Technical Report, Artificial Intelligence Lab,
University of Texas at Austin, January 1996.

Lexical Acquisition: A Novel Machine Learning Problem

Cynthia A. Thompson and Raymond J. Mooney
Department of Computer Sciences
University of Texas
Austin, TX 78712
cthomp@cs.utexas.edu, mooney@cs.utexas.edu

January 19, 1996

Areas: Inductive Learning, Natural Language Acquisition

Abstract

This paper defines a new machine learning problem to which standard machine learning algorithms cannot easily be applied. The problem occurs in the domain of lexical acquisition. The ambiguous and synonymous nature of words causes the difficulty of using standard induction techniques to learn a lexicon. Additionally, negative examples are typically unavailable or difficult to construct in this domain. One approach to solve the lexical acquisition problem is presented, along with preliminary experimental results on an artificial corpus. Future work includes extending the algorithm and performing tests on a more realistic corpus.

1 Introduction

Most work in concept learning assumes a single concept is to be learned. Even when multiple concepts can be handled, it is usually assumed that the concepts do not overlap. This assumption is false in some domains, such as lexicons for natural language processing and speech recognition, where two words can have the same representation, or two different representations can be associated with one word. Traditional machine learning methods do not account for these phenomena. The approach presented in this paper is a first step towards adequately learning multiple, overlapping concepts.

In the lexical acquisition task, the concepts (words) and their definitions are extracted from a unique type of example. Examples are not in the traditional form of positive and negative examples of each concept. Instead, training examples are pairs of sentences and their description(s) in the target representation language. The target representation language might be a database query or case-role representation. Concept definitions must be learned for some of the words in the sentences. The task is to choose which word(s) in the sentences correspond to which part(s) of the representations. The lexicon thus learned can then be used to assist in parsing sentences into their correct representation.

The remainder of the paper is organized as follows. The next section gives some background on the lexical acquisition problem, followed in the next section by a formal definition of the problem. Next, the algorithm description is presented. This is followed by experimental results and future work. After a brief discussion of related work, the paper concludes.

2 Background

The approach described in this paper was developed in the context of learning lexicons for semantic parsing. The definition of *lexicon* as it is used here is a mapping from words to representations of their semantic meanings. The particular representation is determined by the domain at hand and the representation of entire sentences. The initial motivation for learning lexicons was so they could be used to bootstrap a parser acquisition system, CHILL (Zelle & Mooney, 1993, 1994; Zelle, 1995); they could then be used by the parser to map novel sentences into representations of their meanings.

A major assumption of our research has been the assumption of *compositionality*. This assumption states that the meaning representation of a sentence is composed from the meaning representations of the individual words and phrases in that sentence, in addition, perhaps, to some "connecting" information specific to the representation at hand. Conversely, each portion of the sentence representation is generated by the meaning of only one word in the

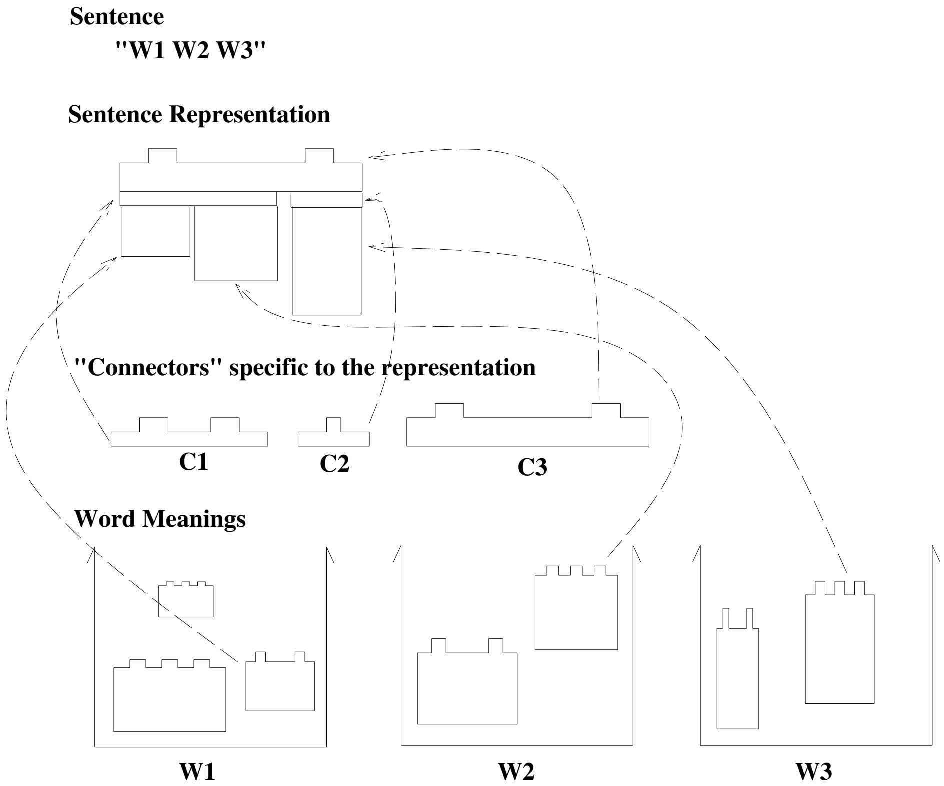

Figure 1: The Compositionality Assumption

sentence. Thus, during lexical acquisition, word meanings are derived from the components of the representations of sentences in which each word appears.

The compositional assumption allows the same learning method to be used for many different representation formalisms. One only needs to specify for each new representation the manner in which individual word meanings are built up (composed together) to form the meaning of sentences. A useful analogy is Lego blocks. First, for each representation formalism, there is a fixed set of "connector" Lego blocks, which hold together the structure of a sentence's meaning representation. Next, each word in the language may have several possible meanings, corresponding to different Lego blocks. For each sentence, one of these Lego blocks is chosen for each word, along with some of the "connector" Lego blocks; together these construct a meaning for the sentence. Figure 1 illustrates how the meanings of the words in a sentence fit together to form the meaning of the sentence.

3 The Lexical Acquisition Problem

The definition of the lexical acquisition problem as we view it is as follows:
Given:

- $I = \{(s_1, r_2), (s_2, r_2), \ldots, (s_n, r_n)\}$, a set of *(sentence, representation)* pairs, where each sentence contains an ordered list of words, and $R = \{r_1, r_2, \ldots, r_n\}$ is a set of strings of simple or structured symbols.

- P, the set of all components of the elements of R.

- $C \subseteq 2^P \times R$, a constructor relation from elements of the power set of P to elements of R.

Find:

M, a set of *(word, meaning)* pairs, where the words and their meanings are extracted from the sentences and their representations, respectively, such that the total number of *(word, meaning)* pairs is minimized. For each pair $(s_i, r_i) \in I$, where $s_i = (w_{i_1}, w_{i_2}, \ldots, w_{i_m})$, it must be the case that if p_j is chosen from M such that $(w_{i_j}, p_j) \in M$ for $1 \leq j \leq m$, then $(\{p_1, p_2, \ldots, p_m\}, r_i) \in C$. □

Less formally, a learner is presented with a set of sentences paired with their meanings, and a procedure for breaking down sentence meanings into their components and building up sentence meanings given their components. The goal is to find a lexicon which will simplify parsing. We hypothesize that minimizing the number of $(word, meaning)$ pairs in the learned lexicon will achieve this objective. The elements of M must also satisfy the constraint that the meaning representation of a sentence can be built up from the meanings of the words in the sentence, using the constructor relation, C.

To give a simple example of the lexical learning problem with a Conceptual Dependency (CD) representation (Schank, 1975), let I be:

{ ([**the,man,ate**], *[ingest,agt:[person,sex:male,age:adult]]*),
([**the,woman,ate**], *[ingest,agt:[person,sex:female,age:adult]]*),
([**the,sheep,ate**], *[ingest,agt:[animal,type:sheep]]*)}.

One M which satisfies the above criteria is

{ (**ate**, *[ingest]*), (**man**, *[person,sex:male,age:adult]*),
(**woman**, *[person,sex:female,age:adult]*),
(**sheep**, *[animal,type:sheep]*), (**the**, *[]*) }.

This is better than an alternative such as

{ (**ate**, *[ingest]*), (**the**, *[person,age:adult]*), (**the**, *[]*),
(**man**, *[male]*), (**woman**, *[female]*), (**sheep**, *[animal,type:sheep]*) }.

The first alternative has five *(word, meaning)* pairs and the second has six, so the first is preferred. In this example, the symbol *agt* is ignored, since it is one of the "connectors" in the representation formalism. Another possibility is to pair **ate** with *[ingest,agt:X]*, instead of ignoring this information.

For a first-order logical representation, an example of I might be

{ ([**the,man,ate**], {*ingest(man1,X), person(man1), male(man1), adult(man1)*}),
([**the,woman,ate**], {*ingest(woman1,X), person(woman1), female(woman1), adult(woman1)*}) }

In this case, an appropriate choice for M is

{ (**ate**, {*ingest(X,Y)*}), (**the**, {}), (**man**, {*person(X), male(X), adult(X)*}),
(**woman**, {*person(X), female(X), adult(X)*}) }.

In a representation in which each word in the sentence is mapped in the same order as, and one-to-one onto, the symbols in the representation, the above problem would be trivial. For example, an element of I could be the pair ([**the,man,ate**],*[the,man,ate]*). The problem takes on greater complexity as soon as the symbol order is permuted, or if some words in the sentence map to no symbol or to multiple symbols in the representation. Further complexity is introduced when a group of words in a sentence can map to one symbol. For example, a phrase can have its own atomic meaning, such as "kick the bucket," meaning to die. Finally, some representations can have complex representations in which a word's meaning is embedded within a (larger) structure. An example of the last type is one in which *[person,sex:male,age:adult]*, the meaning for `man` above, is embedded in the second element of the larger list *[ingest,agt:[person,sex:male,age:adult]]*. This is in contrast to a representation in which each word meaning corresponds to one or more elements in a simple list representing the meaning of a sentence, as in the first-order logical representation above.

Part of the input to the problem is the procedure, C, for building up and breaking down sentence meanings. The semantic representation currently used is a tree-based representation, derived from the CD representational theory and using case-role structures (Fillmore, 1968). For this representation, the method for breaking down sentence meanings is to enumerate all connected subgraphs of the sentence-representation tree. Because of the compositional assumption, these subgraphs are possible meanings for the individual words in the sentence. The number of these is exponential in the size of the representation tree, so some method is needed to eliminate many of the possibilities. This is done by initially insisting that a subgraph can only be one of the hypothesized meanings for a word if it appears in at least two representations of sentences in which the word appears. This result of finding common subgraphs is called a Tree Least General Generalization (TLGG), due to its similarity to the Least General Generalizations of (Plotkin, 1970).

4 Algorithm Description

Standard machine learning methods do not directly apply to this problem for two main reasons, synonymy (multiple words mapping to the same meaning) and polysemy (one word mapping to multiple meanings). Current systems which learn multiple concepts commonly use examples for other concepts as negative examples of the concept currently being learned. The implicit assumption is that concepts are disjoint, an unwarranted assumption in the lexical domain.

One attempt to apply standard machine learning techniques to the lexical acquisition problem might proceed as follows. First, it is noticed that only positive examples are available. One might think of using most specific conjunctive learning, or finding the intersection of all the representations for each word, as proposed by Anderson (1977). However, because of polysemy, the meaning for each word is potentially disjunctive, and this intersection may be empty.

Another attempt might involve analyzing the usage of different words to determine their respective definitions. For example, a list of hypothesized meanings for each word could be formed by enumerating the representations of all sentences in which the word appears, along with all components of these representations. A standard induction algorithm could then attempt to use this list as a source of training examples for each word. However, any attempt to use the list of one word as negative examples for another would be flawed. The learner could not know in advance which words are possibly synonymous, and thus which word lists to use as negative examples of other word meanings. Also, many representation components would be present in the lists of more than one word. This is a source of conflicting evidence for a learner, even without the presence of synonymy.

The current approach eliminates these problems by allowing the definitions of multiple words to overlap and by allowing multiple definitions to be learned for one word. Because of the compositionality assumption, constraints are imposed on the learned word meanings, so that the final lexicon is not overly general. Otherwise, one could hypothesize a lexicon such that each word maps to the entire representation of each sentence in which it appears. Due to compositionality, the learned meaning of one word in a sentence constrains the possible meanings of the other words.

Our algorithm uses a combination of TLGGs and greedy covering to solve the lexical acquisition problem. The choices to make at each iteration are which concept (word) to learn next and what a likely definition (meaning) for the word could be. TLGGs between pairs of sentence representations are used to help narrow down the number of hypothesized meanings for a word. The maximum number of pairings per word is a parameter provided by the user. This effectively builds generalizations of sentence representations, in a manner similar to GOLEM (Muggleton & Feng, 1992). Each choice of a *(word, meaning)* pair eliminates some possible meanings for other words in I, by removing elements from their TLGG list.

A sample corpus that could be used as input to the learning problem is as follows:

1. The boy hit the bat.
   [propel,agt:[person,sex:male,age:child],pat:[obj,type:baseball-bat]]
2. The boy hit the bat.
   [propel,agt:[person,sex:male,age:child],pat:[animal,type:flying-bat]]
3. The hammer hit the pasta.
   [propel,inst:[obj,type:hammer],pat:[food,type:pasta]]
4. The hammer moved.
   [ptrans,pat:[obj,type:hammer]]
5. The boy ate the pasta with the cheese.
   [ingest,agt:[person,sex:male,age:child],pat:[food,type:pasta,accomp:[food,type:cheese]]]
6. The boy ate the pasta with the fork.
   [ingest,agt:[person,sex:male,age:child],pat:[food,type:pasta],inst:[inst,type:fork]]
7. The man ate the pasta with the cheese.
   [ingest,agt:[person,sex:male,age:adult],pat:[food,type:pasta,accomp:[food,type:cheese]]]
8. The man ate the pasta with the fork.
   [ingest,agt:[person,sex:male,age:adult],pat:[food,type:pasta],inst:[inst,type:fork]]
9. The bat ate.
   [ingest,agt:[animal,type:flying-bat]]
10. The bat hit the pasta.
    [propel,inst:[obj,type:baseball-bat],pat:[food,type:pasta]]
11. The bat hit the pasta.
    [propel,agt:[animal,type:flying-bat],pat:[food,type:pasta]]

Our system utilizes a simple, greedy algorithm, illustrated in Figure 2. First, a table T is built from the training input. Each word, w, in S is entered into T, along with pointers

Build a table, T, from the input, consisting of each word, w in S, its TLGGs,
  and pointers, w_P, to the representations of sentences in which it appears.
Loop doing
  Add to the output and remove from T the $(word, meaning)$ pair which covers the
    highest percentage of sentences in which *word* appears;
    break ties within a word by choosing the largest *meaning*;
    break ties between words by choosing the least ambiguous word so far.
  Remove from $word_P$ entries to representations in which *meaning* appears.
  Mark *meaning* in these representations as being covered by *word*.
  Check and rederive if needed the TLGGs for words that appeared in
    sentences marked in this iteration.
Until no entries have potential meanings associated with them.

Figure 2: Algorithm Overview

(w_P) to the representations of the sentences in I in which it appears (w_R). Next, for each word, TLGGs of a random sample of pairs from w_R are derived and entered into T.

Then, the main loop is entered and greedy selection of the best word-TLGG (meaning) pair is performed. A TLGG is a good candidate for a word's meaning if it is part of the representation of a large percentage of sentences in which the word appears. The best word-TLGG pair in T, denoted (w, t), is the one with the highest percentage of this overlap. In some cases, multiple pairs have the same percentage overlap. First, if two TLGGs for one word have the same coverage, the one containing the most nodes is preferred. Second, if two different words have pairs with equal coverage, words which have fewer meanings up to this point are preferred.

Given the sample corpus above, a portion of T at this stage of the algorithm follows, with percentages added and TLGGs given in their preferred order.

W	TLGGs	W_P
boy	[person,sex:male,age:child] (100%), [male] (100%), [child] (100%),	[1,2,5,6]
	[ingest,agt:[person,sex:male,age:child],pat:[food,type:pasta]] (50%),	
	[propel,agt:[person,sex:male,age:child]] (50%)	
	[food,type:pasta] (50%), [pasta] (50%), [food] (50%),	
pasta	[food,type:pasta] (100%), [pasta] (100%), [food] (100%),	[3,5,6,7,8,10,11]
	[ingest,agt:[person,sex:male],pat:[food,type:pasta]] (57.1%),...	

ate	[ingest] (100%), [ingest,agt:[person,sex:male],pat:[food,type:pasta]] (80%), [food,type:pasta] (80%), [person,sex:male] (80%), [pasta] (80%),...	[5,6,7,8,9]
cheese	[ingest,agt:[person,sex:male], pat:[food,type:pasta,accomp[food,type:cheese]]] (100%), [food,type:pasta,accomp:[food,type:cheese]] (100%), [person,sex:male] (100%), [food,type:cheese] (100%), [male] (100%), [pasta] (100%), [cheese] (100%)	[5,7]
bat	[propel] (80%), [animal,type:flying-bat] (60%), [flying-bat] (60%), [propel,agt:[person,sex:male,age:child]] (40%), [person,sex:male,age:child] (40%), [propel,pat:[food,type:pasta]] (40%), [obj,type:baseball-bat] (40%), [male] (40%),...	[1,2,9,10,11]

The meanings *[person,sex:male,age:child]*, *[child]*, and *[male]* all have 100% coverage for `boy`, since they appear in every sentence in which `boy` appears. *[person,sex:male,age:child]* is preferred over the others since it contains more nodes. In the first iteration, many of the above words (and several others, not included in the partial table shown) have a TLGG that covers 100% of the sentence representations for that word. Given more examples there would be fewer such cases. To preserve clarity in the remainder of the example, let us choose (**boy**,*[person,sex:male,age:child]*) as the best (w, t) pair for the first iteration.

In the second step of each iteration, some sentence representations are marked to reflect the meaning just learned. For each element r in w_R, the portion of r that matches t, if any, is marked off as being covered. The meaning, t, may not occur in some elements of w_R since w may be a polysemous word.

The representations for sentences one, two, five, and six get marked as follows, where the portion in `typeface` is the portion marked off as being learned:

1. [propel,agt:`[person,sex:male:age:child]`,pat:[obj,type:baseball-bat]]
2. [propel,agt:`[person,sex:male:age:child]`,pat:[animal,type:flying-bat]]
5. [ingest,agt:`[person,sex:male:age:child]`,pat:[food,type:pasta, accomp:[food,type:cheese]]]
6. [ingest,agt:`[person,sex:male:age:child]`,pat:[food,type:pasta],inst:[inst,type:fork]]

After the sentence representations have been marked, w's entry is modified. Once the meaning for w is chosen for a sentence in which it appears, w's meaning for this sentence has been covered, and the representation no longer has to be in w_P. This is due to the assumption that the meaning of each word in a sentence appears at most once in its representation. Thus, if t occurs n times in one of these representations, the pointer is removed n times. If w_P becomes empty after this step, w is removed from T, since all of its meanings have been learned.

In our example, boy_P becomes empty, since *[person,sex:male,age:child]* appears in every sentence pointed to. Therefore, **boy** is removed from the table.

Finally, for each *word* $\in T$, if *word* appears in one of the sentences whose representation was marked in the second step, the algorithm checks that the TLGG list for *word* is still valid with respect to $word_R$ and removed if it is not. A TLGG is valid if it is in an unmarked part of some representation in $word_R$. Those TLGGs remaining are reordered if needed. This step is taken because of the assumption that each part of a sentence representation is due to the meaning only one word or group of words in the sentence.

Several words occur in sentences whose representations were marked above. The new entries for **ate**, **bat**, **cheese**, and **pasta** after removing invalid TLGGs and reordering those remaining are:

W	*TLGGs*
ate	[ingest] (100%), [ingest,pat:[food,type:pasta]] (80%), [food,type:pasta] (80%), [pasta] (80%),...
pasta	[food,type:pasta] (100%), [food] (100%), [pasta] (100%), [ingest,pat:[food,type:pasta]] (57.1%), [ingest,pat:[food,type:pasta,accomp:[food,type:cheese]]] (28.6%),...
cheese	[food,type:pasta,accomp:[food,type:cheese]] (100%), [food,type:cheese] (100%),...
bat	[propel] (80%), [animal,type:flying-bat] (60%),[flying-bat] (60%), [propel,pat:[food,type:pasta]] (40%), [obj,type:baseball-bat] (40%), [food,type:pasta] (40%), [pasta] (40%), [baseball-bat] (40%)

If the removal of invalid TLGGs leaves an empty TLGG list, more TLGGs are derived from the unmarked portion of the representations still pointed to by $word_P$. If all portions of all elements of $word_R$ contain fully marked representations, *word* is removed from T. Iteration continues until all $S \in I$ have all portions of their representations marked, meaning they can be assembled from learned word meanings.

The remaining iterations of the algorithm for the running example will not be illustrated here, but the final learned lexicon for this example, given good choices in tie-breaking situations, would be:
(**boy**, *[person,sex:male,age:child]*), (**ate**, *[ingest]*), (**pasta**, *[food,type:pasta]*),
(**man**, *[person,sex:male,age:adult]*), (**hammer**, *[obj,type:hammer]*), (**hit**, *[propel]*),
(**moved**, *[ptrans]*), (**fork**, *[inst,type:fork]*), (**cheese**, *[food,type:cheese]*),
(**bat**, *[animal,type:flying-bat]*), (**bat**, *[obj,type:baseball-bat]*).

5 Experimental Results

Our system has been tested on a corpus based on that of McClelland and Kawamoto (1986). This corpus consists of 1475 sentence/case-structure pairs, artificially produced from a set of 19 sentence templates. The case-structure portion of these pairs was modified to produce deeper semantic representations. Examples of sentences and their representation are given in the small corpus of the previous section. The larger corpus was also translated into Japanese, pairing the Japanese sentences with these same representations.

A different random set of training examples was chosen for each of three trials. To measure the success of the system, the percentage of correct word meanings learned was calculated. There are many ways to calculate this metric, and the one used here was based on a quantification of the similarity between the learned representation(s) for a word and the correct one(s). For example, if the correct meaning is *[person,sex:female,age:adult]*, the learned meaning *[person,sex:female]* is closer to the correct meaning than *[person]*. The accuracy results below are based on the percent overlap between the learned lexicon and the correct one.

For the English corpus, the average accuracy was 91.7% at 950 examples. An attempt was made to improve this result by artificially introducing more search into the algorithm. Three tests were run on each training set, causing random permutations in the choices made in tie-breaking situations. The test which returned the smallest lexicon was the one chosen as the output for that training set. The accuracy at 650 examples for these trials was 97.5%, an obvious improvement.

The accuracy for the Japanese corpus was 92.8% at 650 examples. Next, we appended the Japanese and English versions of the corpus, and ran similar experiments on this combined corpus. By combining the two corpora, bilingual learning is simulated. After 1900 examples, the average accuracy was 84%.

In order to determine whether the lexicons learned above were useful, we next used them as background knowledge for the parser acquisition system, CHILL. The generalization accuracy of the resulting parsers was compared to parsers learned using the correct lexicon as background knowledge. Figure 3 shows the resulting accuracies with up to 650 training examples. Again, accuracies were computed by measuring the overlap between parses generated by the learned parser and the correct parses. The upper curve shows the results when given the correct lexicon as background knowledge, while the other curve shows the results when given the lexicons learned in the previous tests. The accuracy at 650 examples is 92.1% for the former curve and 87.3% for the latter.

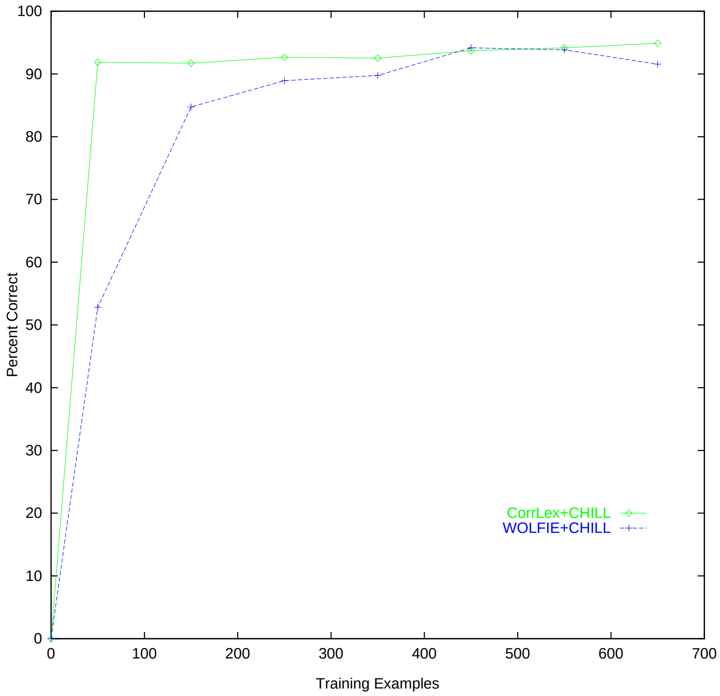

Figure 3: Generalization accuracy of CHILL

6 Future Work

The experimental results to date have been encouraging, but there are several opportunities for improvement. The algorithm could be improved, for example, by making better choices when multiple *(word, meaning)* pairs have equal coverage. Another possible improvement is to make the choice of a pair probabilistic rather than deterministic. With enough future evidence, changes can be made and a different pair may be learned. Finally, incorporating the ability to use background knowledge into the current technique may prove to be useful.

More tests should be performed on the current domain, as well as with alternate representations such as database queries. For this, something analogous to TLGGs will be used to derive the hypothesized meanings for each word. Another possibility is to attempt to learn a translation lexicon by using sentences from one language as the concept source and sentences from another language as the definition source.

Other domains in which an improved version of this technique might be useful include speech, vision, Optical Character Recognition, planning, diagnosis, or any domain in which there are multiple, overlapping concepts to be learned.

7 Related Work

Within lexical acquisition, the most closely related system is that described by Siskind (1992, 1994). His system learns both the syntax and semantics of words, but assumes that a universal grammar is available to the learner. In addition, his system cannot handle ambiguity as well as ours. It does, however, handle noisy training data in the form of incorrect meaning representations being paired with some of the input sentences, in addition to *referential uncertainty*, which is the assumption that the learner cannot infer a unique meaning for an utterance based on the environment in which it occurs.

8 Conclusion

This paper has presented the problem of lexical acquisition, learning word meanings from examples of sentences paired with semantic representations, as an interesting problem for machine learning. Unfortunately, the presence of both synonymy and polysemy prevents the application of standard inductive learning methods to this problem. However, the assumption of compositionality, i.e. that sentence meanings are constructed by composing selected meanings for each of its words, can be exploited to usefully constrain the learning process. We have presented an algorithm for this problem based on greedy covering and demonstrated it's ability to acquire an accurate lexicon from an artificial natural-language corpus. The acquired lexicon has also been shown to be useful in aiding the subsequent learning of a natural language parser.

References

Anderson, J. R. (1977). Induction of augmented transition networks. *Cognitive Science*, *1*, 125–157.

Fillmore, C. J. (1968). The case for case. In Bach, E., & Harms, R. T. (Eds.), *Universals in Linguistic Theory*. Holt, Reinhart and Winston, New York.

McClelland, J. L., & Kawamoto, A. H. (1986). Mechanisms of sentence processing: Assigning roles to constituents of sentences. In Rumelhart, D. E., & McClelland, J. L. (Eds.), *Parallel Distributed Processing, Vol. II*, pp. 318–362. MIT Press, Cambridge, MA.

Muggleton, S., & Feng, C. (1992). Efficient induction of logic programs. In Muggleton, S. (Ed.), *Inductive Logic Programming*, pp. 281–297. Academic Press, New York.

Plotkin, G. D. (1970). A note on inductive generalization. In Meltzer, B., & Michie, D. (Eds.), *Machine Intelligence (Vol. 5)*. Elsevier North-Holland, New York.

Schank, R. C. (1975). *Conceptual Information Processing*. North-Holland, Oxford.

Siskind, J. M. (1992). *Naive Physics, Event Perception, Lexical Semantics and Language Acquisition*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Siskind, J. M. (1994). Lexical acquisition in the presence of noise and homonymy. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 760–766.

Zelle, J. M. (1995). *Using Inductive Logic Programming to Automate the Construction of Natural Language Parsers*. Ph.D. thesis, University of Texas, Austin, TX.

Zelle, J. M., & Mooney, R. J. (1993). Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 817–822 Washington, D.C.

Zelle, J. M., & Mooney, R. J. (1994). Inducing deterministic Prolog parsers from treebanks: A machine learning approach. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pp. 748–753 Seattle, WA.