Fast Online Lexicon Learning for Grounded Language Acquisition

David L. Chen
Department of Computer Science
The University of Texas at Austin
1616 Guadalupe, Suite 2.408
Austin, TX 78701, USA
dlcc@cs.utexas.edu

Abstract

Learning a semantic lexicon is often an important first step in building a system that learns to interpret the meaning of natural language. It is especially important in language grounding where the training data usually consist of language paired with an ambiguous perceptual context. Recent work by Chen and Mooney (2011) introduced a lexicon learning method that deals with ambiguous relational data by taking intersections of graphs. While the algorithm produced good lexicons for the task of learning to interpret navigation instructions, it only works in batch settings and does not scale well to large datasets. In this paper we introduce a new online algorithm that is an order of magnitude faster and surpasses the state-of-the-art results. We show that by changing the grammar of the formal meaning representation language and training on additional data collected from Amazon's Mechanical Turk we can further improve the results. We also include experimental results on a Chinese translation of the training data to demonstrate the generality of our approach.

1 Introduction

Learning to understand the semantics of human languages has been one of the ultimate goals of natural language processing (NLP). Traditional learning approaches have relied on access to parallel corpora of natural language sentences paired with their meanings (Mooney, 2007; Zettlemoyer and Collins, 2007; Lu et al., 2008; Kwiatkowski et al., 2010). However, constructing such semantic annotations can be difficult and time-consuming. More recently, there has been work on learning from ambiguous supervision where a set of potential sentence meanings are given, only one (or a small subset) of which are correct (Chen and Mooney, 2008; Liang et al., 2009; Bordes et al., 2010; Chen and Mooney, 2011). Given the training data, the system needs to infer the correcting meaning for each training sentence.

Building a lexicon of the formal meaning representations of words and phrases, either implicitly or explicitly, is usually an important step in inferring the meanings of entire sentences. In particular, Chen and Mooney (2011) first learned a lexicon to help them resolve ambiguous supervision of relational data in which the number of choices is exponential. They represent the perceptual context as a graph and allow each sentence in the training data to align to any connected subgraph. Their lexicon learning algorithm finds the common connected subgraph that occurs with a word by taking intersections of the graphs that represent the different contexts in which the word appears. While the algorithm produced a good lexicon for their application of learning to interpret navigation instructions, it only works in batch settings and does not scale well to large datasets. In this paper we introduce a novel online algorithm that is an order of magnitude faster and also produces better results on their navigation task.

In addition to the new lexicon learning algorithm, we also look at modifying the meaning representation grammar (MRG) for their formal semantic language. By using a MRG that correlates better to the structure of natural language, we further improve the performance on the navigation task. Since our algorithm can scale to larger datasets, we present results on collecting and training on additional data from Amazon's Mechanical Turk. Finally, we show the generality of our approach by demonstrating our system's ability to learn from a Chinese translation of the training data.

2 Background

A common way to learn a lexicon across many different contexts is to find the common parts of the formal representations associated with different occurrences of the same words or phrases (Siskind, 1996). For graphical representations, this involves finding the common subgraph between multiple graphs (Thompson and Mooney, 2003; Chen and Mooney, 2011). In this section we review the lexicon learning algorithm introduced by Chen and Mooney (2011) as well as the overall task they designed to test semantic understanding of navigation instructions.

2.1 Navigation Task

The goal of the navigation task is to build a system that can understand free-form natural-language instructions and follow them to move to the intended destination (MacMahon et al., 2006; Shimizu and Haas, 2009; Matuszek et al., 2010; Kollar et al., 2010; Vogel and Jurafsky, 2010; Chen and Mooney, 2011). Chen and Mooney (2011) defined a learning task in which the only supervision the system receives is in the form of observing how humans behave when following sample navigation instructions in a virtual world. Formally, the system is given training data in the form: $\{(e_1, a_1, w_1), (e_2, a_2, w_2), \ldots, (e_n, a_n, w_n)\}$, where e_i is a written natural language instruction, a_i is an observed action sequence, and w_i is a description of the virtual world. The goal is then to build a system that can produce the correct a_j given a previously unseen (e_j, w_j) pair.

Since the observed actions a_i only consists of low-level actions (e.g. turn left, turn right, walk forward) and not high-level concepts (e.g. turn your back against the wall and walk to the couch), Chen and Mooney first use a set of rules to automatically construct the space of reasonable plans using the action trace and knowledge about the world. The space is represented compactly using a graph as shown in

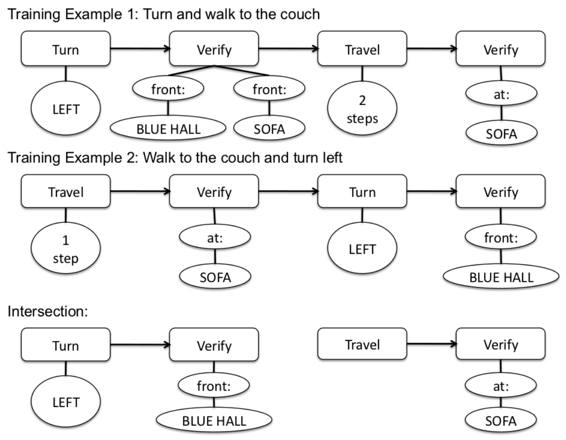

Figure 1: Examples of *landmarks plans* constructed by Chen and Mooney (2011) and how they computed the intersection of two graphs.

Figure 1. This is what they called a *landmarks plan* and consists of the low-level observed actions interleaved with verification steps indicating what objects should be observed after each action.

Given that these *landmarks plans* contain a lot of extraneous details, Chen and Mooney learn a lexicon and use it to identify and remove the irrelevant parts of the plans. They use a greedy method to remove nodes from the graphs that are not associated with any of the words in the instructions. The remaining *refined landmarks plans* are then treated as supervised training data for a semantic-parser learner, KRISP (Kate and Mooney, 2006). Once a semantic parser is trained, it can be used at test time to transform novel instructions into formal navigation plans which are then carried out by a virtual robot (MacMahon et al., 2006).

2.2 Lexicon Learning

The central component of the system is the lexicon learning process which associates words and short phrases (n-grams) to their meanings (connected graphs). To learn the meaning of an n-gram w, Chen and Mooney first collect all navigation plans g that co-occur with w. This forms the initial candidate meaning set for w. They then repeatedly take the intersections between the candidate meanings to generate additional candidate meanings. They use the term intersection to mean a maximal common subgraph (i.e. it is not a subgraph of any other common subgraphs). In general, there are multiple possible intersections between two graphs. In this case, they bias toward finding large connected components by greedily removing the largest common connected subgraph from both graphs until the two graphs have no overlapping nodes. The output of the intersection process consists of all the removed subgraphs. An example of the intersection operation is shown in Figure 1.

Once the list of candidate meanings are generated, they are ranked by the following scoring metric for an n-gram w and a graph g:

$$Score(w, g) = p(g|w) - p(g|\neg w)$$

Intuitively, the score measures how much more likely a graph g appears when w is present compared to when it is not. The probabilities are estimated by counting how many examples contain the word w or graph g, ignoring multiple occurrences in a single example.

3 Online Lexicon Learning Algorithm

While the algorithm presented by Chen and Mooney (2011) produced good lexicons, it only works in batch settings and does not scale well to large datasets. The bottleneck of their algorithm is the intersection process which is time-consuming to perform. Moreover, their algorithm requires taking many intersections between many different graphs. Even though they use beam-search to limit the size of the candidate set, if the initial candidate meaning set for a n-gram is large, it can take a long time to take just one pass through the list of all candidates. Moreover, reducing the beam size could also hurt the quality of the lexicon learned.

In this section, we present another lexicon learning algorithm that is much faster and works in an online setting. The main insight is that most words or short phrases correspond to small graphs. Thus, we concentrate our attention on only candidate meanings that are less than a certain size. Using this constraint, we generate all the potential small connected subgraphs for each navigation plan in the training examples and discard the original graph. Pseudocode for the new algorithm, Subgraph Generation Online Lexicon Learning (SGOLL) algorithm, is shown in Algorithm 1.

As we encounter each new training example which consists of a written navigation instruction

Algorithm 1 SUBGRAPH GENERATION ONLINE LEXICON LEARNING (SGOLL)

```
input A sequence of navigation instructions and the corresponding navigation plans $(e_1, p_1), \ldots, (e_n, p_n)$
output $Lexicon$, a set of phrase-meaning pairs
1:  main
2:    for training example $(e_i, p_i)$ do
3:      Update($(e_i, p_i)$)
4:    end for
5:    OutputLexicon()
6:  end main
7:
8:  function Update(training example $(e_i, p_i)$)
9:    for n-gram $w$ that appears in $e_i$ do
10:     for connected subgraph $g$ of $p_i$ such that the size of $g$ is less than or equal to $m$ do
11:       Increase the co-occurrence count of $g$ and $w$ by 1
12:     end for
13:   end for
14:   Increase the count of examples, each n-gram $w$ and each subgraph $g$
15: end function
16:
17:
18: function OutputLexicon()
19:   for n-gram $w$ that has been observed do
20:     if Number of times $w$ has been observed is less than $minSup$ then
21:       skip $w$
22:     end if
23:     for subgraph $g$ that has co-occurred with $w$ do
24:       if score($w$, $g$) $>$ threshold $t$ then
25:         add $(w, g)$ to $Lexicon$
26:       end if
27:     end for
28:   end for
29: end function
```

and the corresponding navigation plan, we first segment the instruction into word tokens and construct n-grams from them. From the corresponding navigation plan, we find all connected subgraphs of size less than or equal to m. We then update the co-occurrence counts between all the n-grams w and all the connected subgraphs g. We also update the counts of how many examples we have encountered so far and counts of the n-grams w and subgraphs g. At any given time, we can compute a lexicon using these various counts. Specifically, for each n-gram w, we look at all the subgraphs g that co-occurred with it, and compute a score for the pair (w, g). If the score is higher than the threshold t, we add the entry (w, g) to the lexicon. We use the same scoring function as Chen and Mooney, which can be computed efficiently using the counts we keep. In contrast to Chen and Mooney's algorithm though, we add the constraint of minimum support by not creating lexical entries for any n-gram w that appeared in less than $minSup$ training examples. This is to prevent rarely seen n-grams from receiving high scores in our lexicon simply due to their sparsity. Unless otherwise specified, we compute lexical entries for up to 4-grams with threshold $t = 0.4$, maximum subgraph size $m = 3$, and minimum support $minSup = 10$.

It should be noted that SGOLL can also become computationally intractable if the sizes of the navigations plans are large or if we set the maximum subgraph size m to a large number. Moreover, the memory requirement can be quite high if there are many different subgraphs g associated with each n-gram w. To deal with such scalability issues, we could use beam-search and only keep the top k candidates associated with each w. Another important step is to define canonical orderings of the nodes in the graphs. This allows us to determine if two graphs are identical in constant time and also lets us use a hash table to quickly update the co-occurrence and subgraph counts. Thus, even given a large number of subgraphs for each training example, each subgraph can be processed very quickly. Finally, this algorithm readily lends itself to being parallelized. Each processor would get a fraction of the training data and compute the counts individually. Then the counts can be merged together at the end to produce the final lexicon.

3.1 Changing the Meaning Representation Grammar

In addition to introducing a new lexicon learning algorithm, we also made another modification to the original system proposed by Chen and Mooney (2011). To train a semantic parser using KRISP (Kate and Mooney, 2006), they had to supply a MRG, a context-free grammar, for their formal navigation plan language. KRISP learns string-kernel classifiers that maps natural language substrings to MRG production rules. Consequently, it is important that the production rules in the MRG mirror the structure of natural language (Kate, 2008).

The original MRG used by Chen and Mooney is a compact grammar that contains many recursive rules that can be used to generate an infinite number of actions or arguments. While these rules are quite expressive, they often do not correspond well to any words or phrases in natural language. To alleviate this problem, we designed another MRG by expanding out many of the rules. For example, the original MRG contained the following production rules for generating an infinite number of travel actions from the root symbol S.

```
⋆S -> ⋆Action
⋆Action -> ⋆Action, ⋆Action
⋆Action -> ⋆Travel
⋆Travel -> Travel( )
⋆Travel -> Travel( steps: ⋆Num )
```

We expand out the production rules as shown below to map S directly to specific travel actions so they correspond better to patterns such as "go forward" or "walk N steps".

```
⋆S -> Travel( )
⋆S -> Travel( steps: ⋆Num )
⋆S -> Travel( ), ⋆Action
⋆S -> Travel( steps: ⋆Num ), ⋆Action
⋆Action -> ⋆Action, ⋆Action
⋆Action -> Travel( )
⋆Action -> Travel( steps: ⋆Num )
```

While this process of expanding the production rules resulted in many more rules, these expanded rules usually correspond better with words or phrases in natural language. We still retain some of the recursive rules to ensure that the formal language remains as expressive as before.

4 Collecting Additional Data with Mechanical Turk

One of the motivations for studying ambiguous supervision is the potential ease of acquiring large amounts of training data. Without requiring semantic annotations, a human only has to demonstrate how language is used in context which is generally simple to do. We validate this claim by collecting additional training data for the navigation domain using Mechanical Turk (Snow et al., 2008).

There are two types of data we are interested in collecting: natural language navigation instructions and follower data. Thus, we created two tasks on Mechanical Turk. The first one asks the workers to supply instructions for a randomly generated sequence of actions. The second one asks the workers to try to follow a given navigation instruction in our virtual environment. The latter task is used to generate the corresponding action sequences for instructions collected from the first task.

4.1 Task Descriptions

To facilitate the data collection, we first recreated the 3D environments used to collect the original data (MacMahon et al., 2006). We built a Java application that allows the user to freely navigate the three virtual worlds constructed by MacMahon et al. (2006) using the discrete controls of turning left, turning right, and moving forward one step.

The follower task is fairly straightforward using our application. The worker is given a navigation instruction and placed at the starting location. They are asked to follow the navigation instruction as best as they could using the three discrete controls. They could also skip the problem if they did not understand the instruction or if the instruction did not describe a viable route. For each Human Intelligence Task (HIT), we asked the worker to complete 5 follower problems. We paid them $0.05 for each HIT, or 1 cent per follower problem. The instructions used for the follower problems were mainly collected from our Mechanical Turk instructor task with some of the instructions coming from data collected by MacMahon (2007) that was not used by Chen and Mooney (2011).

The instructor task is slightly more involved because we ask the workers to provide new navigation instructions. The worker is shown a 3D simulation of a randomly generated action sequence between length 1 to 4 and asked to write short, free-form instructions that would lead someone to perform those actions. Since this task requires more time to complete, each HIT consists of only 3 instructor problems. Moreover, we pay the workers $0.10 for each HIT, or about 3 cents for each instruction they write.

To encourage quality contributions, we use a tiered payment structure (Chen and Dolan, 2011) that rewards the good workers. Workers who have been identified to consistently provide good instructions were allowed to do higher-paying version of the same HITs that paid $0.15 instead of $0.10.

	Chen & Mooney	MTurk
# instructions	3236	1011
Vocabulary size	629	590
Avg. # words	7.8 (5.1)	7.69 (7.12)
Avg. # actions	2.1 (2.4)	1.84 (1.24)

Table 1: Statistics about the navigation instruction corpora. The average statistics for each instruction are shown with standard deviations in parentheses.

4.2 Data Statistics

Over a 2-month period we accepted 2,884 follower HITs and 810 instructor HITs from 653 workers. This corresponds to over 14,000 follower traces and 2,400 instructions with most of them consisting of single sentences. For instructions with multiple sentences, we merged all the sentences together and treated it as a single sentence. The total cost of the data collection was $277.92. While there were 2,400 instructions, we filtered them to make sure they were of reasonable quality. First, we discarded any instructions that did not have at least 5 follower traces. Then we looked at all the follower traces and discarded any instruction that did not have majority agreement on what the correct path is.

Using our strict filter, we were left with slightly over 1,000 instructions. Statistics about the new corpus and the one used by Chen and Mooney can be seen in Table 1. Overall, the new corpus has a slightly smaller vocabulary, and each instruction is slightly shorter both in terms of the number of words and the number of actions.

5 Experiments

We evaluate our new lexicon learning algorithm as well as the other modifications to the navigation system using the same three tasks as Chen and Mooney (2011). The first task is disambiguating the training data by inferring the correct navigation plans associated with each training sentence. The second task is evaluating the performance of the semantic parsers trained on the disambiguated data. We measure the performance of both of these tasks by comparing to gold-standard data using the same partial correctness metric used by Chen and Mooney which gives credit to a parse for producing the correct action type and additional credit if the arguments were also correct. Finally, the third task is to complete the end-to-end navigation task. There are two versions of this task, the complete task uses the original instructions which are several sentences long and the other version uses instructions that have been manually split into single sentences. Task completion is measured by the percentage of trials in which the system reached the correct destination (and orientation in the single-sentence version).

We follow the same evaluation scheme as Chen and Mooney and perform leave-one-map-out experiments. For the first task, we build a lexicon using ambiguous training data from two maps, and then use the lexicon to produce the best disambiguated semantic meanings for those same data. For the second and third tasks, we train a semantic parser on the automatically disambiguated data, and test on sentences from the third, unseen map.

For all comparisons to the Chen and Mooney results, we use the performance of their *refined landmarks plans* system which performed the best overall. Moreover, it provides the most direct comparison to our approach since both use a lexicon to refine the *landmarks plans*. Other than the modifications discussed, we use the same components as their system including using KRISP to train the semantic parsers and using the execution module from MacMahon et al. (2006) to carry out the navigation plans.

5.1 Inferring Navigation Plans

First, we examine the quality of the refined navigation plans produced using SGOLL's lexicon. The precision, recall, and F1 (harmonic mean of precision and recall) of these plans are shown in Table 2. Compared to Chen and Mooney, the plans produced by SGOLL has higher precision and lower recall. This is mainly due to the additional minimum support constraint we added which discards many noisy lexical entries from infrequently seen n-grams.

	Precision	Recall	F1
Chen and Mooney	78.54	78.10	78.32
SGOLL	87.32	72.96	79.49

Table 2: Partial parse accuracy of how well each algorithm can infer the gold-standard navigation plans.

	Precision	Recall	F1
Chen and Mooney	90.22	55.10	68.37
SGOLL	92.22	55.70	69.43
SGOLL with new MRG	88.36	57.03	69.31

Table 3: Partial parse accuracy of the semantic parsers trained on the disambiguated navigation plans.

5.2 Training Semantic Parsers

Next we look at the performance of the semantic parsers trained on the inferred navigation plans. The results are shown in Table 3. Here SGOLL performs almost the same as Chen and Mooney, with slightly better precision. We also look at the effect of changing the MRG. Using the new MRG for KRISP to train the semantic parser produced slightly lower precision but higher recall, with similar overall F1 score.

5.3 Executing Navigation Plans

Finally, we evaluate the system on the end-to-end navigation task. In addition to SGOLL and SGOLL with the new MRG, we also look at augmenting each of the training splits with the data we collected using Mechanical Turk.

Completion rates for both the single-sentence tasks and the complete tasks are shown in Table 4. Here we see the benefit of each of our modifications. SGOLL outperforms Chen and Mooney's system on both versions of the navigation task. Using the new MRG to train the semantic parsers further improved performance on both tasks. Finally, augmenting the

	Single-sentence	Complete
Chen and Mooney	54.40%	16.18%
SGOLL	57.09%	17.56%
SGOLL with new MRG	57.28%	19.18%
SGOLL with new MRG and MTurk data	57.62%	20.64%

Table 4: End-to-end navigation task completion rates.

	Computation Time
Chen and Mooney (2011)	2,227.63
SGOLL	157.30
SGOLL with MTurk data	233.27

Table 5: The time (in seconds) it took to build the lexicon.

training data with additional instructions and follower traces collected from Mechanical Turk produced the best results.

5.4 Computation Times

Having established the superior performance of our new system compared to Chen and Mooney's, we next look at the computational efficiency of SGOLL. The average time (in seconds) it takes for each algorithm to build a lexicon is shown in Table 5. All the results are obtained running the algorithms on Dell PowerEdge 1950 servers with 2x Xeon X5440 (quad-core) 2.83GHz processors and 32GB of RAM. Here SGOLL has a decidedly large advantage over the lexicon learning algorithm from Chen and Mooney, requiring an order of magnitude less time to run. Even after incorporating the new Mechanical Turk data into the training set, SGOLL still takes much less time to build a lexicon. This shows how inefficient it is to perform graph intersection operations and how our online algorithm can more realistically scale to large datasets.

5.5 Experimenting with Chinese Data

In addition to evaluating the system on English data, we also translated the corpus used by Chen and Mooney into Mandarin Chinese.[1] To run our system, we first segmented the sentences using the Stanford Chinese Word Segmenter (Chang et al., 2008). We evaluated using the same three tasks as before. This resulted in a precision, recall, and F1 of 87.07, 71.67, and 78.61, respectively for the inferred plans. The trained semantic parser's precision, recall, and F1 were 88.87, 58.76, and 70.74, respectively. Finally, the system completed 58.70% of the single-sentence task and 20.13% of the complete task. All of these numbers are very similar to the English results, showing the generality of the system in its ability to learn other languages.

[1]The translation can be downloaded at `http://www.cs.utexas.edu/~ml/clamp/navigation/`

5.6 Discussion

We have introduced a novel, online lexicon learning algorithm that is much faster than the one proposed by Chen and Mooney and also performs better on the navigation tasks they devised. Having a computationally efficient algorithm is critical for building systems that learn from ambiguous supervision. Compared to systems that train on supervised semantic annotations, a system that only receives weak, ambiguous training data is expected to have to train on much larger datasets to achieve similar performance. Consequently, such system must be able to scale well in order to keep the learning process tractable. Not only is SGOLL much faster in building a lexicon, it can also be easily parallelized. Moreover, the online nature of SGOLL allows the lexicon to be continually updated while the system is in use. A deployed navigation system can gather new instructions from the user and receive feedback about whether it is performing the correct actions. As new training examples are collected, we can update the corresponding n-gram and subgraph counts without rebuilding the entire lexicon.

One thing to note though is that while SGOLL makes the lexicon learning step much faster and scalable, another bottleneck in the overall system is training the semantic parser. Existing semantic-parser learners such as KRISP were not designed to scale to very large datasets and have trouble training on more than a few thousand examples. Thus, designing new scalable algorithms for learning semantic parsers is critical to scaling the entire system.

We have performed a pilot data collection of new training examples using Mechanical Turk. Even though the instructions were collected from very different sources (paid human subjects from a university for the original data versus workers recruited over the Internet), we showed that adding the new data into the training set improved the system's performance on interpreting instructions from the original corpus. It verified that we are indeed collecting useful information and that non-experts are fully capable of training the system by demonstrating how to use natural language in relevant contexts.

6 Related Work

The earliest work on cross-situational word learning was by Siskind (1996) who developed a rule-based system to solve the referential ambiguity problem. However, it did not handle noise and was tested only on artificial data. More recently, Fazly et al. (2010) proposed a probabilistic incremental model that can learn online similar to our algorithm and was tested on transcriptions of child-directed speech. However, they generated the semantic representations from the text itself rather than from the environment. Moreover, the referential ambiguity was introduced artificially by including the correct semantic representation of the neighboring sentence.

Our work falls into the larger framework of learning the semantics of language from weak supervision. This problem can be seen as an alignment problem where each sentence in the training data needs to be aligned to one or more records that represent its meaning. Chen and Mooney (2008) previously introduced another task that aligns sportscasting commentaries to events in a simulated soccer game. Using an EM-like retraining method, they alternated between building a semantic parser and estimating the most likely alignment. Liang et al. (2009) developed an unsupervised approach using a generative model to solve the alignment problem. They demonstrated improved results on matching sentences and events on the sportscasting task and also introduced a new task of aligning weather forecasts to weather information. Kim and Mooney (2010) further improved the generative alignment model by incorporating the full semantic parsing model from Lu et al. (2008). This resulted in a joint generative model that outperformed all previous results. In addition to treating the ambiguous supervision problem as an alignment problem, there have been other approaches such as treating it as a ranking problem (Bordes et al., 2010), or a PCFG learning problem (Borschinger et al., 2011).

Parallel to the work of learning from ambiguous supervision, other recent work has also looked at training semantic parsers from supervision other than logical-form annotations. Clarke et al. (2010) and Liang et al. (2011) trained systems on question and answer pairs by automatically finding semantic interpretations of the questions that would generate the correct answers. Artzi and Zettlemoyer (2011) use conversation logs between a computer system and a human user to learn to interpret the human utterances. Finally, Goldwasser et al. (2011) presented an unsupervised approach of learning a semantic parser by using an EM-like retraining loop. They use confidence estimation as a proxy for the model's prediction quality, preferring models that have high confidence about their parses.

7 Conclusion

Learning the semantics of language from the perceptual context in which it is uttered is a useful approach because only minimal human supervision is required. In this paper we presented a novel online algorithm for building a lexicon from ambiguously supervised relational data. In contrast to the previous approach that computed common subgraphs between different contexts in which an n-gram appeared, we instead focus on small, connected subgraphs and introduce an algorithm, SGOLL, that is an order of magnitude faster. In addition to being more scalable, SGOLL also performed better on the task of interpreting navigation instructions. In addition, we showed that changing the MRG and collecting additional training data from Mechanical Turk further improve the performance of the overall navigation system. Finally, we demonstrated the generality of the system by using it to learn Chinese navigation instructions and achieved similar results.

Acknowledgments

The research in this paper was supported by the National Science Foundation (NSF) under the grants IIS-0712097 and IIS-1016312. We thank Lu Guo for performing the translation of the corpus into Mandarin Chinese.

References

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*.

Antoine Bordes, Nicolas Usunier, and Jason Weston. 2010. Label ranking under ambiguous supervision for learning semantic correspondences. In *Proceedings of the 27th International Conference on Machine Learning (ICML-2010)*.

Benjamin Borschinger, Bevan K. Jones, and Mark Johnson. 2011. Reducing grounded learning tasks to grammatical inference. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*.

Pi-Chuan Chang, Michel Galley, and Chris Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the ACL Third Workshop on Statistical Machine Translation*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-2011)*, Portland, OR, June.

David L. Chen and Raymond J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of 25th International Conference on Machine Learning (ICML-2008)*, Helsinki, Finland, July.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11)*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the worlds response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010)*, pages 18–27.

Afsaneh Fazly, Afra Alishahi, and Suzanne Stevenson. 2010. A probabilistic computational model of cross-situational word learning. *Cognitive Science*, 34(6):1017–1063.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11)*.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*, pages 913–920, Sydney, Australia, July.

Rohit J. Kate. 2008. Transforming meaning representation grammars to improve semantic parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 33–40, Manchester, UK, August.

Joohyun Kim and Raymond J. Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*.

Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11)*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, Honolulu, HI, October.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.

Matt MacMahon. 2007. *Following Natural Language Route Instructions*. Ph.D. thesis, Electrical and Computer Engineering Department, University of Texas at Austin.

Cynthia Matuszek, Dieter Fox, and Karl Koscher. 2010. Following directions using statistical machine translation. In *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*.

Raymond J. Mooney. 2007. Learning for semantic parsing. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing: Proceedings of the 8th International Conference, CICLing 2007, Mexico City*, pages 311–324. Springer Verlag, Berlin.

Nobuyuki Shimizu and Andrew Haas. 2009. Learning to follow navigational route instructions. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-2009)*.

Jeffrey M. Siskind. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*, 61(1):39–91, October.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*.

Cynthia A. Thompson and Raymond J. Mooney. 2003. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, 18:1–44.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-07)*, pages 678–687, Prague, Czech Republic, June.