Learning a Compositional Semantics for Freebase with an Open Predicate Vocabulary

Jayant Krishnamurthy
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
jayantk@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
tom.mitchell@cmu.edu

Abstract

We present an approach to learning a model-theoretic semantics for natural language tied to Freebase. Crucially, our approach uses an open predicate vocabulary, enabling it to produce denotations for phrases such as "Republican front-runner from Texas" whose semantics cannot be represented using the Freebase schema. Our approach directly converts a sentence's syntactic CCG parse into a logical form containing predicates derived from the words in the sentence, assigning each word a consistent semantics across sentences. This logical form is evaluated against a learned probabilistic database that defines a distribution over denotations for each textual predicate. A training phase produces this probabilistic database using a corpus of entity-linked text and probabilistic matrix factorization with a novel ranking objective function. We evaluate our approach on a compositional question answering task where it outperforms several competitive baselines. We also compare our approach against manually annotated Freebase queries, finding that our open predicate vocabulary enables us to answer many questions that Freebase cannot.

1 Introduction

Traditional knowledge representation assumes that world knowledge can be encoded using a closed vocabulary of formal predicates. In recent years, semantic parsing has enabled us to build compositional models of natural language semantics using such a closed predicate vocabulary (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005). These semantic parsers map natural language statements to database queries, enabling applications such as answering questions using a large knowledge base (Yahya et al., 2012; Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013; Kwiatkowski et al., 2013; Berant et al., 2013; Berant and Liang, 2014; Reddy et al., 2014). Furthermore, the model-theoretic semantics provided by such parsers have the potential to improve performance on other tasks, such as information extraction and coreference resolution.

However, a closed predicate vocabulary has inherent limitations. First, its coverage will be limited, as such vocabularies are typically manually constructed. Second, it may abstract away potentially relevant semantic differences. For example, the semantics of "Republican front-runner" cannot be adequately encoded in the Freebase schema because it lacks the concept of a "front-runner." We could choose to encode this concept as "politician" at the cost of abstracting away the distinction between the two. As this example illustrates, these two problems are prevalent in even the largest knowledge bases.

An alternative paradigm is an *open* predicate vocabulary, where each natural language word or phrase is given its own formal predicate. This paradigm is embodied in both open information extraction (Banko et al., 2007) and universal schema (Riedel et al., 2013). Open predicate vocabularies have the potential to capture subtle semantic distinctions and achieve high coverage. However, we have yet to develop compelling approaches to compositional semantics within this paradigm.

This paper takes a step toward compositional se-

Transactions of the Association for Computational Linguistics, vol. 3, pp. 257–270, 2015. Action Editor: Katrin Erk.
Submission batch: 12/2014; Revision batch 3/2015; Published 5/2015.

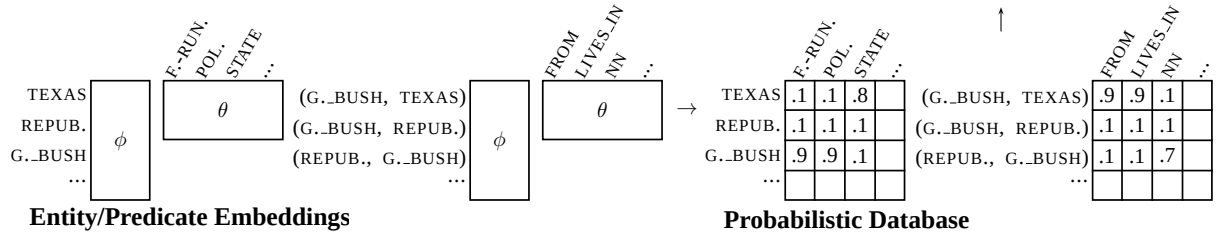

Figure 1: Overview of our approach. Top left: the text is converted to logical form by CCG syntactic parsing and a collection of manually-defined rules. Bottom: low-dimensional embeddings of each entity (entity pair) and category (relation) are learned from an entity-linked web corpus. These embeddings are used to construct a probabilistic database. The labels of these matrices are shortened for space reasons. Top right: evaluating the logical form on the probabilistic database computes the marginal probability that each entity is an element of the text's denotation.

mantics with an open predicate vocabulary. Our approach defines a distribution over denotations (sets of Freebase entities) given an input text. The model has two components, shown in Figure 1. The first component is a rule-based semantic parser that uses a syntactic CCG parser and manually-defined rules to map entity-linked texts to logical forms containing predicates derived from the words in the text. The second component is a probabilistic database with a possible worlds semantics that defines a distribution over denotations for each textually-derived predicate. This database assigns independent probabilities to individual predicate instances, such as $P(\text{FRONT-RUNNER}(\text{/EN/GEORGE\_BUSH})) = 0.9$. Together, these components define an exponentially-large distribution over denotations for an input text; to simplify this output, we compute the marginal probability, over all possible worlds, that each entity is an element of the text's denotation.

The learning problem in our approach is to train the probabilistic database to predict a denotation for each predicate. We pose this problem as probabilistic matrix factorization with a novel query/answer ranking objective. This factorization learns a low-dimensional embedding of each entity (entity pair) and category (relation) such that the denotation of a predicate is likely to contain entities or entity pairs with nearby vectors. To train the database, we first collect training data by analyzing entity-linked sentences in a large web corpus with the rule-based semantic parser. This process generates a collection of logical form queries with observed entity answers. The query/answer ranking objective, when optimized, trains the database to rank the observed answers for each query above unobserved answers.

We evaluate our approach on a question answering task, finding that our approach outperforms several baselines and that our new training objective improves performance over a previously-proposed objective. We also evaluate the trade-offs between open and closed predicate vocabularies by comparing our approach to a manually-annotated Freebase query for each question. This comparison reveals that, when Freebase contains predicates that cover the question, it achieves higher precision and recall than our approach. However, our approach can correctly answer many questions not covered by Freebase.

2 System Overview

The purpose of our system is to predict a denotation γ for a given natural language text s. The denotation γ is the set of Freebase entities that s refers to; for example, if $s =$ "president of the US," then $\gamma = \{\text{/EN/OBAMA}, \text{/EN/BUSH}, ...\}$.[1] Our system

[1]This paper uses a simple model-theoretic semantics where the denotation of a noun phrase is a set of entities and the denotation of a sentence is either true or false. However, for notational convenience, denotations γ will be treated as sets of entities throughout.

represents this prediction problem using the following probabilistic model:

$$P(\gamma|s) = \sum_{w} \sum_{\ell} P(\gamma|\ell, w)P(w)P(\ell|s)$$

The first term in this factorization, $P(\ell|s)$, is a distribution over logical forms ℓ given the text s. This term corresponds to the rule-based semantic parser (Section 3). This semantic parser is deterministic, so this term assigns probability 1 to a single logical form for each text. The second term, $P(w)$, represents a distribution over possible worlds, where each world is an assignment of truth values to all possible predicate instances. The distribution over worlds is represented by a probabilistic database (Section 4). The final term, $P(\gamma|\ell, w)$, deterministically evaluates the logical form ℓ on the world w to produce a denotation γ. This term represents query evaluation against a fixed database, as in other work on semantic parsing.

Section 5 describes inference in our model. To produce a ranked list of entities (Figure 1, top right) from $P(\gamma|s)$, our system computes the marginal probability that each entity is an element of the denotation γ. This problem corresponds to query evaluation in a probabilistic database, which is known to be tractable in many cases (Suciu et al., 2011).

Section 6 describes training, which estimates parameters for the probabilistic database $P(w)$. This step first automatically generates training data using the rule-based semantic parser. This data is used to formulate a matrix factorization problem that is optimized to estimate the database parameters.

3 Rule-Based Semantic Parser

The first part of our compositional semantics system is a rule-based system that deterministically computes a logical form ℓ for a text s. This component is used during inference to analyze the logical structure of text, and during training to generate training data (see Section 6.1). Several input/output pairs for this system are shown in Figure 2.

The conversion to logical form has 3 phases:

1. **CCG syntactic parsing** parses the text and applies several deterministic syntactic transformations to facilitate semantic analysis.
2. **Entity linking** marks known Freebase entities in the text.
3. **Semantic analysis** assigns a logical form to each word, then composes them to produce a logical form for the complete text.

3.1 Syntactic Parsing

The first step in our analysis is to syntactically parse the text. We use the ASP-SYN parser (Krishnamurthy and Mitchell, 2014) trained on CCGBank (Hockenmaier and Steedman, 2002). We then automatically transform the resulting syntactic parse to make the syntactic structure more amenable to semantic analysis. This step marks NPs in conjunctions by replacing their syntactic category with $NP[\text{conj}]$. This transformation allows semantic analysis to distinguish between appositives and comma-separated lists. It also transforms all verb arguments to core arguments, i.e., using the category PP/NP as opposed to $((S\backslash NP)\backslash(S\backslash NP))/NP$. This step simplifies the semantic analysis of verbs with prepositional phrase arguments. The final transformation adds a word feature to each PP category, e.g., mapping PP to $PP[\text{by}]$. These features are used to generate verb-preposition relation predicates, such as DIRECTED_BY.

3.2 Entity Linking

The second step is to identify mentions of Freebase entities in the text. This step could be performed by an off-the-shelf entity linking system (Ratinov et al., 2011; Milne and Witten, 2008) or string matching. However, our training and test data is derived from Clueweb 2009, so we rely on the entity linking for this corpus provided by Gabrilovich et. al (2013).

Our system incorporates the provided entity links into the syntactic parse provided that they are consistent with the parse structure. Specifically, we require that each mention is either (1) a constituent in the parse tree with syntactic category N or NP or (2) a collection of N/N or NP/NP modifiers with a single head word. The first case covers noun and noun phrase mentions, while the second case covers noun compounds. In both cases, we substitute a single multi-word terminal into the parse tree spanning the mention and invoke special semantic rules for mentions described in the next section.

Dan Hesse, CEO of Sprint
$\lambda x.\exists y.x = /\text{EN/DAN\_HESSE} \wedge \text{CEO}(x) \wedge \text{OF}(x, y) \wedge y = /\text{EN/SPRINT}$

Yankees pitcher
$\lambda x.\exists y.\text{PITCHER}(x) \wedge \text{NN}(y, x) \wedge y = /\text{EN/YANKEES}$

Tom Cruise plays Maverick in the movie Top Gun.
$\exists x, y, z.x = /\text{EN/TOM\_CRUISE} \wedge \text{PLAYS}(x, y) \wedge y = /\text{EN/MAVERICK\_(CHARACTER)} \wedge \text{PLAYS\_IN}(x, z) \wedge z = /\text{EN/TOP\_GUN}$

Figure 2: Example input/output pairs for our semantic analysis system. Mentions of Freebase entities in the text are indicated by underlines.

3.3 Semantic analysis

The final step uses the syntactic parse and entity links to produce a logical form for the text. The system induces a logical form for every word in the text based on its syntactic CCG category. Composing these logical forms according to the syntactic parse produces a logical form for the entire text.

Our semantic analyses are based on a relatively naïve model-theoretic semantics. We focus on language whose semantics can be represented with existentially-quantified conjunctions of unary and binary predicates, ignoring, for example, temporal scope and superlatives. Generally, our system models nouns and adjectives as unary predicates, and verbs and prepositions as binary predicates. Special multi-word predicates are generated for verb-preposition combinations. Entity mentions are mapped to the mentioned entity in the logical form. We also created special rules for analyzing conjunctions, appositives, and relativizing conjunctions. The complete list of rules used to produce these logical forms is available online.[2]

We made several notable choices in designing this component. First, multi-argument verbs are analyzed using pairwise relations, as in the third example in Figure 2. This analysis allows us to avoid reasoning about entity triples (quadruples, etc.), which are challenging for the matrix factorization due to sparsity. Second, noun-preposition combinations are analyzed as a category and relation, as in the first example in Figure 2. We empirically found that combining the noun and preposition in such instances resulted in worse performance, as it dramatically increased the sparsity of training instances for the combined relations. Third, entity mentions with the N/N category are analyzed using a special noun-noun relation, as in the second example in Figure 2. Our intuition is that this relation shares instances with other relations (e.g., "city in Texas" implies "Texan city"). Finally, we lowercased each word to create its predicate name, but performed no lemmatization or other normalization.

[2] http://rtw.ml.cmu.edu/tacl2015_csf

3.4 Discussion

The scope of our semantic analysis system is somewhat limited relative to other similar systems (Bos, 2008; Lewis and Steedman, 2013) as it only outputs existentially-quantified conjunctions of predicates. Our goal in building this system was to analyze noun phrases and simple sentences, for which this representation generally suffices. The reason for this focus is twofold. First, this subset of language is sufficient to capture much of the language surrounding Freebase entities. Second, for various technical reasons, this restricted semantic representation is easier to use (and more informative) for training the probabilistic database (see Section 6.3).

Note that this system can be straightforwardly extended to model additional linguistic phenomena, such as additional logical operators and generalized quantifiers, by writing additional rules. The semantics of logical forms including these operations are well-defined in our model, and the system does not even need to be re-trained to incorporate these additions.

4 Probabilistic Database

The second part of our compositional semantics system is a probabilistic database. This database represents a distribution over *possible worlds*, where each world is an assignment of truth values to every predicate instance. Equivalently, the probabilistic database can be viewed as a distribution over databases or knowledge bases.

Formally, a probabilistic database is a collection of random variables, each of which represents the truth value of a single predicate instance. Given entities $e \in E$, categories $c \in C$, and relations $r \in R$ the probabilistic database contains boolean random

variables $c(e)$ and $r(e_1, e_2)$ for each category and relation instance, respectively. All of these random variables are assumed to be independent. Let a world w represent an assignment of truth values to all of these random variables, where $c(e) = w_{c,e}$ and $r(e_1, e_2) = w_{r,e_1,e_2}$. By independence, the probability of a world can be written as:

$$P(w) = \prod_{e \in E} \prod_{c \in C} P(c(e) = w_{c,e}) \times \prod_{e_1 \in E} \prod_{e_2 \in E} \prod_{r \in R} P(r(e_1, e_2) = w_{r,e_1,e_2})$$

The next section discusses how probabilistic matrix factorization is used to model the probabilities of these predicate instances.

4.1 Matrix Factorization Model

The probabilistic matrix factorization model treats the truth of each predicate instance as an independent boolean random variable that is true with probability:

$$P(c(e) = \text{TRUE}) = \sigma(\theta_c^T \phi_e)$$
$$P(r(e_1, e_2) = \text{TRUE}) = \sigma(\theta_r^T \phi_{(e_1,e_2)})$$

Above, $\sigma(x) = \frac{e^x}{1+e^x}$ is the logistic function. In these equations, θ_c and θ_r represent k-dimensional vectors of per-predicate parameters, while ϕ_e and $\phi_{(e_1,e_2)}$ represent k-dimensional vector embeddings of each entity and entity pair. This model contains a low-dimensional embedding of each predicate and entity such that each predicate's denotation has a high probability of containing entities with nearby vectors. The probability that each variable is false is simply 1 minus the probability that it is true.

This model can be viewed as matrix factorization, as depicted in Figure 1. The category and relation instance probabilities can be arranged in a pair of matrices of dimension $|E| \times |C|$ and $|E|^2 \times |R|$. Each row of these matrices represents an entity or entity pair, each column represents a category or relation, and each value is between 0 and 1 and represents a truth probability (Figure 1, bottom right). These two matrices are factored into matrices of size $|E| \times k$ and $k \times |C|$, and $|E|^2 \times k$ and $k \times |R|$, respectively, containing k-dimensional embeddings of each entity, category, entity pair and relation (Figure 1, bottom left). These low-dimensional embeddings are represented by the parameters ϕ and θ.

5 Inference: Computing Marginal Probabilities

Inference computes the marginal probability, over all possible worlds, that each entity is an element of a text's denotation. In many cases – depending on the text – these marginal probabilities can be computed exactly in polynomial time.

The inference problem is to calculate $P(e \in \gamma | s)$ for each entity e. Because both the semantic parser $P(\ell | s)$ and query evaluation $P(\gamma | \ell, w)$ are deterministic, this problem can be rewritten as:

$$P(e \in \gamma | s) = \sum_{\gamma} \mathbf{1}(e \in \gamma) P(\gamma | s)$$
$$= \sum_{w} \mathbf{1}(e \in [\![\ell]\!]_w) P(w)$$

Above, ℓ represents the logical form for the text s produced by the rule-based semantic parser, and $\mathbf{1}$ represents the indicator function. The notation $[\![\ell]\!]_w$ represents denotation produced by (deterministically) evaluating the logical form ℓ on world w. This inference problem corresponds to query evaluation in a probabilistic database, which is #P-hard in general. Intuitively, this problem can be difficult because $P(\gamma | s)$ is a joint distribution over sets of entities that can be exponentially large in the number of entities.

However, a large subset of probabilistic database queries, known as *safe* queries, permit polynomial time evaluation (Dalvi and Suciu, 2007). Safe queries can be evaluated extensionally using a probabilistic notion of a denotation that treats each entity as independent. Let $[\![\ell]\!]_P$ denote a probabilistic denotation, which is a function from entities (or entity pairs) to probabilities, i.e., $[\![\ell]\!]_P(e) \in [0, 1]$. The denotation of a logical form is then computed recursively, in the same manner as a non-probabilistic denotation, using probabilistic extensions of the typical rules, such as:

$$[\![c]\!]_P(e) = \sum_{w} P(w) \mathbf{1}(w_{c,e})$$
$$[\![r]\!]_P(e_1, e_2) = \sum_{w} P(w) \mathbf{1}(w_{r,e_1,e_2})$$
$$[\![c_1(x) \wedge c_2(x)]\!]_P(e) = [\![c_1]\!]_P(e) \times [\![c_2]\!]_P(e)$$
$$[\![\exists y.r(x, y)]\!]_P(e) = 1 - \prod_{y \in E} (1 - [\![r]\!]_P(e, y))$$

The first two rules are base cases that simply retrieve predicate probabilities from the probabilistic database. The remaining rules compute the probabilistic denotation of a logical form from the denotations of its parts.[3] The formula for the probabilistic computation on the right of each of these rules is a straightforward consequence of the (assumed) independence of entities. For example, the last rule computes the probability of an OR of a set of independent random variables (indexed by y) using the identity $A \vee B = \neg(\neg A \wedge \neg B)$. For safe queries, $[\![\ell]\!]_P(e) = P(e \in \gamma|s)$, that is, the probabilistic denotation computed according to the above rules *is equal to* the marginal probability distribution. In practice, all of the queries in the experiments are safe, because they contain only one query variable and do not contain repeated predicates. For more information on query evaluation in probabilistic databases, we refer the reader to Suciu et al. (2011).

Note that inference does not compute the most probable denotation, $\max_\gamma P(\gamma|s)$. In some sense, the most probable denotation is the correct output for a model-theoretic semantics. However, it is highly sensitive to the probabilities in the database, and in many cases it is empty (because a conjunction of independent boolean random variables is unlikely to be true). Producing a ranked list of entities is also useful for evaluation purposes.

6 Training

The training problem in our approach is to learn parameters θ and ϕ for the probabilistic database. We consider two different objective functions for learning these parameters that use slightly different forms of training data. In both cases, training has two phases. First, we generate training data, in the form of observed assertions or query-answer pairs, by applying the rule-based semantic parser to a corpus of entity-linked web text. Second, we optimize the parameters of the probabilistic database to rank observed assertions or answers above unobserved assertions or answers.

[3]This listing of rules is partial as it does not include, e.g., negation or joins between one-argument and two-argument logical forms. However, the corresponding rules are easy to derive.

6.1 Training Data

Training data is generated by applying the process illustrated in Figure 3 to each sentence in an entity-linked web corpus. First, we apply our rule-based semantic parser to the sentence to produce a logical form. Next, we extract portions of this logical form where every variable is bound to a particular Freebase entity, resulting in a simplified logical form. Because the logical forms are existentially-quantified conjunctions of predicates, this step simply discards any conjuncts in the logical form containing a variable that is not bound to a Freebase entity. From this simplified logical form, we generate two types of training data: (1) predicate instances, and (2) queries with known answers (see Figure 3). In both cases, the corpus consists entirely of assumed-to-be-true statements, making obtaining negative examples a major challenge for training.[4]

6.2 Predicate Ranking Objective

Riedel et al. (2013) introduced a ranking objective to work around the lack of negative examples in a similar matrix factorization problem. Their objective is a modified version of Bayesian Personalized Ranking (Rendle et al., 2009) that aims to rank observed predicate instances above unobserved instances.

This objective function uses observed predicate instances (Figure 3, bottom left) as training data. This data consists of two collections, $\{(c_i, e_i)\}_{i=1}^n$ and $\{(r_j, t_j)\}_{j=1}^m$, of observed category and relation instances. We use t_j to denote a tuple of entities, $t_j = (e_{j,1}, e_{j,2})$, to simplify notation. The predicate ranking objective is:

$$O_P(\theta, \phi) = \sum_{i=1}^{n} \log \sigma(\theta_{c_i}^T(\phi_{e_i} - \phi_{e_i'})) + \sum_{j=1}^{m} \log \sigma(\theta_{r_j}^T(\phi_{t_j} - \phi_{t_j'}))$$

where e_i' is a randomly sampled entity such that (c_i, e_i') does not occur in the training data. Similarly, t_j' is a random entity tuple such that (r_j, t_j') does not occur. Maximizing this function attempts

[4]A seemingly simple solution to this problem is to randomly generate negative examples; however, we empirically found that this approach performs considerably worse than both of the proposed ranking objectives.

Original sentence and logical form

General Powell, appearing Sunday on CNN 's Late Edition, said ...

$\exists w, x, y, z.\ \ w = /\text{EN/POWELL} \wedge \text{GENERAL}(w) \wedge \text{APPEARING}(w, x) \wedge \text{SUNDAY}(x) \wedge \text{APPEARING\_ON}(w, y) \wedge y = /\text{EN/LATE} \wedge \text{'S}(z, y) \wedge z = /\text{EN/CNN} \wedge \text{SAID}(w, ...)$

Simplified logical form

$\exists w, y, z.\ \ w = /\text{EN/POWELL} \wedge \text{GENERAL}(w) \wedge \text{APPEARING\_ON}(w, y) \wedge y = /\text{EN/LATE} \wedge \text{'S}(z, y) \wedge z = /\text{EN/CNN}$

Instances	**Queries**	**Answers**
$\text{GENERAL}(/\text{EN/POWELL})$	$\lambda w.\text{GENERAL}(w) \wedge \text{APPEARING\_ON}(w, /\text{EN/LATE})$	/EN/POWELL
$\text{APPEARING\_ON}(/\text{EN/POWELL}, /\text{EN/LATE})$	$\lambda y.\text{APPEARING\_ON}(/\text{EN/POWELL}, y) \wedge \text{'S}(/\text{EN/CNN}, y)$	/EN/LATE
$\text{'S}(/\text{EN/CNN}, /\text{EN/LATE})$	$\lambda z.\text{'S}(z, /\text{EN/LATE})$	/EN/CNN

Figure 3: Illustration of training data generation applied to a single sentence. We generate two types of training data, predicate instances and queries with observed answers, by semantically parsing the sentence and extracting portions of the generated logical form with observed entity arguments. The predicate instances are extracted from the conjuncts in the simplified logical form, and the queries are created by removing a single entity from the simplified logical form.

to find θ_{c_i}, ϕ_{e_i} and $\phi_{e'_i}$ such that $P(c_i(e_i))$ is larger than $P(c_i(e'_i))$ (and similarly for relations). During training, e'_i and t'_j are resampled on each pass over the data set according to each entity or tuple's empirical frequency.

6.3 Query Ranking Objective

The previous objective aims to rank the entities within each predicate well. However, such within-predicate rankings are insufficient to produce correct answers for queries containing multiple predicates – the scores for each predicate must further be calibrated to work well with each other given the independence assumptions of the probabilistic database.

We introduce a new training objective that encourages good rankings for entire queries instead of single predicates. The data for this objective consists of tuples, $\{(\ell_i, e_i)\}_{i=1}^n$, of a query ℓ_i with an observed answer e_i (Figure 3, bottom right). Each ℓ_i is a function with exactly one entity argument, and $\ell_i(e)$ is a conjunction of predicate instances. For example, the last query in Figure 3 is a function of one argument z, and $\ell(e)$ is a single predicate instance, $\text{'S}(e, /\text{EN/LATE})$. The new objective aims to rank the observed entity answer above unobserved entities for each query:

$$O_Q(\theta, \phi) \quad = \quad \sum_{i=1}^{n} \log P_{rank}(\ell_i, e_i, e'_i)$$

P_{rank} generalizes the approximate ranking probability defined by the predicate ranking objective to more general queries. The expression $\sigma(\theta_c^T(\phi_e - \phi_{e'}))$ in the predicate ranking objective can be viewed as an approximation of the probability that e is ranked above e' in category c. P_{rank} uses this approximation for each individual predicate in the query. For example, given the query $\ell = \lambda x.c(x) \wedge r(x, y)$ and entities (e, e'), $P_{rank}(\ell, e, e') = \sigma(\theta_c(\phi_e - \phi_{e'})) \times \sigma(\theta_r(\phi_{(e,y)} - \phi_{(e',y)}))$. For this objective, we sample e'_i such that (ℓ_i, e'_i) does not occur in the training data.

When ℓ's body consists of a conjunction of predicates, the query ranking objective simplifies considerably. In this case, ℓ can be described as three sets of one-argument functions: categories $C(\ell) = \{\lambda x.c(x)\}$, left arguments of relations $R_L(\ell) = \{\lambda x.r(x, y)\}$, and right arguments of relations $R_R(\ell) = \{\lambda x.r(y, x)\}$. Furthermore, P_{rank} is a product so we can distribute the log:

$$\begin{aligned} O_Q(\theta, \phi) \quad = \quad & \sum_{i=1}^{n} \sum_{\lambda x.c(x) \in C(\ell_i)} \log \sigma(\theta_c(\phi_{e_i} - \phi_{e'_i})) \\ & + \sum_{\lambda x.r(x,y) \in R_L(\ell_i)} \log \sigma(\theta_r(\phi_{(e_i,y)} - \phi_{(e'_i,y)})) \\ & + \sum_{\lambda x.r(y,x) \in R_R(\ell_i)} \log \sigma(\theta_r(\phi_{(y,e_i)} - \phi_{(y,e'_i)})) \end{aligned}$$

This simplification reveals that the main difference between O_Q and O_P is the sampling of the unobserved entities e' and tuples t'. O_P samples them in an unconstrained fashion from their empirical distributions for every predicate. O_Q considers

the larger context in which each predicate occurs, with two major effects. First, more negative examples are generated for categories because the logical forms ℓ are more specific. For example, both "president of Sprint" and "president of the US" generate instances of the PRESIDENT predicate; O_Q will use entities that only occur with one of these as negative examples for the other. Second, the relation parameters are trained to rank tuples with a shared argument, as opposed to tuples in general.

Note that, although P_{rank} generalizes to more complex logical forms than existentially-quantified conjunctions, training with these logical forms is more difficult because P_{rank} is no longer a product. In these cases, it becomes necessary to perform inference within the gradient computation, which can be expensive. The restriction to conjunctions makes inference trivial, enabling the factorization above.

7 Evaluation

We evaluate our approach to compositional semantics on a question answering task. Each test example is a (compositional) natural language question whose answer is a set of Freebase entities. We compare our open domain approach to several baselines based on prior work, as well as a human-annotated Freebase query for each example.

7.1 Data

We used Clueweb09 web corpus[5] with the corresponding Google FACC entity linking (Gabrilovich et al., 2013) to create the training and test data for our experiments. The training data is derived from 3 million webpages, and contains 2.1m predicate instances, 1.1m queries, 172k entities and 181k entity pairs. Predicates that appeared fewer than 6 times in the training data were replaced with the predicate UNK, resulting in 25k categories and 2.2k relations.

Our test data consists of fill-in-the-blank natural language questions such as "Incan emperor __" or "Cunningham directed Auchtre's second music video __." These questions were created by applying the training data generation process (Section 6.1) to a collection of held-out webpages. Each natural language question has a corresponding logical form query containing at least one category and relation.

[5] http://www.lemurproject.org/clueweb09.php

# of questions	220
Avg. # of predicates / query	2.77
Avg. # of categories / query	1.75
Avg. # of relations / query	1.02
Avg. # of answers / query	1.92
# of questions with ≥ 1 answer (found by at least one system)	116

Table 1: Statistics of the test data set.

We chose not to use existing data sets for semantic parsing into Freebase as our goal is to model the semantics of language that cannot necessarily be modelled using the Freebase schema. Existing data sets, such as Free917 (Cai and Yates, 2013) and WebQuestions (Berant et al., 2013), would not allow us to evaluate performance on this subset of language. Consequently, we evaluate our system on a new data set with unconstrained language. However, we do compare our approach against manually-annotated Freebase queries on our new data set (Section 7.5).

All of the data for our experiments is available at http://rtw.ml.cmu.edu/tacl2015_csf.

7.2 Methodology

Our evaluation methodology is inspired by information retrieval evaluations (Manning et al., 2008). Each system predicts a ranked list of 100 answers for each test question. We then pool the top 30 answers of each system and manually judge their correctness. The correct answers from the pool are then used to evaluate the precision and recall of each system. In particular, we compute average precision (AP) for each question and report the mean average precision (MAP) across all questions. We also report a weighted version of MAP, where each question's AP is weighted by its number of annotated correct answers. Average precision is computed as $\frac{1}{m}\sum_{k=1}^{m} \mathrm{Prec}(k) \times \mathrm{Correct}(k)$, where $\mathrm{Prec}(k)$ is the precision at rank k, $\mathrm{Correct}(k)$ is an indicator function for whether the kth answer is correct, and m is the number of returned answers (at most 100).

Statistics of the annotated test set are shown in Table 1. A consequence of our unconstrained data generation approach is that some test questions are difficult to answer: of the 220 queries, at least one system was able to produce a correct answer for 116. The remaining questions are mostly unanswerable

	MAP	Weighted MAP
CLUSTERING	0.224	0.266
CORPUSLOOKUP	0.246	0.296
FACTORIZATION (O_P)	0.299	0.473
FACTORIZATION (O_Q)	0.309	0.492
ENSEMBLE (O_P)	0.391	0.614
ENSEMBLE (O_Q)	0.406	0.645
Upper bound	0.527	1.0

Table 2: Mean average precision for our question answering task. The difference in MAP between each pair of adjacent models is statistically significant ($p < .05$) via the sign test.

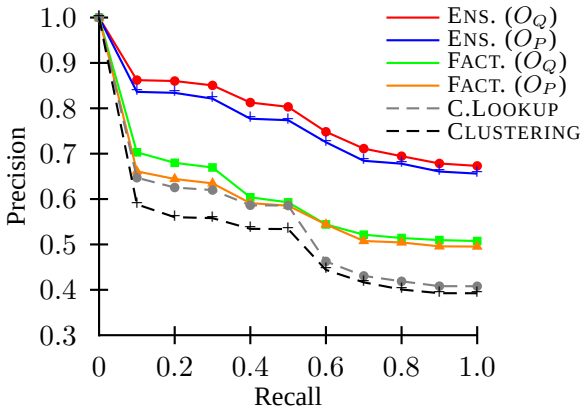

Figure 4: Averaged 11-point precision/recall curves for the 116 answerable test questions.

because they reference rare entities unseen in the training data.

7.3 Models and Baselines

We implemented two baseline models based on existing techniques. The CORPUSLOOKUP baseline answers test questions by directly using the predicate instances in the training data as its knowledge base. For example, given the query $\lambda x.\text{CEO}(x) \wedge \text{OF}(x, /\text{EN/SPRINT})$, this model will return the set of entities e such that $\text{CEO}(e)$ and $\text{OF}(e, /\text{EN/SPRINT})$ both appear in the training data. All answers found in this fashion are assigned probability 1.

The CLUSTERING baseline first clusters the predicates in the training corpus, then answers questions using the clustered predicates. The clustering aggregates predicates with similar denotations, ideally identifying synonyms to smooth over sparsity in the training data. Our approach is closely based on Lewis and Steedman (2013), though is also conceptually related to approaches such as DIRT (Lin and Pantel, 2001) and USP (Poon and Domingos, 2009). We use the Chinese Whispers clustering algorithm (Biemann, 2006) and calculate the similarity between predicates as the cosine similarity of their TF-IDF weighted entity count vectors. The denotation of each cluster is the union of the denotations of the clustered predicates, and each entity in the denotation is assigned probability 1.

We also trained two probabilistic database models, FACTORIZATION (O_P) and FACTORIZATION (O_Q), using the two objective functions described in Sections 6.2 and 6.3, respectively. We optimized both objectives by performing 100 passes over the training data with AdaGrad (Duchi et al., 2011) using an L2 regularization parameter of $\lambda = 10^{-4}$. The predicate and entity embeddings have 300 dimensions. These parameters were selected on the basis of preliminary experiments with a small validation set.

Finally, we observed that CORPUSLOOKUP has high precision but low recall, while both matrix factorization models have high recall with somewhat lower precision. This observation suggested that an ensemble of CORPUSLOOKUP and FACTORIZATION could outperform either model individually. We created two ensembles, ENSEMBLE (O_P) and ENSEMBLE (O_Q), by calculating the probability of each predicate as a 50/50 mixture of each model's predicted probability.

7.4 Results

Table 2 shows the results of our MAP evaluation, and Figure 4 shows a precision/recall curve for each model. The MAP numbers are somewhat low because almost half of the test questions have no correct answers and all models get an average precision of 0 on these questions. The upper bound on MAP is the fraction of questions with at least 1 correct answer. Note that the models perform well on the answerable questions, as reflected by the ratio of the achieved MAP to the upper bound. The weighted MAP metric also corrects for these unanswerable questions, as they are assigned 0 weight in the weighted average.

These results demonstrate several findings. First, we find that both FACTORIZATION models outperform the baselines in both MAP and weighted MAP.

# of questions w/ an annotated MQL query	142
query returns > 1 answer	95
query returns no answers	47
# of questions w/o an MQL query	78

Table 3: Statistics of the Freebase MQL queries annotated for the test data set.

	MAP
ENSEMBLE (O_Q)	0.263
Freebase	0.385

Table 4: Mean average precision of our best performing model compared to a manually annotated Freebase query for each test question.

The performance improvement seems to be most significant in the high recall regime (right side of Figure 4). Second, we find that the query ranking objective O_Q improves performance over the predicate ranking objective O_P by 2-4% on the answerable queries. The precision/recall curves show that this improvement is concentrated in the low recall regime. Finally, the ensemble models are considerably better than their component models; however, even in the ensembled models, we find that O_Q outperforms O_P by a few percent.

7.5 Comparison to Semantic Parsing to Freebase

A natural question is whether our open vocabulary approach outperforms a closed approach for the same problem, such as semantic parsing to Freebase (e.g., Reddy et al. (2014)). In order to answer this question, we compared our best performing model to a manually-annotated Freebase query for each test question. This comparison allows us to understand the relative advantages of open and closed predicate vocabularies.

The first author manually annotated a Freebase MQL query for each natural language question in the test data set. This annotation is somewhat subjective, as many of the questions can only be inexactly mapped on to the Freebase schema. We used the following guidelines in performing the mapping: (1) all relations in the text must be mapped to one or more Freebase relations, (2) all entities mentioned in the text must be included in the query, (3) adjective modifiers can be ignored and (4) entities not mentioned in the text may be included in the query. The fourth condition is necessary because many one-place predicates, such as MAYOR(x), are represented in Freebase using a binary relation to a particular entity, such as GOVERNMENT_OFFICE/TITLE$(x,$ /EN/MAYOR$)$.

Statistics of the annotated queries are shown in Table 3. Coverage is reasonably high: we were able to annotate a Freebase query for 142 questions (65% of the test set). The remaining unannotatable questions are due to missing predicates in Freebase, such as a relation defining the emperor of the Incan empire. Of the 142 annotated Freebase queries, 95 of them return at least one entity answer. The queries with no answers typically reference uncommon entities which have few or no known relation instances in Freebase. The annotated queries contain an average of 2.62 Freebase predicates.

We compared our best performing model, ENSEMBLE (O_Q), to the manually annotated Freebase queries using the same pooled evaluation methodology. The set of correct answers contains the correct predictions of ENSEMBLE (O_Q) from the previous evaluation along with all answers from Freebase.

Results from this evaluation are shown in Table 4.[6] In terms of overall MAP, Freebase outperforms our approach by a fair margin. However, this initial impression belies a more complex reality, which is shown in Table 5. This table compares both approaches by their relative performance on each test question. On approximately one-third of the questions, Freebase has a higher AP than our approach. On another third, our approach has a higher AP than Freebase. On the final third, both approaches perform equally well – these are typically questions where neither approach returns any correct answers (67 of the 75). Freebase outperforms in the overall MAP evaluation because it tends to return more correct answers to each question.

Note that the annotated Freebase queries have several advantages in this evaluation. First, Freebase contains significantly more predicate instances than our training data, which allows it to produce more complete answers. Second, the Freebase queries

[6]The numbers in this table are not comparable to the numbers in Table 2 as the correct answers for each question are different.

	# of queries
Freebase higher AP	75 (34%)
equal AP	75 (34%)
ENSEMBLE (O_Q) higher AP	70 (31%)

Table 5: Question-by-question comparison of model performance. Each test question is placed into one of the three buckets above, depending on whether Freebase or ENSEMBLE (O_Q) achieves a better average precision (AP) for the question.

correspond to the performance of a perfect semantic parser, while current semantic parsers achieve accuracies around 68% (Berant and Liang, 2014).

The results from this experiment suggest that closed and open predicate vocabularies are complementary. Freebase produces high quality answers when it covers a question. However, many of the remaining questions can be answered correctly using an open vocabulary approach like ours. This evaluation also suggests that recall is a limiting factor of our approach; in the future, recall can be improved by using a larger corpus or including Freebase instances during training.

8 Related Work

Open Predicate Vocabularies

There has been considerable work on generating semantic representations with an open predicate vocabulary. Much of the work is non-compositional, focusing on identifying similar predicates and entities. DIRT (Lin and Pantel, 2001), Resolver (Yates and Etzioni, 2007) and other systems (Yao et al., 2012) cluster synonymous expressions in a corpus of relation triples. Matrix factorization is an alternative approach to clustering that has been used for relation extraction (Riedel et al., 2013; Yao et al., 2013) and finding analogies (Turney, 2008; Speer et al., 2008). All of this work is closely related to distributional semantics, which uses distributional information to identify semantically similar words and phrases (Turney and Pantel, 2010; Griffiths et al., 2007).

Some work has considered the problem of compositional semantics with an open predicate vocabulary. Unsupervised semantic parsing (Poon and Domingos, 2009; Titov and Klementiev, 2011) is a clustering-based approach that incorporates composition using a generative model for each sentence that factors according to its parse tree. Lewis and Steedman (2013) also present a clustering-based approach that uses CCG to perform semantic composition. This approach is similar to ours, except that we use matrix factorization and Freebase entities.

Finally, some work has focused on the problem of textual inference within this paradigm. Fader et al. (2013) present a question answering system that learns to paraphrase a question so that it can be answered using a corpus of Open IE triples (Fader et al., 2011). Distributional similarity has also been used to learn weighted logical inference rules that can be used for recognizing textual entailment or identifying semantically similar text (Garrette et al., 2011; Garrette et al., 2013; Beltagy et al., 2013). This line of work focuses on performing inference between texts, whereas our work computes a text's denotation.

A significant difference between our work and most of the related work above is that our work computes denotations containing Freebase entities. Using these entities has two advantages: (1) it enables us to use entity linking to disambiguate textual mentions, and (2) it facilitates a comparison against alternative approaches that rely on a closed predicate vocabulary. Disambiguating textual mentions is a major challenge for previous approaches, so an entity-linked corpus is a much cleaner source of data. However, our approach could also work with automatically constructed entities, for example, created by clustering mentions in an unsupervised fashion (Singh et al., 2011).

Semantic Parsing

Several semantic parsers have been developed for Freebase (Cai and Yates, 2013; Kwiatkowski et al., 2013; Berant et al., 2013; Berant and Liang, 2014). Our approach is most similar to that of Reddy et al. (2014), which uses fixed syntactic parses of unlabeled text to train a Freebase semantic parser. Like our approach, this system automatically-generates query/answer pairs for training. However, this system, like all Freebase semantic parsers, uses a closed predicate vocabulary consisting of only Freebase predicates. In contrast, our approach uses an open predicate vocabulary and can learn denotations for words whose semantics cannot be represented using

Freebase predicates. Consequently, our approach can answer many questions that these Freebase semantic parsers cannot (see Section 7.5).

The rule-based semantic parser used in this paper is very similar to several other rule-based systems that produce logical forms from syntactic CCG parses (Bos, 2008; Lewis and Steedman, 2013). We developed our own system in order to have control over the particulars of the analysis; however, our approach is compatible with these systems as well.

Probabilistic Databases

Our system assigns a model-theoretic semantics to statements in natural language (Dowty et al., 1981) using a learned distribution over possible worlds. This distribution is concisely represented in a probabilistic database, which can be viewed as a simple Markov Logic Network (Richardson and Domingos, 2006) where all of the random variables are independent. This independence simplifies query evaluation: probabilistic databases permit efficient exact inference for safe queries (Suciu et al., 2011), and approximate inference for the remainder (Gatterbauer et al., 2010; Gatterbauer and Suciu, 2015).

9 Discussion

This paper presents an approach for compositional semantics with an open predicate vocabulary. Our approach defines a probabilistic model over denotations (sets of Freebase entities) conditioned on an input text. The model has two components: a rule-based semantic parser that produces a logical form for the text, and a probabilistic database that defines a distribution over denotations for each predicate. A training phase learns the probabilistic database by applying probabilistic matrix factorization with a query/answer ranking objective to logical forms derived from a large, entity-linked web corpus. An experimental analysis demonstrates that this approach outperforms several baselines and can answer many questions that cannot be answered by semantic parsing into Freebase.

Our approach learns a model-theoretic semantics for natural language text tied to Freebase, as do some semantic parsers, except with an open predicate vocabulary. This difference influences several other aspects of the system's design. First, because no knowledge base with the necessary knowledge exists, the system is forced to *learn* its knowledge base (in the form of a probabilistic database). Second, the system can directly map syntactic CCG parses to logical forms, as it is no longer necessary to map words to a closed vocabulary of knowledge base predicates. In some sense, our approach is the exact opposite of the typical semantic parsing approach: usually, the semantic parser is learned and the knowledge base is fixed; here, the knowledge base is learned and the semantic parser is fixed. From a machine learning perspective, training a probabilistic database via matrix factorization is easier than training a semantic parser, as there are no difficult inference problems. However, it remains to be seen whether a learned knowledge base can achieve similar recall as a fixed knowledge base on the subset of language it covers.

There are two limitations of this work. The most obvious limitation is the restriction to existentially quantified conjunctions of predicates. This limitation is not inherent to the approach, however, and can be removed in future work by using a system like Boxer (Bos, 2008) for semantic parsing. A more serious limitation is the restriction to one- and two-argument predicates, which prevents our system from representing events and n-ary relations. Conceptually, a similar matrix factorization approach could be used to learn embeddings for n-ary entity tuples; however, in practice, the sparsity of these tuples makes learning challenging. Developing methods for learning n-ary relations is an important problem for future work.

A direction for future work is scaling up the size of the training corpus to improve recall. Low recall is the main limitation of our current system as demonstrated by the experimental analysis. Both stages of training, the data generation and matrix factorization, can be parallelized using a cluster. All of the relation instances in Freebase can also be added to the training corpus. It should be feasible to increase the quantity of training data by a factor of 10-100, for example, to train on all of ClueWeb. Scaling up the training data may allow a semantic parser with an open predicate vocabulary to outperform comparable closed vocabulary systems.

Acknowledgments

This research was supported in part by DARPA under contract number FA8750-13-2-0005, and by a generous grant from Google. We additionally thank Matt Gardner, Ndapa Nakashole, Amos Azaria and the anonymous reviewers for their helpful comments.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets markov: Deep semantics with probabilistic logical form. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

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

Chris Biemann. 2006. Chinese whispers: An efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*.

Qingqing Cai and Alexander Yates. 2013. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nilesh Dalvi and Dan Suciu. 2007. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4), October.

David R. Dowty, Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Anthony Fader, 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*.

Evgeniy Gabrilovich, 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). http://lemurproject.org/clueweb09/.

Dan Garrette, Katrin Erk, and Raymond Mooney. 2011. Integrating logical representations with probabilistic information using markov logic. In *Proceedings of the International Conference on Computational Semantics*.

Dan Garrette, Katrin Erk, and Raymond J. Mooney. 2013. A formal approach to linking logical form and vector-space lexical semantics. In Harry Bunt, Johan Bos, and Stephen Pulman, editors, *Computing Meaning*, volume 4, pages 27–48.

Wolfgang Gatterbauer and Dan Suciu. 2015. Approximate lifted inference with probabilistic databases. *Proceedings of the VLDB Endowment*, 8(5), January.

Wolfgang Gatterbauer, Abhay Kumar Jha, and Dan Suciu. 2010. Dissociation and propagation for efficient query evaluation over probabilistic databases. In *Proceedings of the Fourth International VLDB workshop on Management of Uncertain Data (MUD 2010) in conjunction with VLDB 2010, Singapore, September 13, 2010.*

Thomas L. Griffiths, Joshua B. Tenenbaum, and Mark Steyvers. 2007. Topics in semantic representation. *Psychological Review 114*.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of Third International Conference on Language Resources and Evaluation*.

Jayant Krishnamurthy and Tom M. 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*.

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*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the*

2013 Conference on Empirical Methods in Natural Language Processing.

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

Dekang Lin and Patrick Pantel. 2001. DIRT — discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics – Volume 2, Issue 1*.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*.

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

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Association for Computational Linguistics: Human Language Technologies (ACL HLT)*.

Robert Speer, Catherine Havasi, and Henry Lieberman. 2008. AnalogySpace: Reducing the dimensionality of common sense knowledge. In *AAAI*.

Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. 2011. Probabilistic databases. *Synthesis Lectures on Data Management*, 3(2):1–180.

Ivan Titov and Alexandre Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1), January.

Peter D. Turney. 2008. The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, 33(1):615–655, December.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*.

Alexander Yates and Oren Etzioni. 2007. Unsupervised resolution of objects and relations on the web. In *Proceedings of the 2007 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.