



Lexical Acquisition Made by Machine

A simulation of how a machine learns the meaning of words

Jesper Alvelid
Svandammsvägen 43
126 35 Hägersten
0762117797
jalvelid@kth.se

Fredrik Frantzén
Hampvägen 46
178 35 Ekerö
0737545165
ffra@kth.se

Degree Project in Computer Science, First Level, DD143X
CSC, Royal Institute of Technology
Supervisor: Michael Minock
Examiner: Mårten Björkman

2012-04-12

Abstract

Learning the meaning of words is a complicated task with many problems. In this study an algorithm to map words to meanings was developed regarding the three problems: handling of sentences (not only singular words), distinguishing the correct of multiple events in a scene and building a lexicon with no entries at the beginning. The aim of this study was to implement an algorithm that would replicate the results of a previous study. The results acquired confirmed the work previously done, the same percentage of word meanings (100%) were learned with equal conditions. To further develop the algorithm problems with words that are spelled identically but mean different things and contexts where events are not describing the utterances said need to be solved. This would make the algorithm more applicable in real world situations.

Sammanfattning

Att lära sig betydelsen av ord är en väldigt komplicerad uppgift med många problem som behöver lösas. I denna studie utvecklades en algoritm som parar ihop ord med betydelser med avseende på de tre problemen: att kunna hantera meningar (inte bara enstaka ord), att kunna välja ut den rätta händelsen i ett sammanhang samt att kunna lära sig ord utan att tidigare ha kännedom om några ord. Målet med studien var att implementera en algoritm som skulle kunna replikera resultaten i en tidigare rapport på ämnet. De erhållna resultaten fastställde de i ett tidigare arbete, samma andel av betydelser av ord (100%) lärdes in under samma förhållanden. För att ytterligare utveckla algoritmen måste två ytterligare problem lösas: ord som stavas likadant men har olika betydelser och sammanhang där händelserna inte beskriver vad som sades i sammanhanget. Detta skulle göra algoritmen mer användbar i tillämpningar inom ämnet.

Contents

1	Preliminaries	1
1.1	Word definitions	1
1.2	Sentences and symbols	1
1.3	Statement of collaboration	1
2	Introduction	3
2.1	Language learning	3
2.2	Purpose	4
2.3	Overview of the report	4
3	Background	7
3.1	Problems with lexical acquisition	7
3.2	Siskind's results	8
3.3	Principles of exclusion	8
4	Approach	11
5	Implementation	13
5.1	Structure of input	13
5.2	Utterance generator	14
5.3	The algorithm	16
5.4	Possible and necessary symbols	17
5.5	Principles	18
5.5.1	Principle 1: Hypothesis coinciding with known words	18
5.5.2	Principle 2: Cross-sentence learning	18
5.5.3	Principle 3: Meaning only possible for one word	19
5.5.4	Principle 4: Meaning of words not overlapping	20
5.6	Output from the algorithm	20
6	Results	23
6.1	Tests	23
6.2	Tests without corruption	25
7	Discussion	27

7.1	Problems with the implementation	27
7.2	Homonymy and noise	28
7.3	Implementing other senses	28
7.4	Applications	28
8	Conclusion	31
9	Bibliography	33

Chapter 1

Preliminaries

1.1 Word definitions

Bootstrapping: To create something which from the beginning is empty.

Corpus size: The number of utterances in input.

False negative: A word where the correct meaning is falsely excluded.

False positive: A word where a false meaning is concluded to be the correct one.

Homonymy: A word which has several diverging meanings.

Hypothesis: A grouping of symbols into a possible meaning for a whole sentence.

Lexical acquisition: The process of learning the meaning of a word.

Necessary symbol: Symbols that are partly-known to be a meaning of a word, every word has its own list of necessary symbols.

Noise: Symbols that describe what is happening in a scene, but not what is explained by the corresponding sentence.

Possible symbol: Symbols that could be the meaning of a word, every word has its own list of possible symbols.

Referential uncertainty: The amount of events occurring (hypotheses) whenever an utterance is heard.

Symbol: Describes the meaning of a word.

1.2 Sentences and symbols

Throughout this text many examples will be given of sentences run through the algorithm and symbols representing the meanings of the words. To make it easy for the reader natural language sentences will always be seen in *italics* and symbols will always be seen inside curly brackets { }.

1.3 Statement of collaboration

In this study both code writing and report writing has been done as a collaborative work meaning that both authors have written in every section and the coding has

CHAPTER 1. PRELIMINARIES

been done together.

Chapter 2

Introduction

Most people learn their first complex language in the early stages of their life. Their influences are people around them like their parents, grandparents and other people related to their family. Some even might learn multiple languages if they have multilingual influences. Very small children are not able to learn languages the same way adults do when abroad or students do in school, because older people have already learned a first language. A child cannot read and has no experiences from other languages when they learn their first language. So how do children learn without experience?

2.1 Language learning

One theory is that a child pairs what was said with what it feels in that instance. This would mean that when a child sees a ball bouncing on the floor and hears her mother say "The ball is bouncing!" the child assumes what was said is related to anything of what she memorized at that moment. The child was maybe not looking at the ball, but instead saw dad walk in from the kitchen. Not only sight but hearing a fly buzz or feeling the cold wooden floor could be the child's impression of the situation. The human body is a complex machine with many different methods of perception, so there is no doubt that pairing words with the world is a complex problem.

Another interesting similarity is when a grown-up travels abroad. The unavoidable task is to learn the language of the visited country. Although most people would have had an introduction to the new language beforehand and might be actively studying it, much of the active learning will be from the exposure of conversations with other people. Some languages, like Spanish and Italian, are highly related and will therefore be learned faster by many people, and some are that much different it is hard to recognize any words at all. Although, if a child is able to learn a language only by exposure of conversations then so should also adults unless this ability is lost on the way to adulthood.

2.2 Purpose

Jeffrey Mark Siskind (Ph.D. in Computer Science) built a system and developed an algorithm [1] that tests the learning acquisition task young children are faced with. The system is very slimmed down from factors of the process as it only covers the sensory ability sight and not for example touch or taste. This paper is in essence a replication of the problem Siskind treated.

In this study an exploration of what possibilities there are for machine learning of the meaning of words, simply called lexical acquisition, was conducted. The aim was to implement an algorithm which given a natural language sentence and a list of events occurring (hypotheses with symbols such as {GO}), could decide the meaning of each word in the sentence, i.e. which symbols should be linked to each word.

The main purpose of the thesis was to see if a replication of Siskind's results could be acquired.

2.3 Overview of the report

This report begins with a presentation of the different problems that need to be solved in order to receive a mapping of meanings to words as well as how the algorithm developed [1] works which was closely followed in this study.

As the aim of this study was to replicate Siskind's results, a system that works similarly had to be developed. Some interesting factors, though not acknowledged in this study's implementation, will be presented here to later be discussed in the discussion section.

In the approach section it is described what was needed to take into account in order to implement the system and the algorithm. Additionally, it shortly shows how the algorithm will be tested and how the natural language sentences will be created.

In the implementation section, it is explained in detail how the principles of learning the meaning of words were implemented, which are the parts of the algorithm, and how they work on an example. Also in this section it is shown how the generator constructs the sentences along with their hypotheses, which become the input to the algorithm, how the input is structured and how the algorithm handles the input until the words meanings are concluded.

The result section shows how effective the method used was and if the algorithm actually learned the correct meaning of the words, and to which extent. The initial tests show whether the implementation is able to acquire a lexicon and the results of another test show what difference it makes to exclude some problematic words.

The discussion section mainly focuses on factors that are of importance in the real world when learning languages and how these factors could be integrated in the algorithm to make it more efficient. Homonymy and noise is also to a great extent discussed here and what difference it would make to implement it.

2.3. OVERVIEW OF THE REPORT

Finally a comparison between the results acquired and Siskind's is made to conclude any similarities and the conclusions which can be drawn from the results and the discussion is pointed out.

Chapter 3

Background

3.1 Problems with lexical acquisition

The implementation in this study uses a stripped down approach of the lexical acquisition task a normal child encounters, though to learn the meaning of words is still a complicated task. To parse the meaning of a sentence, and break it down into the meaning of individual words from just a scene is quite difficult. Because given a sentence there is not an easy way to figure out what the meaning of a single word is. Any word in a sentence is a candidate of describing anything in the particular context, particularly at the beginning when not a single word is known.

Thereby, for lexical acquisition to work on a machine, specification of how to map meanings to words and of which problems exist is of high importance. The different critical problems needed to be acknowledged and solved by the algorithm are:

- Handling of sentences
- Distinguish the correct event of multiple events in a scene (referential uncertainty)
- Build a lexicon from having an empty lexicon (bootstrapping)

Siskind developed two algorithms in his study. The first solves the three problems described above and the aim of this study was to implement an algorithm handling these problems.

However, Siskind also constructed another algorithm which handles two additional problems which makes the lexical acquisition task even more complicated, these are:

- Ignore instances where the events in the scene are not related to what is being said (noise)
- Handle words with multiple meanings (homonymy)

3.2 Siskind's results

Siskind found that there was a correlation between the amount of unique words used in the simulation and the corpus size: the more unique words in the simulation, the larger corpus was needed to build a lexicon where 95% of the words had converged, but the corpus size grew linearly to the amount of unique words. Increasing the number of events in the scene (referential uncertainty) or the number of objects in the scene did not affect the corpus size significantly.

When excluding noise and homonymy from the test the number of false positives, e.g. words to which the algorithm concludes a false meaning, was always zero, which means it learned all words correctly. As the noise rate or homonymy occurrences increased, the number of false positives grew and an exponentially larger corpus was needed.

3.3 Principles of exclusion

Applying the principles of learning the meaning of words is to a large extent a case of excluding possible meanings. Because when introduced to a new word it can have so many possible meanings, in fact all possible meanings for the sentence the word occurred in are possible meanings for the word itself. To conclude the correct meaning of the word the only way to go is to accordingly exclude meanings which cannot be the explanation of the word.

Solving of the lexical acquisition problem is done by specifying and following principles of which the implementation in this study has four. Hereafter follows explanations of them, including them written in logical expressions.

The symbols used, which will be thoroughly explained in the implementation chapter, are the following: s = word, S = sentence, m = main hypothesis constructed from the sentence, M = all hypotheses to a sentence, $P(s)$ = possible symbols list for s , $N(s)$ = necessary symbols list for s , $F(m)$ = set of word symbols in m , $F1(m)$ = all symbols existing only once in m

- **Hypotheses coinciding with known words** - The first principle (by Siskind called *constraining hypotheses with partial knowledge* [1]): discard every hypothesis to an utterance where one or more words are partly known but do not coincide with the hypothesis.

$$M \leftarrow \{m \in M \mid \cup_{s \in S} N(s) \subseteq F(m) \wedge F(m) \subseteq \cup_{s \in S} P(s)\}$$

- **Cross-sentence learning** - The second principle (by Siskind called *cross-situational interference* [1]) describes that by comparing possible meanings of a word that exists in multiple sentences, often only one meaning is possible for the word in all sentences and must therefore be the correct one. This is probably the most important principle as it quite easily can determine the correct meaning of a word with just a few sentences.

$$\text{for } s \in S \text{ do } P(s) \leftarrow P(s) \cap \cup_{m \in M} F(m)$$

3.3. PRINCIPLES OF EXCLUSION

- **Meaning only possible for one word** - The third principle (by Siskind called *covering constraints* [1]) states that if a symbol can be excluded as a possible symbol for all but one word in a sentence, it must be the meaning, or part, of that word.

$$\text{for } s \in SdoN(s) \leftarrow N(s) \cup [(\cap_{m \in M} F(m)) \setminus \cup_{s' \in S, s \neq s'} P(s')]$$

- **Meaning of words not overlapping** - The last principle (by Siskind called *the principle of exclusivity* [1]) works in a sense similar to the third principle. When a symbol is known to be part of the meaning of a word, it can be excluded as the meaning of all other words in the sentence to prevent overlapping.

$$\text{for } s \in SdoP(s) \leftarrow P(s) \setminus [(\cap_{m \in M} F_1(m)) \cap \cup_{s' \in S, s \neq s'} N(s')]$$

Chapter 4

Approach

The goal of this study was to replicate Siskind's results, therefore a program with an algorithm similar to his for mapping meanings to words needed to be developed. In addition, an utterance generator that maps utterances to combinations of meanings, hypotheses, was needed. The algorithm includes the first four of Siskind's rules [1], which were reformatted into the principles described in section 3.3.

The principles used in the algorithm that needed to be implemented were clearly specified by Siskind in both text and by the means of logical expressions. As such they were relatively easy to break down into pieces thus possible to implement step by step.

To test the principles we needed a large file with utterances paired with hypotheses. Generation of utterances therefore had to be done dynamically and because of that a format based on the Backus-Naur Form [3] was developed. The format is built up by different rules for expressions, like "Subject verb object". Words are grouped with similar ones enabling them to be randomly chosen from the group to then be inserted into the expressions. Whenever a meaningful word is selected to be included, the corresponding symbols are paired with the expression.

Chapter 5

Implementation

The implementation in this study consists of two parts, the utterance generator and the algorithm. The utterance generator's purpose is to generate natural language sentences with related hypotheses and the algorithm's, in turn, to parse the input and conclude the meaning of words. Both the utterance generator and the algorithm were implemented in the programming language java and can be run on any standard PC even from many years back.

5.1 Structure of input

The input to the algorithm had to be constructed in a way that was easy to interpret and accordingly be able to use the rules on. The sentences describing the scene are simply put in text strings, which then in the algorithm are split into individual words.

However, the most important part of the structure of the input is the meanings, called symbols individually and called hypotheses when grouped into possible meanings of whole sentences. One example is the sentence *John goes to school* which meaning can be represented by the hypothesis {GO, John, TO, school} in which {GO} and {John} are symbols. In this case {GO, TO} is the meaning of *goes* where {TO} signifies just as *to* in natural language in the case of *go to*, that something is moving towards something else. The symbol {TO} however has only this meaning unlike the word *to*, and does not vary depending on the context. Some words might not be paired with any symbol and *to* is one of them, as *to* does not have a meaning by itself.

Many symbols can be used to describe the meaning of a word, for example {GIVE, TO} could have been used instead of {CAUSE, GO, TO} to describe *give*. But to keep the amount of symbols to a minimum and to test the algorithm, the decision was made to use symbols that can be reused in other meanings e.g. {GO} symbolizing movement and {TO} symbolizing direction.

5.2 Utterance generator

The first significant part of the implementation is the utterance generator. In order to produce sentences of natural language, the generator needed to be extensively developed. The utterance generator developed in this study breaks down the building of a sentence into words and expressions and is through that, with a large set of rules defined in its input file (a sample is given below in listing 5.1), able to generate natural and varying sentences.

Listing 5.1. The input with rules to the utterance generator.

```

/*defined verbs at the top in different tenses*/
GO = goes went
EAT = eats ate

U { I Vf }
Vf {
    V[0]
    V[1]
}
V[0,1] {
    GO to I | GO TO
    EAT F FOOD | EAT
}
I {
    PERSON
    F ANIMAL
}
PERSON {
    Carl | Carl
    Susan | Susan
}
ANIMAL {
    cat | cat
}
FOOD {
    banana | banana
    apple | apple
}
F {
    the
    a
}
/* possible sentences with this format
"Susan went to the cat" or "Carl eats a banana"*/

```

5.2. UTTERANCE GENERATOR

Firstly the generator builds an utterance by fetching a random instance belonging to the utterance start clause ("Subject verb"). If "branch" symbols occur in that instance, the generator will fetch a random instance from that branch to include in the utterance. Simultaneously the generator fetches symbols paired with the selected instance in the branch, which later comes together as a hypothesis.

After the first hypothesis has been created (which most likely is the only correct one) the generator continues generating utterances (as many as the referential uncertainty number) but discards the utterance and only adds the new hypothesis as a hypothesis to the first utterance. This is then repeated to create a rather large corpus of utterances with related hypothesis. An example of the output of the utterance generator, which is then used as input to the algorithm, is shown in figure 5.2.

Following is an example of how a tree of "branches" develops when an utterance is generated. Construction of different utterances is possible here, for example *John gives a cat a ball* or *John gave the cat a ball*, which is then paired with {John, CAUSE, GO, TO, cat, spherical-toy}, as a main hypothesis. This as *John* is coupled with {John}, *give a ... a ...* with {CAUSE, GO, TO}, *cat* with {cat} and *ball* with {spherical-toy} as seen in figure 5.1. Additional hypothesis can then be added by constructing a new utterance *John gives the cat a jelly-rat*, discard the utterance and keep the hypothesis {John, CAUSE, GO, TO, cat, jellyrat}. The more hypotheses the higher the referential uncertainty.

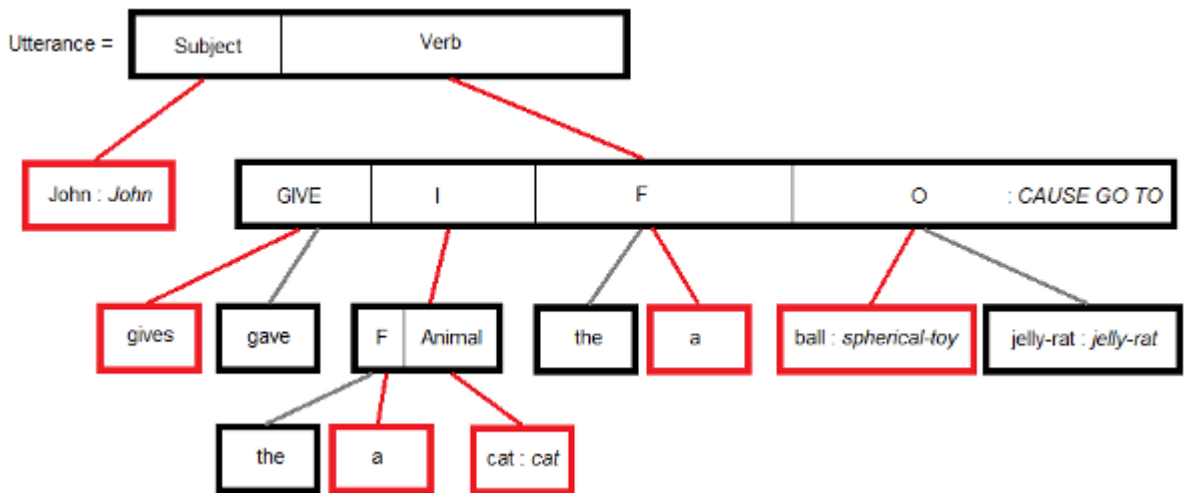


Figure 5.1. A small example of how a sentence is constructed. Boxes in red are the ones chosen, the rest were possibilities. *F* and *O* are examples of "branch" symbols.

```
frazz@delluxe: ~/Dropbox/KEX/latestforre:
File Edit View Terminal Help
Carl lifts a bowl
{
Carl CAUSE GO UP bowl
Mary DRINK water
cat WALK FAST
Mary ROTATE-ROLL
John GO FROM school
cat ROTATE-ROLL ON floor
}
Susan eats the apple
{
Susan EAT apple
cat EAT apple
cat DRINK wine
John WALK
Carl DRINK glass water
cat GO FROM school
}
:
```

Figure 5.2. The output file from the utterance generator can look like this with the natural language sentence first, and following inside curly brackets its hypotheses.

5.3 The algorithm

The other important part of the implementation is the algorithm. It is constructed of the four principles mentioned earlier and hereafter explained in detail how the implementation uses them on an example to conclude the meaning of the words.

The algorithm takes as input the output from the utterance generator. Then it goes through the input, sentence by sentence one at a time, looking for new words and words which meaning has not been concluded. If there exists such words in the current sentence, the principles are run on the sentence and the new words are added to the wordlist. When possible and necessary symbols for a word coincide, the word is added to the dictionary. Below in listing 5.2 follows pseudocode of the algorithm:

5.4. POSSIBLE AND NECESSARY SYMBOLS

Listing 5.2. Pseudocode of the algorithm.

```
Algorithm input /* input is a list of tuples containing a
                sentence and a list of hypotheses as in figure 5.2*/
{
  wordlist = NULL /* has type: (word, N(word),
                          P(word), Concluded(word))*/

  for each tuple in input
    // tuple = (sentence, hypotheses[ ])
    {
      if( sentence has new words )
      {
        add these words to the wordlist with an
        empty N(word) and Concluded(word)

        add all symbols in all hypotheses,
        relating to the sentence, to P(word)
      }

      remove hypotheses according to principle 1

      for each remaining hypothesis
      {
        apply principle 2
        apply principle 3
        apply principle 4
        if( N(word) == P(word) ) //word learned
          add collection of symbols in N(word)
          to Concluded(word)
      }
    }
}
```

5.4 Possible and necessary symbols

The implementation of the principles is built up by removing and filling lists paired with the words. Every word is paired with two lists: the necessary symbols list and the possible symbols list, representing the sets N and P respectively seen in the logical expressions of the principles in section 3.3. The possible symbols list contains all symbols that cannot be excluded as a meaning for the word. The necessary symbols list contains only the symbols that the algorithm has found to partly be the meaning of the word. When the necessary symbols list contains the

same symbols as the possible symbols list it is concluded which symbols are to be paired with the word and therefore the meaning of the word have been learned.

5.5 Principles

5.5.1 Principle 1: Hypothesis coinciding with known words

Example: *John walks very fast.*

Hypotheses:

- (a) {WALK, John, FAST}
- (b) {GO, John, TO, school}
- (c) {WALK, school, TO, John, GO}

After the algorithm has processed a number of utterances, the lists for the words in the example might look something like table 5.1.

Word	Necessary symbols	Possible symbols
John	{John}	{John, Mary, GO}
walks	{WALK}	{WALK, TO}
very	{}	{TO, John}
fast	{}	{FAST, TO}

Table 5.1. After principle 1

If a hypothesis excludes any necessary symbol that is known to be in the sentence, the hypothesis will be discarded as it is obviously not a valid hypothesis to the sentence. The necessary symbols {John} and {WALK} are both in (a) and (c) which means it fulfilled the first criterion to be a valid hypothesis. Hypothesis (b) is missing the symbol {WALK} and should therefore be discarded.

The second criterion that needs to be fulfilled is that all the symbols in the hypothesis need to be included in the combined list of possible symbols. Symbols not in any of the possible symbols lists are known to not be part of the meaning of the sentence. All symbols in (a) are found in the union of the possible symbols lists, (c) however contains the symbol school which cannot be found in the possible symbols list which leaves us with only one valid hypothesis.

5.5.2 Principle 2: Cross-sentence learning

When one or more hypotheses have passed the first principle, what is being done is removal of the possible symbols for each word that are not contained in the combined list of symbols of the hypotheses. This means that the possible symbols

5.5. PRINCIPLES

of a word, that was gathered from other sentences, which does not match with the hypothesis for the latest sentence will be removed, and only the symbols that all sentences had in common will be kept.

Word	Necessary symbols	Possible symbols
John	{John}	{John}
walks	{WALK}	{WALK}
very	{}	{John}
fast	{}	{FAST}

Table 5.2. After principle 2

After applying the second principle with (a), table 5.1 is updated to table 5.2. Comparing table 5.1 and 5.2 shows that the symbols {Mary} and {GO} are removed from the possible symbols list for *John*. The same is applied to the rest of the words which removes {TO} from their lists.

5.5.3 Principle 3: Meaning only possible for one word

A symbol that is apparent in all hypotheses and only appears in one of the words' possible symbols lists, is part of the meaning of that word and will be moved to the necessary symbols list.

Word	Necessary symbols	Possible symbols
John	{John}	{John}
walks	{WALK}	{WALK}
very	{}	{John}
fast	{FAST}	{FAST}

Table 5.3. After principle 3

In this case the symbol {FAST} only occurred in the possible symbols list of the word *fast* and is therefore moved into the necessary symbols list of that word. The symbol {John} appears for the word *John* and *very* so the symbol {John} is not necessarily the meaning of *very*. {WALK} appears only for the word *walks* in this sentence and must be the meaning of that word (the algorithm already had that figured out from a previous run in this case).

5.5.4 Principle 4: Meaning of words not overlapping

For each word in the sentence, the symbols which only appear once in every hypothesis and also are necessary symbols in any of the other words of the utterance are removed from the possible symbols lists.

The symbol {John} only appeared once in the hypothesis and was already known to be the meaning of one of the words in the utterance. Thus the symbol {John} can be removed from all other possible symbols lists. In this case from the word *very*.

When a row looks like any of those in table 5.4, when the necessary symbols set is equal to the possible symbols set, the algorithm will assign that set as the meaning for the word.

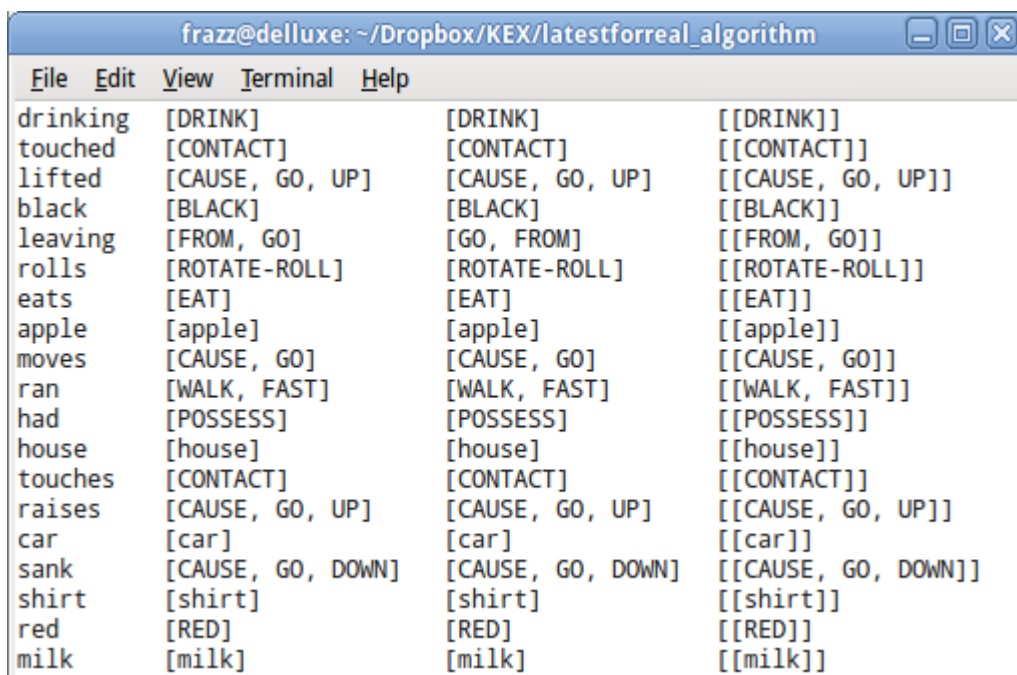
Word	Necessary symbols	Possible symbols
John	{John}	{John}
walks	{WALK}	{WALK}
very	{}	{}
fast	{FAST}	{FAST}

Table 5.4. After principle 4

5.6 Output from the algorithm

After the algorithm have been run on all the input, the different lists have been filled up with words and symbols. Here the output from the algorithm is shown in figure 5.3 and 5.4. The figure shows the output when having 500 utterances as input (figure 5.3) and 100 utterances as input (figure 5.4). A referential uncertainty of 5 was used in both examples. npd The first column has the word, the second necessary symbols for the words meaning, the third possible symbols for the words meaning and lastly the fourth (if not empty) the concluded meaning of the word.

5.6. OUTPUT FROM THE ALGORITHM



A terminal window titled "frazz@delluxe: ~/Dropbox/KEX/latestforreal_algorithm" displays the output of an algorithm. The output is a table with four columns: the original word, its learned representation in square brackets, the word in all caps, and the word in double square brackets. The words listed are: drinking, touched, lifted, black, leaving, rolls, eats, apple, moves, ran, had, house, touches, raises, car, sank, shirt, red, and milk.

Word	Learned Representation	Word (Caps)	Word (Double Brackets)
drinking	[DRINK]	[DRINK]	[[DRINK]]
touched	[CONTACT]	[CONTACT]	[[CONTACT]]
lifted	[CAUSE, GO, UP]	[CAUSE, GO, UP]	[[CAUSE, GO, UP]]
black	[BLACK]	[BLACK]	[[BLACK]]
leaving	[FROM, GO]	[GO, FROM]	[[FROM, GO]]
rolls	[ROTATE-ROLL]	[ROTATE-ROLL]	[[ROTATE-ROLL]]
eats	[EAT]	[EAT]	[[EAT]]
apple	[apple]	[apple]	[[apple]]
moves	[CAUSE, GO]	[CAUSE, GO]	[[CAUSE, GO]]
ran	[WALK, FAST]	[WALK, FAST]	[[WALK, FAST]]
had	[POSSESS]	[POSSESS]	[[POSSESS]]
house	[house]	[house]	[[house]]
touches	[CONTACT]	[CONTACT]	[[CONTACT]]
raises	[CAUSE, GO, UP]	[CAUSE, GO, UP]	[[CAUSE, GO, UP]]
car	[car]	[car]	[[car]]
sank	[CAUSE, GO, DOWN]	[CAUSE, GO, DOWN]	[[CAUSE, GO, DOWN]]
shirt	[shirt]	[shirt]	[[shirt]]
red	[RED]	[RED]	[[RED]]
milk	[milk]	[milk]	[[milk]]

Figure 5.3. A selection of the output from the algorithm, where all words have been learned.

File	Edit	View	Terminal	Help
glass	[glass]	[glass]	[[glass]]	
mary	[Mary]	[Mary]	[[Mary]]	
is	[]	[]	[[[]]]	
having	[POSSESS]	[POSSESS, spherical-toy]	[]	
ball	[spherical-toy]	[spherical-toy]	[[spherical-toy]]	
touched	[CONTACT]	[CONTACT]	[[CONTACT]]	
shirt	[shirt]	[shirt]	[[shirt]]	
john	[John]	[John]	[[John]]	
taking	[]	[CAUSE, GO, TO]	[]	
shoe	[]	[John, CAUSE, GO, TO, shoe]	[]	
took	[TO]	[CAUSE, GO, TO]	[]	
red	[RED]	[RED]	[[RED]]	
pen	[pen]	[pen]	[[pen]]	
lifted	[UP]	[cat, CAUSE, GO, UP]	[]	
green	[]	[cat, GREEN]	[]	
sees	[SEE]	[SEE]	[[SEE]]	
rolls	[ROTATE-ROLL]	[ROTATE-ROLL]	[[ROTATE-ROLL]]	
on	[ON]	[ON]	[[ON]]	
floor	[floor]	[ROTATE-ROLL, ON, floor]	[]	
moving	[CAUSE, GO]	[CAUSE, GO]	[[CAUSE, GO]]	
rolled	[ROTATE-ROLL]	[ROTATE-ROLL]	[[ROTATE-ROLL]]	
had	[POSSESS]	[POSSESS]	[[POSSESS]]	
apple	[apple]	[apple]	[[apple]]	
walking	[WALK]	[WALK]	[[WALK]]	

Figure 5.4. A selection of the output from the algorithm, where not all words have been learned as there was too few sentences. For example can shoe still include {John}, {CAUSE}, {GO}, {TO} and {shoe} in its meaning.

Chapter 6

Results

6.1 Tests

Tests were made with varied amount of utterances and varied amount of sets of symbols for each utterance to find how they affected the learning task. There were about 100 different words in total for the system to learn and about 50 different symbols. About 250 utterances were needed to acquire the meanings of 80 words. The rest of the words could not be acquired no matter the corpus size (see table 6.1 and illustrated in figure 6.1). Probably as a result of the lexicon being corrupt because the words *to*, *is* and *was* in the utterances are homonymous. Another possible cause is that some words only appears with the same set of words and with about the same set of symbols e.g. *John eats the banana*, where the word banana is said only when the symbols {EAT} and {banana} occur. Changing the number of symbol sets paired with the utterances (referential uncertainty) did not affect the learning task noticeably when using a corpus size of 250 (see table 6.2 and illustrated in figure 6.2) which is proof that the principles of exclusion is working.

Corpus size	50	150	250	500
% of words acquired	50%	70%	80%	80%

Table 6.1. Showing average lexicon size acquired using a referential uncertainty of 1 when varying corpus size.

Referential uncertainty	1	5	10	20
% of words acquired	80%	80%	80%	80%

Table 6.2. Showing average lexicon size acquired using a corpus size of 250 when varying the referential uncertainty.

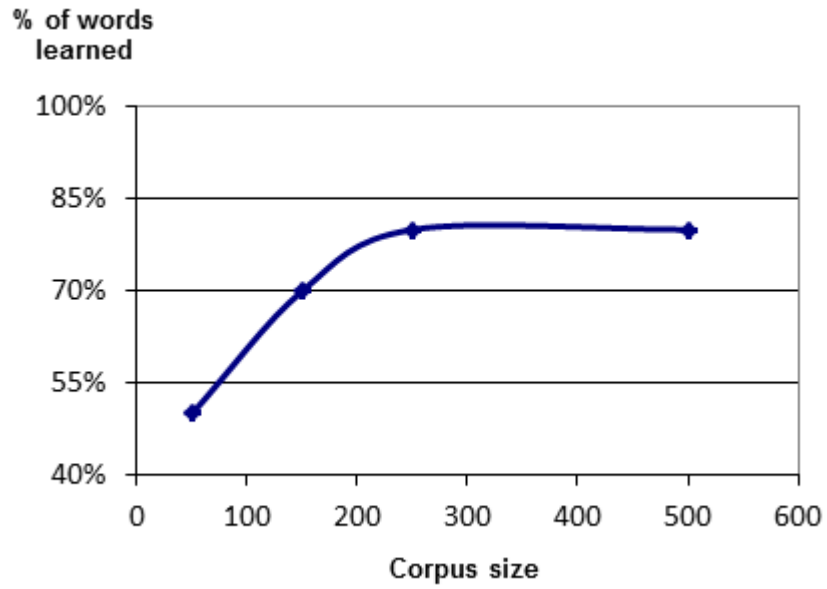


Figure 6.1. An illustration of table 6.1

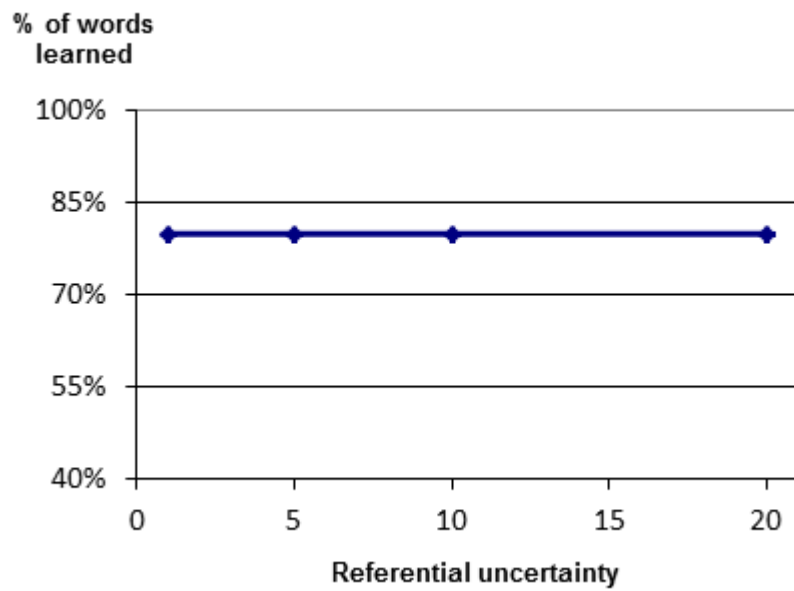


Figure 6.2. An illustration of table 6.2

6.2. TESTS WITHOUT CORRUPTION

6.2 Tests without corruption

Further tests were conducted where input that could corrupt the lexicon was excluded (described in 6.1), as well as input where the algorithm could not determine the meaning of words because two or more words appeared only together. When the corpus size was large enough, around 250 utterances (see table 6.3 and illustrated in figure 6.3), the meanings of all words were learned, which confirms Siskind's results [1] that without homonymy all words shall be learned.

Corpus size	50	150	250	500
% of words acquired	50%	85%	99%	100%

Table 6.3. Tests without corrupting input.

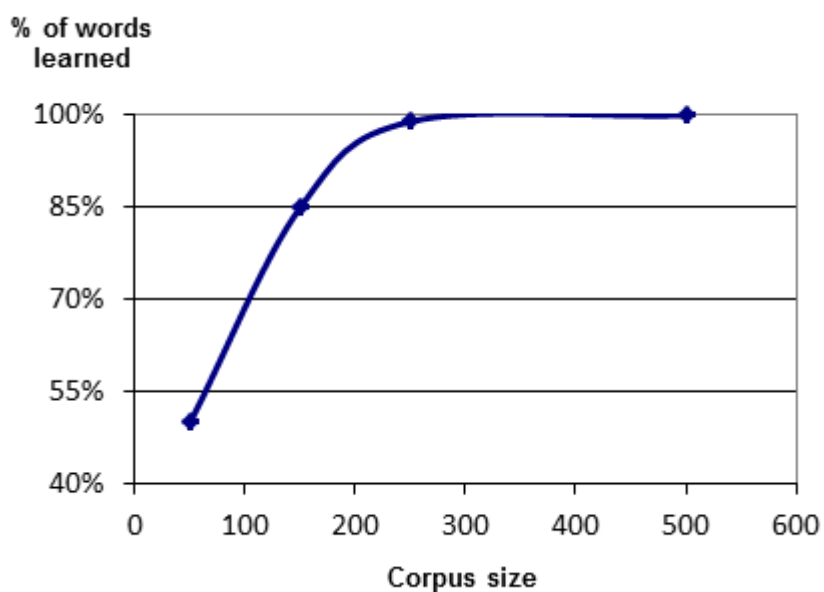


Figure 6.3. An illustration of table 6.3

Chapter 7

Discussion

7.1 Problems with the implementation

Because the implementation does not handle homonymy several of the words will not receive meanings, for example the word *is*. In many instances the word would be mapped to the symbol {BE} (i.e. *John is a man*) but in others it is just a redundant word not mapped to any symbol (i.e. *John is walking*).

The words *is*, *was* and *to* were identified to corrupt the lexicon. In Siskind's simulations [1] without noise and homonymy all words were paired with the right meaning. Prevention of that happening is easiest done by excluding these words or at least only using the words in instances where they have the same meaning, a problem though as that means fewer utterances can be generated.

Another problem is that it is time consuming to add new utterance formats that make sense. Separate words such as verbs, nouns and adjectives are easy to add individually, but it is not possible to add these dynamically which makes it quite time consuming to add a reasonable amount of words (that does not corrupt the lexicon).

In this implementation there are multiple symbols assigned to each word but there is no way of pairing multiple words to one symbol unless the words are not separated by spaces, where the algorithm will identify the combination as an entirely new word instead.

The algorithm does not know how to arrange the symbols in a manner that it can create a sensible functional expression i.e. { GO(John, TO(school)) }. This is not a problem for the actual lexical acquisition task, but it limits the amount of applications based on this algorithm.

However, to solve this problem two additional methods was developed by Siskind [1] and can be implemented though that is outside the scope of this study.

7.2 Homonymy and noise

The implementation in this study handles the three problems: handling of sentences, referential uncertainty and bootstrapping. There exists even more possible problems such as that a scene can be falsely described, which is called noise. A description of a scene in just one sentence is often not complete as it can be done in many, many ways. Which event is actually referred to in the sentence? For example the utterances *It is sunny* and *The horse stood in the meadow* can both describe the same scene, but signifies two completely different events. Noise however is the case when the utterance does not refer to any of the events in the scene.

Another possible problem called homonymy is that a word, depending on the context, could have different meanings. An example is the word *right*. In some cases it means that something is correct, in some it explains that someone should be able to do something and it can also signify a direction.

Siskind has a solution for these problems [1], but according to his own simulation a homonymy rate of 1.68 and a noise rate of 5% with a vocabulary size of 10,000 words results in about 20 % false positives and false negatives in the lexicon, a result which is far from reliable.

7.3 Implementing other senses

This implementation mainly focuses on visual input, most utterances are structured to what is visually observable (i.e. *John walks to the store*). What this means is that the degree of referential uncertainty increases if more sensory input is implemented (i.e. {CAUSE, sun, WARM, face} as a hypothesis to *John walks to the store*). However, simulations of this algorithm have shown that the rate of lexical acquisition is unaffected by the increase in referential uncertainty (see table 6.2), Siskind acquired similar results [1]. Consequently there should not be a problem to implement symbols for hearing, touch, taste, smell and balance. The same applies for facial expressions and body language. The referential uncertainty increases but does not affect the acquisition task.

7.4 Applications

One present-day application would be to combine this algorithm with a scene interpreter- and a speech recognition system. This scene interpreter would have to be able to detect events in a scene or from the real world, in which area Siskind also has made some work [2]. The events would then be converted to a set of symbols which the algorithm is able to understand. These events would then be paired with words identified by the speech recognition system. Utilization of this could be in very complex systems like conversation robots able to interact with the world. Other computer vision modules could be integrated like a nervous system able to detect pressure (i.e. touch).

7.4. APPLICATIONS

A problem that might arise is if the system learns a word incorrectly. Currently this means that the lexicon is broken, the algorithm cannot reset incorrect meanings because the algorithm has no way of telling a correctly learned word apart from an incorrectly learned word. The system must therefore be able to relearn a word to be practical. Thinking about possible applications like this gives ideas about how the algorithm can be developed further.

Chapter 8

Conclusion

Conclusions that can be drawn are that the results acquired in this study are absolutely comparable with that of Siskind's [1]. If a comparison is made between the implementation in this study and the results Siskind acquired with only the use of the principles used in this study, there is no difference in the percentage of words learned (100% in both cases) if compared to the test cases without homonyms in this study (see table 6.3 and illustrated in figure 6.3), as consideration regarding homonymy was not part of this implementation. Consequently the aim for this study, to replicate Siskind's results was achieved.

Moreover, another interesting conclusion is that referential uncertainty does not affect how many words are learned as the results acquired show no difference regardless of the number of hypotheses for an utterance, precisely what Siskind's study [1] also showed. Thus the algorithm has been very well implemented in this study regarding this problem.

In addition, to further optimize the lexical acquisition algorithm what could be done is to implement solutions to homonymy and noise. This is absolutely necessary in order to be able to use the algorithm on real world languages, with much more complicated texts and sentences than the ones treated in this study, and achieve a passable grade of word meanings acquired.

Chapter 9

Bibliography

1. Siskind JM. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition*. 1996;61(1-2):39-91.
2. Siskind JM. Grounding language in perception. *Artificial Intelligence Review*. 1995;8:371-91.
3. Treutwein B. Syntax summary. [updated - ; cited 2012 Apr 10]. Available from: <http://www.lrz.de/~bernhard/Algol-BNF.html>
4. Anick P, Pustejovsky J. An application of lexical semantics to knowledge acquisition from corpora. *COLNG 1990 Volume 1*. 1990.
5. Ogino M, Kikushi M, Asada M. How can humanoid acquire lexicon? 2006.
6. Kit C. How Does Lexical Acquisition Begin? A cognitive perspective. *Cognitive Science*. 2003;1(1):1-50.
7. Yu C, Ballard DH, Asalin RN. The Role of Embodied Intention in Early Lexical Acquisition. *Cognitive Science*. 2003.