Confidence Driven Unsupervised Semantic Parsing

Dan Goldwasser * **Roi Reichart** † **James Clarke** * **Dan Roth** *
* Department of Computer Science, University of Illinois at Urbana-Champaign
{goldwas1,clarkeje,danr}@illinois.edu
† Computer Science and Artificial Intelligence Laboratory, MIT
roiri@csail.mit.edu

Abstract

Current approaches for semantic parsing take a supervised approach requiring a considerable amount of training data which is expensive and difficult to obtain. This supervision bottleneck is one of the major difficulties in scaling up semantic parsing.

We argue that a semantic parser can be trained effectively without annotated data, and introduce an unsupervised learning algorithm. The algorithm takes a self training approach driven by confidence estimation. Evaluated over Geoquery, a standard dataset for this task, our system achieved 66% accuracy, compared to 80% of its fully supervised counterpart, demonstrating the promise of unsupervised approaches for this task.

1 Introduction

Semantic parsing, the ability to transform Natural Language (NL) input into a formal Meaning Representation (MR), is one of the longest standing goals of natural language processing. The importance of the problem stems from both theoretical and practical reasons, as the ability to convert NL into a formal MR has countless applications.

The term *semantic parsing* has been used ambiguously to refer to several semantic tasks (e.g., semantic role labeling). We follow the most common definition of this task: finding a mapping between NL input and its interpretation expressed in a well-defined formal MR language. Unlike shallow semantic analysis tasks, the output of a semantic parser is complete and unambiguous to the extent it can be understood or even executed by a computer system.

Current approaches for this task take a data driven approach (Zettlemoyer and Collins, 2007; Wong and Mooney, 2007), in which the learning algorithm is given a set of NL sentences as input and their corresponding MR, and learns a statistical semantic parser — a set of parameterized rules mapping lexical items and syntactic patterns to their MR. Given a sentence, these rules are applied recursively to derive the most probable interpretation.

Since semantic interpretation is limited to the syntactic patterns observed in the training data, in order to work well these approaches require considerable amounts of annotated data. Unfortunately annotating sentences with their MR is a time consuming task which requires specialized domain knowledge and therefore minimizing the supervision effort is one of the key challenges in scaling semantic parsers.

In this work we present the first unsupervised approach for this task. Our model compensates for the lack of training data by employing a self training protocol based on identifying high confidence self labeled examples and using them to retrain the model. We base our approach on a simple observation: semantic parsing is a difficult structured prediction task, which requires learning a complex model, however identifying good predictions can be done with a far simpler model capturing repeating patterns in the predicted data. We present several simple, yet highly effective confidence measures capturing such patterns, and show how to use them to train a semantic parser without manually annotated sentences.

Our basic premise, that predictions with high confidence score are of high quality, is further used to improve the performance of the unsupervised train-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1486–1495,
Portland, Oregon, June 19-24, 2011.

ing procedure. Our learning algorithm takes an EM-like iterative approach, in which the predictions of the previous stage are used to bias the model. While this basic scheme was successfully applied to many unsupervised tasks, it is known to converge to a sub optimal point. We show that by using confidence estimation as a proxy for the model's prediction quality, the learning algorithm can identify a better model compared to the default convergence criterion.

We evaluate our learning approach and model on the well studied Geoquery domain (Zelle and Mooney, 1996; Tang and Mooney, 2001), consisting of natural language questions and their prolog interpretations used to query a database consisting of U.S. geographical information. Our experimental results show that using our approach we are able to train a good semantic parser without annotated data, and that using a confidence score to identify good models results in a significant performance improvement.

2 Semantic Parsing

We formulate semantic parsing as a structured prediction problem, mapping a NL input sentence (denoted $\mathbf{x}$), to its highest ranking MR (denoted $\mathbf{z}$). In order to correctly parametrize and weight the possible outputs, the decision relies on an intermediate representation: an alignment between textual fragments and their meaning representation (denoted $\mathbf{y}$). Fig. 1 describes a concrete example of this terminology. In our experiments the input sentences $\mathbf{x}$ are natural language queries about U.S. geography taken from the Geoquery dataset. The meaning representation $\mathbf{z}$ is a formal language database query, this output representation language is described in Sec. 2.1.

The prediction function, mapping a sentence to its corresponding MR, is formalized as follows:

$$\hat{\mathbf{z}} = F_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y},\mathbf{z}\in\mathcal{Z}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) \qquad (1)$$

Where Φ is a feature function defined over an input sentence $\mathbf{x}$, alignment $\mathbf{y}$ and output $\mathbf{z}$. The weight vector $\mathbf{w}$ contains the model's parameters, whose values are determined by the learning process.

We refer to the $\arg\max$ above as the inference problem. Given an input sentence, solving this inference problem based on Φ and $\mathbf{w}$ is what compromises our semantic parser. In practice the parsing decision is decomposed into smaller decisions (Sec. 2.2). Sec. 4 provides more details about the feature representation and inference procedure used.

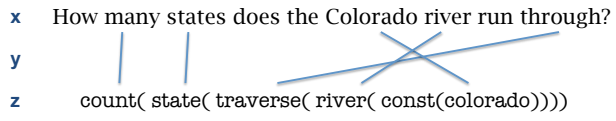

Figure 1: Example of an input sentence ($\mathbf{x}$), meaning representation ($\mathbf{z}$) and the alignment between the two ($\mathbf{y}$) for the Geoquery domain

Current approaches obtain $\mathbf{w}$ using annotated data, typically consisting of ($\mathbf{x}$, $\mathbf{z}$) pairs. In Sec. 3 we describe our unsupervised learning procedure, that is how to obtain $\mathbf{w}$ without annotated data.

2.1 Target Meaning Representation

The output of the semantic parser is a logical formula, grounding the semantics of the input sentence in the domain language (i.e., the Geoquery domain). We use a subset of first order logic consisting of typed constants (corresponding to specific states, etc.) and functions, which capture relations between domains entities and properties of entities (e.g., `population` : $E \rightarrow N$). The semantics of the input sentence is constructed via functional composition, done by the substitution operator. For example, given the function `next_to(x)` and the expression `const(texas)`, substitution replaces the occurrence of the free variable `x` with the expression, resulting in a new formula: `next_to(const(texas))`. For further details we refer the reader to (Zelle and Mooney, 1996).

2.2 Semantic Parsing Decisions

The inference problem described in Eq. 1 selects the top ranking output formula. In practice this decision is decomposed into smaller decisions, capturing local mapping of input tokens to logical fragments and their composition into larger fragments. These decisions are further decomposed into a feature representation, described in Sec. 4.

The first type of decisions are encoded directly by the alignment ($\mathbf{y}$) between the input tokens and their corresponding predicates. We refer to these as **first**

order decisions. The pairs connected by the alignment ($\mathbf{y}$) in Fig. 1 are examples of such decisions.

The final output structure $\mathbf{z}$ is constructed by composing individual predicates into a complete formula. For example, consider the formula presented in Fig. 1: `river( const(colorado))` is a composition of two predicates `river` and `const(colorado)`. We refer to the composition of two predicates, associated with their respective input tokens, as **second order decisions**.

In order to formulate these decisions, we introduce the following notation. c is a constituent in the input sentence $\mathbf{x}$ and $\mathcal{D}$ is the set of all function and constant symbols in the domain. The alignment $\mathbf{y}$ is a set of mappings between constituents and symbols in the domain $\mathbf{y} = \{(c, s)\}$ where $s \in \mathcal{D}$.

We denote by s_i the i-th output predicate composition in $\mathbf{z}$, by $s_{i-1}(s_i)$ the composition of the $(i-1)$-th predicate on the i-th predicate and by $y(s_i)$ the input word corresponding to that predicate according to the alignment $\mathbf{y}$.

3 Unsupervised Semantic Parsing

Our learning framework takes a self training approach in which the learner is iteratively trained over its own predictions. Successful application of this approach depends heavily on two important factors - how to select high quality examples to train the model on, and how to define the learning objective so that learning can halt once a good model is found.

Both of these questions are trivially answered when working in a supervised setting: by using the labeled data for training the model, and defining the learning objective with respect to the annotated data (for example, loss-minimization in the supervised version of our system).

In this work we suggest to address both of the above concerns by approximating the quality of the model's predictions using a confidence measure computed over the statistics of the self generated predictions. Output structures which fall close to the center of mass of these statistics will receive a high confidence score.

The first issue is addressed by using examples assigned a high confidence score to train the model, acting as labeled examples.

We also note that since the confidence score provides a good indication for the model's prediction performance, it can be used to approximate the overall *model* performance, by observing the model's total confidence score over all its predictions. This allows us to set a performance driven goal for our learning process - return the model maximizing the confidence score over all predictions. We describe the details of integrating the confidence score into the learning framework in Sec. 3.1.

Although using the model's prediction score (i.e., $\mathbf{w}^T\Phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$) as an indication of correctness is a natural choice, we argue and show empirically, that unsupervised learning driven by confidence estimation results in a better performing model. This empirical behavior also has theoretical justification: training the model using examples selected according to the model's parameters (i.e., the top ranking structures) may not generalize much further beyond the existing model, as the training examples will simply reinforce the existing model. The statistics used for confidence estimation are different than those used by the model to create the output structures, and can therefore capture additional information unobserved by the prediction model. This assumption is based on the well established idea of multi-view learning, applied successfully to many NL applications (Blum and Mitchell, 1998; Collins and Singer, 1999). According to this idea if two models use different views of the data, each of them can enhance the learning process of the other.

The success of our learning procedure hinges on finding good confidence measures, whose confidence prediction correlates well with the true quality of the prediction. The ability of unsupervised confidence estimation to provide high quality confidence predictions can be explained by the observation that prominent prediction patterns are more likely to be correct. If a non-random model produces a prediction pattern multiple times it is likely to be an indication of an underlying phenomenon in the data, and therefore more likely to be correct. Our specific choice of confidence measures is guided by the intuition that unlike structure prediction (i.e., solving the inference problem) which requires taking statistics over complex and intricate patterns, identifying high quality predictions can be done using much simpler patterns that are significantly easier to capture.

In the reminder of this section we describe our

Algorithm 1 Unsupervised Confidence driven Learning

Input: Sentences $\{\mathbf{x}^l\}_{l=1}^N$,
initial weight vector $\mathbf{w}$

1: **define** $Confidence : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \rightarrow \mathcal{R}$,
$i = 0,\ S_i = \emptyset$
2: **repeat**
3: **for** $l = 1, \ldots, N$ **do**
4: $\hat{\mathbf{y}}, \hat{\mathbf{z}} = \arg\max_{\mathbf{y},\mathbf{z}} \mathbf{w}^T \Phi(\mathbf{x}^l, \mathbf{y}, \mathbf{z})$
5: $S_i = S_i \cup \{\mathbf{x}^l, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$
6: **end for**
7: $Confidence$ = compute confidence statistics
8: S_i^{conf} = select from S_i using $Confidence$
9: $\mathbf{w_i} \leftarrow Learn(\cup_i S_i^{conf})$
10: $i = i + 1$
11: **until** S_i^{conf} has no new unique examples
12: $best = \arg\max_{\mathbf{i}}(\sum_{s \in S_i} Confidence(s))/|S|$
13: **return** $\mathbf{w_{best}}$

learning approach. We begin by introducing the overall learning framework (Sec. 3.1), we then explain the rational behind confidence estimation over self-generated data and introduce the confidence measures used in our experiments (Sec. 3.2). We conclude with a description of the specific learning algorithms used for updating the model (Sec. 3.3).

3.1 Unsupervised Confidence-Driven Learning

Our learning framework works in an EM-like manner, iterating between two stages: making predictions based on its current set of parameters and then retraining the model using a subset of the predictions, assigned high confidence. The learning process "discovers" new high confidence training examples to add to its training set over multiple iterations, and converges when the model no longer adds new training examples.

While this is a natural convergence criterion, it provides no performance guarantees, and in practice it is very likely that the quality of the model (i.e., its performance) fluctuates during the learning process. We follow the observation that confidence estimation can be used to approximate the performance of the entire model and return the model with the highest overall prediction confidence.

We describe this algorithmic framework in detail in Alg. 1. Our algorithm takes as input a set of natural language sentences and a set of parameters used for making the initial predictions[1]. The algorithm then iterates between the two stages - predicting the output structure for each sentence (line 4), and updating the set of parameters (line 9). The specific learning algorithms used are discussed in Sec. 3.3. The training examples required for learning are obtained by selecting high confidence examples - the algorithm first takes statistics over the current predicted set of output structures (line 7), and then based on these statistics computes a confidence score for each structure, selecting the top ranked ones as positive training examples, and if needed, the bottom ones as negative examples (line 8). The set of top confidence examples (for either correct or incorrect prediction), at iteration i of the algorithm, is denoted S_i^{conf}. The exact nature of the confidence computation is discussed in Sec. 3.2.

The algorithm iterates between these two stages, at each iteration it adds more self-annotated examples to its training set, learning therefore converges when no new examples are added (line 11). The algorithm keeps track of the models it trained at each stage throughout this process, and returns the one with the highest averaged overall confidence score (lines 12-13). At each stage, the overall confidence score is computed by averaging over all the confidence scores of the predictions made at that stage.

3.2 Unsupervised Confidence Estimation

Confidence estimation is calculated over a batch of input ($\mathbf{x}$) - output ($\mathbf{z}$) pairs. Each pair decomposes into smaller first order and second order decisions (defined Sec. 2.2). Confidence estimation is done by computing the statistics of these decisions, over the entire set of predicted structures. In the rest of this section we introduce the confidence measures used by our system.

Translation Model The first approach essentially constructs a simplified translation model, capturing word-to-predicate mapping patterns. This can be considered as an abstraction of the prediction model: we collapse the intricate feature representation into

[1] Since we commit to the max-score output prediction, rather than summing over all possibilities, we require a reasonable initialization point. We initialized the weight vector using simple, straight-forward heuristics described in Sec. 5.

high level decisions and take statistics over these decisions. Since it takes statistics over considerably less variables than the actual prediction model, we expect this model to make reliable confidence predictions. We consider two variations of this approach, the first constructs a unigram model over the first order decisions and the second a bigram model over the second order decisions. Formally, given a set of predicted structures we define the following confidence scores:

Unigram Score:

$$p(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{|z|} p(s_i|y(s_i))$$

Bigram Score:

$$p(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{|z|} p(s_{i-1}(s_i)|y(s_{i-1}), y(s_i))$$

Structural Proportion Unlike the first approach which decomposes the predicted structure into individual decisions, this approach approximates the model's performance by observing global properties of the structure. We take statistics over the proportion between the number of predicates in $\mathbf{z}$ and the number of words in $\mathbf{x}$.

Given a set of structure predictions S, we compute this proportion for each structure (denoted as $Prop(\mathbf{x}, \mathbf{z})$) and calculate the average proportion over the entire set (denoted as $AvProp(S)$). The confidence score assigned to a given structure $(\mathbf{x}, \mathbf{y})$ is simply the difference between its proportion and the averaged proportion, or formally

$$PropScore(S, (\mathbf{x}, \mathbf{z})) = AvProp(S) - Prop(x, z)$$

This measure captures the global complexity of the predicted structure and penalizes structures which are too complex (high negative values) or too simplistic (high positive values).

Combined The two approaches defined above capture different views of the data, a natural question is then - *can these two measures be combined to provide a more powerful estimation?* We suggest a third approach which combines the first two approaches. It first uses the score produced by the latter approach to filter out unlikely candidates, and then ranks the remaining ones with the former approach and selects those with the highest rank.

3.3 Learning Algorithms

Given a set of self generated structures, the parameter vector can be updated (line 9 in Alg. 1). We consider two learning algorithm for this purpose.

The first is a **binary learning** algorithm, which considers learning as a classification problem, that is finding a set of weights $\mathbf{w}$ that can best separate correct from incorrect structures. The algorithm decomposes each predicted formula and its corresponding input sentence into a feature vector $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$ normalized by the size of the input sentence $|\mathbf{x}|$, and assigns a binary label to this vector[2]. The learning process is defined over both positive and negative training examples. To accommodate that we modify line 8 in Alg. 1, and use the confidence score to select the top ranking examples as positive examples, and the bottom ranking examples as negative examples. We use a linear kernel SVM with squared-hinge loss as the underlying learning algorithm.

The second is a **structured learning** algorithm which considers learning as a ranking problem, i.e., finding a set of weights $\mathbf{w}$ such that the "gold structure" will be ranked on top, preferably by a large margin to allow generalization.The structured learning algorithm can directly use the top ranking predictions of the model (line 8 in Alg. 1) as training data. In this case the underlying algorithm is a structural SVM with squared-hinge loss, using hamming distance as the distance function. We use the cutting-plane method to efficiently optimize the learning process' objective function.

4 Model

Semantic parsing as formulated in Eq. 1 is an inference procedure selecting the top ranked output logical formula. We follow the inference approach in (Roth and Yih, 2007; Clarke et al., 2010) and formalize this process as an Integer Linear Program (ILP). Due to space consideration we provide a brief description, and refer the reader to that paper for more details.

[2]Without normalization longer sentences would have more influence on binary learning problem. Normalization is therefore required to ensure that each sentence contributes equally to the binary learning problem regardless of its length.

4.1 Inference

The inference decision (Eq. 1) is decomposed into smaller decisions, capturing mapping of input tokens to logical fragments (first order) and their composition into larger fragments (second order). We encode a first-order decision as α_{cs}, a binary variable indicating that constituent c is aligned with the logical symbol $\mathtt{s}$. A second-order decision $\beta_{cs,dt}$, is encoded as a binary variable indicating that the symbol $\mathtt{t}$ (associated with constituent d) is an argument of a function $\mathtt{s}$ (associated with constituent c). We frame the inference problem over these decisions:

$$F_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\alpha,\beta} \sum_{c \in \mathbf{x}} \sum_{s \in D} \alpha_{cs} \cdot \mathbf{w}^T \Phi_1(\mathbf{x}, c, s) + \sum_{c,d \in \mathbf{x}} \sum_{s,t \in D} \beta_{cs,dt} \cdot \mathbf{w}^T \Phi_2(\mathbf{x}, c, s, d, t) \quad (2)$$

We restrict the possible assignments to the decision variables, forcing the resulting output formula to be *syntactically* legal, for example by restricting active β-variables to be type consistent, and force the resulting functional composition to be acyclic. We take advantage of the flexible ILP framework, and encode these restrictions as global constraints over Eq. 2. We refer the reader to (Clarke et al., 2010) for a full description of the constraints used.

4.2 Features

The inference problem defined in Eq. (2) uses two feature functions: Φ_1 and Φ_2.

First-order decision features Φ_1 Determining if a logical symbol is aligned with a specific constituent depends mostly on lexical information. Following previous work (e.g., (Zettlemoyer and Collins, 2005)) we create a small lexicon, mapping logical symbols to surface forms.[3] Existing approaches rely on annotated data to extend the lexicon. Instead we rely on external knowledge (Miller et al., 1990) and add features which measure the lexical similarity between a constituent and a logical symbol's surface forms (as defined by the lexicon).

[3]The lexicon contains on average 1.42 words per function and 1.07 words per constant.

Model	Description
INITIAL MODEL	Manually set weights (Sec. 5.1)
PRED. SCORE	normalized prediction (Sec. 5.1)
ALL EXAMPLES	All top structures (Sec. 5.1)
UNIGRAM	Unigram score (Sec. 3.2)
BIGRAM	Bigram score (Sec. 3.2)
PROPORTION	Words-predicate prop (Sec. 3.2)
COMBINED	Combined estimators (Sec. 3.2)
RESPONSE BASED	Supervised (binary) (Sec. 5.1)
SUPERVISED	Fully Supervised (Sec. 5.1)

Table 1: Compared systems and naming conventions.

Second-order decision features Φ_2 Second order decisions rely on syntactic information. We use the dependency tree of the input sentence. Given a second-order decision $\beta_{cs,dt}$, the dependency feature takes the normalized distance between the head words in the constituents c and d. In addition, a set of features indicate which logical symbols are usually composed together, without considering their alignment to the text.

5 Experiments

In this section we describe our experimental evaluation. We compare several confidence measures and analyze their properties. Tab. 1 defines the naming conventions used throughout this section to refer to the different models we evaluated. We begin by describing our experimental setup and then proceed to describe the experiments and their results. For the sake of clarity we focus on the best performing models (COMBINED using BIGRAM and PROPORTION) first and discuss other models later in the section.

5.1 Experimental Settings

In all our experiments we used the Geoquery dataset (Zelle and Mooney, 1996), consisting of U.S. geography NL questions and their corresponding Prolog logical MR. We used the data split described in (Clarke et al., 2010), consisting of 250 queries for evaluation purposes. We compared our system to several supervised models, which were trained using a disjoint set of queries. Our learning system had access only to the NL questions, and the logical forms were only used to evaluate the system's performance. We report the proportion of correct structures (accuracy). Note that this evaluation cor-

responds to the 0/1 loss over the predicted structures.

Initialization Our learning framework requires an initial weight vector as input. We use a straight forward heuristic and provide uniform positive weights to three features. This approach is similar in spirit to previous works (Clarke et al., 2010; Zettlemoyer and Collins, 2007). We refer to this system as INITIAL MODEL throughout this section.

Competing Systems We compared our system to several other systems:

(1) **PRED. SCORE**: An unsupervised framework using the model's internal prediction score ($\mathbf{w}^T\Phi(\mathbf{x},\mathbf{y},\mathbf{z})$) for confidence estimation.

(2) **ALL EXAMPLES**: Treating all predicted structures as correct, i.e., at each iteration the model is trained over all the predictions it made. The reported score was obtained by selecting the model at the training iteration with the highest overall confidence score (see line 12 in Alg. 1).

(3) **RESPONSE BASED**: A natural upper bound to our framework is the approach used in (Clarke et al., 2010). While our approach is based on assessing the correctness os the model's predictions according to unsupervised confidence estimation, their framework is provided with external supervision for these decisions, indicating if the predicted structures are correct.

(4) **SUPERVISED**: A fully supervised framework trained over 250 $(\mathbf{x},\mathbf{z})$ pairs using structured SVM.

5.2 Results

Our experiments aim to clarify three key points:

(1) Can a semantic parser indeed be trained without any form of external supervision? this is our key question, as this is the first attempt to approach this task with an unsupervised learning protocol.[4] In order to answer it, we report the overall performance of our system in Tab. 2.

The manually constructed model INITIALMODEL achieves a performance of 0.22. We can expect learning to improve on this baseline. We compare three self-trained systems, ALL EXAMPLES, PREDICTIONSCORE and COMBINED, which differ in their sample selection strategy, but all use confidence estimation for selecting the final semantic parsing model. The ALL EXAMPLES approach achieves an accuracy score of 0.656. PREDICTIONSCORE only achieves a performance of 0.164 using the binary learning algorithm and 0.348 using the structured learning algorithm. Finally, our confidence-driven technique COMBINED achieved a score of 0.536 for the binary case and 0.664 for the structured case, the best performing models in both cases. As expected, the supervised systems RESPONSE BASED and SUPERVISED achieve the best performance.

These results show that training the model with training examples selected carefully will improve learning - as the best performance is achieved with perfect knowledge of the predictions correctness (RESPONSE BASED). Interestingly the difference between the structured version of our system and that of RESPONSE BASED is only 0.07, suggesting that we can recover the binary feedback signal with high precision. The low performance of the PREDICTIONSCORE model is also not surprising, and it demonstrates one of the key principles in confidence estimation - the score should be comparable across predictions done over different inputs, and not the same input, as done in PREDICTIONSCORE model.

(2) How does confidence driven sample selection contribute to the learning process? Comparing the systems driven by confidence sample-selection to the ALL EXAMPLES approach uncovers an interesting tradeoff between training with more (noisy) data and selectively training the system with higher quality examples. We argue that carefully selecting high quality training examples will result in better performance. The empirical results indeed support our argument, as the best performing model (RESPONSE BASED) is achieved by sample selection with perfect knowledge of prediction correctness. The confidence-based sample selection system (COMBINED) is the best performing system out of all the self-trained systems. Nonetheless, the ALL EXAMPLES strategy performs well when compared to COMBINED, justifying a closer look at that aspect of our system.

We argue that different confidence measures capture different properties of the data, and hypothe-

[4] While unsupervised learning for various semantic tasks has been widely discussed, this is the first attempt to tackle this task. We refer the reader to Sec. 6 for further discussion of this point.

size that combining their scores will improve the resulting model. In Tab. 3 we compare the results of the COMBINED measure to the results of its individual components - PROPORTION and BIGRAM. We compare these results both when using the binary and structured learning algorithms. Results show that using the COMBINED measure leads to an improved performance, better than any of the individual measures, suggesting that it can effectively exploit the properties of each confidence measure. Furthermore, COMBINED is the only sample selection strategy that outperforms ALL EXAMPLES.

(3) Can confidence measures serve as a good proxy for the model's performance? In the unsupervised settings we study the learning process may not converge to an optimal model. We argue that by selecting the model that maximizes the averaged confidence score, a better model can be found. We validate this claim empirically in Tab. 4. We compare the performance of the model selected using the confidence score to the performance of the final model considered by the learning algorithm (see Sec. 3.1 for details). We also compare it to the best model achieved in any of the learning iterations.

Since these experiments required running the learning algorithm many times, we focused on the binary learning algorithm as it converges considerably faster. In order to focus the evaluation on the effects of learning, we ignore the initial model generated manually (INITIAL MODEL) in these experiments. In order to compare models performance across the different iterations fairly, a uniform scale, such as UNIGRAM and BIGRAM, is required. In the case of the COMBINED measure we used the BIGRAM measure for performance estimation, since it is one of its underlying components. In the PRED. SCORE and PROPORTION models we used both their confidence prediction, and the simple UNIGRAM confidence score to evaluate *model* performance (the latter appear in parentheses in Tab. 4).

Results show that the over overall confidence score serves as a reliable proxy for the model performance - using UNIGRAM and BIGRAM the framework can select the best performing model, far better than the performance of the default model to which the system converged.

Algorithm	Supervision	Acc.
INITIAL MODEL	—	0.222
SELF-TRAIN: (*Structured*)		
PRED. SCORE	—	0.348
ALL EXAMPLES	—	0.656
COMBINED	—	**0.664**
SELF-TRAIN: (*Binary*)		
PRED. SCORE	—	0.164
COMBINED	—	0.536
RESPONSE BASED		
BINARY	250 (binary)	0.692
STRUCTURED	250 (binary)	0.732
SUPERVISED		
STRUCTURED	250 (struct.)	0.804

Table 2: Comparing our *Self-trained* systems with Response-based and supervised models. Results show that our COMBINED approach outperforms all other unsupervised models.

Algorithm	Accuracy
SELF-TRAIN: (*Structured*)	
PROPORTION	0.6
BIGRAM	0.644
COMBINED	0.664
SELF-TRAIN: (*Binary*)	
BIGRAM	0.532
PROPORTION	0.504
COMBINED	0.536

Table 3: Comparing COMBINED to its components BIGRAM and PROPORTION. COMBINED results in a better score than any of its components, suggesting that it can exploit the properties of each measure effectively.

Algorithm	Best	Conf. estim.	Default
PRED. SCORE	0.164	0.128 (0.164)	0.134
UNIGRAM	0.52	0.52	0.4
BIGRAM	0.532	0.532	0.472
PROPORTION	0.504	0.27 (0.504)	0.44
COMBINED	0.536	0.536	0.328

Table 4: Using confidence to approximate model performance. We compare the best result obtained in any of the learning algorithm iterations (Best), the result obtained by approximating the best result using the averaged prediction confidence (Conf. estim.) and the result of using the default convergence criterion (Default). Results in parentheses are the result of using the UNIGRAM confidence to approximate the model's performance.

6 Related Work

Semantic parsing has attracted considerable interest in recent years. Current approaches employ various machine learning techniques for this task, such as Inductive Logic Programming in earlier systems (Zelle and Mooney, 1996; Tang and Mooney, 2000) and statistical learning methods in modern ones (Ge and Mooney, 2005; Nguyen et al., 2006; Wong and Mooney, 2006; Kate and Mooney, 2006; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Zettlemoyer and Collins, 2009).

The difficulty of providing the required supervision motivated learning approaches using weaker forms of supervision. (Chen and Mooney, 2008; Liang et al., 2009; Branavan et al., 2009; Titov and Kozhevnikov, 2010) ground NL in an external world state directly referenced by the text. The NL input in our setting is not restricted to such grounded settings and therefore we cannot exploit this form of supervision. Recent work (Clarke et al., 2010; Liang et al., 2011) suggest using response-based learning protocols, which alleviate some of the supervision effort. This work takes an additional step in this direction and suggest an unsupervised protocol.

Other approaches to unsupervised semantic analysis (Poon and Domingos, 2009; Titov and Klementiev, 2011) take a different approach to semantic representation, by clustering semantically equivalent dependency tree fragments, and identifying their predicate-argument structure. While these approaches have been applied successfully to semantic tasks such as question answering, they do not ground the input in a well defined output language, an essential component in our task.

Our unsupervised approach follows a self training protocol (Yarowsky, 1995; McClosky et al., 2006; Reichart and Rappoport, 2007b) enhanced with constraints restricting the output space (Chang et al., 2007; Chang et al., 2009). A Self training protocol uses its own predictions for training. We estimate the quality of the predictions and use only high confidence examples for training. This selection criterion provides an additional view, different than the one used by the prediction model. Multi-view learning is a well established idea, implemented in methods such as co-training (Blum and Mitchell, 1998).

Quality assessment of a learned model output was explored by many previous works (see (Caruana and Niculescu-Mizil, 2006) for a survey), and applied to several NL processing tasks such as syntactic parsing (Reichart and Rappoport, 2007a; Yates et al., 2006), machine translation (Ueffing and Ney, 2007), speech (Koo et al., 2001), relation extraction (Rosenfeld and Feldman, 2007), IE (Culotta and McCallum, 2004), QA (Chu-Carroll et al., 2003) and dialog systems (Lin and Weng, 2008).

In addition to sample selection we use confidence estimation as a way to approximate the overall quality of the model and use it for model selection. This use of confidence estimation was explored in (Reichart et al., 2010), to select between models trained with different random starting points. In this work we integrate this estimation deeper into the learning process, thus allowing our training procedure to return the best performing model.

7 Conclusions

We introduced an unsupervised learning algorithm for semantic parsing, the first for this task to the best of our knowledge. To compensate for the lack of training data we use a self-training protocol, driven by unsupervised confidence estimation. We demonstrate empirically that our approach results in a high preforming semantic parser and show that confidence estimation plays a vital role in this success, both by identifying good training examples as well as identifying good over all performance, used to improve the final model selection.

In future work we hope to further improve unsupervised semantic parsing performance. Particularly, we intend to explore new approaches for confidence estimation and their usage in the unsupervised and semi-supervised versions of the task.

Acknowledgments We thank the anonymous reviewers for their helpful feedback. This material is based upon work supported by DARPA under the Bootstrap Learning Program and Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT*.

S.R.K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *ACL*.

R. Caruana and A. Niculescu-Mizil. 2006. An empirical comparison of supervised l earning algorithms. In *ICML*.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the ACL*.

M. Chang, D. Goldwasser, D. Roth, and Y. Tu. 2009. Unsupervised constraint driven learning for transliteration discovery. In *NAACL*.

D. Chen and R. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *ICML*.

J. Chu-Carroll, J. Prager K. Czuba, and A. Ittycheriah. 2003. In question answering, two heads are better than on. In *HLT-NAACL*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*, 7.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *EMNLP–VLC*.

A. Culotta and A. McCallum. 2004. Confidence estimation for information extraction. In *HLT-NAACL*.

R. Ge and R. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *CoNLL*.

R. Kate and R. Mooney. 2006. Using string-kernels for learning semantic parsers. In *ACL*.

Y. Koo, C. Lee, and B. Juang. 2001. Speech recognition and utterance verification based on a generalized confidence score. *IEEE Transactions on Speech and Audio Processing*, 9(8):821–832.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*.

P. Liang, M.I. Jordan, and D. Klein. 2011. Deep compositional semantics from shallow supervision. In *ACL*.

F. Lin and F. Weng. 2008. Computing confidence scores for all sub parse trees. In *ACL*.

D. McClosky, E. Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *HLT-NAACL*.

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*.

L. Nguyen, A. Shimazu, and X. Phan. 2006. Semantic parsing with structured svm ensemble classification models. In *ACL*.

H. Poon and P. Domingos. 2009. Unsupervised semantic parsing. In *EMNLP*.

R. Reichart and A. Rappoport. 2007a. An ensemble method for selection of high quality parses. In *ACL*.

R. Reichart and A. Rappoport. 2007b. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL*.

R. Reichart, R. Fattal, and A. Rappoport. 2010. Improved unsupervised pos induction using intrinsic clustering quality and a zipfian constraint. In *CoNLL*.

B. Rosenfeld and R. Feldman. 2007. Using corpus statistics on entities to improve semi–supervised relation extraction from the web. In *ACL*.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*.

L. Tang and R. Mooney. 2000. Automated construction of database interfaces: integrating statistical and relational learning for semantic parsing. In *EMNLP*.

L. R. Tang and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *ECML*.

I. Titov and A. Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *ACL*.

I. Titov and M. Kozhevnikov. 2010. Bootstrapping semantic analyzers from non-contradictory texts. In *ACL*.

N. Ueffing and H. Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

Y.W. Wong and R. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *NAACL*.

Y.W. Wong and R. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised method. In *ACL*.

A. Yates, S. Schoenmackers, and O. Etzioni. 2006. Detecting parser errors using web-based semantic filters. In *EMNLP*.

J. M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In *AAAI*.

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

L. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *CoNLL*.

L. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *ACL*.