Open-Vocabulary Semantic Parsing
with both Distributional Statistics and Formal Knowledge

Matt Gardner and Jayant Krishnamurthy
Allen Institute for Artificial Intelligence
Seattle, Washington, USA
{mattg,jayantk}@allenai.org

Abstract
Semantic parsers map language onto executable statements in a fixed schema. This mapping allows them to effectively leverage the information contained in large, formal knowledge bases (e.g., Freebase) to answer questions, but it is also fundamentally limiting—semantic parsers can only represent language that falls within their manually-produced schema. Recently proposed methods for open vocabulary semantic parsing overcome this limitation by learning execution models for arbitrary language. However, all prior approaches to open vocabulary semantic parsing are purely distributional, making no use of any underlying knowledge base. We show how to combine the benefits of both of these approaches by incorporating knowledge base information into open vocabulary semantic parsing models, improving mean average precision on an open-domain natural language query task by more than 120%.

1 Introduction
Semantic parsing is the task of mapping a phrase in natural language onto formal statements in some fixed schema, which can then be executed against a database (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005). For example, the phrase “Who is the president of the United States?” might be mapped onto the formal statement $\lambda(x)$.GOVERNMENT/PRESIDENT_OF(x, USA), which could then be executed against a knowledge base (KB) such as Freebase (Bollacker et al., 2008) to return BARACK OBAMA. By mapping phrases to executable statements, semantic parsers can leverage large, curated sources of knowledge to answer questions (Berant et al., 2013).

This benefit comes with an inherent limitation, however—semantic parsers can only produce executable statements within their manually-produced schema. There is no query against Freebase that can answer questions such as “Who are the Democratic front-runners in the US election?”, as Freebase does not encode any information about front-runners. Semantic parsers trained for Freebase fail on these kinds of questions.

To overcome this limitation, recent work has proposed methods for open vocabulary semantic parsing, where language is mapped onto semi-formal statements with predicates derived directly from the language itself (Lewis and Steedman, 2013; Krishnamurthy and Mitchell, 2015). For instance, the question above might be mapped to $\lambda(x)$.president_of(x, USA). This query is not executable against any knowledge base, however, and so open vocabulary semantic parsers must learn execution models for the predicates found in the text. They do this with a distributional approach similar to word embedding methods, giving them broad coverage, but lacking access to the large, curated KBs available to traditional semantic parsers.

Prior work in semantic parsing, then, has either had direct access to the information in a knowledge base, or broad coverage over all of natural language using distributional techniques, but not both.

In this work, we show how to combine these two approaches by incorporating KB information into open vocabulary semantic parsing models. Our key
### Table: Candidate entities and Probability that entity is in denotation

| Input Text | Logical Form | Probability that entity is in denotation |
|------------|--------------|----------------------------------------|
| Italian architect ___ | $\lambda x.\text{architect}(x) \land \text{architect} \_N/(\text{ITALY}, x)$ | $p(\text{architect}(\text{PALLADIO})) \times p(\text{architect}_N/\text{ITALY, PALLADIO}) = 0.79$ |
| ... | ... | $p(\text{architect}(\text{OBAMA})) \times p(\text{architect}_N/\text{ITALY, OBAMA}) = 0.01$ |

**Category models:**

- Predicate parameters for `architect`:
  - $\theta$: $[.2, -.6, . . .]$  
  - $\omega$:  
    - TYPE:ARCHITECT $\rightarrow .52$  
    - TYPE:DESIGNER $\rightarrow .32$  
    - NATIONALITY:ITALY $\rightarrow .20$  
  - Entity parameters for PALLADIO:
    - $\phi$: $[.15, -.8, . . .]$  
    - $\psi$: TYPE:ARCHITECT $\rightarrow 1$  
    - NATIONALITY:ITALY $\rightarrow 1$  

- $p(\text{architect}(\text{PALLADIO})) = \sigma(\theta^T \phi + \omega^T \psi)$

**Relation models:**

- Predicate parameters for $\text{architect}_N/\text{ITALY}$:
  - $\theta$: $[-.9, .1, . . .]$  
  - $\omega$:  
    - /PERSON/NATIONALITY$^{-1} \rightarrow .29$  
    - /STRUCTURE/ARCHITECT $\rightarrow .11$  
    - /PERSON/ETHNICITY$^{-1} \rightarrow .05$  
  - Entity pair parameters for (ITALY, PALLADIO):
    - $\phi$: $[-.8, 2, . . .]$  
    - $\psi$: /PERSON/NATIONALITY$^{-1} \rightarrow 1$  
    - /STRUCTURE/ARCHITECT $\rightarrow 0$  

- $p(\text{architect}_N/\text{ITALY, PALLADIO}) = \sigma(\theta^T \phi + \omega^T \psi)$

*Figure 1: Overview of the components of our model. Given an input text, we use a CCG parser and an entity linker to produce a logical form with predicates derived from the text. For each predicate, we learn a distributional vector $\theta$, as well as weights $\omega$ associated with a set of selected Freebase queries. For each entity and entity pair, we learn a distributional vector $\phi$, and we extract a binary feature vector $\psi$ from Freebase, indicating whether each entity or entity pair is in the set returned by the selected Freebase queries. These models are combined to assign probabilities to candidate entities.*

Insight is that the executable statements produced by traditional semantic parsers can be converted into features that can be added to the learned execution models of open vocabulary semantic parsers. This allows open vocabulary models to use the KB fact `/GOVERNMENT/PRESIDENT_OF(BarackObama, USA)` when scoring `president_of(BarackObama, USA)`, without requiring the model to map the language onto a single formal statement. Crucially, this also allows the model to use these KB facts even when they only provide partial information about the language being modeled. Knowing that an entity is a politician, e.g., is very helpful information for deciding whether that entity is a front-runner. Our approach, outlined in Figure 1, effectively learns the meaning of a word as a distributional vector plus a weighted combination of Freebase queries, a considerably more expressive representation than both traditional and open vocabulary semantic parsers.

Additionally, to allow open vocabulary semantic parsing models to make better use of the KB information, we present improved logical forms compared to those used in prior work, and a simple technique for generating candidate entities from the KB. We demonstrate our approach on the task of answering open-domain fill-in-the-blank natural language queries. By giving open vocabulary semantic parsers direct access to KB information, we improve mean average precision on this task by over 120%.

## 2 Open vocabulary semantic parsing

In this section, we briefly describe current models for open vocabulary semantic parsing, following Krishnamurthy and Mitchel (2015). That work introduced a technique for learning semantic parsers with open predicate vocabularies. Instead of mapping text to Freebase queries, their method parses text to a surface logical form whose predicates are derived directly from the words in the text (see Figure 1). Next, a distribution over denotations for each predicate is learned using a matrix factorization approach similar to that of Riedel et al. (2013). This distribution is concisely represented using a probabilistic database, which also enables efficient probabilistic
execution of logical form queries.

The matrix factorization has two sets of parameters: each category or relation has a learned \( k \)-dimensional embedding \( \theta \), and each entity or entity pair has a learned \( k \)-dimensional embedding \( \phi \). The probability assigned to a category instance \( c(e) \) or relation instance \( r(e_1, e_2) \) is given by:

\[
p(c(e)) = \sigma(\theta^T \phi_e) \\
p(r(e_1, e_2)) = \sigma(\theta^T \phi(e_1, e_2))
\]

The probability of a predicate instance is the sigmoided inner product of the corresponding predicate and entity embeddings. Thus, predicates with nearby embeddings will have similar distributions over the entities in their denotation. The parameters \( \theta \) and \( \phi \) are learned using a query ranking objective that optimizes them to rank entities observed in the denotation of a logical form above unobserved entities.\(^1\) Given the trained predicate and entity parameters, the system is capable of efficiently computing the marginal probability that an entity is an element of a logical form’s denotation using approximate inference algorithms for probabilistic databases.

The model presented in this section is purely distributional, with predicate and entity models that draw only on co-occurrence information found in a corpus. In the following sections, we will show how to augment this model with the information contained in large, curated KBs such as Freebase.

3 Converting Freebase queries to features

The key insight of this paper is that we can convert the executable queries used by traditional semantic parsers into features that provide KB information to the learned execution models of open vocabulary semantic parsers. Here we describe how this is done.

Traditional semantic parsers map words onto distributions over executable queries, then select one to actually execute, returning sets of entities or entity pairs from a knowledge base as a result. Instead of executing a single query, we can simply execute all of the possible queries and use an entity’s (or entity pair’s) membership in each set as a feature in our predicate models.

There are two problems with this approach: (1) the set of all possible queries is intractably large, so we need a mechanism similar to a semantic parser’s lexicon to select a small set of queries for each word; and (2) executing hundreds or thousands of queries at runtime for each predicate and entity is not computationally tractable. To solve these problems, we use a graph-based technique called subgraph feature extraction (SFE)\(^2\) (Gardner and Mitchell, 2015).

3.1 Subgraph feature extraction

SFE is a technique for generating feature matrices over node pairs in a graph with labeled edges. When the graph corresponds to a formal KB such as Freebase, the features generated by SFE are isomorphic to statements in the KB schema.\(^3\) This means that we can use SFE to generate a feature vector for each entity (and entity pair) which succinctly captures the set of all statements in whose denotations the entity (or entity pair) appears. Using this feature vector as part of the semantic parser’s entity models solves problem (2) above, and performing feature selection over these feature vectors for each predicate solves problem (1).

Some example features extracted by SFE are shown in Figure 2. For entity pairs, these features include the sequence of edges (or paths) connecting the nodes corresponding to the entity pair. For entities, these features include the set of paths connected to the node, optionally including the node at the end of the path. Note the correspondence between these features and Freebase queries: the path \( \langle \text{DESIGNED} \rightarrow \text{LOCATED} \rangle \) can be executed as a query against Freebase, returning a set of (architect, location) entity pairs, where the architect designed a structure in the location.\(^4\) (PALLADIO, ITALY) is one such entity pair, so this pair has a feature value of 1 for this query.

3.2 Feature selection

The feature vectors produced by SFE contain tens of millions of possible formal statements. Out of these tens of millions of formal statements, only a handful represent relevant Freebase queries for any particular predicate. We therefore select a small number of

\(^1\)Note that this is implicitly factorizing a co-occurrence matrix between predicates and entities (Levy and Goldberg, 2014).

\(^2\)In a restricted class, which contains horn clauses and a few other things; see Gardner (2015) for more details.
features with the highest PMI values for each predicate), and feature \( \psi(e) \) or relation instance \( r(e_1, e_2) \) is thus given by:

\[
p(c(e)) = \sigma(\theta^T_c \phi_c + \omega^T_c \psi_c(e))
\]
\[
p(r(e_1, e_2)) = \sigma(\theta^T_r \phi_r(e_1, e_2) + \omega^T_r \psi_r(e_1, e_2))
\]

In the above equations, the \( \theta \) and \( \phi \) terms are learned predicate and entity embeddings, as described in Section 2. The second term in the sum represents our new features and their learned weights. \( \psi_c(e) \) and \( \psi_r(e_1, e_2) \) are SFE feature vectors for each entity and entity pair; a different set of features is chosen for each predicate \( c \) and \( r \), as described in Section 3.2. The terms \( \omega_c \) and \( \omega_r \) represent learned weights for these features.

In our model, there are now three sets of parameters to be learned: (1) \( \theta \), low-dimensional distributional vectors trained for each predicate; (2) \( \phi \), low-dimensional distributional vectors trained for each entity and entity pair; and (3) \( \omega \), weights associated with the selected formal SFE features for each predicate. All of these parameters are optimized jointly, using the same method described in Section 2.

Note here that, as stated in Section 3, each SFE feature corresponds to a query over the formal schema, defining a set of entities (or entity pairs). The associated feature weight measures the likelihood that an entity in this set is also in the denotation of the surface predicate. Our models include many such features for each surface predicate, effectively mapping each surface predicate onto a weighted combination of Freebase queries.

4 Combined predicate models

Here we present our approach to incorporating knowledge base information into open vocabulary semantic parsers. Having described how we use SFE to generate features that correspond to statements in a formal schema, adding these features to the models described in Section 2 is straightforward.

We saw in Section 2 that open vocabulary semantic parsers learn distributional vectors for each category, relation, entity and entity pair. We augment these vectors with the feature vectors described in Section 3. Each category and relation receives a weight \( \omega \) for each selected Freebase query, and each entity and entity pair has an associated feature vector \( \psi \). The truth probability of a category instance \( c(e) \) or relation instance \( r(e_1, e_2) \) is thus given by:

\[
p(c(e)) = \sigma(\theta^T_c \phi_c + \omega^T_c \psi_c(e))
\]
\[
p(r(e_1, e_2)) = \sigma(\theta^T_r \phi_r(e_1, e_2) + \omega^T_r \psi_r(e_1, e_2))
\]

5 Making full use of KB information

In addition to improving predicate models, as just described, adding knowledge base information to open vocabulary semantic parsers suggests two other simple improvements: (1) using more specific
logical forms, and (2) generating candidate entities from the knowledge base.

5.1 Logical form generation

Krishnamurthy and Mitchell (2015), whose work on open vocabulary semantic parsing we are following in this paper, generate logical forms from natural language statements by computing a syntactic CCG parse, then applying a collection of rules to produce logical forms. However, their logical form analyses do not model noun-mediated relations well. For example, given the phrase “Italian architect Andrea Palladio,” their system’s logical form would include the relation $N/N$(ITALY, PALLADIO). Here, the $N/N$ predicate represents a generic noun modifier relation; however, this relation is too vague for the predicate model to accurately learn its denotation. A similar problem occurs with prepositions and possessives, e.g., it is similarly hard to learn the denotation of the predicate of.

Our system improves the analysis of noun-mediated relations by simply including the noun in the predicate name. In the architect example above, our system produces the relation architect.$N/N$. It does this by concatenating all intervening noun modifiers between two entity mentions and including them in the predicate name; for example, “Illinois attorney general Lisa Madigan” produces the predicate attorney.general.$N/N$). We similarly improve the analyses of prepositions and possessives, e.g., it is similarly hard to learn the denotation of the predicate of.

Unfortunately, for some entities, such as the United States, the set of connected entities can be intractably large; we therefore limit this expansion to only those entities with fewer than 100 directly connected entities. Our motivation for this is that this candidate entity generation is most useful for rarely seen entities, for which we have few or no related entities seen during training. These entities also tend to have relatively few connected entities in Freebase.

5.2 Candidate entity generation

A key benefit of our predicate models is that they are able to assign scores to entity pairs that were never seen in the training data. Distributional models have no basis for assigning these scores and therefore assume $p(r(e_1,e_2)) = 0$ for unseen entity pairs $(e_1,e_2)$. This assumption limits the recall of these models when it is applied to question answering, as entity pairs will not have been observed for many correct, but rare entity answers. In contrast, because our models have access to a large knowledge base, the formal component of the model can always give a score to any entity pair in the knowledge base. This benefit allows our model to considerably improve question answering performance on rare entities.

However, this benefit has an associated drawback: given a logical form query, it is computationally and statistically undesirable to consider all Freebase entities as answers. Scoring millions of entities is slow, and furthermore makes it more difficult to identify correct answers (as their relative frequency in the candidate set decreases). Therefore, when answering a query, we compute a set of candidate entities and only score these candidates. Our candidate set consists of all entities seen with the query entities in the training set as well as all entities directly connected to the query entities in Freebase, or connected by a mediator node.

We evaluate our open-vocabulary semantic parser on a fill-in-the-blank natural language query task. Each test example is a natural language phrase containing at least two Freebase entities, one of which is held out. The system must propose a ranked list of Freebase entities to fill in the blank left by the held out entity, and the predicted entities are then judged manually for correctness. We compare our proposed models, which combine distributional and formal elements, with a purely distributional baseline from prior work. All of the data and code used in these experiments is available at [url withheld for review].
6.1 Data

We use the dataset introduced by Krishnamurthy and Mitchell (2015), which consists of the ClueWeb09 web corpus along with Google’s FACC entity linking of that corpus to Freebase (Gabrilovich et al., 2013). For training data, 3 million webpages from this corpus were processed with a CCG parser to produce logical forms (Krishnamurthy and Mitchell, 2014). This produced 2.1m predicate instances involving 142k entity pairs and 184k entities. After removing infrequently-seen predicates (seen fewer than 6 times), there were 25k categories and 4.2k relations.

We also used the test set created by Krishnamurthy and Mitchell, which contains 220 queries generated in the same fashion as the training data from a separate section of ClueWeb. However, as they did not release a development set with their data, we used this set as a development set. For a final evaluation, we generated another, similar test set from a different held out section of ClueWeb, in the same fashion as done by Krishnamurthy and Mitchell. This final test set contains 307 queries. We report results on both of these sets below.

6.2 Models

We compare three models in our experiments: (1) the distributional model described in Section 2; (2) a formal model (new to this work), where the distributional parameters θ and φ in Section 4 are fixed at zero; and (3) the combined model described in Section 4 (also new to this work). Except where noted, all experiments use our modified logical forms (Section 5.1) and our entity proposal mechanism (Section 5.2).

6.3 Methodology

Given a fill-in-the-blank query such as “Italian architect ____”, each system produces a ranked list of 100 candidate entities. To compare the output of the systems, we follow a pooled evaluation protocol commonly used in relation extraction and information retrieval (West et al., 2014; Riedel et al., 2013).

We take the top 30 predictions from each system and manually annotate whether they are correct, and use those annotations to compute the average precision (AP) and reciprocal rank (RR) of each system on the query. Average precision is defined as \( \frac{1}{m} \sum_{k=1}^{m} \text{Prec}(k) \times \text{Correct}(k) \), where \( \text{Prec}(k) \) is the precision at rank \( k \) and \( \text{Correct}(k) \) is an indicator function for whether the \( k \)th answer is correct, and \( m \) is number of returned answers (up to 100 in this evaluation). AP is equivalent to calculating the area under a precision-recall curve. Reciprocal rank is computed by first finding the rank \( r \) of the first correct prediction made by a system. Reciprocal rank is then \( \frac{1}{r} \), ranging from 1 (if the first prediction is correct) to 0 (if there is no correct answer returned). In the tables below we report mean average precision (MAP) and mean reciprocal rank (MRR), averaged over all of the queries in the test set. We also report a weighted version of MAP, where the AP of each query is scaled by the number of annotated correct answers to the query (shown as W-MAP in the tables for space considerations).

6.4 Results

We first show the effect of using the new logical forms introduced in Section 5.1. As can be seen in Table 1 with our improved logical forms, all models are better able to capture the semantics of language. This improvement is most pronounced in the formal models, which have more capacity to get specific features from Freebase with the new logical forms. As our logical forms are able to give all models better performance, the remaining experiments we present all use these logical forms.

We next show the improvement gained by using

| Model       | K&M’s LFs | Our LFs | Delta |
|-------------|-----------|---------|-------|
| Distributional | .269      | .284    | +.015 |
| Formal      | .231      | .276    | +.045 |
| Combined    | .313      | .335    | +.022 |

Table 1: Comparison of models using logical forms from prior work versus those introduced in this paper. The metric shown is mean average precision on the dev set. With our logical forms, performance of all models improves, quite substantially in the case of the formal model.
| Method                        | MAP  | W-MAP | MRR  |
|-------------------------------|------|-------|------|
| Distributional model         | .163 | .163  | .288 |
| With Freebase entities       | .229 | .275  | .312 |
| Relative improvement          | 40%  | 69%   | 8%   |

Table 2: Allowing Krishnamurthy and Mitchell’s model to propose candidate entities from Freebase improves mean average precision substantially, by allowing the model to have higher recall.

| Method                           | MAP  | W-MAP | MRR  |
|----------------------------------|------|-------|------|
| Prior work (distributional)      | .284 | .371  | .379 |
| Our formal model                 | .276 | .469  | .334 |
| Our combined model               | .335 | .477  | .429 |
| Relative improvement             | 18%  | 29%   | 13%  |

Table 3: Results on the development set for our fill-in-the-blank task. The combined model significantly improves MAP over prior work.

the simple candidate entity generation outlined in Section 5.2. By simply appending the list of connected entities in Freebase to the end of the rankings returned by the distributional model, MAP improves by 40% (see Table 2). The connectedness of an entity pair in Freebase is very informative, especially for rare entities that are not seen together during training.

Table 3 shows a comparison between the semantic parsing models we have discussed on the development set. As can be seen, the combined model significantly improves performance over prior work, giving a relative gain in weighted MAP of 29%.

Table 4 shows that these improvements are consistent on the final test set, as well. The performance improvement seen by the combined model is actually larger on this set, with gains on our metrics ranging from 50% to 87%.

On both of these datasets, the difference in MAP between the combined model and the distributional model is statistically significant (by a paired permutation test, $p < 0.05$). The differences between the combined model and the formal model, and between the formal model and the distributional model, are not statistically significant, as each method has certain kinds of queries that it performs well on. Only the combined model is able to consistently outperform the distributional model on all kinds of queries.

**6.5 Discussion**

Our model tends to outperform the distributional model on queries containing predicates with exact or partial correlates in Freebase. For example, our model obtains nearly perfect average precision on the queries “French newspaper ___” and “Israeli prime minister ___”, both of which can be exactly expressed in Freebase. The top features for newspaper($x$) all indicate that $x$ has type NEWSPAPER in Freebase, and the top features for newspaper,N/N($x$, $y$) indicate that $y$ is a newspaper, and that $x$ is either the circulation area of $y$ or the language of $y$.

The model also performs well on queries with partial Freebase correlates, such as “Microsoft head honcho ___”, “The United States, ___’s closest ally”, and “Patriots linebacker ___” although with somewhat lower average precision. The high weight features in these cases tend to provide useful hints, even though there is no direct correlate; for example, the model learns that “honchos” are people, and that they tend to be CEOs and film producers.

There are also some areas where our model can be improved. First, in some cases, the edge sequence features used by the model are not expressive enough to identify the correct relation in Freebase. An example of this problem is the “linebacker” example above, where the features for linebacker,N/N...
can capture which athletes play for which teams, but not the positions of those athletes. Second, our model can under-perform on predicates with no close mapping to Freebase. An example where this problem occurs is the query “___ is a NASA mission.” Third, there remains room to further improve the logical forms produced by the semantic parser, specifically for multi-word expressions. One problem occurs with multi-word noun modifiers, e.g., “Vice president Al Gore” is mapped to \text{vice} (\text{AL GORE}) \land \text{president} (\text{AL GORE}). Another problem is that there is no back-off with multi-word relations. For example, the predicate \text{head\_honcho\_N/N} was never seen in the training data, so it is replaced with \text{unknown}; however, it would be better to replace it with \text{honcho\_N/N}, which was seen in the training data. Finally, although using connected entities in Freebase as additional candidates during inference is helpful, it often over- or under-generates candidates. A more tailored, per-query search process could improve performance.

7 Related work

In addition to work on traditional and open vocabulary semantic parsing, which we have already discussed, there are two other classes of work that are related to what we have presented in this paper. First, the task of learning a probabilistic database in an open vocabulary semantic parser has a strong connection with the task of knowledge base completion. In addition to SFE (Gardner and Mitchell, 2015), our work draws inspiration from work on embedding the entities and relations in a knowledge base (Riedel et al., 2013; Nickel et al., 2011; Nickel et al., 2014; Bordes et al., 2013; Toutanova et al., 2015), as well as work on graph-based methods for reasoning with knowledge bases (Lao and Cohen, 2010; Gardner et al., 2014; Neelakantan et al., 2015). Indeed, our combination of embedding methods with graph-based methods in this paper is suggestive of how one could proceed to combine the two in methods for knowledge base completion. Preliminary work exploring this direction has already been done by Toutanova and Chen (2015).

Second, our work is conceptually related to many methods that aim to learn word embeddings that are informed by some kind of external knowledge (Faruqui et al., 2015; Rocktäschel et al., 2015; Schwartz et al., 2015; Yu and Dredze, 2014). A key difference between these approaches and ours is that they use the structured information only to modify distributional representations, while we include the structured information directly in our models, training feature weights jointly with distributional representations.

8 Conclusion

Prior work in semantic parsing has either been able to leverage large knowledge bases to answer questions, or used distributional techniques to gain broad coverage over all of natural language. In this paper, we have shown how to gain both of these benefits by converting the queries generated by traditional semantic parsers into features which are then used in open vocabulary semantic parsing models. We presented a technique to do this conversion in a way that is scalable using graph-based feature extraction methods. Our combined model achieved relative gains of over 50% in mean average precision and mean reciprocal rank versus a purely distributional approach. We also introduced a better mapping from surface text to logical forms, and a simple method for using a knowledge base to find candidate entities during inference. Taken together, the methods introduced in this paper improved mean average precision on our task from .163 to .370, a 127% relative improvement over prior work.

A consequence of this work is that it suggests a new direction for semantic parsing research. Existing semantic parsers map language to a single knowledge base query, an approach that successfully leverages a knowledge base’s predicate instances, but is fundamentally limited by its schema. In contrast, our approach maps language to a \text{weighted combination of queries} plus a distributional component; this approach is capable of representing a much broader class of concepts while still using the knowledge base when it is helpful. Furthermore, it is capable of using the knowledge base even when the meaning of the language cannot be exactly represented by a knowledge base predicate, which is a common occurrence. We believe that this kind of
approach could significantly expand the applicability of semantic parsing techniques to more complex domains where the assumptions of traditional techniques are too limiting.

References

[Berant et al.2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In EMNLP, pages 1533–1544.

[Bollacker et al.2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In Proceedings of SIGMOD.

[Bordes et al.2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In Advances in Neural Information Processing Systems, pages 2787–2795.

[Faruqui et al.2015] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In NAACL.

[Gabrilovich et al.2013] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Fac1: Freebase annotation of cluweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). Note: http://lemurproject.org/cluweb09/FAcc1/Cited by. 5.

[Gardner and Mitchell2015] Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In Proceedings of EMNLP. Association for Computational Linguistics.

[Gardner et al.2014] Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In Proceedings of EMNLP. Association for Computational Linguistics.

[Gardner2015] Matt Gardner. 2015. Reading and Reasoning with Knowledge Graphs. Ph.D. thesis, Carnegie Mellon University.

[Krishnamurthy and Mitchell2014] Jayant Krishnamurthy and Tom M. Mitchell. 2014. Joint syntactic and semantic parsing with combinatory categorial grammar. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL).

[Krishnamurthy and Mitchell2015] Jayant Krishnamurthy and Tom M. Mitchell. 2015. Learning a compositional semantics for Freebase with an open predicate vocabulary. Transactions of the Association for Computational Linguistics, 3:257–270.

[Lao and Cohen2010] Ni Lao and William W Cohen. 2010. Fast query execution for retrieval models based on path-constrained random walks. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 881–888. ACM.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In NIPS.

[Lewis and Steedman2013] Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. TACL, 1:179–192.

[Neelakantan et al.2015] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In ACL 2015. Association for Computational Linguistics.

[Nickel et al.2011] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In Proceedings of the 28th international conference on machine learning (ICML-11), pages 809–816.

[Nickel et al.2014] Maximilian Nickel, Xueyan Jiang, and Volker Tresp. 2014. Reducing the rank in relational factorization models by including observable patterns. In Advances in Neural Information Processing Systems, pages 1179–1187.

[Riedel et al.2013] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In Proceedings of NAACL-HLT.

[Rocktäschel et al.2015] Tim Rocktaeschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In NAACL.

[Schwartz et al.2015] Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric patterns and coordinations: Fast and enhanced representations of verbs and adjectives. In NAACL.

[Toutanova and Chen2015] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In ACL workshop on on Continuous Vector Space Models and Their Compositionality.

[Toutanova et al.2015] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In EMNLP.

[West et al.2014] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In WWW.
[Yu and Dredze2014] Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In ACL.

[Zelle and Mooney1996] John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In AAAI.

[Zettlemoyer and Collins2005] Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In UAI.