Lexical Acquisition as Constraint Satisfaction

Jeffrey Mark Siskind*
University of Pennsylvania
Institute for Research in Cognitive Science
3401 Walnut Street, Room 411C
Philadelphia PA 19104
215/898–0367
internet: Qobi@CIS.UPenn.EDU

February 16, 1993

Abstract

In this paper we present a computational study of lexical acquisition. We attempt to characterize the lexical acquisition task faced by children by defining a simplified formal approximation of this task which we term the mapping problem. We then present a novel strategy for solving large instances of this mapping problem. This strategy is capable of learning the word-to-meaning mappings for as many as 10,000 words given corpora of 20,000 utterances. Such lexical acquisition is accomplished in a language independent fashion without any reference to the syntax of the language being learned.

Topic Area: Lexical Acquisition

1 Introduction

When learning their native language, children must learn a lexicon mapping the words in that language to their meanings. A common conjecture, dating as far back as Locke (1690, cf. Gleitman 1990) and Saint Augustine (cf. Bruner 1983), is that children learn word-to-meaning mappings by hearing isolated words in a context where their meanings are readily apparent. For example, they would learn that *ball* means 'ball' by hearing the word *ball* while being shown a ball. If this conjecture were true, lexical acquisition would be a non-problem. Children would simply be presented with the lexicon as input.

Unfortunately, the aforementioned folk theory appears to be false. A simple examination of parent-to-child speech corpora such as ChildeS (MacWhinney and Snow 1985) reveals that children rarely if ever hear single word utterances. Instead, they hear multi-word utterances such as *Mommy picked up the ball.* If they heard this utterance in a context where it was clear that the whole utterance meant that mommy picked up the ball, they would face the task of determining that *Mommy*, *picked up*, and *ball* meant 'mommy,' 'picked up,' and 'ball' respectively and not vice versa.

The task faced by children is—in fact—even more complex than this. The above scenario assumed that children could unambiguously determine the meaning of each utterance from context even if the meaning of each word was unclear. More likely, children face a situation where they are also uncertain of the meanings of the utterances that they hear. When hearing the utterance *Mommy picked up the ball* they might be uncertain as to whether this utterance meant that mommy picked up the ball, that mommy was holding the ball, or for that matter that mommy wanted the ball. We call this *referential uncertainty*. Children therefore face a two-fold task of first disambiguating the referential uncertainty to determine what

*This research was supported in part by an AT&T Bell Laboratories Ph.D. scholarship to the author, by a Presidential Young Investigator Award to Professor Robert C. Berwick under National Science Foundation Grant DCR–85552543, by a grant from the Siemens Corporation, and by the Kapor Family Foundation. This research was also supported in part by ARO grant DAAL 03–89–C–0031, by DARPA grant N00014–90–J–1863, by NSF grant IRI 90–16592 and by Ben Franklin grant 91S.3078C–1. Part of this research was performed while the author was visiting Xerox PARC as a research intern and as a consultant.

utterances mean and then determining which words in those utterances correspond to which components of those meanings.

In this paper we present a computational study of lexical acquisition. We attempt to characterize the lexical acquisition task faced by children by defining a simplified formal approximation of this task which we term the *mapping problem*. In this problem, a learner is presented with a sequence of utterances—each being a sequence of words—where each utterance is paired with a set of expressions which denote the referentially uncertain hypothesized meanings of that utterance. For example, the learner might be presented with the utterance *Mommy picked up the ball* paired with expressions such as CAUSE(**mother**, GO(**ball**, UP)), GRASP(**mother**, **ball**), or WANT(**mother**, **ball**). Let us assume that a language learner can bring an elaborate cognitive apparatus to bear to hypothesize potential meanings for an utterance from an extended visual and psychological context.[1] From a corpus containing such utterances paired with sets of hypothesized meanings, the learner must infer—among other things—that CAUSE(**mother**, GO(**ball**, UP)) is the correct meaning of *Mommy picked up the ball* and that **mother**, CAUSE(x, GO(y, UP)), and **ball** are the correct meanings of *Mommy*, *picked-up*, and *ball* respectively.

We present a novel strategy for solving large instances of the mapping problem. An important characteristic of this algorithm is that it scales up to very large tasks, correctly identifying the word-to-meaning mappings for as many as 10,000 words from a corpus of 20,000 utterances, each paired with four meaning expressions. In some ways, this task is as large as the task faced by real children. We also demonstrate how the algorithm scales up to handle a high degree of referential uncertainty. Philosophers such as Quine (1960) have pointed out the apparent difficulty or even impossibility of the language acquisition task if a child associates a large, or perhaps infinite set of referentially uncertain meanings with each utterance. This has prompted researchers such as Gleitman (1990) and Fisher et al. (1991) to suggest that children do—and in fact must—use syntactic and prosodic information when learning word meanings. In this paper we offer counter evidence to the claim that syntactic and prosodic information is necessary by demonstrating effective procedures which can acquire large lexica without knowledge or use of syntax or prosody. We do not claim that children actually employ any of the techniques discussed in this paper; only that they could—in principle—do so.

The remainder of this paper is organized as follows. Section 2 gives a precise definition of the mapping problem. Section 3 reviews a divide-and-conquer technique for finding solutions to instances of the mapping problem which has been previously reported in the literature. While this technique has been successful in solving small instances of the mapping problem, it becomes intractable for larger instances. Section 4 reviews cross-situational learning, another well known technique which has been proposed by numerous authors as the basis of lexical acquisition in children. We demonstrate that while this technique is computationally feasible even for large tasks, it is incomplete in that it will never converge to a single solution to an instance of the mapping problem, even when a unique solution exists. We review these prior techniques as they form the basis for our new approach. Section 5 shows how the mapping problem can be viewed as a constraint satisfaction problem and discusses a novel technique based on arc consistency for solving instances of the mapping problem. Section 6 describes a number of experiments which have been performed to demonstrate the effectiveness of this new technique on very large corpora. Section 7 discusses the relevance of these results to both child language acquisition research and automated lexicography and presents some unsolved issues for future research.

2 The Mapping Problem

Throughout this paper we assume that one can represent the meanings of words, phrases, and utterances as *terms*. A term is either a *constant symbol* (e.g. **John**), a *variable* (e.g. x), or a *function symbol* (e.g. GO) applied to one or more terms (e.g. GO(**John**, TO(x))). It is important to note that variables denote unfilled argument positions in words or phrases, and not pronouns, traces, or entities bound by quantification. Accordingly, the meanings of utterances will be represented by ground (i.e. variable-free) terms since utterances typically have no unfilled argument positions.

We further assume that an external cognitive apparatus provides an inventory of constant and function symbols out of which one can construct such terms. For expository purposes, in this paper we adopt symbols

[1] In this paper we do not address the issue of how such a cognitive apparatus might work; we simply presuppose its existence.

reminiscent to those proposed by Jackendoff (1983, 1990). Nothing turns on this decision, however, since the algorithm we present does not rely on any interpretation given to terms or their constituent symbols. Rather, it simply manipulates terms as symbolic entities, combining smaller terms to form larger ones, breaking terms down into their constituents, and pairing those constituents with words, phrases, and utterances as part of the lexical acquisition process. Thus our algorithm works equally well for *any* method of representing meanings as terms.

Many languages have words which play a purely syntactic role and lack any semantic content. Examples of such words are case markers such as the English word *of* and the Japanese word *ga*, and complementizers such as the English word *that*. To indicate that such words lack any semantic content, we represent their meaning via the distinguished symbol $\perp$. For expedience, we also use $\perp$ to represent the meanings of words which fall outside the semantic space of the chosen inventory of primitives. Thus, since the Jackendovian notation we adopt in this paper does not represent definite and indefinite reference, determiners such as the English word *the* will take on $\perp$ as their meaning.[2]

Terms can be used to represent the meanings of words and phrases in addition to whole utterances. For example, we might represent the meanings of the words *to* and *school* as $\text{TO}(x)$ and **school**, and the phrase *walked to school* as $\text{GO}(x, \text{TO}(\textbf{school}))$. With such a representation it appears natural to represent the meaning of a phrase such as *to school* with the term $\text{TO}(\textbf{school})$ since it can be derived compositionally from the meanings of its constituents. We formalize this intuitive compositional process by defining a *linking rule*, a function $\textsc{Link}(s, t)$ that takes two terms s and t as input and returns the set of all terms that can be formed by appropriately combining s and t. We refer to s as the *head* and to t as the *complement*. The linking rule we adopt is defined as follows:

ALGORITHM To compute $\textsc{Link}(s, t)$:

1. If t is not a ground term, return the empty set.
   - For example, $\textsc{Link}(\text{GO}(x, y), \text{TO}(x)) \Rightarrow \{\}$.
2. Otherwise, if s is just a variable, return the empty set.
   - For example, $\textsc{Link}(x, \textbf{school}) \Rightarrow \{\}$.
3. Otherwise, if $t = \perp$, return the singleton set $\{s\}$.
   - For example, $\textsc{Link}(\textbf{school}, \perp) \Rightarrow \{\textbf{school}\}$.
4. Otherwise, if s is a ground term, return the empty set.
   - For example, $\textsc{Link}(\textbf{John}, \textbf{school}) \Rightarrow \{\}$.
5. Otherwise, return the set of all terms that can be derived by choosing some variable in s and replacing all instances of that variable with t.
   - For example, $\textsc{Link}(\text{GO}(x, y), \textbf{John}) \Rightarrow \{\text{GO}(\textbf{John}, y), \text{GO}(x, \textbf{John})\}$. □

Step 1 stipulates that complements must be variable-free. This avoids issues of undesired variable capture. Step 2 stipulates that heads must not be identity functions. Step 3 is a special case for dealing with $\perp$. Step 4 states that only constituents with a free argument position can take arguments. Step 5 is the essence of the linking rule which is reminiscent of β-reduction. Note that step 5 does not specify which variable in the head is to be replaced with the complement. For example, $\textsc{Link}(\text{GO}(x, y), \textbf{John})$ returns both $\text{GO}(\textbf{John}, y)$ and $\text{GO}(x, \textbf{John})$ as possible linkings. Others (cf. Pinker 1989) have suggested far more specific linking rules which eliminate such ambiguity. The learning algorithm we present in this paper converges to unique solutions despite the fact that we adopt such a highly underspecified linking rule.

We can iterate this process of deriving the meanings of phrases from the meanings of their constituents to derive the meaning of a whole utterance from the meanings of the words comprising that utterance. We formalize this notion via the following *semantic interpretation rule*. A *lexicon* is a map from words to terms representing their meanings. An *interpretation tree* for a sequence of words $w_1 \cdots w_m$ under a lexicon L is any binary tree with m leaf nodes, where $v_1, \ldots, v_m$ are those leaf nodes taken from left to right, such that

- each leaf node v_j is labeled with $L(w_j)$ and

[2] This is simply an expository issue and not an inherent limitation of our technique. Adopting a richer inventory of semantic primitives will allow our technique to represent and learn the meanings of determiners.

- each non-leaf node v with daughters v_{left} and v_{right} is labeled either with an element of $\textsc{Link}(s,t)$ or with an element of $\textsc{Link}(t,s)$ where s is the label of v_{left} and t is the label of v_{right}.

We say that s is a *possible interpretation* of an utterance u under a lexicon L if s is the label of the root of some interpretation tree for the sequence of words $w_1 \cdots w_m$ comprising u under L. By existentially quantifying over all binary branching trees we insure that our notion of possible interpretation—and thus our learning algorithm—makes only minimal use of syntax.

The mapping problem can now be stated formally as follows. Given a corpus consisting of a sequence of utterances—each being a sequence of words—where each utterance is paired with a set of ground terms representing hypothesized meanings for that utterance, find a lexicon mapping each word in the corpus unambiguously to a single term such that at least one of the hypothesized meanings paired with each utterance is a possible interpretation of that utterance under that lexicon.

One important characteristic of the mapping problem is that it requires each word to have a single meaning. We refer to this as the *monosemy constraint*. If we did not enforce the monosemy constraint—and allowed an unbounded number of polysemous meanings per word—the mapping problem would be highly unconstrained. It would admit trivial solutions where each occurrence of a word would take on a different meaning. One such trivial solution would assign the first hypothesized meaning of each utterance as the meaning of the first word in that utterance and assign $\bot$ as the meaning of the remaining words. Experience has shown that adopting the monosemy constraint limits almost all instances of the mapping problem to a single unique solution. One drawback of the algorithm we have studied is that it will fail to find a solution to an instance of the mapping problem if that solution requires a polysemous lexicon. Admittedly, a strict monosemy constraint is far too strong since human languages exhibit polysemy and children succeed in learning in the presence of such polysemy. Future work will address ways of relaxing the strict monosemy constraint to allow learning a limited degree of polysemy.

As far as lexical acquisition is concerned, one important characteristic of the linking rule is that it is reversible. It is possible to define the procedure $\textsc{Unlink}(r)$ which returns the set of all pairs of terms $\langle s,t\rangle$ such that $r \in \textsc{Link}(s,t)$. For example, $\textsc{Unlink}(\text{GO}(\textbf{ball}, \text{TO}(\textbf{John})))$ yields the following four head/complement pairs:

Head	Complement
$\text{GO}(x, \text{TO}(\textbf{John}))$	**ball**
$\text{GO}(\textbf{ball}, x)$	$\text{TO}(\textbf{John})$
$\text{GO}(\textbf{ball}, \text{TO}(x))$	**John**
$\text{GO}(\textbf{ball}, \text{TO}(\textbf{John}))$	$\bot$

Note specifically the last possibility of assigning $\bot$ as the meaning of the complement, leaving the entire original term r as the meaning of the head. This option will always be present when computing $\textsc{Unlink}(r)$ for any term r.

The function $\textsc{Unlink}(r)$ can be computed by the following algorithm:

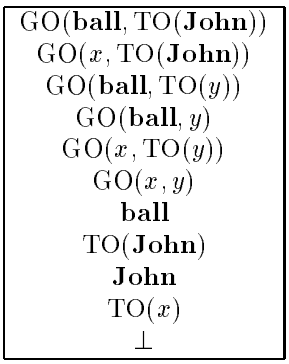
Algorithm To compute $\textsc{Unlink}(r)$ as a set of pairs $\langle s,t\rangle$:

Either return $\langle r, \bot\rangle$ or perform the following two steps:

1. Select some ground subterm of r and assign it to t.
2. Replace one or more occurrences of t in r with a new variable to yield s. □

Each alternate choice of subterm of r, in step 1, or choice of which occurrences of t to replace in r, in step 2, will yield a different pair $\langle s,t\rangle$ in the computed result. Computing $\textsc{Unlink}(r)$ can be viewed as a special case of higher-order unification (Huet 1975), or more specifically, higher-order matching.

Given the function $\textsc{Unlink}(r)$, one can define the function $\textsc{Unlink}^*(r)$ to compute the set of all possible terms which could appear as the label of some node in some interpretation tree whose root is labeled r. For example, $\textsc{Unlink}^*(\text{GO}(\textbf{ball}, \text{TO}(\textbf{John})))$ yields the following set:



Such a set would be useful since any word appearing in an utterance whose meaning was GO(**ball**, TO(**John**)) would have to take on a member of the above set as its meaning. We refer to a member of UNLINK$^*(r)$ as a *submeaning* of r. Note that UNLINK$^*(r)$ will always include both r itself and $\perp$ as members. This is because $\langle r, \perp \rangle$ is always a member of UNLINK(r). The fact that UNLINK$^*(r)$ will always contain $\perp$ has important consequences for the cross-situational learning technique we discuss in section 4.

The function UNLINK$^*(r)$ can be defined recursively as follows:

$$\text{UNLINK}^*(r) \triangleq \{r\} \cup \bigcup_{\langle s,t \rangle \in \text{UNLINK}(r)} (\text{UNLINK}^*(s) \cup \text{UNLINK}^*(t))$$

The following algorithm, however, is a more efficient method for enumerating the elements of UNLINK$^*(r)$ when r is a ground term:

> ALGORITHM To compute UNLINK$^*(r)$ for a ground term r:
>
> Either return $\perp$ or let s be some subexpression of r and repeat the following two steps an arbitrary number of times:
>
> 1. Select some proper subexpression t of s not containing any variables introduced in step 2.
> 2. Replace one or more occurrences of t in s with a new variable.
>
> Upon completion, s is a member of UNLINK$^*(r)$. □

The functions UNLINK(r) and UNLINK$^*(r)$ will play a prominent role in the algorithm we present for solving the mapping problem.

3 A Divide-and-Conquer Solution to the Mapping Problem

One can attempt to solve an instance of the mapping problem by enumerating all possible interpretation trees for all possible pairings of each utterance in the corpus to one of its hypothesized meanings. While it is intractable to enumerate all binary branching interpretation trees, one can adopt some syntactic theory and enumerate only those interpretation trees which are valid parse trees under that syntactic theory. Siskind (1990, 1991, 1992) described such a technique and applied it successfully to small corpora in both English and Japanese, learning a single correct word-to-meaning mapping for each word in the those corpora. We will refer to this technique as divide-and-conquer. Since this divide-and-conquer technique incorporates syntactic knowledge, it does not solve exactly the same formulation of the mapping problem as presented here. Nonetheless, the problems are similar enough to warrant comparison.

Despite the success of the divide-and-conquer technique in learning small fragments of English and Japanese, it suffers from a major shortcoming. Siskind (1992) reports that processing the 9 utterance English corpus takes about an hour of elapsed time, while processing the 9 utterance Japanese corpus takes about twelve hours of elapsed time on a Symbolics XL1200$^{\text{TM}}$ computer. Furthermore, the divide-and-conquer technique does not scale well. It can not process any larger corpora in a reasonable amount of time. The remainder of this paper presents a new algorithm which alleviates this shortcoming. A key feature of this new algorithm is that not only does it scale up to handle substantially larger instances of the mapping problem, it furthermore does not rely on any syntactic knowledge.

4 Cross-Situational Learning

A simple explanation for lexical acquisition in children is often proposed. This scheme suggests that when a child hears a word she can hypothesize all of the potential meanings for that word from the non-linguistic context of the utterance containing that word. Upon hearing that word in several different utterances—each in a different context—she can intersect the corresponding sets to find those meanings which are consistent across the different occurrences of that word. Presumably, hearing words in enough different situations would enable the child to rule out all incorrect hypotheses and uniquely determine word meanings.

We can attempt to formalize this procedure as follows. In the context of the mapping problem, the learner hears a sequence of utterances each paired with a set of terms. A word in an utterance u_i must take on as its meaning a submeaning of one of the hypothesized meanings paired with that utterance, i.e. a member of $\textsc{Unlink}^*(t_{ik})$ where t_{ik} is one of the hypothesized meanings paired with u_i. Thus, the learner could maintain a map D from words to sets of terms. At any point in time, $D(w)$ would be the set of all meanings for w consistent with the utterances heard so far. We refer to $D(w)$ as the *meaning domain* of w. Initially, D would map each word to the universal set of all terms. Upon hearing an utterance u_i consisting of the words $w_{i1} \cdots w_{i,m(i)}$ paired with the terms $t_{i1}, \ldots, t_{i,l(i)}$, the learner would form the set $\textsc{Unlink}^*(t_{i1}) \cup \cdots \cup \textsc{Unlink}^*(t_{i,l(i)})$ and update the meaning domain $D(w_{ij})$ for each word w_{ij} in the utterance to be the intersection of its old value and this set. Thus we have the following simple procedure:

```
1   for ⟨u_i, {t_i1, ..., t_i,l(i)}⟩ ∈ CORPUS
2       let w_i1 ··· w_i,m(i) = u_i in
3           for j from 1 to m
4               D(w_ij) ← D(w_ij) ∩ ⋃_{k=1}^{l(i)} UNLINK*(t_ik)
```

While executing the above procedure, the size of each meaning domain decreases monotonically. It is easy to see that if any meaning domain ever became empty the instance of the mapping problem would have no solution. Likewise, if the procedure reduces all meaning domains to singleton sets then the elements of those singleton sets would constitute a unique solution to the instance of the mapping problem. If however, the procedure terminates without reducing all of the meaning domains to singletons, then uncertainty would remain as to which—if any—of the meanings for a given word is correct. Each element in the cross-product of the meaning domains would constitute a potential solution to the instance of the mapping problem. Each potential solution must be checked for global consistency across the entire corpus leading to uncertainty as to whether zero, one, or several of these potential solutions is an actual solution.

Informal versions of the above procedure have been suggested by numerous authors for hundreds, perhaps thousands of years. Locke (1690, cf. Gleitman 1990), Saint Augustine (cf. Bruner 1983) and Socrates all offer some variant of the above procedure as an explanation for child language acquisition. More recently, Pinker (1989) informally discusses this procedure, calling it 'event category labeling.' If this procedure is applied to instances of the mapping problem with the particular linking rule discussed in section 2, $\textsc{Unlink}^*(r)$ will contains $\perp$ for every term r. Thus this strategy will never be able to rule out $\perp$ as a potential meaning for any word and accordingly will never uniquely determine the meaning of any word that does not mean $\perp$. It should be stressed that this limitation is not simply an artifact of our formulation. Simply stated, cross-situational learning can never uniquely determine the meaning of *any* meaningful word so long as there is but a single language which has but a single word devoid of any semantic content. Even ignoring this theoretical limitation, cross-situational learning suffers from other problems. It converges very slowly, requiring a larger corpus to converge than other methods such as the divide-and-conquer technique discussed in section 3. For example, while the divide-and-conquer technique is able to uniquely determine the word-to-meaning mappings for both the English and Japanese corpora from Siskind (1991, 1992), cross-situational learning terminates with:

```
(29 43 5 21 59 39 43 22 29 4 28)
```

meanings per word for the English corpus and with:

```
(59 21 22 29 43 3 3 5 43 4 39)
```

meanings per word for the Japanese corpus. Nonetheless, processing these corpora using the cross-situational technique takes only 3.45 and 3.75 CPU seconds respectively.[3] Thus we see a substantial

[3] Unless otherwise noted, all CPU times in this paper were measured on a four processor Sun SPARCserver™ 690MP

tradeoff between the complexity and completeness of both the divide-and-conquer and cross-situational techniques. In section 6 we further investigate the convergence properties of cross-situational learning experimentally.

5 The Mapping Problem as a Constraint Satisfaction Problem

We can view the mapping problem as a constraint satisfaction problem (CSP). A CSP is a finite set of variables $\{x_1, \ldots, x_W\}$, where each variable x_w ranges over a corresponding finite domain $D(w)$, along with a finite set of constraints $\{P_1, \ldots, P_n\}$, each applied to a subset of the variables (e.g. $P_1(x_3, x_5, x_9)$). A solution to a CSP consists of a map from variables to elements of their domains such that all of the constraints are satisfied. The mapping problem can be viewed as a CSP where the variables are the words appearing in the corpus, the domains are all of the submeanings of all of the terms appearing in the corpus, and each utterance in the corpus acts as a constraint between the words appearing in that utterance, constraining those words to take on meanings for which one of the hypothesized meanings associated with that utterance is a possible interpretation of that utterance. Figure 1 illustrates the CSP that corresponds to the English corpus from Siskind (1991, 1992).

Numerous techniques have been proposed for solving CSPs (cf. Mackworth 1992). One such technique is *node consistency*. If x_w is a variable that appears as an argument to some unary constraint P_i we say that the pair $\langle x_w, P_i \rangle$ is node consistent if $(\forall y \in D(w))P_i(y)$. An entire CSP is node consistent if all pairs $\langle x_w, P_i \rangle$ in the CSP are node consistent. A pair $\langle x_w, P_i \rangle$ can be made be made node consistent by removing from $D(w)$ those elements which prevent that pair from being node consistent.

$$D(w) \leftarrow D(w) \cap \{y | P_i(y)\} \tag{1}$$

It is sound to remove such elements as they could not appear in any solution. An entire CSP can be made node consistent by applying (1) once to each pair $\langle x_w, P_i \rangle$ in the CSP. The cross-situational learning procedure discussed in section 4 is equivalent to this node consistency technique where the unary constraints $P_i(x_w)$ are taken to be the constraint that the term x_w is a submeaning of one of the hypothesized meanings paired with utterance u_i.

Node consistency is a fast but incomplete technique for solving CSPs. This incompleteness explains—in part—the convergence problems of cross-situational learning. Slower, more powerful techniques are known for solving CSPs. One such technique is *arc consistency*.[4] If $x_{w_{ij}}$ is a variable that appears as the j-th argument to some constraint $P_i(x_{w_{i1}}, \ldots, x_{w_{i,m(i)}})$ of $m(i)$ arguments, we say that the pair $\langle x_{w_{ij}}, P_i \rangle$ is arc consistent if

$$\begin{aligned}(\forall y_j \in D(w_{ij}))(\exists y_1 \in D(w_{i1})) \cdots (\exists y_{j-1} \in D(w_{i,j-1})) \\ (\exists y_{j+1} \in D(w_{i,j+1})) \cdots (\exists y_{m(i)} \in D(w_{i,m(i)}))P_i(y_1, \ldots, y_{m(i)}).\end{aligned} \tag{2}$$

Likewise, an entire CSP is arc consistent if all pairs $\langle x_{w_{ij}}, P_i \rangle$ in the CSP are arc consistent. A pair $\langle x_{w_{ij}}, P_i \rangle$ can be made be made arc consistent by removing from $D(w_{ij})$ those elements which prevent that pair from being arc consistent.

$$\begin{aligned}D(w_{ij}) \leftarrow D(w_{ij}) \cap \{y_j | (\exists y_1 \in D(w_{i1})) \cdots (\exists y_{j-1} \in D(w_{i,j-1})) \\ (\exists y_{j+1} \in D(w_{i,j+1})) \cdots (\exists y_{m(i)} \in D(w_{i,m(i)}))P_i(y_1, \ldots, y_{m(i)})\}\end{aligned} \tag{3}$$

Again, it is sound to remove such elements as they could not appear in any solution. An entire CSP can be made arc consistent by applying (3) to each pair $\langle x_{w_{ij}}, P_i \rangle$ in the CSP and repeating this process until no domain is reduced. Unlike node consistency, making an entire CSP arc consistent may require multiple applications of (3) to a given variable-constraint pair.

In the context of the mapping problem, the key difference between node consistency and arc consistency is that node consistency views the words of an utterance in isolation while arc consistency views such words in the context of the other words in that utterance. For example, suppose that we knew that *John walked to school* meant GO(**John**, TO(**school**)) and we knew that *John* meant **John**. Node consistency would allow

model 140 with 512 megabytes of 80 nanosecond main memory running SunOS version 4.1.2. The algorithm was implemented in Lucid™ Common Lisp version 4.1 and compiled with using the production compiler with settings `speed 3`, `safety 0`, `space 0`, `debug 0`, and `compilation-speed 0`.

[4] Arc consistency is traditionally defined only for two-argument constraints. The definition given here is a straightforward generalization of the traditional definition.

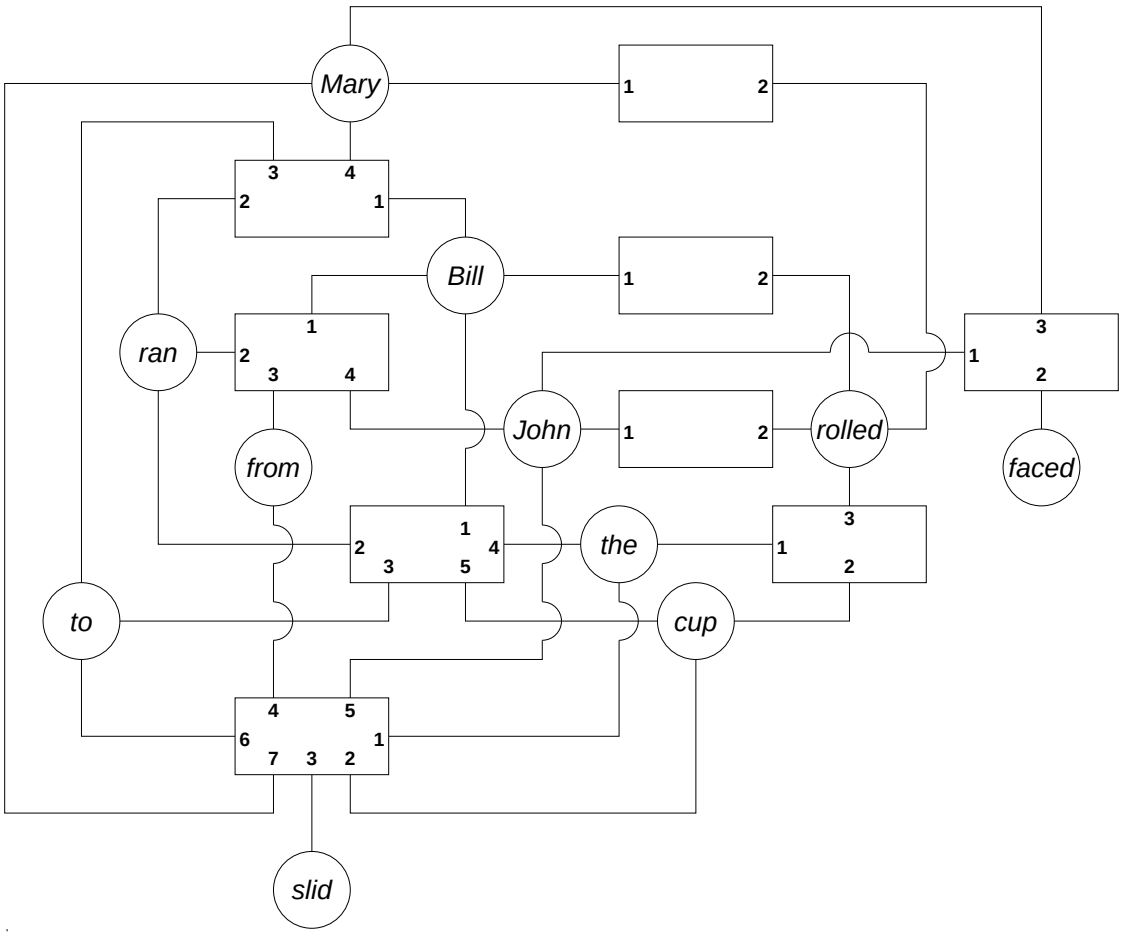


Figure 1: The English corpus from Siskind (1991, 1992) viewed as a constraint satisfaction problem. The circles depict the variables of the CSP while the rectangles depict the constraints. Each rectangle denotes an utterance and constitutes the constraint that a given set of word-to-meaning mappings allows that utterance to have one of the hypothesized meanings as a possible interpretation.

both **John** and GO(**John**, TO(**school**)) as potential meanings for *walked*, *to*, and *school* (along with other terms as well) since both **John** and GO(**John**, TO(**school**)) are submeanings of GO(**John**, TO(**school**)). But arc consistency would rule out such terms as potential meanings for each of the words *walked*, *to*, and *school* since there is no way to combine **John**—the known meaning for *John*—with the meanings of *walked*, *to*, and *school* to form GO(**John**, TO(**school**)) as the meaning of *John walked to school* if any of the words *walked*, *to*, or *school* took on **John** or GO(**John**, TO(**school**)) as their meaning.

Evaluating (3) to making a pair $\langle x_{w_{ij}}, P_i \rangle$ arc consistent requires repeated evaluation of the formula $P_i(y_1, \ldots, y_{m(i)})$. Recall that in the context of the mapping problem, the formula $P_i(y_1, \ldots, y_{m(i)})$ is true if and only if one of the hypothesized meanings $t_{i1}, \ldots, t_{i,l(i)}$ associated with utterance u_i is a possible interpretation of u_i taking $y_1, \ldots, y_{m(i)}$ as the meanings of the words $w_{i1} \cdots w_{i,m(i)}$ comprising u_i. This may appear to be intractable since determining whether a term is a possible interpretation of an utterance—according to the definition given in section 2—requires existentially quantifying over all binary branching interpretation trees for that utterance. Furthermore, applying (3) to determine which y's should be removed from $D(w_{ij})$ to make a pair $\langle x_{w_{ij}}, P_i \rangle$ arc consistent will require evaluating $P_i(y_1, \ldots, y_{m(i)})$ for every tuple $\langle y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_{m(i)} \rangle$ in the cross product $D(w_{i1}) \times \cdots \times D(w_{i,j-1}) \times D(w_{i,j+1}) \times \cdots \times D(w_{i,m(i)})$. On the surface, this would appear to require time exponential in the length of the utterance.

Both of these sources of intractability can be removed by using a variant of the CKY algorithm (Kasami 1965, Younger 1967). To determine whether for a given y_j the formula

$$(\exists y_1 \in D(w_{i1})) \cdots (\exists y_{j-1} \in D(w_{i,j-1}))(\exists y_{j+1} \in D(w_{i,j+1})) \cdots (\exists y_{m(i)} \in D(w_{i,m(i)}))P_i(y_1, \ldots, y_{m(i)}) \quad (4)$$

is true, one first applies the following recurrence relation to build a chart M, (an $m(i)+1$ by $m(i)+1$ matrix):

$$M_{j,j+1} = \{y_j\} \quad (5)$$

$$M_{j_1,j_1+1} = D(w_{ij_1}) \ \forall j_1 \neq j \quad (6)$$

$$M_{j_1 j_2} = \left(\bigcap_{k=1}^{l(i)} \textsc{Unlink}^*(t_{ik}) \right) \cap \left(\bigcup_{j_3=j_1+1}^{j_2-1} \bigcup_{s \in M_{j_1 j_3}} \bigcup_{t \in M_{j_3 j_2}} (\textsc{Link}(s,t) \cup \textsc{Link}(t,s)) \right) \quad (7)$$

The entry M_{j_1,j_2+1} contains all possible interpretations for the utterance fragment $w_{j_1} \cdots w_{j_2}$, while the entry $M_{1,m(i)+1}$ contains all possible interpretations for the whole utterance. Formula (4) then is true if and only if $M_{1,m(i)+1} \cap \bigcup_{k=1}^{l(i)} t_{ik}$ is not empty. Note that since we are concerned only with interpretations that are submeanings of one of the hypothesized meanings associated with the u_i, at each step of the recursion we intersect M_{ij} with $\bigcap_{k=1}^{l(i)} \textsc{Unlink}^*(t_{ik})$, the set of all submeanings of the hypothesized meanings $t_{i1}, \ldots, t_{i,l(i)}$ associated with u_i. This puts an upper bound on the cardinality of the sets contained in each entry of the chart. Given this bound, making a pair $\langle x_{w_{ij}}, P_i \rangle$ arc consistent can be done in time cubic in the length $m(i)$ of the utterance u_i.

There is a further optimization one can perform. Nominally, to make a pair $\langle x_{w_{ij}}, P_i \rangle$ arc consistent one must evaluate (4) for each $y \in D(w_{ij})$. Furthermore, if P_i is applied to the variables $x_{w_{i1}}, \ldots, x_{w_{i,m(i)}}$ then each of the pairs $\langle x_{w_{i1}}, P_i \rangle, \ldots, \langle x_{w_{i,m(i)}}, P_i \rangle$ must be made arc consistent to make the whole CSP arc consistent. All of the ensuant evaluations of (4) can share much of the chart M constructed from prior evaluations.

As when applying node consistency, the size of each meaning domain decreases monotonically when applying arc consistency. Arc consistency, like node consistency, is a sound but incomplete technique for solving constraint satisfaction problems. If any meaning domain ever became empty when applying arc consistency, the instance of the mapping problem would have no solution. Likewise, if arc consistency reduces all meaning domains to singleton sets then the elements of those singleton sets constitute a unique solution to the instance of the mapping problem. If however, arc consistency terminates without reducing all of the meaning domains to singletons, then uncertainty remains as to which—if any—of the meanings for a given word is correct. Each element in the cross-product of the meaning domains constitutes a potential solution to the instance of the mapping problem. Each potential solution must be checked for global consistency across the entire corpus leading to uncertainty as to whether zero, one, or several of these potential solutions is an actual solution.

The incompleteness of arc consistency can be mitigated by applying the divide-and-conquer technique as a post-processing step after the completion of arc consistency. In this case, the information gained by

arc consistency can be used to speed up the divide-and-conquer technique. The CKY algorithm is used to precompute the set of all possible meanings for each subsequence of words in each utterance in the corpus given the meaning domains produced by arc consistency. At each recursive step of the divide-and-conquer technique, the recursion simply fails if a phrase is paired with term which cannot be a possible interpretation of that phrase. Such filtering dramatically improves the performance of the divide-and-conquer technique. While the incompleteness of arc consistency is a theoretical concern, it is not a problem in practise with any but the smallest of corpora. While the post-processing step is needed to process the English and Japanese corpora from Siskind (1991, 1992), few of the larger random corpora we have generated have required divide-and-conquer post-processing, and then only for a small number of words.

Both node and arc consistency are techniques which apply on an utterance by utterance basis. While (1) need only be applied once to each utterance to make a CSP node consistent, (3) may need to be applied multiple times to an utterance to make a CSP arc consistent. The standard way to do this would be to maintain a queue of utterances. This queue would be initialized to contain the entire corpus. Rule (3) would be applied to utterances as they are removed from the queue. When processing an utterance, any other utterance not already on the queue that contains a word whose meaning domain is reduced would be placed back on the queue for further processing. Processing would continue until the queue is empty.

While the above method is particularly efficient, we adopt a much simpler strategy. The corpus is processed repeatedly, applying (3) to each utterance in order. While in theory, this may apply (3) unnecessarily to some utterances, in practise we find that for practically all corpora of sufficient size, arc consistency converges to singleton meaning domains for all words in the corpus with a single application of (3) to each utterance. If we limit ourselves in this fashion to a single pass of arc consistency, the resulting lexical acquisition algorithm is *on-line* in the sense that neither an utterance nor its potential meanings need be remembered after processing. The meaning domains constitute the only long term memory required between utterances.

Arc consistency strictly dominates node consistency. In other words, any element y of the meaning domain $D(w_{ij})$ which would be removed by applying node consistency (i.e. rule (1)) to utterance u_i would also be removed by applying arc consistency (i.e. rule (3)) to the same utterance. In theory, one need never apply node consistency if one is also applying arc consistency. In practise, however, arc consistency can be very slow if applied to an utterance that contains many words with large meaning domains. This is likely to occur when processing the first utterances in the corpus. Node consistency does not suffer from this limitation and can quickly process an utterance independent of the size of the meaning domains of its constituent words. To alleviate this problem, we adopt a two-pass strategy whereby we first process the entire corpus using node consistency—to quickly reduce the size of all meaning domains—and then process the corpus a second time using arc consistency. While this two-stage process is formally equivalent to performing arc consistency alone, it is much faster.

The above two-stage process is at odds with the attempt to formulate an on-line learning strategy. One possible on-line variant of the two-stage strategy would be to process each utterance only once and choose whether to apply node consistency or arc consistency depending on the sizes of the meaning domains of the words in the utterance. While we have not tried this approach, it appears feasible to construct a fast on-line learning strategy with only a slight increase in the size of the corpus needed for convergence when compared to any complete strategy for solving the mapping problem such as the divide-and-conquer technique.

We have processed both the English and Japanese corpora from Siskind (1991, 1992) using the arc consistency technique. More specifically, we perform a first pass of node consistency, followed by as many complete passes of arc consistency as needed for convergence, followed by a post-processing application of the divide-and-conquer technique. This combined technique yields the same results as reported by Siskind, but requires a total of only 136 seconds of CPU time for the English corpus and 91 seconds of CPU time for the Japanese corpus. When processing the English corpus, node consistency produces

```
(29 43 5 21 59 39 43 22 29 4 28)
```

meanings per word. After the first pass of arc consistency there are

```
(2 7 1 1 15 5 1 1 2 1 4)
```

meanings per word. After the second pass of arc consistency there are

```
(2 4 1 1 14 4 1 1 2 1 4)
```

$$
\begin{aligned}
\text{NP} &\rightarrow \text{N}_{\text{PROPER}} \\
\text{NP} &\rightarrow \text{N}_{\text{SPEC}}\ \text{N} \\
\text{PP} &\rightarrow \text{P NP} \\
\text{S} &\rightarrow \text{NP V COMPLEMENTS} \\
\text{COMPLEMENTS} &\rightarrow \\
\text{COMPLEMENTS} &\rightarrow \text{NP} \\
\text{COMPLEMENTS} &\rightarrow \text{PP} \\
\text{COMPLEMENTS} &\rightarrow \text{PP PP}
\end{aligned}
$$

Figure 2: A small grammar used to generate random English-like corpora.

meanings per word. A third pass of arc consistency yields no reduction in the number of meanings per word so arc consistency terminates. The divide-and-conquer post-processing step produces, of course, one meaning per word.

6 Experiments

Ideally, one would test the effectiveness of the arc consistency technique on large, naturally produced corpora. Unfortunately, this is not feasible since our lexical acquisition model requires that utterances be paired with sets of hypothesized meanings produced from the non-linguistic context of the utterance and we know of no corpora containing such annotations. As an alternative testing methodology, we have constructed large randomly generated corpora of English-like and Japanese-like text and processed these corpora with our arc consistency technique. The English utterances are produced by the grammar shown in figure 2 while the Japanese utterance are produced by the grammar shown in figure 3. The English grammar has the terminal categories N_{SPEC}, N, N_{PROPER}, P, and V. Since this language is artificial—and only intended to resemble English—the words in this language are taken to be *N1*, *N2*, ..., *V1*, *V2*, ... and the like. The Japanese grammar assigns words to its terminal categories, namely N_{SPEC}, N, N_{PROPER}, P, V_{SPEC} and V by a similar process. The desired number of words in each category is a parameter of the corpus generator though we typically specify a desired total vocabulary size independently from the percentage of words in each category. For the experiments described in this paper, we stipulate that 0.4% of the words in the lexicon are of category N_{SPEC}, 24% are common nouns, 24% are proper nouns, 2% are prepositions, and 49.6% are verbs. For Japanese we stipulate that there are two words of category V_{SPEC}. Verbs fall into one of four subcategorization classes, namely intransitive, transitive, those that take a single PP complement, and those that take two PP complements. We can independently specify the relative size of each class, though in this paper we take them all to be of equal size.

A random utterance of category S is generated by applying the rules of the grammar with equal probability except that the choice of which rule to apply for COMPLEMENTS depends on the subcategorization class of the chosen verb. Lexical items are chosen according to a specified probability distribution. The first experiment we describe uses a uniform distribution.

Along with the utterance, we generate a term to denote its meaning. A proper or common noun is taken to be a constant symbol denoting the meaning of a noun phrase containing that noun. A preposition is taken to be a one argument function symbol applied to its NP complement to form the meaning of a prepositional phrase. A verb is taken to be a function symbol applied to its subject and complements to form the meaning of a sentence. Such sentential function symbols take one argument, in the case of intransitive verbs, or two arguments, in other cases. The subject becomes the first argument of the sentential function symbol. If the verb has a single complement, it becomes the second argument. If the verb has two arguments x and y, they must be distinct, and the second argument is taken to be $[\text{Path}\ x, y]$. To generate a referentially uncertain pairing of an utterance with several terms we generate multiple utterance-meaning pairs and

$$
\begin{aligned}
\text{NP} &\rightarrow \text{N}_{\text{PROPER}} \\
\text{NP} &\rightarrow \text{N N}_{\text{SPEC}} \\
\text{PP} &\rightarrow \text{NP P} \\
\text{S} &\rightarrow \text{NP V}_{\text{SPEC}}\ \text{COMPLEMENTS V} \\
\text{COMPLEMENTS} &\rightarrow \\
\text{COMPLEMENTS} &\rightarrow \text{NP} \\
\text{COMPLEMENTS} &\rightarrow \text{PP} \\
\text{COMPLEMENTS} &\rightarrow \text{PP PP}
\end{aligned}
$$

Figure 3: A small grammar used to generate random Japanese-like corpora.

throw away all but one of the utterances, pairing that utterance with all of the generated meanings.[5] Thus, the English-like corpora resemble the following:

```
(((NSPEC0 N16 V35 P0 N-PROPER8 P1 NSPEC0 N15)
  ((V35 N16 (PATH (P0 N-PROPER8) (P1 N15)))
   (V38 N-PROPER6 (PATH (P0 N-PROPER4) (P1 N-PROPER2)))
   (V34 N-PROPER2 (PATH (P0 N-PROPER20) (P0 N-PROPER4)))
   (V3 N2 (PATH))))
 ((N-PROPER7 V2)
  ((V2 N-PROPER7 (PATH))
   (V13 N14 (P1 N3))
   (V40 N-PROPER20 N-PROPER1)
   (V41 N-PROPER23 N-PROPER15)))
 ⋮)
```

while the Japanese-like corpora resemble the following:

```
(((NSPEC0 N10 VSPEC1 NSPEC0 N1 V43)
  ((V43 N10 N1)
   (V45 N-PROPER21 N11)
   (V27 N-PROPER7 (PATH (P0 N-PROPER14) (P0 N-PROPER22)))
   (V5 N-PROPER7 (PATH))))
 ((N-PROPER11 VSPEC0 N-PROPER19 P0 V14)
  ((V14 N-PROPER11 (P0 N-PROPER19))
   (V50 N19 N15)
   (V36 N9 (PATH (P1 N-PROPER6) (P1 N22)))
   (V24 N23 (P1 N-PROPER10))))
 ⋮).
```

Note that while the same tokens are used to represent both words and their meanings, the lexical acquisition algorithm has no knowledge of this fact as it never compares the equality of a lexical token with a meaning token. This artifact of our process for generating random corpora, however, affords us a convenient way for quickly determining whether our learning algorithm has converged to the correct answer without human intervention, allowing us to test this algorithm on numerous large corpora.

We have generated three English-like and three Japanese-like corpora of increasing size. The statistics of these corpora are summarized in figure 4. We processed each of these corpora using a first pass of node consistency, followed by as many passes of arc consistency as necessary for convergence. Since node consistency will never rule out $\perp$ as a possible meaning for a word, figure 4 reports the number of words

[5] Thus the unintended meanings are uncorrelated with the intended one. A more realistic model would be for there to be some correlation between the intended and unintended meanings.

	English-like			Japanese-like		
Number of utterances	200	2000	20000	200	2000	20000
Number of words in lexicon	102	992	9922	105	995	9911
Number of words in corpus	955	9489	95094	1122	11487	115018
Minimum utterance length	2	2	2	3	3	3
Maximum utterance length	9	9	9	10	10	10
Average utterance length	4.78	4.74	4.75	5.61	5.74	5.75
Referential uncertainty	4	4	4	4	4	4
Minimum occurrences per word	1	1	1	1	1	1
Maximum occurrences per word	193	523	543	199	1009	10022
Average occurrences per word	9.36	9.57	9.58	10.69	11.54	11.61
Ambiguous words after node consistency	64	389	3938	61	405	3878
Ambiguous words after one pass of arc consistency	0	4	51	0	9	48
Ambiguous words after two passes of arc consistency		0	0		0	0
Total processing time (CPU seconds)	293	2481	26070	238	2621	27784

Figure 4: Statistics of three English-like and three Japanese-like corpora generated with uniform lexical selection using the grammars of figures 2 and 3.

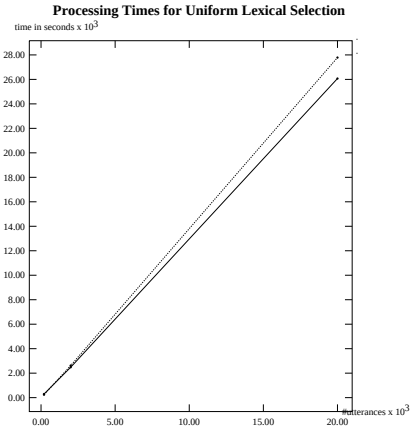


Figure 5: CPU Times for processing various English-like and Japanese-like corpora generated with uniform lexical selection.

that remain ambiguous even after ignoring $\perp$. No corpus exhibited residual ambiguity after arc consistency so no divide-and-conquer post-processing steps were needed. The correct word-to-meaning mappings were discovered in all cases. No corpus required more than two passes of arc consistency, and in fact, arc consistency always converged to unique meanings for at least 99% of the words in each corpus after only a single pass. In figure 5 we have plotted the processing times for each of these corpora. Notice that the processing time scales linearly with the corpus size.

Mitch Marcus (p.c.) has suggested that lexical acquisition is difficult because words appear with roughly a Zipf's law distribution where their frequency is inversely proportional to their rank causing numerous words to appear very infrequently. To test the sensitivity of our learning strategy to such situations we ran a second series of experiments processing three English-like and three Japanese-like corpora of increasing size, generated using a Zipf's law lexical selection criteria. The results of this experiment are shown in figure 6. While arc consistency no longer converges to a single word-to-meaning mapping for all words in the corpora, typically very few words remain ambiguous after arc consistency has run to completion though a greater number of words remain ambiguous after only a single pass of arc consistency.[6] Typically,

[6] We did not perform a divide-and-conquer post-processing step after arc consistency as this proved intractable.

	English-like			Japanese-like		
Number of utterances	200	2000	20000	200	2000	20000
Number of words in lexicon	87	810	7587	96	830	7450
Number of words in corpus	848	9244	103242	1107	11884	107699
Minimum utterance length	2	2	2	3	3	3
Maximum utterance length	9	9	9	10	10	10
Average utterance length	4.24	4.62	5.16	5.54	5.94	5.38
Referential uncertainty	4	4	4	4	4	4
Minimum occurrences per word	1	1	1	1	1	1
Maximum occurrences per word	170	931	5094	185	1379	13371
Average occurrences per word	9.75	11.41	13.61	11.53	14.32	14.46
Ambiguous words after node consistency	69	531	5742	79	557	5188
Ambiguous words after one pass of arc consistency	3	42	339	5	33	413
Ambiguous words after two passes of arc consistency	2	19	164	1	11	186
Total processing time (CPU seconds)	306	2986	51636	345	3875	32377

Figure 6: Statistics of three English-like and three Japanese-like corpora generated with Zipf's law lexical selection using the grammars of figures 2 and 3.

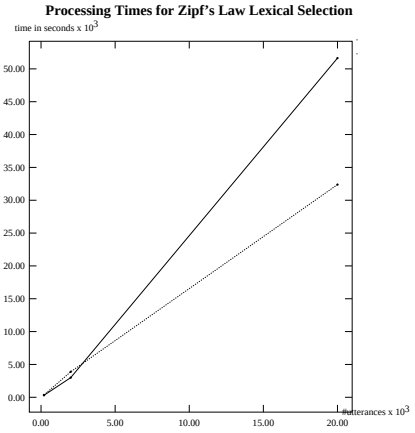


Figure 7: CPU Times for processing various English-like and Japanese-like corpora generated with Zipf's law lexical selection.

however, less than 5% of the lexicon remains unlearned. Furthermore, as figure 7 shows, the processing time still scales well—though not quite linearly—in the size of the corpus.

The full paper will contain a series of experiments to show that arc consistency can process corpora with higher degrees of referential uncertainty. These experiments have been performed but I wish to redo them in a controlled fashion before publishing the data.

7 Conclusion

Two metrics can be used to measure the performance of a learning algorithm. The first is how much data is needed to learn some information while the second is the processing time needed to learn that information given the data. It is well known within the machine learning community that often an algorithm that requires more data to learn some information may require less processing time to learn the same information than a more sophisticated algorithm which processes less data. We see this here with arc consistency requiring more data for convergence than the divide-and-conquer and conquer technique but requiring far less processing time. In this paper we have demonstrated an effective technique for learning a lexicon of

word-to-meaning mappings by processing a corpus of utterances paired with a referentially uncertain set of hypothesized meanings. This technique can successfully learn the meanings of as many as 10,000 words by processing 20,000 utterances where individual words appear—on the average—ten times in the corpus and often as infrequently as once. It is often said that children learn an average of ten new words a day. If they achieve basic fluency within the first three years of their life, they will have learned roughly 10,000 words. During this time they will have heard at least 20,000 utterances. Thus the size of the problems we can handle are within the range of those faced by children. The processing requirements are well within the capabilities of current computer hardware and thus likely to be within the capacity of the human brain. While we make no claim that children actually employ the techniques we described in this paper, they could—in principle—do so.

These results have important consequences for child language acquisition research. Some researchers (cf. Gleitman 1990, Fisher et al. 1991) have argued that children use syntactic information to acquire word meanings and derive such information from prosodic cues in the input. Two sources of evidence are used to support this claim. One is the assumed difficulty or impossibility, voiced by Quine (1960), of learning word meanings from contextual evidence alone. The other is experimental evidence that shows that children are sensitive to syntactic and prosodic information in the input signal. The fact that children are sensitive to such information is interpreted as evidence that they use such information to acquire word meanings in light of the claim that they could not be doing it any other way. The results in this paper substantially weaken the first source of evidence and demonstrate that they could be learning word meanings without syntactic or prosodic information. While it still might be the case that children actually do use syntax and prosody, support for this argument is weakened. This should motivate researchers both to seek additional supporting evidence of this fact, as well as evidence for alternate approaches.

On the engineering side, the results in this paper might also find application in natural language processing. NLP systems such as machine translators require large lexica mapping words to useful representations of their meanings. If some corpus can be found which has utterances annotated with their meanings, the techniques described in this paper can be used to derive a lexicon from that corpus.

Several outstanding problems remain with the techniques described in this paper. First, while arc consistency scales linearly in the number of utterances in the corpus, cubicly in the utterance length, and linearly in the degree of referential uncertainty, it scales exponentially in the size of the terms used to represent hypothesized meanings. This is because UNLINK$^*(r)$ will produce a set whose cardinality is exponential in the size of r. Thus arc consistency cannot be used when utterance meanings are represented by large terms. Second, arc consistency cannot learn a polysemous lexicon. It will fail and report inconsistency when a corpus contains polysemous words. Third, while arc consistency makes very little use of syntax, it does rely in a limited fashion on word order. In particular, it cannot process utterances such as *Who did John give the ball to?* which contain two discontinuous words (i.e. *Who* and *to*) which link to form a single semantic entity such as TO(**who**). Arc consistency will fail when presented with a corpus containing such utterances. Note that this does not rule out all forms of movement. Utterances such as *What did John give to Mary?* can be handled correctly, since they don't involve a discontinuous semantic entity. Finally, arc consistency will fail if presented with a noisy corpus containing utterances paired with a sets of hypothesized meanings missing the correct meaning. Current work in progress is addressing all of these issues and will be reported in a future paper.

References

[1] Jerome Bruner. *Child's Talk.* W. W. Norton & Co., New York, 1983.

[2] Cynthia Fisher, Geoffrey Hall, Susan Rakowitz, and Lila Gleitman. When it is better to receive than to give: syntactic and conceptual constraints on vocabulary growth. Unpublished manuscript received directly from author, 1991.

[3] Lila Gleitman. The structural sources of verb meanings. *Language Acquisition*, 1(1):3–55, 1990.

[4] Gérard Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[5] Ray Jackendoff. *Semantics and Cognition.* M. I. T. Press, Cambridge, MA, 1983.

[6] Ray Jackendoff. *Semantic Structures.* M. I. T. Press, Cambridge, MA, 1990.

[7] T. Kasami. An efficient recognition and syntax algorithm for context-free languages. Scientific Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford MA, 1965.

[8] John Locke. *An essay concerning human understanding.* Meridian Books, Cleveland, 1964. (Original work published 1690).

[9] Alan K. Mackworth. Constraint satisfaction. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 285–293. John Wiley & Sons, Inc., New York, 1992.

[10] Brian MacWhinney and Catherine Snow. The child language data exchange system. *Journal of Child Language*, 12, 1985.

[11] Mitch Marcus. Personal communication.

[12] Steven Pinker. *Learnability and Cognition.* M. I. T. Press, Cambridge, MA, 1989.

[13] W. V. O. Quine. *Word and object.* M. I. T. Press, Cambridge, MA, 1960.

[14] Jeffrey Mark Siskind. Acquiring core meanings of words, represented as Jackendoff-style conceptual structures, from correlated streams of linguistic and non-linguistic input. In *Proceedings of the* 28^{th} *Annual Meeting of the Association for Computational Linguistics*, pages 143–156, University of Pittsburgh, Pittsburgh, PA, June 1990.

[15] Jeffrey Mark Siskind. Dispelling myths about language bootstrapping. In *The AAAI Spring Symposium Workshop on Machine Learning of Natural Language and Ontology*, pages 157–164, March 1991.

[16] Jeffrey Mark Siskind. *Naive Physics, Event Perception, Lexical Semantics, and Language Acquisition.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, January 1992.

[17] D. H. Younger. Recognition and parsing of context-free languages in time $O(n^3)$. *Information and Control*, 10(2):189–208, 1967.