Unsupervised Semantic Parsing

Hoifung Poon **Pedro Domingos**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{hoifung,pedrod}@cs.washington.edu

Abstract

We present the first unsupervised approach to the problem of learning a semantic parser, using Markov logic. Our USP system transforms dependency trees into quasi-logical forms, recursively induces lambda forms from these, and clusters them to abstract away syntactic variations of the same meaning. The MAP semantic parse of a sentence is obtained by recursively assigning its parts to lambda-form clusters and composing them. We evaluate our approach by using it to extract a knowledge base from biomedical abstracts and answer questions. USP substantially outperforms TextRunner, DIRT and an informed baseline on both precision and recall on this task.

1 Introduction

Semantic parsing maps text to formal meaning representations. This contrasts with semantic role labeling (Carreras and Marquez, 2004) and other forms of shallow semantic processing, which do not aim to produce complete formal meanings. Traditionally, semantic parsers were constructed manually, but this is too costly and brittle. Recently, a number of machine learning approaches have been proposed (Zettlemoyer and Collins, 2005; Mooney, 2007). However, they are supervised, and providing the target logical form for each sentence is costly and difficult to do consistently and with high quality. Unsupervised approaches have been applied to shallow semantic tasks (e.g., paraphrasing (Lin and Pantel, 2001), information extraction (Banko et al., 2007)), but not to semantic parsing.

In this paper we develop the first unsupervised approach to semantic parsing, using Markov logic (Richardson and Domingos, 2006). Our USP system starts by clustering tokens of the same type, and then recursively clusters expressions whose subexpressions belong to the same clusters. Experiments on a biomedical corpus show that this approach is able to successfully translate syntactic variations into a logical representation of their common meaning (e.g., USP learns to map active and passive voice to the same logical form, etc.). This in turn allows it to correctly answer many more questions than systems based on TextRunner (Banko et al., 2007) and DIRT (Lin and Pantel, 2001).

We begin by reviewing the necessary background on semantic parsing and Markov logic. We then describe our Markov logic network for unsupervised semantic parsing, and the learning and inference algorithms we used. Finally, we present our experiments and results.

2 Background

2.1 Semantic Parsing

The standard language for formal meaning representation is first-order logic. A term is any expression representing an object in the domain. An atomic formula or atom is a predicate symbol applied to a tuple of terms. Formulas are recursively constructed from atomic formulas using logical connectives and quantifiers. A *lexical entry* defines the logical form for a lexical item (e.g., a word). The semantic parse of a sentence is derived by starting with logical forms in the lexical entries and recursively composing the meaning of larger fragments from their parts. In traditional approaches, the lexical entries and meaning-composition rules are both manually constructed. Below are sample rules in a definite clause grammar (DCG) for parsing the sentence: "Utah borders Idaho".

$$
\begin{array}{l}
Verb[\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{y})] \rightarrow borders \\
NP[\mathtt{Utah}] \rightarrow Utah \\
NP[\mathtt{Idaho}] \rightarrow Idaho \\
VP[\mathtt{rel}(\mathtt{obj})] \rightarrow Verb[\mathtt{rel}] \quad NP[\mathtt{obj}] \\
S[\mathtt{rel}(\mathtt{obj})] \rightarrow NP[\mathtt{obj}] \quad VP[\mathtt{rel}]
\end{array}
$$

The first three lines are lexical entries. They are fired upon seeing the individual words. For example, the first rule applies to the word "borders" and generates syntactic category *Verb* with the meaning $\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{y})$ that represents the next-to relation. Here, we use the standard lambda-calculus notation, where $\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{y})$ represents a function that is true for any $(\mathtt{x},\mathtt{y})$-pair such that $\mathtt{borders}(\mathtt{x},\mathtt{y})$ holds. The last two rules compose the meanings of sub-parts into that of the larger part. For example, after the first and third rules are fired, the fourth rule fires and generates $VP[\lambda \mathtt{y} \lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{y})(\mathtt{Idaho})]$; this meaning simplifies to $\lambda \mathtt{x}.\mathtt{borders}(\mathtt{x},\mathtt{Idaho})$ by the λ-*reduction rule*, which substitutes the argument for a variable in a functional application.

A major challenge to semantic parsing is syntactic variations of the same meaning, which abound in natural languages. For example, the aforementioned sentence can be rephrased as "Utah is next to Idaho,""Utah shares a border with Idaho," etc. Manually encoding all these variations into the grammar is tedious and error-prone. Supervised semantic parsing addresses this issue by learning to construct the grammar automatically from sample meaning annotations (Mooney, 2007). Existing approaches differ in the meaning representation languages they use and the amount of annotation required. In the approach of Zettlemoyer and Collins (2005), the training data consists of sentences paired with their meanings in lambda form. A probabilistic combinatory categorial grammar (PCCG) is learned using a log-linear model, where the probability of the final logical form L and meaning-derivation tree T conditioned on the sentence S is $P(L,T|S) = \frac{1}{Z}\exp\left(\sum_i w_i f_i(L,T,S)\right)$. Here Z is the normalization constant and f_i are the feature functions with weights w_i. Candidate lexical entries are generated by a domain-specific procedure based on the target logical forms.

The major limitation of supervised approaches is that they require meaning annotations for example sentences. Even in a restricted domain, doing this consistently and with high quality requires nontrivial effort. For unrestricted text, the complexity and subjectivity of annotation render it essentially infeasible; even pre-specifying the target predicates and objects is very difficult. Therefore, to apply semantic parsing beyond limited domains, it is crucial to develop unsupervised methods that do not rely on labeled meanings.

In the past, unsupervised approaches have been applied to some semantic tasks, but not to semantic parsing. For example, DIRT (Lin and Pantel, 2001) learns paraphrases of binary relations based on distributional similarity of their arguments; TextRunner (Banko et al., 2007) automatically extracts relational triples in open domains using a self-trained extractor; SNE applies relational clustering to generate a semantic network from TextRunner triples (Kok and Domingos, 2008). While these systems illustrate the promise of unsupervised methods, the semantic content they extract is nonetheless shallow and does not constitute the complete formal meaning that can be obtained by a semantic parser.

Another issue is that existing approaches to semantic parsing learn to parse syntax and semantics together.[1] The drawback is that the complexity in syntactic processing is coupled with semantic parsing and makes the latter even harder. For example, when applying their approach to a different domain with somewhat less rigid syntax, Zettlemoyer and Collins (2007) need to introduce new combinators and new forms of candidate lexical entries. Ideally, we should leverage the enormous progress made in syntactic parsing and generate semantic parses directly from syntactic analysis.

2.2 Markov Logic

In many NLP applications, there exist rich relations among objects, and recent work in statistical relational learning (Getoor and Taskar, 2007) and structured prediction (Bakir et al., 2007) has shown that leveraging these can greatly improve accuracy. One of the most powerful representations for this is Markov logic, which is a probabilistic extension of first-order logic (Richardson and Domingos, 2006). Markov logic makes it

[1]The only exception that we are aware of is Ge and Mooney (2009).

possible to compactly specify probability distributions over complex relational domains, and has been successfully applied to unsupervised coreference resolution (Poon and Domingos, 2008) and other tasks. A *Markov logic network (MLN)* is a set of weighted first-order clauses. Together with a set of constants, it defines a Markov network with one node per ground atom and one feature per ground clause. The weight of a feature is the weight of the first-order clause that originated it. The probability of a state x in such a network is given by the log-linear model $P(x) = \frac{1}{Z}\exp\left(\sum_i w_i n_i(x)\right)$, where Z is a normalization constant, w_i is the weight of the ith formula, and n_i is the number of satisfied groundings.

3 Unsupervised Semantic Parsing with Markov Logic

Unsupervised semantic parsing (**USP**) rests on three key ideas. First, the target predicate and object constants, which are pre-specified in supervised semantic parsing, can be viewed as clusters of syntactic variations of the same meaning, and can be learned from data. For example, `borders` represents the next-to relation, and can be viewed as the cluster of different forms for expressing this relation, such as "borders", "is next to", "share the border with"; `Utah` represents the state of Utah, and can be viewed as the cluster of "Utah", "the beehive state", etc.

Second, the identification and clustering of candidate forms are integrated with the learning for meaning composition, where forms that are used in composition with the same forms are encouraged to cluster together, and so are forms that are composed of the same sub-forms. This amounts to a novel form of relational clustering, where clustering is done not just on fixed elements in relational tuples, but on arbitrary forms that are built up recursively.

Third, while most existing approaches (manual or supervised learning) learn to parse both syntax and semantics, unsupervised semantic parsing starts directly from syntactic analyses and focuses solely on translating them to semantic content. This enables us to leverage advanced syntactic parsers and (indirectly) the available rich resources for them. More importantly, it separates the complexity in syntactic analysis from the semantic one, and makes the latter much easier to perform. In particular, meaning composition does not require domain-specific procedures for generating candidate lexicons, as is often needed by supervised methods.

The input to our USP system consists of dependency trees of training sentences. Compared to phrase-structure syntax, dependency trees are the more appropriate starting point for semantic processing, as they already exhibit much of the relation-argument structure at the lexical level.

USP first uses a deterministic procedure to convert dependency trees into quasi-logical forms (QLFs). The QLFs and their sub-formulas have natural lambda forms, as will be described later. Starting with clusters of lambda forms at the atom level, USP recursively builds up clusters of larger lambda forms. The final output is a probability distribution over lambda-form clusters and their compositions, as well as the MAP semantic parses of training sentences.

In the remainder of the section, we describe the details of USP. We first present the procedure for generating QLFs from dependency trees. We then introduce their lambda forms and clusters, and show how semantic parsing works in this setting. Finally, we present the Markov logic network (MLN) used by USP. In the next sections, we present efficient algorithms for learning and inference with this MLN.

3.1 Derivation of Quasi-Logical Forms

A *dependency tree* is a tree where nodes are words and edges are dependency labels. To derive the QLF, we convert each node to an unary atom with the predicate being the lemma plus POS tag (below, we still use the word for simplicity), and each edge to a binary atom with the predicate being the dependency label. For example, the node for Utah becomes $\mathtt{Utah}(\mathtt{n}_1)$ and the subject dependency becomes $\mathtt{nsubj}(\mathtt{n1},\mathtt{n2})$. Here, the $\mathtt{n_i}$ are Skolem constants indexed by the nodes. The QLF for a sentence is the conjunction of the atoms for the nodes and edges, e.g., the sentence above will become $\mathtt{borders}(\mathtt{n}_1) \wedge \mathtt{Utah}(\mathtt{n}_2) \wedge \mathtt{Idaho}(\mathtt{n}_3) \wedge \mathtt{nsubj}(\mathtt{n}_1,\mathtt{n}_2) \wedge \mathtt{dobj}(\mathtt{n}_1,\mathtt{n}_3)$.

3.2 Lambda-Form Clusters and Semantic Parsing in USP

Given a QLF, a relation or an object is represented by the conjunction of a subset of the atoms. For example, the next-to relation is represented by $\mathtt{borders}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1,\mathtt{n}_2) \wedge \mathtt{dobj}(\mathtt{n}_1,\mathtt{n}_3)$, and the states of Utah and Idaho are represented by $\mathtt{Utah}(\mathtt{n}_2)$ and $\mathtt{Idaho}(\mathtt{n}_3)$. The meaning composition of two sub-formulas is simply their conjunction. This allows the maximum flexibility in learning. In particular, lexical entries are no longer limited to be adjacent words as in Zettlemoyer and Collins (2005), but can be arbitrary fragments in a dependency tree.

For every sub-formula F, we define a corresponding lambda form that can be derived by replacing every Skolem constant $\mathtt{n}_\mathtt{i}$ that does not appear in any unary atom in F with a unique lambda variable $\mathtt{x}_\mathtt{i}$. Intuitively, such constants represent objects introduced somewhere else (by the unary atoms containing them), and correspond to the arguments of the relation represented by F. For example, the lambda form for $\mathtt{borders}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1, \mathtt{n}_2) \wedge \mathtt{dobj}(\mathtt{n}_1, \mathtt{n}_3)$ is $\lambda \mathtt{x}_2 \lambda \mathtt{x}_3.\ \mathtt{borders}(\mathtt{n}_1) \wedge \mathtt{nsubj}(\mathtt{n}_1, \mathtt{x}_2) \wedge \mathtt{dobj}(\mathtt{n}_1, \mathtt{x}_3)$.

Conceptually, a lambda-form cluster is a set of semantically interchangeable lambda forms. For example, to express the meaning that Utah borders Idaho, we can use any form in the cluster representing the next-to relation (e.g., "borders", "shares a border with"), any form in the cluster representing the state of Utah (e.g., "the beehive state"), and any form in the cluster representing the state of Idaho (e.g., "Idaho"). Conditioned on the clusters, the choices of individual lambda forms are independent of each other.

To handle variable number of arguments, we follow Davidsonian semantics and further decompose a lambda form into the *core form*, which does not contain any lambda variable (e.g., $\mathtt{borders}(\mathtt{n}_1)$), and the *argument forms*, which contain a single lambda variable (e.g., $\lambda \mathtt{x}_2.\mathtt{nsubj}(\mathtt{n}_1, \mathtt{x}_2)$ and $\lambda \mathtt{x}_3.\mathtt{dobj}(\mathtt{n}_1, \mathtt{x}_3)$). Each lambda-form cluster may contain some number of *argument types*, which cluster distinct forms of the same argument in a relation. For example, in Stanford dependencies, the object of a verb uses the dependency dobj in the active voice, but nsubjpass in passive.

Lambda-form clusters abstract away syntactic variations of the same meaning. Given an instance of cluster T with arguments of argument types $\mathtt{A}_1, \cdots, \mathtt{A}_\mathtt{k}$, its *abstract lambda form* is given by $\lambda \mathtt{x}_1 \cdots \lambda \mathtt{x}_\mathtt{k}.\mathtt{T}(\mathtt{n}) \wedge \bigwedge_{\mathtt{i}=1}^{\mathtt{k}} \mathtt{A}_\mathtt{i}(\mathtt{n}, \mathtt{x}_\mathtt{i})$.

Given a sentence and its QLF, semantic parsing amounts to partitioning the atoms in the QLF, dividing each part into core form and argument forms, and then assigning each form to a cluster or an argument type. The final logical form is derived by composing the abstract lambda forms of the parts using the λ-reduction rule.[2]

3.3 The USP MLN

Formally, for a QLF Q, a semantic parse L partitions Q into parts $\mathtt{p}_1, \mathtt{p}_2, \cdots, \mathtt{p}_\mathtt{n}$; each part p is assigned to some lambda-form cluster c, and is further partitioned into core form f and argument forms $\mathtt{f}_1, \cdots, \mathtt{f}_\mathtt{k}$; each argument form is assigned to an argument type a in c. The USP MLN defines a joint probability distribution over Q and L by modeling the distributions over forms and arguments given the cluster or argument type.

Before presenting the predicates and formulas in our MLN, we should emphasize that they should not be confused with the atoms and formulas in the QLFs, which are represented by reified constants and variables.

To model distributions over lambda forms, we introduce the predicates $\mathtt{Form}(\mathtt{p}, \mathtt{f}!)$ and $\mathtt{ArgForm}(\mathtt{p}, \mathtt{i}, \mathtt{f}!)$, where p is a part, i is the index of an argument, and f is a QLF subformula. $\mathtt{Form}(\mathtt{p}, \mathtt{f})$ is true iff part p has core form f, and $\mathtt{ArgForm}(\mathtt{p}, \mathtt{i}, \mathtt{f})$ is true iff the ith argument in p has form f.[3] The "f!" notation signifies that each part or argument can have only one form.

To model distributions over arguments, we introduce three more predicates: $\mathtt{ArgType}(\mathtt{p}, \mathtt{i}, \mathtt{a}!)$ signifies that the ith argument of p is assigned to argument type a; $\mathtt{Arg}(\mathtt{p}, \mathtt{i}, \mathtt{p}')$ signifies that the ith argument of p is $\mathtt{p}'$; $\mathtt{Number}(\mathtt{p}, \mathtt{a}, \mathtt{n})$ signifies that there are n arguments of p that are assigned to type a. The truth value of $\mathtt{Number}(\mathtt{p}, \mathtt{a}, \mathtt{n})$ is determined by the ArgType atoms.

Unsupervised semantic parsing can be captured by four formulas:

$$\begin{gathered}
\mathtt{p} \in +\mathtt{c} \wedge \mathtt{Form}(\mathtt{p}, +\mathtt{f}) \\
\mathtt{ArgType}(\mathtt{p}, \mathtt{i}, +\mathtt{a}) \wedge \mathtt{ArgForm}(\mathtt{p}, \mathtt{i}, +\mathtt{f}) \\
\mathtt{Arg}(\mathtt{p}, \mathtt{i}, \mathtt{p}') \wedge \mathtt{ArgType}(\mathtt{p}, \mathtt{i}, +\mathtt{a}) \wedge \mathtt{p}' \in +\mathtt{c}' \\
\mathtt{Number}(\mathtt{p}, +\mathtt{a}, +\mathtt{n})
\end{gathered}$$

All free variables are implicitly universally quantified. The "+" notation signifies that the MLN contains an instance of the formula, with a separate weight, for each value combination of the

[2]Currently, we do not handle quantifier scoping or semantics for specific closed-class words such as determiners. These will be pursued in future work.

[3]There are hard constraints to guarantee that these assignments form a legal partition. We omit them for simplicity.

variables with a plus sign. The first formula models the mixture of core forms given the cluster, and the others model the mixtures of argument forms, argument clusters, and argument numbers, respectively, given the argument type.

To encourage clustering and avoid overfitting, we impose an exponential prior with weight α on the number of parameters.[4]

The MLN above has one problem: it often clusters expressions that are semantically opposite. For example, it clusters antonyms like "elderly/young", "mature/immature". This issue also occurs in other semantic-processing systems (e.g., DIRT). In general, this is a difficult open problem that only recently has started to receive some attention (Mohammad et al., 2008). Resolving this is not the focus of this paper, but we describe a general heuristic for fixing this problem. We observe that the problem stems from the lack of negative features for discovering meanings in contrast. In natural languages, parallel structures like conjunctions are one such feature.[5] We thus introduce an exponential prior with weight β on the number of conjunctions where the two conjunctive parts are assigned to the same cluster. To detect conjunction, we simply used the Stanford dependencies that begin with "conj". This proves very effective, fixing the majority of the errors in our experiments.

4 Inference

Given a sentence and the quasi-logical form Q derived from its dependency tree, the conditional probability for a semantic parse L is given by $Pr(L|Q) \propto \exp\left(\sum_i w_i n_i(L,Q)\right)$. The MAP semantic parse is simply $\arg\max_L \sum_i w_i n_i(L,Q)$. Enumerating all L's is intractable. It is also unnecessary, since most partitions will result in parts whose lambda forms have no cluster they can be assigned to. Instead, USP uses a greedy algorithm to search for the MAP parse. First we introduce some definitions: a partition is called *λ-reducible from* p if it can be obtained from the current partition by recursively λ-reducing the part containing p with one of its arguments; such a partition is called *feasible* if the core form of the new part is contained in some cluster. For example, consider the QLF of "Utah borders Idaho" and assume that the current partition is $\lambda x_2 x_3.\mathtt{borders}(n_1) \wedge \mathtt{nsubj}(n_1, x_2) \wedge \mathtt{dobj}(n_1, x_3)$, $\mathtt{Utah}(n_2)$, $\mathtt{Idaho}(n_3)$. Then the following partition is λ-reducible from the first part in the above partition: $\lambda x_3.\mathtt{borders}(n_1) \wedge \mathtt{nsubj}(n_1, n_2) \wedge \mathtt{Utah}(n_2) \wedge \mathtt{dobj}(n_1, x_3)$, $\mathtt{Idaho}(n_3)$. Whether this new partition is feasible depends on whether the core form of the new part $\lambda x_3.\mathtt{borders}(n_1) \wedge \mathtt{nsubj}(n_1, n_2) \wedge \mathtt{Utah}(n_2) \wedge \mathtt{dobj}(n_1, x_3)$ (i.e. $\mathtt{borders}(n_1) \wedge \mathtt{nsubj}(n_1, n_2) \wedge \mathtt{Utah}(n_2)$) is contained in some lambda-form cluster.

Algorithm 1 USP-Parse(*MLN, QLF*)

```
Form parts for individual atoms in $QLF$ and assign each to its most probable cluster
**repeat**
  **for all** parts p in the current partition **do**
    **for all** partitions that are $\lambda$-reducible from p and feasible **do**
      Find the most probable cluster and argument type assignments for the new part and its arguments
    **end for**
  **end for**
  Change to the new partition and assignments with the highest gain in probability
**until** none of these improve the probability
**return** current partition and assignments
```

Algorithm 1 gives pseudo-code for our algorithm. Given part p, finding partitions that are λ-reducible from p and feasible can be done in time $O(ST)$, where S is the size of the clustering in the number of core forms and T is the maximum number of atoms in a core form. We omit the proof here but point out that it is related to the unordered subtree matching problem which can be solved in linear time (Kilpelainen, 1992). Inverted indexes (e.g., from p to eligible core forms) are used to further improve the efficiency. For a new part p and a cluster that contains p's core form, there are k^m ways of assigning p's m arguments to the k argument types of the cluster. For larger k and m, this is very expensive. We therefore approximate it by assigning each argument to the best type, independent of other arguments.

This algorithm is very efficient, and is used repeatedly in learning.

[4]Excluding weights of ∞ or $-\infty$, which signify hard constraints.

[5]For example, in the sentence "IL-2 inhibits X in A and induces Y in B", the conjunction between "inhibits" and "induces" suggests that they are different. If "inhibits" and "induces" are indeed synonyms, such a sentence will sound awkward and would probably be rephrased as "IL-2 inhibits X in A and Y in B".

5 Learning

The learning problem in USP is to maximize the log-likelihood of observing the QLFs obtained from the dependency trees, denoted by Q, summing out the unobserved semantic parses:

$$\begin{aligned} L_\theta(Q) &= \log P_\theta(Q) \\ &= \log \textstyle\sum_L P_\theta(Q, L) \end{aligned}$$

Here, L are the semantic parses, θ are the MLN parameters, and $P_\theta(Q, L)$ are the completion likelihoods. A serious challenge in unsupervised learning is the identifiability problem (i.e., the optimal parameters are not unique) (Liang and Klein, 2008). This problem is particularly severe for log-linear models with hard constraints, which are common in MLNs. For example, in our USP MLN, conditioned on the fact that $\mathtt{p} \in \mathtt{c}$, there is exactly one value of $\mathtt{f}$ that can satisfy the formula $\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form(p, f)}$, and if we add some constant number to the weights of $\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form(p, f)}$ for all $\mathtt{f}$, the probability distribution stays the same.[6] The learner can be easily confused by the infinitely many optima, especially in the early stages. To address this problem, we impose local normalization constraints on specific groups of formulas that are mutually exclusive and exhaustive, i.e., in each group, we require that $\sum_{i=1}^{k} e^{w_i} = 1$, where w_i are the weights of formulas in the group. Grouping is done in such a way as to encourage the intended mixture behaviors. Specifically, for the rule $\mathtt{p} \in +\mathtt{c} \wedge \mathtt{Form(p, +f)}$, all instances given a fixed $\mathtt{c}$ form a group; for each of the remaining three rules, all instances given a fixed $\mathtt{a}$ form a group. Notice that with these constraints the completion likelihood $P(Q, L)$ can be computed in closed form for any L. In particular, each formula group contributes a term equal to the weight of the currently satisfied formula. In addition, the optimal weights that maximize the completion likelihood $P(Q, L)$ can be derived in closed form using empirical relative frequencies. E.g., the optimal weight of $\mathtt{p} \in \mathtt{c} \wedge \mathtt{Form(p, f)}$ is $\log(n_{\mathtt{c,f}}/n_{\mathtt{c}})$, where $n_{\mathtt{c,f}}$ is the number of parts $\mathtt{p}$ that satisfy both $\mathtt{p} \in \mathtt{c}$ and $\mathtt{Form(p, f)}$, and $n_{\mathtt{c}}$ is the number of parts p that satisfy $\mathtt{p} \in \mathtt{c}$.[7] We leverage this fact for efficient learning in USP.

[6]Regularizations, e.g., Gaussian priors on weights, alleviate this problem by penalizing large weights, but it remains true that weights within a short range are roughly equivalent.

[7]To see this, notice that for a given $\mathtt{c}$, the total contribution to the completion likelihood from all groundings in its formula group is $\sum_{\mathtt{f}} w_{\mathtt{c,f}} n_{\mathtt{c,f}}$. In addition, $\sum_{\mathtt{f}} n_{\mathtt{c,f}} = n_{\mathtt{c}}$ and there is the local normalization constraint $\sum_{\mathtt{f}} e^{w_{\mathtt{c,f}}} = 1$. The optimal weights $w_{\mathtt{c,f}}$ are easily derived by solving this constrained optimization problem.

Algorithm 2 USP-Learn(*MLN*, *QLFs*)

```
Create initial clusters and semantic parses
Merge clusters with the same core form
Agenda ← ∅
repeat
  for all candidate operations $O$ do
    Score $O$ by log-likelihood improvement
    if score is above a threshold then
      Add $O$ to agenda
    end if
  end for
  Execute the highest scoring operation $O^*$ in the agenda
  Regenerate MAP parses for affected QLFs and update agenda and candidate operations
until agenda is empty
return the MLN with learned weights and the semantic parses
```

Another major challenge in USP learning is the summation in the likelihood, which is over all possible semantic parses for a given dependency tree. Even an efficient sampler like MC-SAT (Poon and Domingos, 2006), as used in Poon & Domingos (2008), would have a hard time generating accurate estimates within a reasonable amount of time. On the other hand, as already noted in the previous section, the lambda-form distribution is generally sparse. Large lambda-forms are rare, as they correspond to complex expressions that are often decomposable into smaller ones. Moreover, while ambiguities are present at the lexical level, they quickly diminish when more words are present. Therefore, a lambda form can usually only belong to a small number of clusters, if not a unique one. We thus simplify the problem by approximating the sum with the mode, and search instead for the L and θ that maximize $\log P_\theta(Q, L)$. Since the optimal weights and log-likelihood can be derived in closed form given the semantic parses L, we simply search over semantic parses, evaluating them using log-likelihood.

Algorithm 2 gives pseudo-code for our algorithm. The input consists of an MLN without weights and the QLFs for the training sentences. Two operators are used for updating semantic parses. The first is to merge two clusters, denoted by $\mathtt{MERGE}(\mathtt{C_1}, \mathtt{C_2})$ for clusters $\mathtt{C_1}, \mathtt{C_2}$, which does the following:

1. Create a new cluster $\mathtt{C}$ and add all core forms in $\mathtt{C}_1, \mathtt{C}_2$ to $\mathtt{C}$;
2. Create new argument types for $\mathtt{C}$ by merging those in $\mathtt{C}_1, \mathtt{C}_2$ so as to maximize the log-likelihood;
3. Remove $\mathtt{C}_1, \mathtt{C}_2$.

Here, merging two argument types refers to pooling their argument forms to create a new argument type. Enumerating all possible ways of creating new argument types is intractable. USP approximates it by considering one type at a time and either creating a new type for it or merging it to types already considered, whichever maximizes the log-likelihood. The types are considered in decreasing order of their numbers of occurrences so that more information is available for each decision. MERGE clusters syntactically different expressions whose meanings appear to be the same according to the model.

The second operator is to create a new cluster by composing two existing ones, denoted by $\mathtt{COMPOSE}(\mathtt{C_R}, \mathtt{C_A})$, which does the following:

1. Create a new cluster $\mathtt{C}$;
2. Find all parts $\mathtt{r} \in \mathtt{C_R}, \mathtt{a} \in \mathtt{C_A}$ such that $\mathtt{a}$ is an argument of $\mathtt{r}$, compose them to $\mathtt{r}(\mathtt{a})$ by λ-reduction and add the new part to $\mathtt{C}$;
3. Create new argument types for $\mathtt{C}$ from the argument forms of $\mathtt{r}(\mathtt{a})$ so as to maximize the log-likelihood.

COMPOSE creates clusters of large lambda-forms if they tend to be composed of the same subforms (e.g., the lambda form for "is next to"). These lambda-forms may later be merged with other clusters (e.g., borders).

At learning time, USP maintains an *agenda* that contains operations that have been evaluated and are pending execution. During initialization, USP forms a part and creates a new cluster for each unary atom $\mathtt{u}(\mathtt{n})$. It also assigns binary atoms of the form $\mathtt{b}(\mathtt{n}, \mathtt{n}')$ to the part as argument forms and creates a new argument type for each. This forms the initial clustering and semantic parses. USP then merges clusters with the same core form (i.e., the same unary predicate) using MERGE.[8] At each step, USP evaluates the candidate operations and adds them to the agenda if the improvement is above a threshold.[9] The operation with the highest score is executed, and the parameters are updated with the new optimal values. The QLFs which contain an affected part are reparsed, and operations in the agenda whose score might be affected are re-evaluated. These changes are done very efficiently using inverted indexes. We omit the details here due to space limitations. USP terminates when the agenda is empty, and outputs the current MLN parameters and semantic parses.

USP learning uses the same optimization objective as hard EM, and is also guaranteed to find a local optimum since at each step it improves the log-likelihood. It differs from EM in directly optimizing the likelihood instead of a lower bound.

6 Experiments

6.1 Task

Evaluating unsupervised semantic parsers is difficult, because there is no predefined formal language or gold logical forms for the input sentences. Thus the best way to test them is by using them for the ultimate goal: answering questions based on the input corpus. In this paper, we applied USP to extracting knowledge from biomedical abstracts and evaluated its performance in answering a set of questions that simulate the information needs of biomedical researchers. We used the GENIA dataset (Kim et al., 2003) as the source for knowledge extraction. It contains 1999 PubMed abstracts and marks all mentions of biomedical entities according to the GENIA ontology, such as cell, protein, and DNA. As a first approximation to the questions a biomedical researcher might ask, we generated a set of two thousand questions on relations between entities. Sample questions are: "What regulates MIP-1alpha?", "What does anti-STAT 1 inhibit?". To simulate the real information need, we sample the relations from the 100 most frequently used verbs (excluding the auxiliary verbs *be*, *have*, and *do*), and sample the entities from those annotated in GENIA, both according to their numbers of occurrences. We evaluated USP by the number of answers it provided and the accuracy as determined by manual labeling.[10]

[8]Word-sense disambiguation can be handled by including a new kind of operator that splits a cluster into subclusters. We leave this to future work.

[9]We currently set it to 10 to favor precision and guard against errors due to inexact estimates.

[10]The labels and questions are available at http://alchemy.cs.washington.edu/papers/poon09.

6.2 Systems

Since USP is the first unsupervised semantic parser, conducting a meaningful comparison of it with other systems is not straightforward. Standard question-answering (QA) benchmarks do not provide the most appropriate comparison, because they tend to simultaneously emphasize other aspects not directly related to semantic parsing. Moreover, most state-of-the-art QA systems use supervised learning in their key components and/or require domain-specific engineering efforts. The closest available system to USP in aims and capabilities is TextRunner (Banko et al., 2007), and we compare with it. TextRunner is the state-of-the-art system for open-domain information extraction; its goal is to extract knowledge from text without using supervised labels. Given that a central challenge to semantic parsing is resolving syntactic variations of the same meaning, we also compare with RESOLVER (Yates and Etzioni, 2009), a state-of-the-art unsupervised system based on TextRunner for jointly resolving entities and relations, and DIRT (Lin and Pantel, 2001), which resolves paraphrases of binary relations. Finally, we also compared to an informed baseline based on keyword matching.

Keyword: We consider a baseline system based on keyword matching. The question substring containing the verb and the available argument is directly matched with the input text, ignoring case and morphology. We consider two ways to derive the answer given a match. The first one (**KW**) simply returns the rest of sentence on the other side of the verb. The second one (**KW-SYN**) is informed by syntax: the answer is extracted from the subject or object of the verb, depending on the question. If the verb does not contain the expected argument, the sentence is ignored.

TextRunner: TextRunner inputs text and outputs relational triples in the form (R, A_1, A_2), where R is the relation string, and A_1, A_2 the argument strings. Given a triple and a question, we first match their relation strings, and then match the strings for the argument that is present in the question. If both match, we return the other argument string in the triple as an answer. We report results when exact match is used (**TR-EXACT**), or when the triple string can contain the question one as a substring (**TR-SUB**).

RESOLVER: RESOLVER (Yates and Etzioni, 2009) inputs TextRunner triples and collectively resolves coreferent relation and argument strings. On the GENIA data, using the default parameters, RESOLVER produces only a few trivial relation clusters and no argument clusters. This is not surprising, since RESOLVER assumes high redundancy in the data, and will discard any strings with fewer than 25 extractions. For a fair comparison, we also ran RESOLVER using all extractions, and manually tuned the parameters based on eyeballing of clustering quality. The best result was obtained with 25 rounds of execution and with the entity multiple set to 200 (the default is 30). To answer questions, the only difference from TextRunner is that a question string can match any string in its cluster. As in TextRunner, we report results for both exact match (**RS-EXACT**) and substring (**RS-SUB**).

DIRT: The DIRT system inputs a path and returns a set of similar paths. To use DIRT in question answering, we queried it to obtain similar paths for the relation of the question, and used these paths while matching sentences. We first used MINIPAR (Lin, 1998) to parse input text using the same dependencies as DIRT. To determine a match, we first check if the sentence contains the question path or one of its DIRT paths. If so, and if the available argument slot in the question is contained in the one in the sentence, it is a match, and we return the other argument slot from the sentence if it is present. Ideally, a fair comparison will require running DIRT on the GENIA text, but we were not able to obtain the source code. We thus resorted to using the latest DIRT database released by the author, which contains paths extracted from a large corpus with more than 1GB of text. This puts DIRT in a very advantageous position compared with other systems. In our experiments, we used the top three similar paths, as including more results in very low precision.

USP: We built a system for knowledge extraction and question answering on top of USP. It generated Stanford dependencies (de Marneffe et al., 2006) from the input text using the Stanford parser, and then fed these to USP-Learn[11], which produced an MLN with learned weights and the MAP semantic parses of the input sentences. These MAP parses formed our knowledge base (KB). To answer questions, the system first parses the questions[12] using USP-Parse with the

[11]α and β are set to -5 and -10.

[12]The question slot is replaced by a dummy word.

Table 1: Comparison of question answering results on the GENIA dataset.

	# Total	# Correct	Accuracy
KW	150	67	45%
KW-SYN	87	67	77%
TR-EXACT	29	23	79%
TR-SUB	152	81	53%
RS-EXACT	53	24	45%
RS-SUB	196	81	41%
DIRT	159	94	59%
USP	**334**	**295**	**88%**

learned MLN, and then matches the question parse to parses in the KB by testing subsumption (i.e., a question parse matches a KB one iff the former is subsumed by the latter). When a match occurs, our system then looks for arguments of type in accordance with the question. For example, if the question is "What regulates MIP-1alpha?", it searches for the argument type of the relation that contains the argument form "nsubj" for subject. If such an argument exists for the relation part, it will be returned as the answer.

6.3 Results

Table 1 shows the results for all systems. USP extracted the highest number of answers, almost doubling that of the second highest (RS-SUB). It obtained the highest accuracy at 88%, and the number of correct answers it extracted is three times that of the second highest system. The informed baseline (KW-SYN) did surprisingly well compared to systems other than USP, in terms of accuracy and number of correct answers. TextRunner achieved good accuracy when exact match is used (TR-EXACT), but only obtained a fraction of the answers compared to USP. With substring match, its recall substantially improved, but precision dropped more than 20 points. RESOLVER improved the number of extracted answers by sanctioning more matches based on the clusters it generated. However, most of those additional answers are incorrect due to wrong clustering. DIRT obtained the second highest number of correct answers, but its precision is quite low because the similar paths contain many errors.

6.4 Qualitative Analysis

Manual inspection shows that USP is able to resolve many nontrivial syntactic variations without user supervision. It consistently resolves the syntactic difference between active and passive voices. It successfully identifies many distinct argument forms that mean the same (e.g., "X stimulates Y" $\approx$ "Y is stimulated with X", "expression of X" $\approx$ "X expression"). It also resolves many nouns correctly and forms meaningful groups of relations. Here are some sample clusters in core forms:

{investigate, examine, evaluate, analyze, study, assay}

{diminish, reduce, decrease, attenuate}

{synthesis, production, secretion, release}

{dramatically, substantially, significantly}

An example question-answer pair, together with the source sentence, is shown below:

Q: What does IL-13 enhance?

A: The 12-lipoxygenase activity of murine macrophages.

Sentence: The data presented here indicate that (1) the 12-lipoxygenase activity of murine macrophages is upregulated in vitro and in vivo by IL-4 and/or IL-13, . . .

7 Conclusion

This paper introduces the first unsupervised approach to learning semantic parsers. Our USP system is based on Markov logic, and recursively clusters expressions to abstract away syntactic variations of the same meaning. We have successfully applied USP to extracting a knowledge base from biomedical text and answering questions based on it.

Directions for future work include: better handling of antonyms, subsumption relations among expressions, quantifier scoping, more complex lambda forms, etc.; use of context and discourse to aid expression clustering and semantic parsing; more efficient learning and inference; application to larger corpora; etc.

8 Acknowledgements

We thank the anonymous reviewers for their comments. This research was partly funded by ARO grant W911NF-08-1-0242, DARPA contracts FA8750-05-2-0283, FA8750-07-D-0185, HR0011-06-C-0025, HR0011-07-C-0060 and NBCH-D030010, NSF grants IIS-0534881 and IIS-0803481, and ONR grant N00014-08-1-0670. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, DARPA, NSF, ONR, or the United States Government.

References

G. Bakir, T. Hofmann, B. B. Schölkopf, A. Smola, B. Taskar, S. Vishwanathan, and (eds.). 2007. *Predicting Structured Data*. MIT Press, Cambridge, MA.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 2670–2676, Hyderabad, India. AAAI Press.

Xavier Carreras and Luis Marquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages 89–97, Boston, MA. ACL.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454, Genoa, Italy. ELRA.

Ruifang Ge and Raymond J. Mooney. 2009. Learning a compositional semantic parser using an existing syntactic parser. In *Proceedings of the Forty Seventh Annual Meeting of the Association for Computational Linguistics*, Singapore. ACL.

Lise Getoor and Ben Taskar, editors. 2007. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Pekka Kilpelainen. 1992. *Tree Matching Problems with Applications to Structured Text databases*. Ph.D. Thesis, Department of Computer Science, University of Helsinki.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:180–82.

Stanley Kok and Pedro Domingos. 2008. Extracting semantic networks from text via relational clustering. In *Proceedings of the Nineteenth European Conference on Machine Learning*, pages 624–639, Antwerp, Belgium. Springer.

Percy Liang and Dan Klein. 2008. Analyzing the errors of unsupervised learning. In *Proceedings of the Forty Sixth Annual Meeting of the Association for Computational Linguistics*, pages 879–887, Columbus, OH. ACL.

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*, pages 323–328, San Francisco, CA. ACM Press.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, Granada, Spain. ELRA.

Saif Mohammad, Bonnie Dorr, and Graeme Hirst. 2008. Computing word-pair antonymy. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 982–991, Honolulu, HI. ACL.

Raymond J. Mooney. 2007. Learning for semantic parsing. In *Proceedings of the Eighth International Conference on Computational Linguistics and Intelligent Text Processing*, pages 311–324, Mexico City, Mexico. Springer.

Hoifung Poon and Pedro Domingos. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty First National Conference on Artificial Intelligence*, pages 458–463, Boston, MA. AAAI Press.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 649–658, Honolulu, HI. ACL.

Matt Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62:107–136.

Alexander Yates and Oren Etzioni. 2009. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammers. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh, Scotland. AUAI Press.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 878–887, Prague, Czech. ACL.