsing model works for open-domain semantic parsing. On the other hand, DeSCoG performed on Geoquery equally to many existing parsers: SYN0, SCISSOR, WASP, and KRISP.

Appendix: Search heuristics

We will describe a method for finding the best semantic graph G^* in Equation 6 given a sentence S and its dependency structure D. The problem is indeed searching within an enormous semantic graph space. The search is biased by the dependency structure D and the probability model given above. Here, we use the beam search strategy to cut off branches which could lead to solutions with low probabilities. To reduce the total complexity, we divide the search into two stages. First, the search, with the goal to maximize $Pr(G_c|S,D)Pr(B|G_c,S,D)$, will output a list of N-best (G_c,B)'s. Then, the search will look for the best W for each of those N-best (G_c,B)'s.

Stage 1 The goal of this stage is to find N-best (G_c,B)'s. The search processes one word at a time with the order: (i) from left to right, and (ii) all the dependents of the word were already processed. And, each word is processed only one time (i.e. no backtracking required). For example, given the sentence in Figure 2a, *It* is processed first, then *clear* and *is*; *not* is processed last. When processing a word (e.g. *is*), the search will considers all the candidate partial graphs for the meaning of this word, then use the binding operator to combine them with the available parses yielded after processing the prior words (*it, clear*). When there are no words left, it will remove all but N-best candidates with the highest $Pr(G_c|S,D)Pr(B|G_c,S,D)$.

Stage 2 After finding the set of N-best (G_c,B)'s, the search will look for the most probable W^* for each (G_c,B), i.e. solve $W^* = \arg\max_W Pr_w(W|G_c,B,S,D)$ where $Pr_w(W|G_c,B,S,D)$ is given by Equation 5. Because the normalizing factor Z is the same for given G_c,B, the task is to maximize the third component under the wrapping constraint, which is encoded in $\psi(W)$. It is worth noting that this is indeed a linear optimization problem where the logarithm of the third component, i.e. $\sum_{f\,\hat{\odot}_k o \in W} \log Pr_w\left(f\,\hat{\odot}_k o|G_c(f),G_c(o),POS(path(s(f),s(o)))\right)$, is the objective function. Therefore, we solve it by making use of Integer Linear Programming[10]. In order to calculate Z for each G_c,B, we need to take the sum over all feasible W's, which is intractable. Therefore, Z is approximated on M-best W's $Z \approxeq \left(\sum_{i=1}^{M} e^{F(W_i)}\right)^{-1}$. Together, this give us a set of triples (G_c,B,W) with their probabilities.

Acknowledgments

We thank Remko Scha for valuable discussions and three anonymous reviewers for helpful comments.

[10]Gurobi solver is used. http://www.gurobi.com/

References

Allen, J., Swift, M., and de Beaumont, W. (2008). Deep semantic analysis of text. In *Symposium on Semantics in Systems for Text Processing (STEP)*, volume 2008.

Alshawi, H., Chang, P. C., and Ringgaard, M. (2011). Deterministic statistical mapping of sentences to underspecified semantics. *Proceedings of the Ninth IWCS*, pages 15–24.

Baral, C., Dzifcak, J., Gonzalez, M. A., and Zhou, J. (2011). Using inverse lambda and generalization to translate english to formal languages. *Arxiv preprint arXiv:1108.3843*.

Basile, V., Bos, J., Evang, K., and Venhuizen, N. (2012). Developing a large semantically annotated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12). To appear*.

Blackburn, P. and Bos, J. (2000). Working with discourse representation theory: An advanced course in computational semantics. *Draft available at www. comsem. org*.

Bos, J. (2005). Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics IWCS*, volume 6, pages 42–53.

Bos, J. (2008). Wide-coverage semantic analysis with boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, volume 1, pages 277–286.

Bos, J. (2009). Towards a large-scale formal semantic lexicon for text processing. In *Proceedings of the Biennal GSCL Conference From Form to Meaning: Processing Texts Automatically*, pages 3–14.

Bos, J. (2011). A survey of computational semantics: Representation, inference and knowledge in wide-coverage text understanding. *Language and Linguistics Compass*, 5(6):336–366.

Clark, S. and Curran, J. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Clarke, J., Goldwasser, D., Chang, M., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27.

Covington, M. A. (2001). A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, pages 95–102.

De Marneffe, M., MacCartney, B., and Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Ge, R. (2010). Learning for semantic parsing using statistical syntactic parsing techniques. Technical report, DTIC Document.

Ge, R. and Mooney, R. (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.

Ge, R. and Mooney, R. J. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 9–16.

Geurts, B. and Beaver, D. I. (2011). Discourse representation theory. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Fall 2011 edition.

Goldwasser, D., Reichart, R., Clarke, J., and Roth, D. (2011). Confidence driven unsupervised semantic parsing. In *Proc. of the Meeting of Association for Computational Linguistics (ACL), Portland, OR, USA*.

Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall.

Kamp, H. (1981). A theory of truth and semantic representation. *Formal Semantics*, pages 189–222.

Kate, R. and Mooney, R. (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*, pages 913–920.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (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*, pages 1223–1233.

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Liang, P., Jordan, M., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 590–599. Association for Computational Linguistics.

Montague, R. (1970). Universal grammar. *Theoria*, 36(3):373–398.

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

Poon, H. and Domingos, P. (2009). Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 1–10.

Wong, Y. W. (2007). *Learning for semantic parsing and natural language generation using statistical machine translation techniques*. ProQuest.

Wong, Y. W. and Mooney, R. J. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446.

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

Zettlemoyer, L. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of UAI*, volume 5.

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