Large-scale Semantic Parsing without Question-Answer Pairs

Siva Reddy, Mirella Lapata, Mark Steedman
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
siva.reddy@ed.ac.uk, mlap@inf.ed.ac.uk, steedman@inf.ed.ac.uk

Abstract

In this paper we introduce a novel semantic parsing approach to query Freebase in natural language without requiring manual annotations or question-answer pairs. Our key insight is to represent natural language via semantic graphs whose topology shares many commonalities with Freebase. Given this representation, we conceptualize semantic parsing as a graph matching problem. Our model converts sentences to semantic graphs using CCG and subsequently grounds them to Freebase guided by denotations as a form of weak supervision. Evaluation experiments on a subset of the FREE917 and WEBQUESTIONS benchmark datasets show our semantic parser improves over the state of the art.

1 Introduction

Querying a database to retrieve an answer, telling a robot to perform an action, or teaching a computer to play a game are tasks requiring communication with machines in a language interpretable by them. *Semantic parsing* addresses the specific task of learning to map natural language (NL) to machine interpretable formal meaning representations. Traditionally, sentences are converted into logical form grounded in the symbols of some fixed ontology or relational database.

Approaches for learning semantic parsers have been for the most part supervised, using annotated training data consisting of sentences and their corresponding logical forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Kwiatkowski et al., 2010). More recently, alternative forms of supervision have been proposed to alleviate the annotation burden, e.g., by learning from conversational logs (Artzi and Zettlemoyer, 2011), from sentences paired with system behavior (Chen and Mooney, 2011; Goldwasser and Roth, 2011; Artzi and Zettlemoyer, 2013), via distant supervision (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013), from questions (Goldwasser et al., 2011; Poon, 2013; Fader et al., 2013), and question-answer pairs (Clarke et al., 2010; Liang et al., 2011). Indeed, methods which learn from question-answer pairs have been gaining momentum as a means of scaling semantic parsers to large, open-domain problems (Kwiatkowski et al., 2013; Berant et al., 2013; Berant and Liang, 2014; Yao and Van Durme, 2014). Figure 1 shows an example of a question, its annotated logical form, and answer (or denotation).

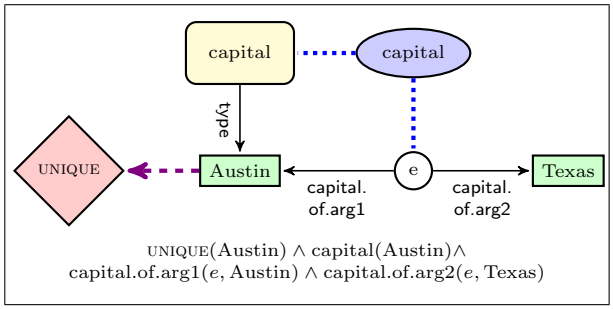
Question	What is the capital of Texas?
Logical Form	$\lambda x.\ \text{city}(x) \wedge \text{capital}(x, \text{Texas})$
Answer	{Austin}

Figure 1: An example question with annotated logical query, and its answer.

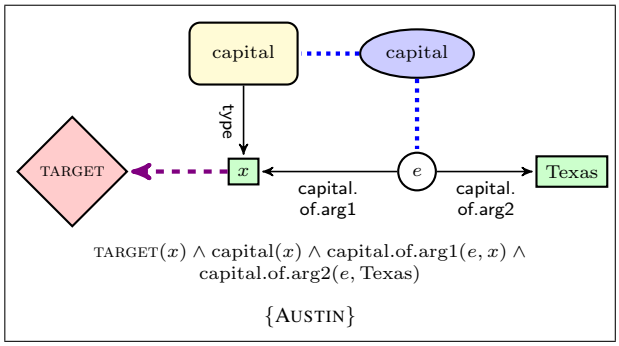
In this paper, we build a semantic parser that does not require example annotations or question-answer pairs but instead learns from a large knowledge base (KB) and web-scale corpora. Specifically, we exploit Freebase, a large community-authored knowledge base that spans many sub-domains and stores real world facts in graphical format, and parsed sentences from a large corpus. We formulate semantic parsing as a graph matching problem. We convert the output of an open-domain combinatory categorial grammar (CCG) parser (Clark and Curran, 2007) into a graphical representation and subsequently map it onto Freebase. The parser's graphs (also called *ungrounded* graphs) are mapped to all possible Freebase subgraphs (also called *grounded* graphs) by replacing edges and nodes with relations and types in Freebase. Each grounded graph corresponds to a unique grounded logical query. During learning, our semantic parser is trained to identify which KB subgraph best corresponds to the NL graph. Problem-

$$\text{capital}(\text{Austin}) \wedge \text{UNIQUE}(\text{Austin}) \wedge \text{capital.of.arg1}(e, \text{Austin}) \wedge \text{capital.of.arg2}(e, \text{Texas})$$

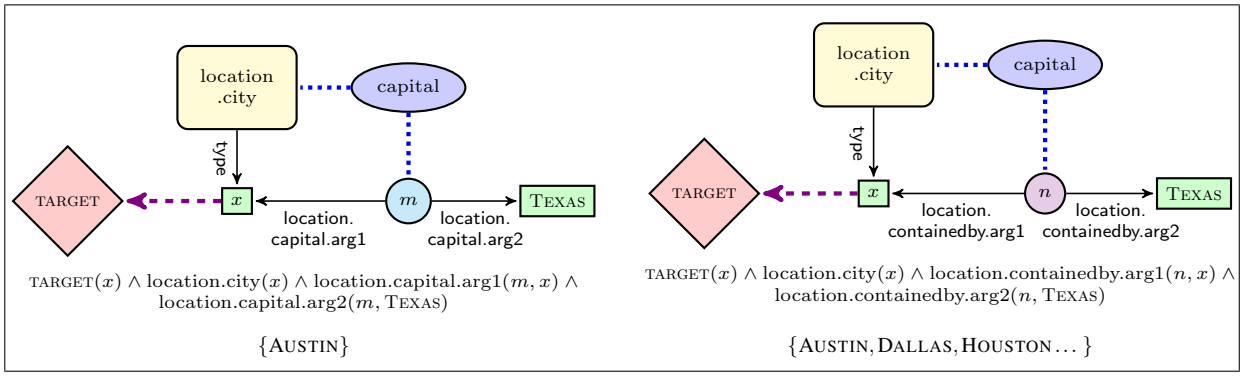
(a) Semantic parse of the sentence *Austin is the capital of Texas.*



(b) Ungrounded graph for semantic parse (a); UNIQUE means that *Austin* is the only capital of *Texas.*



(c) Query graph after removing *Austin* from graph (b) and its denotation.



(d) Freebase graphs for NL graph (c) and their denotations.

Figure 2: Steps involved in converting a natural language sentence to a Freebase grounded graph.

atically, ungrounded graphs may give rise to many grounded graphs. Since we do not make use of manual annotations of sentences or question-answer pairs, we do not know which grounded graphs are correct. To overcome this, we rely on comparisons between denotations of natural language queries and related Freebase queries as a form of *weak supervision* in order to learn the mapping between NL and KB graphs.

Figure 2 illustrates our approach for the sentence *Austin is the capital of Texas.* From the CCG syntactic derivation (which we omit here for the sake of brevity) we obtain a semantic parse (Figure 2a) and convert it to an ungrounded graph (Figure 2b). Next, we select an entity from the graph and replace it with a variable x, creating a graph corresponding to the query *What is the capital of Texas?* (Figure 2c). The math function UNIQUE on *Austin* in Figure 2b indicates *Austin* is the only value of x which can satisfy the query graph in Figure 2c. Therefore, the denotation[1] of the NL query graph is {AUSTIN}. Figure 2d shows two different groundings of the query graph in the Freebase KB. We obtain these by replacing edges and nodes in the query graph with Freebase relations and types. We use the denotation of the NL query as a form of weak supervision to select the best grounded graph. Under the constraint that the denotation of a Freebase query should be the same as the denotation of the NL query, the graph on the left hand-side of Figure 2d is chosen as the correct grounding.

Experimental results on two benchmark datasets consisting of questions to Freebase — FREE917 (Cai and Yates, 2013) and WEBQUESTIONS (Berant

[1]The denotation of a graph is the set of feasible values for the nodes marked with TARGET.

et al., 2013) — show that our semantic parser improves over state-of-the-art approaches. Our contributions include: a novel graph-based method to convert natural language sentences to grounded semantic parses which exploits the similarities in the topology of knowledge graphs and linguistic structure, together with the ability to train using a wide range of features; a proposal to learn from a large scale web corpus, without question-answer pairs, based on denotations of queries from natural language statements as weak supervision; and the development of a scalable semantic parser which besides Freebase uses CLUEWEB09 for training, a corpus of 503.9 million webpages. Our semantic parser can be downloaded from `http://sivareddy.in/downloads`.

2 Framework

Our goal is to build a semantic parser which maps a natural language sentence to a logical form that can be executed against Freebase. We begin with CLUEWEB09, a web-scale corpus automatically annotated with Freebase entities (Gabrilovich et al., 2013). We extract the sentences containing at least two entities linked by a relation in Freebase. We parse these sentences using a CCG syntactic parser, and build semantic parses from the syntactic output. Semantic parses are then converted to semantic graphs which are subsequently grounded to Freebase. Grounded graphs can be easily converted to a KB query deterministically. During training we learn which grounded graphs correspond best to the natural language input. In the following, we provide a brief introduction to Freebase and its graph structure. Next, we explain how we obtain semantic parses from CCG (Section 2.2), how we convert them to graphs (Section 2.3), and ground them in Freebase (Section 2.4). Section 3 presents our learning algorithm.

2.1 The Freebase Knowledge Graph

Freebase consists of 42 million entities and 2.5 billion facts. A fact is defined by a triple containing two entities and a relation between them. Entities represent real world concepts, and edges represent relations, thus forming a graph-like structure.

A Freebase subgraph is shown in Figure 3 with

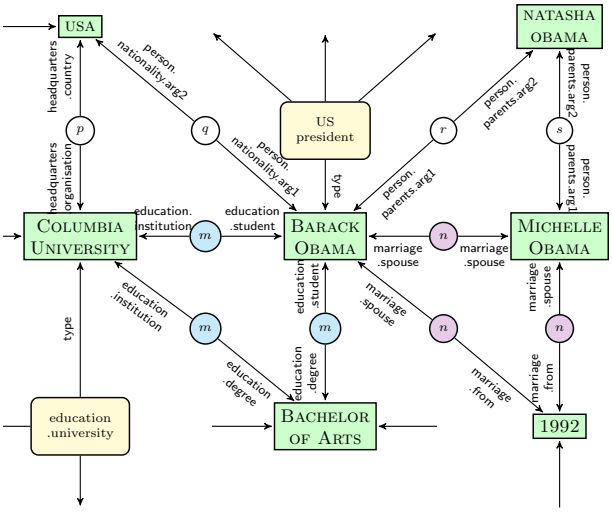


Figure 3: Freebase knowledge graph. Entities are represented by rectangles, relations between entities by edges, mediator nodes by circles, types by rounded rectangles.

rectangles denoting entities. In addition to simple facts, Freebase encodes complex facts, represented by multiple edges (e.g., the edges connecting BARACK OBAMA, COLUMBIA UNIVERSITY and BACHELOR OF ARTS). Complex facts have intermediate nodes called mediator nodes (circles in Figure 3 with the same identifiers e.g., *m* and *n*). For reasons of uniformity, we assume that simple facts are also represented via mediator nodes and split single edges into two with each subedge going from the mediator node to the target node (see person.nationality.arg1 and person.nationality.arg2 in Figure 3). Finally, Freebase also has entity types defining is-a relations. In Figure 3 types are represented by rounded rectangles (e.g., BARACK OBAMA is of type US president, and COLUMBIA UNIVERSITY is of type education.university).

2.2 Combinatory Categorial Grammar

The graph like structure of Freebase inspires us to create a graph like structure for natural language, and learn a mapping between them. To do this we take advantage of the representational power of Combinatory Categorial Grammar (Steedman, 2000). CCG is a linguistic formalism that tightly couples syntax and semantics, and can be used to model a wide range of language phenom-

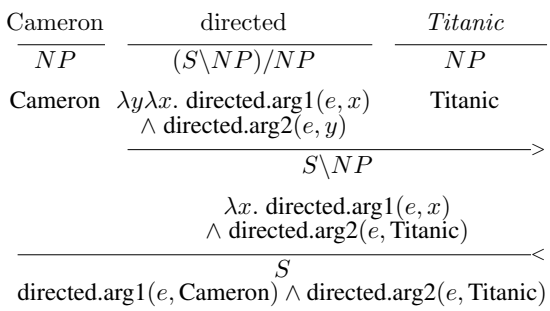
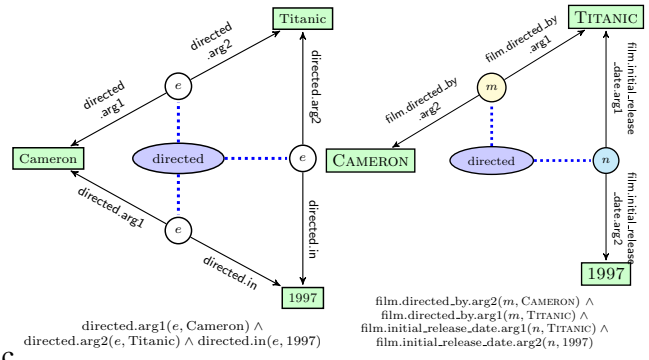


Figure 4: CCG derivation containing both syntactic and semantic parse construction.



(a) Ungrounded graph (b) Grounded graph

Figure 5: Graph representations for the sentence *Cameron directed Titanic in 1997*.

ena. CCG is well known for capturing long-range dependencies inherent in constructions such as coordination, extraction, raising and control, as well as standard local predicate-argument dependencies (Clark et al., 2002), thus supporting wide-coverage semantic analysis. Moreover, due to the transparent interface between syntax and semantics, it is relatively straightforward to built a semantic parse for a sentence from its corresponding syntactic derivation tree (Bos et al., 2004).

In our case, the choice of syntactic parser is motivated by the scale of our problem; the parser must be broad-coverage and robust enough to handle a web-sized corpus. For these reasons, we rely on the C&C parser (Clark and Curran, 2004), a general-purpose CCG parser, to obtain syntactic derivations. To our knowledge, we present the first attempt to use a CCG parser trained on treebanks for grounded semantic parsing. Most previous work has induced task-specific CCG grammars (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010). An example CCG derivation is shown in Figure 4.

Semantic parses are constructed from syntactic CCG parses, with semantic composition being guided by the CCG syntactic derivation.[2] We use a neo-Davidsonian (Parsons, 1990) semantics to represent semantic parses.[3] Each word has a semantic category based on its syntactic category and part of speech. For example, the syntactic category for *directed* is $(S\backslash NP)/NP$, i.e., it takes two argument NPs and becomes S. To represent its semantic category, we use a lambda term $\lambda y \lambda x.\ \text{directed.arg1}(e,x) \wedge \text{directed.arg2}(e,y)$, where e identifies the event of *directed*, and x and y are arguments corresponding to the NPs in the syntactic category.

We obtain semantic categories automatically using the indexed syntactic categories provided by the C&C parser. The latter reveal the bindings of basic constituent categories in more complex categories. For example, in order to convert $((S\backslash NP)\backslash(S\backslash NP))/NP$ to its semantic category, we must know whether all NPs have the same referent and thus use the same variable name. The indexed category $((S_e\backslash NP_x)\backslash(S_e\backslash NP_x))/NP_y$ reveals that there are only two different NPs, x and y, and that one of them (i.e., x) is shared across two subcategories. We discuss the details of semantic category construction in the Appendix.

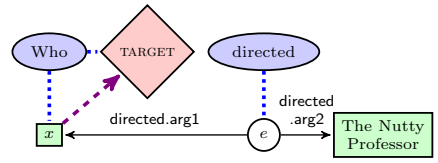
Apart from n-ary predicates representing events (mostly verbs), we also use unary predicates representing types in language (mostly common nouns and noun modifiers). For example, capital(Austin) indicates *Austin* is of type capital. Prepositions, adjectives and adverbs are represented by predicates lexicalized with their head words to provide more information (see capital.of.arg1 instead of of.arg1 in Figure 2a).

2.3 Ungrounded Semantic Graphs

We will now illustrate how we create ungrounded semantic graphs from CCG-derived semantic parses. Figure 5a displays the ungrounded graph for the sen-

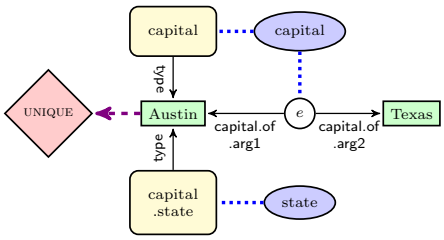
[2]See Bos et al. (2004) for a detailed introduction to semantic representation using CCG.

[3]Neo-Davidsonian semantics is a form of first-order logic that uses event identifiers (e) to connect verb predicates and their subcategorized arguments through conjunctions.



$\text{TARGET}(x) \wedge \text{directed.arg1}(e, x) \wedge$
$\text{directed.arg2}(e, \text{TheNuttyProfessor})$

(a) Who directed The Nutty Professor?



$\text{UNIQUE}(\text{Austin}) \wedge \text{capital}(\text{Austin}) \wedge \text{capital.state}(\text{Austin}) \wedge$
$\text{capital.of.arg1}(e, \text{Austin}) \wedge \text{capital.of.arg2}(e, \text{Texas})$

(b) Austin is the state capital of Texas.

Figure 6: Ungrounded graphs with math functions TARGET and UNIQUE.

tence *Cameron directed Titanic in 1997*. In order to construct ungrounded graphs topologically similar to Freebase, we define five types of nodes:

Word Nodes (Ovals) Word nodes are denoted by ovals. They represent natural language words (e.g., *directed* in Figure 5a, *capital* and *state* in Figure 6b). Word nodes are connected to other word nodes via syntactic dependencies. For readability, we do not show inter-word dependencies.

Entity Nodes (Rectangles) Entity nodes are denoted by rectangles and represent entities e.g., *Cameron* in Figure 5a. In cases where an entity is not known, we use variables e.g., x in Figure 6a. Entity variables are connected to their corresponding word nodes from which they originate by dotted links e.g., x in Figure 6a is connected to the word node *who*.

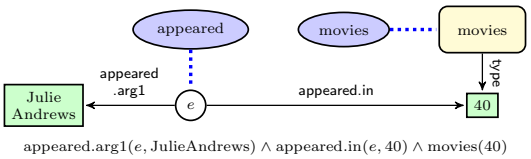
Mediator Nodes (Circles) Mediator nodes are denoted by circles and represent events in language. They connect pairs of entities which participate in an event forming a clique (see the entities *Cameron*, *Titanic* and *1997* in Figure 5a). We define an *edge* as a link that connects any two entities via a mediator. The *subedge* of an edge i.e., the link between a mediator and an entity, corresponds to the predicate denoting the event and taking the entity as its argument (e.g. directed.arg1 links e and *Cameron* in Figure 5a). Mediator nodes are connected to their corresponding word nodes from which they originate by dotted links e.g. mediators in Figure 5a are connected to word node *directed*.

Type nodes (Rounded rectangles) Type nodes are denoted by rounded rectangles. They represent unary predicates in natural language. In Figure 6b type nodes capital and capital.state are attached to *Austin* denoting *Austin* is of type capital and capital.state. Type nodes are also connected to their corresponding word nodes from which they originate by dotted links e.g. type node capital.state and word node *state* in Figure 6b.

Math nodes (Diamonds) Math nodes are denoted by diamonds. They describe functions to be applied on the nodes/subgraphs they attach to. The function TARGET attaches to the entity variable of interest. For example, the graph in Figure 6a represents the question *Who directed The Nutty Professor?*. Here, TARGET attaches to x representing the word *who*. UNIQUE attaches to the entity variable modified by the definite article *the*. In Figure 6b, UNIQUE attaches to *Austin* implying that only *Austin* satisfies the graph. Finally, COUNT attaches to entity nodes which have to be counted. For the sentence *Julie Andrews has appeared in 40 movies* in Figure 7, the KB could either link *Julie Andrews* and *40*, with type node *movies* matching the grounded type *integer*, or it could link *Julie Andrews* to each movie she acted in and the count of these different movies add to 40. In anticipation of this ambiguity, we generate two semantic parses resulting in two ungrounded graphs (see Figures 7a and 7b). We generate all possible grounded graphs corresponding to each ungrounded graph, and leave it up to the learning to decide which ones the KB prefers.

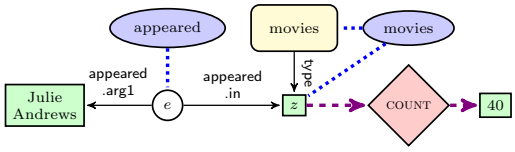
2.4 Grounded Semantic Graphs

We ground semantic graphs in Freebase by mapping edge labels to relations, type nodes to entity types, and entity nodes to Freebase entities. Math nodes remain unchanged. Though word nodes are not present in Freebase, we retain them in our grounded graphs to extract sophisticated features based on words and grounded predicates.



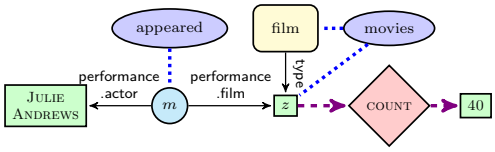
$\text{appeared.arg1}(e, \text{JulieAndrews}) \wedge \text{appeared.in}(e, 40) \wedge \text{movies}(40)$

(a) Ungrounded Graph



$\text{appeared.arg1}(e, \text{JulieAndrews}) \wedge \text{appeared.in}(e, z)$
$\wedge\ \text{movies}(z) \wedge \text{COUNT}(z, 40)$

(b) Alternate Ungrounded Graph



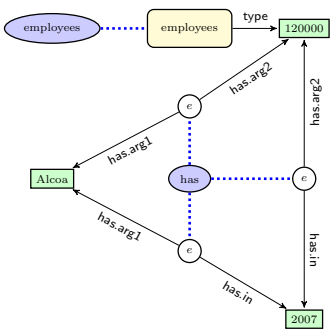
$\text{performance.actor}(m, \text{JulieAndrews}) \wedge \text{performance.film}(m, z)$
$\wedge\ \text{film}(z) \wedge \text{COUNT}(z, 40)$

(c) Grounded graph

Figure 7: Graph representations for the sentence *Julie Andrews has appeared in 40 movies.* Ungrounded graph (a) directly connects *Julie Andrews* and *40*, whereas graph (b) uses the math function COUNT. Ungrounded graph (b) and grounded graph (c) have similar topology.

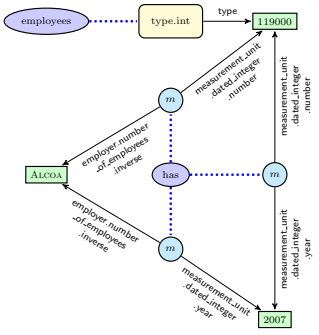
Entity nodes Previous approaches (Cai and Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013) use a manual lexicon or heuristics to ground named entities to Freebase entities. Fortunately, CLUEWEB09 sentences have been automatically annotated with Freebase entities, so we use these annotations to ground proper names to Freebase entities (denoted by uppercase words) e.g., *Cameron* in Figure 5a is grounded to Freebase entity CAMERON in Figure 5b. Common nouns like *movies* (see Figure 7b) are left as variables to be instantiated by the entities satisfying the graph.

Type nodes Type nodes are grounded to Freebase entity types. Type nodes capital and capital.state in Figure 6b are grounded to all possible types of Austin (e.g., location.city, location.capital_city, book.book_subject, broadcast.genre). In cases where entity nodes are not grounded, (e.g., z in Figure 7b),



$\text{has.arg1}(e, \text{Alcoa}) \wedge \text{has.arg2}(e, 120000)$
$\wedge\ \text{has.in}(e, 2007) \wedge\ \text{employees}(120000)$

(a) Ungrounded Graph



$\text{employer.number\_of\_employees.inverse}(m, \text{ALCOA}) \wedge$
$\text{measurement\_unit.dated\_integer.number}(m, 119000)$
$\wedge\ \text{measurement\_unit.dated\_integer.year}(m, 2007) \wedge$
$\text{type.int}(119000)$

(b) Grounded Graph

Figure 8: Graph representations for *Alcoa has 120000 employees in 2007.*

we use an automatically constructed lexicon which maps ungrounded types to grounded ones (see Section 4.2 for details).

Edges An edge between two entities is grounded using all edges linking the two entities in the knowledge graph. For example, to ground the edge between *Titanic* and *Cameron* in Figure 5, we use the following edges linking TITANIC and CAMERON in Freebase: (film.directed_by.arg1, film.directed_by.arg2), (film.produced_by.arg1, film.produced_by.arg2). If only one entity is grounded, we use all possible edges from this grounded entity. If no entity is grounded, we use a mapping lexicon which is automatically created as described in Section 4.2. Given an ungrounded graph with n edges, there are $O((k+1)^n)$ possible grounded graphs, with k being the grounded edges in the knowledge graph for each ungrounded edge together with an additional empty (no) edge.

Mediator nodes In an ungrounded graph, mediator nodes represent semantic event identifiers. In the grounded graph, they represent Freebase fact identifiers. Fact identifiers help distinguish if neighboring edges belong to a single complex fact, which may or may not be coextensive with an ungrounded event. In Figure 8a, the edges corresponding to the event identifier e are grounded to a single complex fact in Figure 8b, with the fact identifier m. However, in Figure 5a, the edges of the ungrounded event e are grounded to different Freebase facts, distinguished in Figure 5b by the identifiers m and n. Furthermore,

the edge in 5a between CAMERON and 1997 is not grounded in 5b, since no Freebase edge exists between the two entities.

We convert grounded graphs to SPARQL queries, but for readability we only show logical expressions. The conversion is deterministic and is exactly the inverse of the semantic parse to graph conversion (Section 2.3). Wherever a node/edge is instantiated with a grounded entity/type/relation in Freebase, we use them in the grounded parse (e.g., type node capital.state in Figure 6b becomes location.capital_city). Math function TARGET is useful in retrieving instantiations of entity variables of interest (see Figure 6a).

3 Learning

A natural language sentence may give rise to several grounded graphs. But only one (or a few) of them will be a faithful representation of the sentence in Freebase. We next describe our algorithm for finding the best Freebase graph for a given sentence, our learning model, and the features it uses.

3.1 Algorithm

Freebase has a large number of relations and entities, and as a result there are many possible grounded graphs g for each ungrounded graph u. We construct and score graphs incrementally, traversing each node in the ungrounded graph and matching its edges and types in Freebase. Given a NL sentence s, we construct from its CCG syntactic derivation all corresponding ungrounded graphs u. Using a beam search procedure (described in Section 4.2), we find the best scoring graphs $(\hat{g},\hat{u})$, maximizing over different graph configurations (g,u) of s:

$$(\hat{g},\hat{u}) = \arg\max_{g,u} \Phi(g,u,s,\mathcal{KB})\cdot\theta \qquad (1)$$

We define the score of $(\hat{g},\hat{u})$ as the dot product between a high dimensional feature representation $\Phi = (\Phi_1,\ldots\Phi_m)$ and a weight vector θ (see Section 3.3 for details on the features we employ).

We estimate the weights θ using the averaged structured perceptron algorithm (Collins, 2002). As shown in Algorithm 1, the perceptron makes several passes over sentences, and in each iteration it computes the best scoring $(\hat{g},\hat{u})$ among the candidate graphs for a given sentence. In line 6, the algorithm updates θ with the difference (if any) between the feature representations of the best scoring graph $(\hat{g},\hat{u})$ and the *gold standard* graph (g^+,u^+). The goal of the algorithm is to rank gold standard graphs higher than the any other graphs. The final weight vector θ is the average of weight vectors over T iterations and N sentences. This averaging procedure avoids overfitting and produces more stable results (Collins, 2002).

Algorithm 1: Averaged Structured Perceptron

Input: Training sentences: $\{s_i\}_{i=1}^N$

1 $\theta \leftarrow 0$
2 **for** $t \leftarrow 1\ldots T$ **do**
3 **for** $i \leftarrow 1\ldots N$ **do**
4 $(\hat{g}_i,\hat{u}_i) = \arg\max_{g_i,u_i} \Phi(g_i,u_i,s_i,\mathcal{KB})\cdot\theta$
5 **if** $(u_i^+,g_i^+) \neq (\hat{u}_i,\hat{g}_i)$ **then**
6 $\theta \leftarrow \theta + \Phi(g_i^+,u_i^+,s_i,\mathcal{KB}) - \Phi(\hat{g}_i,\hat{u}_i,s_i,\mathcal{KB})$
7 **return** $\frac{1}{T}\sum_{t=i}^{T}\frac{1}{N}\sum_{i=1}^{N}\theta^i{}_t$

As we do not make use of question-answer pairs or manual annotations of sentences, gold standard graphs (g^+,u^+) are not available. In the following, we explain how we approximate them by relying on graph denotations as a form of weak supervision.

3.2 Selecting Surrogate Gold Graphs

Let u be an ungrounded semantic graph of s. We select an entity E in u, replace it with a variable x, and make it a target node. Let u^+ represent the resulting ungrounded graph. Next, we obtain all grounded graphs g^+ which correspond to u^+ such that the denotations $[\![u^+]\!]_{\mathcal{KB}} = [\![g^+]\!]_{\mathcal{NL}}$. We use these *surrogate* graphs g^+as gold standard, and the pairs (u^+,g^+) for model training. There is considerable latitude in choosing which entity E to replace. This can be done randomly, according to entity frequency, or some other criterion. We found that substituting the entity with the most connections to other entities in the sentence works well in practice. All the entities that can replace x in u^+ to constitute a valid fact in Freebase will be the denotation of u^+, $[\![u^+]\!]_{\mathcal{NL}}$. While it is straightforward to compute $[\![g^+]\!]_{\mathcal{KB}}$, it is hard to compute $[\![u^+]\!]_{\mathcal{NL}}$ because of the mismatch between our natural language semantic language and the Freebase query language. To ensure that graphs u^+ and g^+ have the same denotations, we impose the following constraints:

Constraint 1 If the math function UNIQUE is attached to the entity being replaced in the ungrounded graph, we assume the denotation of u^+ contains only that entity. For example, in Figure 2b, we replace *Austin* by x, and thus assume $[\![u^+]\!]_{\mathcal{NL}} = \{\text{AUSTIN}\}$.[4] Any grounded graph which results in $[\![g^+]\!]_{\mathcal{KB}} = \{\text{AUSTIN}\}$ will be considered a surrogate gold graph. This allows us to learn entailment relations, e.g., *capital.of* should be grounded to location.capital (left hand-side graph in Figure 2d) and not to location.containedby which results in all locations in *Texas* (right hand-side graph in Figure 2d).

Constraint 2 If the target entity node is a number, we select the Freebase graphs with denotation close to this number. For example, in Figure 8a if 120,000 is replaced by x, and we assume $[\![u^+]\!]_{\mathcal{NL}} = \{120{,}000\}$. However, the grounded graph 8b retrieves $[\![g^+]\!]_{\mathcal{KB}} = \{119{,}000\}$. We treat this as correct if $\frac{\beta}{\gamma} \in [0.9, 1.1]$ where $\beta \in [\![u^+]\!]_{\mathcal{NL}}$ and $\gamma \in [\![g^+]\!]_{\mathcal{KB}}$. Integers can either occur directly in relation with an entity as in Figure 8b, or must be enumerated as in Figure 7c.

Constraint 3 If the target entity node is a date, we select the grounded graph which results in the smallest set containing the date based on the intuition that most sentences in the data describe specific rather than general events.

Constraint 4 If none of the above constraints apply to the target entity E, we know $E \in [\![u^+]\!]_{\mathcal{NL}}$, and hence we select the grounded graphs which satisfy $E \in [\![g^+]\!]_{\mathcal{KB}}$ as surrogate gold graphs.

3.3 Features

Our feature vector $\Phi(g, u, s, \mathcal{KB})$ denotes the features extracted from a sentence s and its corresponding graphs u and g with respect to a knowledge base $\mathcal{KB}$. The elements of the vector $(\phi_1, \phi_2, \ldots)$ take integer values denoting the number of times a feature appeared. We devised the following broad feature classes:

Lexical alignments Since ungrounded graphs are similar in topology to grounded graphs, we extract ungrounded and grounded edge and type alignments. So, from graphs 5a and 5b, we obtain the edge alignment ϕ_{edge}(directed.arg1, directed.arg2, film.directed_by.arg2, film.directed_by.arg1) and the subedge alignments ϕ_{edge}(directed.arg1, film.directed_by.arg2) and ϕ_{edge}(directed.arg2, film.directed_by.arg1). In a similar fashion we extract type alignments (e.g., ϕ_{type}(capital,location.city)).

Contextual features In addition to lexical alignments, we also use contextual features which essentially record words or word combinations surrounding grounded edge labels. Feature ϕ_{event} records an event word and its grounded predicates (e.g., in Figure 7c we extract features ϕ_{event}(appear, performance.film) and ϕ_{event}(appear, performance.actor). Feature ϕ_{arg} records a predicate and its argument words (e.g., ϕ_{arg}(performance.film, movie) in Figure 7c). Word combination features are extracted from the parser's dependency output. The feature ϕ_{dep} records a predicate and the dependencies of its event word (e.g., from the grounded version of Figure 6b we extract features ϕ_{dep}(location.state.capital.arg1, capital, state) and ϕ_{dep}(location.state.capital.arg2, capital, state)). Using such features, we are able to handle multiword predicates.

Lexical similarity We count the number of word stems[5] shared by grounded and ungrounded edge labels e.g., in Figure 5 directed.arg1 and film.directed_by.arg2 have one stem overlap (ignoring the argument labels arg1 and arg2). For a grounded graph, we compute ϕ_{stem}, the aggregate stem overlap count over all its grounded and ungrounded edge labels. We did not incorporate WordNet/Wiktionary-based lexical similarity features but these were found fruitful in Kwiatkowski et al. (2013). We also have a feature for stem overlap count between the grounded edge labels and the context words.

Graph connectivity features These features penalize graphs with non-standard topologies. For example, we do not want a final graph with no edges. The feature value $\phi_{hasEdge}$ is one if there exists at least one edge in the graph. We also have a feature $\phi_{nodeCount}$ for counting the number of connected

[4] We also remove UNIQUE attached to x to exactly mimic the test time setting.

[5] We use the Porter stemmer.

Domain	#Rels	#Types	#Triples	#Train	#Free917	#WebQ
business	226	102	23m	30k	46	49
film	113	75	42m	13k	49	91
people	85	59	68m	56k	29	430
all*	411	210	120m	99k	124	570

Table 1: Domain-specific Freebase statistics (*some relations/types/triples are shared across domains); number of training CLUEWEB09 sentences; number of test questions in FREE917 and WEBQUESTIONS.

nodes in the graph. Finally, feature ϕ_{colloc} captures the collocation of grounded edges (e.g., edges belonging to a single complex fact are likely to co-occur; see Figure 8b).

4 Experimental Setup

In this section we present our experimental set-up for assessing the performance of the semantic parser described above. We present the datasets on which our model was trained and tested, discuss implementation details, and briefly introduce the models used for comparison with our approach.

4.1 Data

We evaluated our approach on the FREE917 (Cai and Yates, 2013) and WEBQUESTIONS (Berant et al., 2013) datasets. FREE917 consists of 917 questions and their meaning representations (written in a variant of lambda calculus) which we, however, do not use. The dataset represents 81 domains covering 635 Freebase relations, with most domains containing fewer than 10 questions. We report results on three domains, namely `film`, `business`, and `people` as these are relatively large in both FREE917 and Freebase. WEBQUESTIONS consists of 5,810 question-answer pairs, 2,780 of which are reserved for testing. Our experiments used a subset of WEBQUESTIONS representing the three target domains. We extracted domain-specific queries semi-automatically by identifying question-answer pairs with entities in target domain relations. In both datasets, named entities were disambiguated to Freebase entities with a named entity lexicon.[6]

Table 1 presents descriptive statistics for each domain. Evaluating on all domains in Freebase would generate a very large number of queries for which denotations would have to be computed (the number of queries is linear in the number of domains and the size of training data). Our system loads Freebase using Virtuoso[7] and queries it with SPARQL. Virtuoso is slow in dealing with millions of queries indexed on the entire Freebase, and is the only reason we did not work with the complete Freebase.

4.2 Implementation

To train our model, we extracted sentences from CLUEWEB09 which contain at least two entities associated with a relation in Freebase, and have an edge between them in the ungrounded graph. These were further filtered so as to remove sentences which do not yield at least one semantic parse without an uninstantiated entity variable. For example, the sentence *Avatar is directed by Cameron* would be used for training, whereas *Avatar directed by Cameron received a critical review* wouldn't. In the latter case, any semantic parse will have an uninstantiated entity variable for *review*. Table 1 (Train) shows the number of sentences we obtained.

In order to train our semantic parser, we initialized the alignment and type features (ϕ_{edge} and ϕ_{type}, respectively) with the alignment lexicon weights. These weights are computed as follows. Let $count(r', r)$ denote the number of pairs of entities which are linked with edge r' in Freebase and edge r in CLUEWEB09 sentences. We then estimate the probability distribution $P(r'/r) = \frac{count(r',r)}{\sum_i count(r'_i,r)}$. Analogously, we created a type alignment lexicon. The counts were collected from CLUEWEB09 sentences containing pairs of entities linked with an edge in Freebase (`business` 390k, `film` 130k, and `people` 490k). Contextual features were initialized to -1 since most word contexts and grounded predicates/types do not appear together. All other features were set to 0.

We used a beam-search algorithm to convert ungrounded graphs to grounded ones. The edges and types of each ungrounded graph are placed in a priority queue. Priority is based on edge/type tf-idf scores collected over CLUEWEB09. At each step, we pop an element from the queue and ground it in Freebase. We rank the resulting grounded graphs us-

[6] FREE917 comes with a named entity lexicon. For WEBQUESTIONS we hand-coded this lexicon.

[7] `http://virtuoso.openlinksw.com`

ing the perceptron model, and pick the n-best ones, where n is the beam size. We continue until the queue is empty. In our experiments we used a beam size of 100. We trained a single model for all the domains combined together. We ran the perceptron for 20 iterations (around 5–10 million queries). At each training iteration we used 6,000 randomly selected sentences from the training corpus.

4.3 Comparison Systems

We compared our graph-based semantic parser (henceforth GRAPHPARSER) against two state-of-the-art systems both of which are open-domain and work with Freebase. The semantic parser developed by Kwiatkowski et al. (2013) (henceforth KCAZ13) is learned from question-answer pairs and follows a two-stage procedure: first, a natural language sentence is converted to a domain-independent semantic parse and then grounded onto Freebase using a set of logical-type equivalent operators. The operators explore possible ways sentential meaning could be expressed in Freebase and essentially transform logical form to match the target ontology. Our approach also has two steps (i.e., we first generate multiple ungrounded graphs and then ground them to different Freebase graphs). We do not use operators to perform structure matching, rather we create multiple graphs and leave it up to the learner to find an appropriate grounding using a rich feature space. To give a specific example, their operator *literal to constant* is equivalent to having named entities for larger text chunks in our case. Their operator *split literal* explores different edge possibilities in an event whereas we start with a clique and remove unwanted edges. Our approach has (almost) similar expressive power but is conceptually simpler.

Our second comparison system was the semantic parser of Berant and Liang (2014) (henceforth PARASEMPRE) which also uses QA pairs for training and makes use of paraphrasing. Given an input NL sentence, they first construct a set of logical forms based on hand-coded rules, and then generate sentences from each logical form (using generation templates and a lexicon). Pairs of logical forms and natural language are finally scored using a paraphrase model consisting of two components. An association model determines whether they contain phrase pairs likely to be paraphrases and a vector space model assigns a vector representation for each sentence, and learns a scoring function that ranks paraphrase candidates. Our semantic parser employs a graph-based representation as a means of handling the mismatch between natural language, whereas PARASEMPRE opts for a text-based one through paraphrasing.

Finally, we compared our semantic parser against a baseline which is based on graphs but employs no learning. The baseline converts an ungrounded graph to a grounded one by replacing each ungrounded edge/type with the highest weighted grounded label creating a *maximum weighted graph*, henceforth MWG. Both GRAPHPARSER and the baseline use the same alignment lexicon (a weighted mapping from ungrounded to grounded labels).

System	Prec	Rec	F1
MWG	52.6	49.1	50.8
KCAZ13	72.6	66.1	69.2
GRAPHPARSER	81.9	76.6	79.2

Table 2: Experimental results on FREE917.

5 Results

Table 2 summarizes our results on FREE917. As described earlier, we evaluated GRAPHPARSER on a subset of the dataset representing three domains (`business`, `film`, and `people`). Since this subset contains a relatively small number of instances (124 in total), we performed 10-fold cross validation with 9 folds as development data[8], and one fold as test data. We report results averaged over all test folds. With respect to KCAZ13, we present results with their cross-domain trained models, where training data from multiple domains is used to test foreign domains.[9] KCAZ13 used generic features like string similarity and knowledge base features which apply across domains and do not require in-domain training data. We do not report results with PARASEMPRE as the small number of training instances would put their method at a disadvantage. We treat a predicted query as correct if its denota-

[8]The development data is only used for model selection and for determining the optimal training iteration.

[9]We are grateful to Tom Kwiatkowski for supplying us with the output of their system.

Features	FREE917	WEBQ
All	79.2	41.4
−Contextual	73.3	42.6
−Alignment	66.7	34.8
−Connectivity	65.0	36.6
−Similarity	62.5	35.0

Table 3: GRAPHPARSER ablation results on FREE917 and WEBQUESTIONS development set.

System	Prec	Rec	F1
MWG	39.4	34.0	36.5
PARASEMPRE	37.5	37.5	37.5
GRAPHPARSER	41.9	37.0	39.3
GRAPHPARSER + PARA	44.7	38.4	41.3

Table 4: Experimental results on WEBQUESTIONS.

tion is exactly equal to the denotation of the manually annotated gold query.

As can be seen, GRAPHPARSER outperforms KCAZ13 and the MWG baseline by a wide margin. This is an encouraging result bearing in mind that our model does not use question-answer pairs. We should also point out that our domain relation set is larger compared to KCAZ13. We do not prune any of the relations in Freebase, whereas KCAZ13 use only 112 relations and 83 types from our three domains (see Table 1). We further performed a feature ablation study to examine the contribution of different feature classes. As shown in Table 3, the most important features are those based on lexical similarity, as also observed in KCAZ13. Graph connectivity and lexical alignments are equally important (these features are absent from KCAZ13). Contextual features are not very helpful over and above alignment features which also encode contextual information. Overall, generic features like lexical similarity are helpful only to a certain extent; the performance of GRAPHPARSER improves considerably when additional graph-related features are taken into account.

We also analyzed the errors GRAPHPARSER makes. 25% of these are caused by the C&C parser and are cases where it either returns no syntactic analysis or a wrong one. 19% of the errors are due to Freebase inconsistencies. For example, our system answered the question *How many stores are in Nittany mall?* with 65 using the relation shopping_center.number_of_stores whereas the gold standard provides the answer 25 counting all stores using the relation shopping_center.store. Around 15% of errors include structural mismatches between natural language and Freebase; for the question *Who is the president of Gap Inc?*, our method grounds *president* to a grounded type whereas in Freebase it is represented as a relation employment.job.title. The remaining errors are miscellaneous. For example, the question *What are some films on Antarctica?* receives two interpretations, i.e., movies filmed in Antarctica or movies with Antarctica as their subject.

We next discuss our results on WEBQUESTIONS. PARASEMPRE was trained with 1,115 QA pairs (corresponding to our target domains) together with question paraphrases obtained from the PARALEX corpus (Fader et al., 2013).[10] While training PARASEMPRE, out-of-domain Freebase relations and types were removed. Both GRAPHPARSER and PARASEMPRE were tested on the same set of 570 in-domain QA pairs with exact answer match as the evaluation criterion. For development purposes, GRAPHPARSER uses 200 QA pairs. Table 4 displays our results. We observe that GRAPHPARSER obtains a higher F1 against MWG and PARASEMPRE. Differences in performance among these systems are less pronounced compared to FREE917. This is for a good reason. WEBQUESTIONS is a challenging dataset, created by non-experts. The questions are not tailored to Freebase in any way, they are more varied and display a wider vocabulary. As a result the mismatch between natural language and Freebase is greater and the semantic parsing task harder.

Error analysis further revealed that parsing errors are responsible for 13% of the questions GRAPHPARSER fails to answer. Another cause of errors is mismatches between natural language and Freebase. Around 7% of the questions are of the type *Where did X come from?*, and our model answers with the individual's nationality, whereas annotators provide the birthplace (city/town/village) as the right answer. Moreover, 8% of the questions are of the type *What does X do?*, which the annotators answer with the individual's profession. In natural language, we rarely attest constructions

[10] We used the SEMPRE package (http://www-nlp.stanford.edu/software/sempre/) which does not use any hand-coded entity disambiguation lexicon.

like *X does dentist/researcher/actor*. The proposed framework assumes that Freebase and natural language are somewhat isomorphic, which is not always true. An obvious future direction would be to paraphrase the questions so as to increase the number of grounded and ungrounded graphs. As an illustration, we rewrote questions like *Where did X come from* to *What is X's birth place*, and *What did X do* to *What is X's profession* and evaluated our model GRAPHPARSER + PARA. As shown in Table 4, even simple paraphrasing can boost performance.

Finally, Table 3 (third column) examines the contribution of different features on the WEBQUESTIONS development dataset. Interestingly, we observe that contextual features are not useful and in fact slightly harm performance. We hypothesize that this is due to the higher degree of mismatch between natural language and Freebase in this dataset. Features based on similarity, graph connectivity, and lexical alignments are more robust and generally useful across datasets.

6 Discussion

In this paper, we introduce a new semantic parsing approach for Freebase. A key idea in our work is to exploit the structural and conceptual similarities between natural language and Freebase through a common graph-based representation. We formalize semantic parsing as a graph matching problem and learn a semantic parser without using annotated question-answer pairs. We have shown how to obtain graph representations from the output of a CCG parser and subsequently learn their correspondence to Freebase using a rich feature set and their denotations as a form of weak supervision. Our parser yields state-of-the art performance on three large Freebase domains and is not limited to question answering. We can create semantic parses for any type of NL sentences.

Our work brings together several strands of research. Graph-based representations of sentential meaning have recently gained some attention in the literature (Banarescu et al., 2013), and attempts to map sentences to semantic graphs have met with good inter-annotator agreement. Our work is also closely related to Kwiatkowski et al. (2013) and Berant and Liang (2014) who present open-domain semantic parsers based on Freebase and trained on QA pairs. Despite differences in formulation and model structure, both approaches have explicit mechanisms for handling the mismatch between natural language and the KB (e.g., using logical-type equivalent operators or paraphrases). The mismatch is handled implicitly in our case via our graphical representation which allows for the incorporation of all manner of powerful features. More generally, our method is based on the assumption that linguistic structure has a correspondence to Freebase structure which does not always hold (e.g., in *Who is the grandmother of Prince William?*, *grandmother* is not directly expressed as a relation in Freebase). Additionally, our model fails when questions are too short without any lexical clues (e.g., *What did Charles Darwin do?*). Supervision from annotated data or paraphrasing could improve performance in such cases. In the future, we plan to explore cluster-based semantics (Lewis and Steedman, 2013) to increase the robustness on unseen NL predicates.

Our work joins others in exploiting the connections between natural language and open-domain knowledge bases. Recent approaches in relation extraction use distant supervision from a knowledge base to predict grounded relations between two target entities (Mintz et al., 2009; Hoffmann et al., 2011; Riedel et al., 2013). During learning, they aggregate sentences containing the target entities, ignoring richer contextual information. In contrast, we learn from each individual sentence taking into account all entities present, their relations, and how they interact. Krishnamurthy and Mitchell (2012) formalize semantic parsing as a distantly supervised relation extraction problem combined with a manually specified grammar to guide semantic parse composition.

Finally, our approach learns a model of semantics guided by denotations as a form of weak supervision. Beyond semantic parsing (Artzi and Zettlemoyer, 2013; Liang et al., 2011; Clarke et al., 2010), feedback-based learning has been previously used for interpreting and following NL instructions (Branavan et al., 2009; Chen and Mooney, 2011), playing computer games (Branavan et al., 2012), and grounding language in the physical world (Krishnamurthy and Kollar, 2013; Matuszek et al., 2012).

Lemma	POS	Semantic Class	Semantic Category
*	VB*, IN, TO, POS	EVENT	directed : $(S_e\backslash NP_x{<}1{>})/NP_y{<}2{>}$: $\lambda Q\lambda P\lambda e.\exists x\exists y.$ directed.arg1$(e,x) \wedge$ directed.arg2$(e,y) \wedge P(x) \wedge Q(y)$
*	NN, NNS	TYPE	movie : NP : $\lambda x.$movie(x)
*	NNP*, PRP*	ENTITY	Obama : NP : $\lambda x.equal(x,$Obama$)$
*	RB*	EVENTMOD	annually : $S_e\backslash S_e$: $\lambda P\lambda e.lex_e.$annually$(e) \wedge P(e)$
*	JJ*	TYPEMOD	state : NP_x/NP_x : $\lambda P\lambda x.lex_x.$state$(x) \wedge P(x)$
be	*	COPULA	be: $(S_y\backslash NP_x)/NP_y$: $\lambda Q\lambda P\lambda y.\exists x.lex_y(x) \wedge P(x) \wedge Q(y)$
the	*	UNIQUE	the : NP_x/NP_x : $\lambda P\lambda x.$UNIQUE$(x) \wedge P(x)$
*	CD	COUNT	twenty : N_x/N_x : $\lambda P\lambda x.$COUNT$(x,20) \wedge P(x)$ twenty : N_x/N_x : $\lambda P\lambda x.equal(x,20) \wedge P(x)$
not, n't	*	NEGATION	not : $(S_e\backslash NP_x)/(S_e\backslash NP_x)$: $\lambda P\lambda Q\lambda e.\exists x.$NEGATION$(e) \wedge P(x,e) \wedge Q(x)$
no	*	COMPLEMENT	no : NP_x/N_x : $\lambda P\lambda x.$COMPLEMENT$(x) \wedge P(x)$
*	WDT, WP*, WRB	QUESTION	what : $S[wq]_e/(S[dcl]_e\backslash NP_x)$: $\lambda P\lambda e.\exists x.$TARGET$(x) \wedge P(x,e)$
*	WDT, WP*, WRB	CLOSED	which : $(NP_x\backslash NP_x)/(S[dcl]_e\backslash NP_x)$: $\lambda P\lambda Q\lambda x.\exists e.P(x,e) \wedge Q(x)$

Table 5: Rules used to classify words into semantic classes. * represents a wild card expression which matches anything. lex_x denotes the lexicalised form of x e.g., when $state : NP_x/NP_x : \lambda P\lambda x.lex_x.state(x) \wedge P(x)$ is applied to $capital : NP : \lambda y.capital(y)$, the lexicalised form of x becomes $capital$, and therefore the predicate $lex_x.state$ becomes $capital.state$. The resulting semantic parse after application is $\lambda x.capital.state(x) \wedge capital(x)$.

Appendix

We use a handful of rules to divide words into semantic classes. Based on a word's semantic class and indexed syntactic category, we construct its semantic category automatically. For example, *directed* is a member of the EVENT class, and its indexed syntactic category is $((S_e\backslash NP_x{<}1{>})/NP_y{<}2{>})$ (here, $<1>$ and $<2>$ indicate that x and y are the first and second arguments of e). We then generate its semantic category as $\lambda Q\lambda P\lambda e.\exists x\exists y.directed.arg1(e,x) \wedge directed.arg2(e,y) \wedge P(x) \wedge Q(y)$. Please refer to Appendix B of Clark and Curran (2007) for a list of their indexed syntactic categories.

The rules are described in Table 5. Syntactic categories are not shown for the sake of brevity. Most rules will match any syntactic category. Exceptions are copula-related rules (see *be* in the sixth row) which apply only to the syntactic category $(S\backslash NP)/NP$, and rules pertaining to *wh*-words (see the last two rows in the table). When more than one rule apply, we end up with multiple semantic parses. There are a few cases like passives, question words, and prepositional phrases where we modified the original indexed categories for better interpretation of the semantics (these are not displayed here). We also handle non-standard CCG operators involving unary and binary rules as described in Appendix A of Clark and Curran (2007).

Acknowledgements

We are grateful to the anonymous reviewers for their valuable feedback on an earlier version of this paper. Thanks to Mike Lewis and the members of ILCC for helpful discussions and comments. We acknowledge the support of EU ERC Advanced Fellowship 249520 GRAMPLUS and EU IST Cognitive Systems IP EC-FP7-270273 "Xperience".

References

Artzi, Yoav and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical*

Methods in Natural Language Processing. Edinburgh, Scotland, pages 421–432.

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

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

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

Berant, Jonathan and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland, USA, pages 1415–1425.

Bos, Johan, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *Proceedings of Coling 2004*. Geneva, Switzerland, pages 1240–1246.

Branavan, S.R.K., Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. 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*. Suntec, Singapore, pages 82–90.

Branavan, S.R.K., Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*. Jeju Island, Korea, pages 126–135.

Cai, Qingqing and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 423–433.

Chen, David L. and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*. San Francisco, California, pages 859–865.

Clark, Stephen and James R Curran. 2004. Parsing the wsj using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Barcelona, Spain, pages 103–111.

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

Clark, Stephen, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. pages 327–334.

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

Collins, Michael. 2002. Discriminative training methods for Hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Philadelphia, Pennsylvania, pages 1–8.

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

Gabrilovich, Evgeniy, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0).

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

Goldwasser, Dan and Dan Roth. 2011. Learning from natural instructions. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. Barcelona, Spain, pages 1794–1800.

Hoffmann, Raphael, Congle Zhang, Xiao Ling, Luke S Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA, pages 541–550.

Krishnamurthy, Jayant and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transations of the Association for Computational Linguistics* 1(1):193–206.

Krishnamurthy, Jayant and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea, pages 754–765.

Kwiatkowski, Tom, 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*. Seattle, Washington, USA, pages 1545–1556.

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

Lewis, Mike and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics* 1:179–192.

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

Matuszek, Cynthia, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. In *Proceedings of the 29th International Conference on Machine Learning*. Edinburgh, Scotland, pages 1671–1678.

Mintz, Mike, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. pages 1003–1011.

Parsons, Terence. 1990. *Events in the Semantics of English*. MIT Press, Cambridge, MA.

Poon, Hoifung. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 933–943.

Riedel, Sebastian, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia, pages 74–84.

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

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

Yao, Xuchen and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of*

the 52nd Annual Meeting of the Association for Computational Linguistics. Baltimore, Maryland, USA, pages 956–966.

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

Zettlemoyer, Luke 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*. Prague, Czech Republic, pages 678–687.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of 21st Conference in Uncertainilty in Artificial Intelligence*. Edinburgh, Scotland, pages 658–666.