UNSUPERVISED PARSING

Dishant Sharma*

Madhur Chanana*

Anupam Mahajan*

KunalSachdeva*

1. *Abstract*

We present the first unsupervised approach for semantic parsing that rivals the accuracy of supervised approaches in translating natural-language questions to database queries. Our GUSP system produces a semantic parse by annotating the dependency-tree nodes and edges with latent states, and learns a probabilistic grammar using EM. To compensate for the lack of example annotations or question-answer pairs, GUSP adopts a novel grounded-learning approach to leverage database for indirect supervision. On the challenging ATIS dataset, GUSP attained an accuracy of 84%, effectivelytying with the best published results by supervised approaches. Our USP system transforms dependency trees into quasi-logical forms, recursively induces lambda forms from these, and clustersthem 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.

Index Terms - DCP, GUSP, QLF, SQL, USP

* Dronacharya College of Engineering, CSE Department, Gurgaon

2. 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 constructedmanually, but this is too costly and brittle. Recently, a number of machine learning approache

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 andMarkov logic. Wethen describe our Markov logic network for unsupervised semantic parsing, and the learning andinference algorithms we used. Finally, we present our experiments and results.

Semantic parsing maps text to a formal meaning representation such as logical forms or structured

queries. Recently, there has been a burgeoning interest in developing machine-learning

approaches for semantic parsing (ZettlemoyerandCollins, 2005; Zettlemoyer and Collins, 2007; Mooney, 2007; Kwiatkowski et al., 2011), but the predominant paradigm uses supervised learning,

which requires example annotations that are costly to obtain. More recently, several groundedlearning

approaches have been proposed to alleviate the annotation burden (Chen and Mooney,

2008; Kim and Mooney, 2010; B¨orschinger et al., 2011; Clarke et al., 2010; Liang et al., 2011). In

particular, Clarke et al. (2010) and Liang et al. (2011) proposed methods to learn from questionanswerpairs alone, which represents a significant advance. However, although these methods exonerateannotators from mastering specialized logical forms, finding the answers for complex questionsstill requires non-trivial effort. Poon &Domingos (2009, 2010) proposed theUSP system for unsupervised semantic parsing,which learns a parser by recursively clusteringand composing synonymous expressions. Whiletheir approach completely obviates the need for directsupervision, their target logic forms are selfinducedclusters, which do not align with existing database or ontology. As a result, USP can not beused directly to answer complex questions againstan existing database. More importantly, it missethe opportunity to leverage database for indirectsupervision.

In this paper, we present the GUSP system,which combines unsupervised semantic parsingwith grounded learning from a database. GUSPstarts with the dependency tree of a sentence and

produces a semantic parse by annotating the nodesand edges with latent semantic states derived from

the database. Given a set of natural-languagequestions and a database, GUSP learns a probabilisticsemantic grammar using EM. To compensatefor the lack of direct supervision, GUSPconstrains the search space using the databaseschema, and bootstraps learning using lexicalscores computed from the names and values ofdatabase elements.Unlike previous grounded-learning approaches,GUSP does not require ambiguous annotationsor oracle answers, but rather focuses on leveragingdatabase contents that are readily available.Unlike USP, GUSP predetermines the targetlogical forms based on the database schema,which alleviates the

difficulty in learning and ensuresthat the output semantic parses can be directlyused in querying the database.

3 Background

3.1 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* definesthe 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 bordersIdaho".

Verb[_y_x.borders(x, y)] ! bordersNP[Utah] ! UtahNP[Idaho] ! Idaho

V P[rel(obj)] ! V erb[rel] NP[obj] S[rel(obj)] ! NP[obj] V P[rel]

The first three lines are lexical entries. They arefired upon seeing the individual words. For example,the first rule applies to the word "borders" andgenerates syntactic category *Verb* with the meaning_y_x.borders(x, y) that represents the nexttorelation. Here, we use the standard lambda calculusnotation, where _y_x.borders(x, y) represents a function that is true for any (x, y)-pair such that borders(x, y) holds. The last two rules compose the meanings of sub-parts into thatof the larger part. For example, after the firstand third rules are fired, the fourth rule fires andgenerates V P[_y_x.borders(x, y)(Idaho)];

thismeaning simplifies to _x.borders(x, Idaho) bythe _-*reduction rule*, which substitutes the argumentfor a variable in a functional application.A major challenge to semantic parsing is syntacticvariations of the same meaning, whichabound in natural languages. For example, theaforementioned 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 categorical grammar (PCCG) is learned using a loglinearmodel, where the probability of the final logical form L and meaning-derivation tree T conditioned on the sentence S is P(L, T|S) =Z exp (Pi wifi(L, T, S)). Here Z is the normalization constant and fi are the feature functions with weights wi. 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. 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.

4 Grounded Unsupervised Semantic Parsing

In this section, we present the GUSP system for grounded unsupervised semantic parsing. GUSP is unsupervised and does not require example logical forms or question-answer pairs. Figure 1 shows an example of end-to-end question answering using GUSP. GUSP produces a semantic parse of the question by annotating its dependency tree with latent semantic states. The semantic tree can then be deterministically converted into SQL to obtain answer from the database. Given a set of natural-language questions and a database, GUSP learns a probabilistic semantic grammar using EM.

```
SELECT flight.flight_id
FROM flight, city, city c2, flight_stop, airport_service, airport_service as2
WHERE flight.from_airport = airport_service.airport_code AND flight.to_airport =
as2.airport_code AND airport_service.city_code = city.city_code AND as2.city_code =
city2.city_code AND city.city_name = 'toronto' AND city2.city_name = 'san diego' AND
flight_stop.flight_id = flight.flight_id AND flight_stop.stop_airport = 'dtw'
```

Figure 1[2]: End-to-end question answering by GUSP for sentence get flight from toronto to san diego stopping in dtw. Top: the dependency tree of the sentence is annotated with latent semantic states by GUSP. For brevity, we omit the edge states. Raising occurs from flight to get and sinking occurs from get to diego. Bottom: the semantic tree is deterministically converted into SQL to obtain answer from the database.

To compensate for the lack of annotated examples, GUSP derives indirect supervision from a novel combination of three key sources. First, GUSP leverages the target database to constrain the search space. Specifically, it defines the semantic states based on the database schema, and derives lexical-trigger scores from database elements to bootstrap learning. Second, in contrast to most existing approaches for semantic parsing, GUSP starts directly from dependency trees and focuses on translating them into semantic parses. While syntax may not always align perfectly with semantics, it is still highly informative about the latter. In particular, dependency edges are often indicative of semantic relations. On the other hand, syntax and semantic often diverge, and synactic parsing errors abound. To combat this problem, GUSP introduces a novel dependency-

based meaning representation with an augmented state space to account for semantic relations that are nonlocal in the dependency tree. GUSP's approach of starting directly from dependency tree is inspired by USP. However, GUSPuses a different meaning representation defined over individual nodes and edges, rather than partitions, which enables linear-time exact inference. GUSP also handles complex linguistic phenomena and syntax-semantics mismatch by explicitly augmenting the state space, whereas USP's capability in handling such phenomena is indirect and more limited.

GUSP represents meaning by a semantic tree, which is similar to DCS (Liang et al., 2011). Their approach to semantic parsing, however, differs from GUSP in that it induced the semantic tree directly from a sentence, rather than starting from a dependency tree and annotating it. Their approach

alleviates some complexity in the meaningrepresentation for handling syntax-semanticsmismatch, but it has to search over a much largersearch space involving exponentially many candidatetrees. This might partially explain why it hasnot yet been scaled up to the ATIS dataset.Finally, GUSP recognizes that certain aspectsin semantic parsing may not be worth learningusing precious annotated examples. Theseare domain-independent and closed-class expressions,such as times and dates (e.g., before 5pmand July seventeenth), logical connectives (e.g.,and, or, not), and numerics (e.g., 200 dollars).GUSP preprocesses the text to detect suchexpressions and restricts their interpretation todatabase elements of compatible types (e.g., before5pm vs. flight.departure time orflight.arrival time). Short of training examples,GUSP also resolves quantifier scopingambiguities deterministically by a fixed ordering.For example, in the phrase cheapest flight to Seattle,the scope of cheapest can be either flight orflight to seattle. GUSP always chooses to applythe superlative at last, amounting to choosing themost restricted scope (flight to seattle), which isusually the correct interpretation.In the remainder of this section, we first formalizethe problem setting and introduce the GUSmeaning representation. We then present theGUSP model and learning and inference algorithms.Finally, we describe how to convert aGUSP semantic parse into SQL.

4.1 Problem Formulation

Let d be a dependency tree, N(d) and E(d) beits nodes and edges. In GUSP, a semantic parse of d is an assignment z : N(d) □ E(d) → S that maps its nodes and edges to semantic statesin S. For example, in the example in Figure 1,z(flight) = E : flight. At the core of GUSPis a joint probability distribution P_(d; z) over thedependency tree and the semantic parse. Semanticparsing in GUSP amounts to finding the mostprobable parse z*= argmaxzPθ(d, z). Givena set of sentences and their dependency trees D,learning in GUSP maximizes the log-likelihood ofD while summing out the latent parses z:θ*= arg max log Pθ(D)

= arg max ∑ logX ∑ Pθ(d, z)

4.2 Simple Semantic States

Node states GUSP creates a state E:X (E short for entity) for each database entity X (i.e., a database table), a state P:Y (P short for property) and V:Y (V short for value) for each database attribute Y (i.e., a database column).[4] Node states are assigned to dependency nodes. Intuitively,they represent database entities, properties, and values. For example, the ATIS domain contains entities such as flight and fare,which may contain properties such as the departuretime flight.departure time or ticket

price fare.one direction cost. The mentions of entities and properties are represented by entity and property states, whereas constants such as 9:25am or 120 dollars are represented by value states. In the semantic parse in Figure 1, for example, flight is assigned to entity state E:flight, where torontois assigned to value state V:city.name. There is a special node state NULL, which signifies that the subtreeheaded by the word contributes no meaning to thesemantic parse (e.g., an auxilliary verb).

Edge states GUSP creates an edge state foreach valid relational join paths connecting twonode states. [3]Edge states are assigned to dependencyedges. GUSP enforces the constraints that the node states of the dependency parent and child must agree with the node states in the edge state. For example, E:flight--V:flight.departure time represents a natural join between the flight entity and the property

value departure time. For a dependency edge e : a ! b, the assignment to E:flight- -V:flight.departure time signifies that a represents a flight entity, and b represents the value of its departure time. An edge state may also represent a relational path consisting of a serial of joins. For example, Zettlemoyer and

Collins (2007) used a predicate from(f,c) to signify that flight f starts from city c. In the ATIS database, however, this amounts to a path of three joins:

flight.from airport-airport

airport-airport service

airport service-city

In GUSP, this is represented by the edgestate flight-flight.from airport- -airport-airport service-city.

GUSP only creates edge states for relational join paths up to length four, as longer paths rarely correspond to meaningful semantic relations.

Composition To handle compositions such as American Airlines and New York City, it helps to distinguish the head words (Airlines and City) from the rest. In GUSP, this is handled by introducing,

for each node state such as E:airline, a new node state such as E:airline:C, where C signifies composition. For example, in Figure 1, diego is assigned to V:city.name, whereas san is assigned to V:city.name:C, since san diego forms a single meaning unit, and should be translated into SQL as a whole.

5 Unsupervised Semantic Parsing with Markov Logic

Unsupervised semantic parsing (**USP**) rests on three key ideas. First, the target predicate and objectconstants, 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, bordersrepresents the next-to relation, and can be viewed as the cluster of different forms for expressing thisrelation, 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 builtup 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 syntacticparsers 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 toperform. In particular, meaning composition does not require domain-specific procedures for generatingcandidate 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.

5.1 Derivation of Quasi-Logical Forms

A *dependency tree* is a tree where nodes are words and edges are dependency labels. [5]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 Utah(n1) and the subject dependency

becomesnsubj(n1, n2). Here, the niare 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 borders(n1) ^ Utah(n2) ^ Idaho(n3) ^nsubj(n1, n2) ^ dobj(n1, n3).

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 verb (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.

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 amatch. The first one (**KW**) simple 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,A1,A2), where R is the relation string, and A1,A2the 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 resolvescoreferent 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 extractionand 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,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 firstparses the questions12 using USP-Parse with the

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 amatch 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.

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.

References

[1]G. Bakir, T. Hofmann, B. B. Sch¨olkopf, A. Smola,B. Taskar, S. Vishwanathan, and (eds.). 2007. *Predicting*
Structured Data. MIT Press, Cambridge,MA.

[2]Michele Banko, Michael J. Cafarella, Stephen Soderland,MattBroadhead, 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.

[3]Xavier Carreras and LuisMarquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling.
In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages89–97, Boston, MA. ACL.

[4]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.

[5]RuifangGe and Raymond J. Mooney. 2009. Learning a compositional semantic parser using an existingsyntactic parser. In *Proceedings of the Forty SeventhAnnual Meeting of the Association for ComputationalLinguistics*, Singapore.ACL.

[6]LiseGetoor and Ben Taskar, editors. 2007. *Introduction to Statistical Relational Learning*. MIT Press,
Cambridge,MA. PekkaKilpelainen. 1992. *Tree Matching Problemswith Applications to Structured Text databases*.

Ph.D. Thesis, Department of Computer Science,University of Helsinki.Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichiTsujii. 2003. GENIA corpus - a semanticall annotated corpus for bio-textmining. *Bioinformatics*,19:180–82.

[7]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.

[8]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.

[9]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.

[10]Dekang Lin. 1998. Dependency-based evaluationof 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.

[11]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.

[12]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.

[13]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.

[14]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.

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