Natural Language for Knowledge Acquisition: Learning from Definite Anaphora

Łucja Iwańska and Kellyn Kruger
Wayne State University
Department of Computer Science
Detroit, Michigan 48202, USA
{lucja, kkk}@cs.wayne.edu
ph. (313) 577-1667, fx: (313) 577-6868

1 Introduction

1.1 Formal Problem of Distinguishing Referring and Non-Referring Definite Noun Phrases

We investigate grammatical inference algorithms capable of automatically acquiring context-free grammars and grammars with limited context-sensitivity that approximate the formal problem of distinguishing English referring and non-referring definite noun phrases. Definite noun phrases are those noun phrases that start with the determiner "the". In the following short text:

"*A dog was found in our backyard. The animal was dirty, but happy. We contacted the Humane Society.*"

there are two definite noun phrases: "the dog" and "the Humane Society"; the first one is a referring definite noun phrase– its antecedent, "a dog", is explicit in the text; the second noun phrase, "the Humane Society", is a non-referring one– the text does not contain any explicit expression to which this noun phrase refers to[1].

1.2 Context-Free Approximation

The formal language encoding the problem of distinguishing referring and non-referring definite noun phrases is unknown; it may or may not be context-free, in fact, it may not even be recursive. We approximate this problem with a possibly weaker notion of context-free. Our approach is feasible because of the nature of the applications we plan for such automatically acquired grammars:

First, computational tractability as well as the actual system performance is absolutely critical because we plan to utilize the acquired grammars for automatic knowledge acquisition from large corpora of English texts.

Second, our knowledge acquisition algorithm that will utilize such learned grammars does not require a perfect coverage of the grammar. In fact, even a very imperfect coverage, the level of 10% or so, is sufficient for acquiring useful knowledge about concepts.

1.3 Hand-Crafted Grammar and Limited Contextual Parameters Facilitate Knowledge Acquisition

We believe that a vast knowledge about the relationships among various concepts as well as knowledge about the applicability of different attributes to describing concepts can be automatically acquired from large corpora of English texts, with very little so-called in-depth, computationally expensive processing[2]. We have devised a simple, yet powerful, algorithm that given a large number, several hundreds or more, of textual documents, allows one to acquire such knowledge automatically. This algorithm utilizes a small, handcrafted grammar, somewhat similar to the grammar reported in [Iwańska, 1991], used there to resolve a general anaphora with weak methods. The right-hand-side of the main top-level production in this small grammar is of the form

$Det\ (Pre\_Modfs)\ Head\ (Post\_Modfs)$

where Det rewrites to simple and complex expressions involving determiners, Pre_Modfs rewrites to noun modifiers such as adjectival phrases and compound nominals, $Head$ is a head noun, and $Post\_Modfs$ are head modifiers such as prepositional phrases and unmarked relative clauses.

The combination of this simple grammar and computing certain contextual, mostly syntactic factors, allows one to distinguish referring and non-referring definite noun phrases with a high degree of accuracy. For the three hundreds texts processed, in 70% of all cases, the system correctly distinguished referring expressions, including definite anaphora, from the non-referring ones. This capability of distinguishing referring and non-referring definite noun phrases allows one to identify parts of texts that constitute learning opportunities.

[1] We ignore references to the entities that are not explicitly mentioned in text.

[2] In-depth processing of natural language often presupposes, among others, a perfect sentential-level parsing.

1.4 Indirectly Referring Definite Noun Phrases Are Learning Opportunities

A subset of referring definite noun phrases, we call them indirectly referring, constitutes learning opportunities. Computation of the resolvents of the indirectly referring noun phrases is nontrivial. It often involves a multi-step reasoning process or an assumption of the existing apriori knowledge.

For example, understanding the coreference relation between "a dog" and "the animal" may require an apriori knowledge that "dogs" are "animals", i.e., understanding the relation, in this case 'set/superset' relation, between the concepts of "dog" and "animal"; understanding the coreference relation between the expressions "CNN's Capital Gang" and "the show" may involve a three-step inference:

1. "CNN is a network";
2. "Networks have shows";
3. "Capital Gang is a show".

We characterize both of these cases as indirectly referring noun phrases.

If one cannot make such a multi-step reasoning connection or if one does not have such an apriori knowledge, then texts containing such indirectly referring definite noun phrases are learning opportunities. One may assume that a text is coherent and on order to support this assumption, one may further assume that it contains unknown relations and decide to learn them.

For example, if one does not know that "dogs" are "animals", i.e., if one's knowledge is incomplete with respect to the relation between the type "dog" and the type "animal", then the reference "the animal" to an individual characterized as "a dog" allows one to acquire this relation.

Similarly, if one never heard of "Capital Gang", then the reference "the show" allows one to acquire knowledge that the entity "CNN's Capital Gang" is an instance of the type "show"; one may additionally make a plausible assumption that other entities of the same type as the entity "CNN", network, are in some relation with the entities of the type "show".

Once an indirectly referring noun phrase is identified, our knowledge acquisition algorithm explores its local context and extracts plausible relations and attributes. When the number of a particular relation or an attribute identified in various texts reaches a certain threshold, this piece of knowledge is acquired. The quality and quantity of the acquired knowledge, so far tested by hand on twenty documents, gives one a serious hope for a significant knowledge base to be automatically acquired from the thousands and millions of textual documents widely available in electronic form.

1.5 From Hand-Crafted to Automatically Acquired Grammars

We are in the process of implementing the knowledge acquisition algorithm supported by the hand-crafted grammar and computation of limited contextual factors. We expect that the acquired knowledge in itself be an achievement; at the very least, it will release us from an extremely unpleasant and time consuming knowledge engineering effort.

However, the fact that both the grammar and the contextual parameters were handcrafted is unsatisfactory. First, it is difficult to improve them. As in case of any handcrafting, the process is tedious, time consuming, the prospect of success is extremely hard to judge; engineering solutions may work, but just as easily they may decrease the performance, something that can be only known after extensive (and expensive) test runs. Second, the question remains as to whether there are better grammars and more appropriate contextual parameters that would significantly improve the quality or quantity of the acquired knowledge.

We have decided to demystify some aspects of this handcrafting process and considered a number of algorithms that allow one to acquire such grammars automatically. We plan to subsequently evaluate each such automatically acquired grammar with respect to the quantity and quality of knowledge it allows the system to acquire from the same texts. This black-box evaluation of the acquired grammar involving a human is a necessity. All but trivial properties of context-free languages are undecidable [Hopcroft and Ullman, 1979]. Unless we identify a special class of grammars that are both good for knowledge acquisition and whose properties of interest are decidable, not much beyond such a black-box evaluation can be done.

1.6 "Knowledge for NLP" Or "NLP for Knowledge"

We believe that natural language facilitates knowledge acquisition. We plan to demonstrate that even a superficial processing of English texts, which very remotely approximates human understanding of natural language, can be an extremely powerful tool for automatic knowledge acquisition.

Our work is loosely psychologically motivated. Our knowledge acquisition algorithm simulates the "natural language learning mode" human behavior of acquiring knowledge signaled by the indirectly referring noun phrases. We believe that this mode of processing natural language is very common, mainly because human knowledge is so very incomplete. People communicate via natural language, engage in dialogs and read texts, in order to learn something new– either to add a new piece of knowledge or to revise the existing one.

Children as young as four years old appear to have

no problems understanding references involving definite noun phrases and appear to be primarily utilizing them to acquire knowledge about concepts, attributes and their relationships. Clearly, at this point of their development, their knowledge of natural language syntax is incomplete and their so-called world knowledge is extremely incomplete.

Adults also learn. An adult who knows nothing whatsoever about dinosaurs when presented with the following text

"*A velociraptor was found in our backyard. The dinosaur was dirty, but happy.*"

is most likely not going to say "*This is incoherent*"; she or he is more likely to conclude that her or his knowledge is incomplete and that there are things of the types "dinosaur" and "velociraptor" and that there is some, as of yet unknown relationship between the two types.

1.7 Is Natural Language A Free Lunch ?

If successful, our work would complement the "Knowledge for NLP" thinking with the "NLP for Knowledge". In both the AI and computational linguistics communities, the generally assumed wisdom of "Knowledge for NLP" represents a belief that in order to process natural language, lots of pre-existing knowledge is necessary. Some of the earliest works adhering to this wisdom is [Schank and Riesbeck, 1981] whose observation about the definite anaphora references to the entities or relationships not mentioned explicitly in the texts sparked the idea of a script. [Hobbs, 1979], [Hobbs et al., 1988], views text understanding as a process of computing text coherence via abductive reasoning. Computing this coherence requires an apriori knowledge of concepts, their relationships and attributes. The same "Knowledge for NLP" wisdom is reconfirmed by the CYC project- one reads [Guha and Lenat, 1990, page 3]

> Natural Language Understanding (NLU) is one tantalizing "free lunch" route: get a program to read English, say, then use it to rapidly build up the large KB (semi)automatically. Unfortunately, NLU itself requires vast amounts of common sense knowledge ... [sentence continues].

Our goal is not to invalidate this "Knowledge for NLP" thinking. Clearly, an apriori existing knowledge about the relations among concepts and knowledge of relevant attributes facilitates understanding texts. We provide a different perspective and addresses a different aspect of human processing of natural language. We do not believe that an apriori existing knowledge is absolutely necessary for processing natural language because natural language understanding is intertwined with the process of knowledge acquisition.

1.8 Content of This Paper

In this paper, we present some preliminary results. We stress that the work is very much in progress. Much remains to be done in order to furnish answers to these puzzling questions and fully support our beliefs about the "natural language learning mode". What we have done so far is the following:

1. We have collected a significant amount of artificial and real-life data. Section 2 discusses a sample artificial language, English corpora of real-life data, sample English texts and our data collection methodology.
2. We have considered a number of grammatical inference algorithms, loosely motivated by inductive machine learning and formal theories of learnability. Section 3 contains a formal statement of the problem; it discusses generic steps of the class of learning algorithms considered, outlines issues, idealizations and assumptions involved, presents three algorithms and shows their output (acquired grammars) for the sample artificial and natural language data.

The details about the algorithm for acquiring knowledge about the relationships among various concepts knowledge about the applicability of different attributes to describing concepts as well as the results of knowledge acquisition such as concepts and concept/attributes relations learned, are not discussed here.

```
E+ = { aeh                          E- = { meh
       bffh                                pppi
       aeeh                                pa
       bfeh                                bfe
       bfffffefeeh                         bfffffefee
       ah                                  a
       aeehme                              aehm
       ahneef                              ahnf
       ahofffef                            aadd
       affffffeeehofffef                   abffffffeeehofffef
     }                                   }
```

Figure 1: Hand-generated positive and negative examples of an artificial, character-based language. Such data constitute test data for the general-purpose grammatical inference algorithms considered.

2 Development and Test Data

The algorithms presented, those implemented and those designed and verified so far by-hand only, are developed based on artificial languages such as languages in standard textbooks on Theory of Computation [Hopcroft and Ullman, 1979] and real linguistic data, electronic versions of written English texts.

2.1 Sample Artificial Language Data

Figure 1 shows a sample artificial, character-based language whose positive and negative examples constitute the development and test data for the general-purpose grammatical inference algorithms considered.

2.2 Sources of Natural Language Real-Life Data

We have collected real-life development and test English data, positive and negative examples of referring definite noun phrases, from the following two sources:

1. Wall Street Journal and Tipster corpora available on a CD-ROM through the LDC initiative; We chose 86 articles that exhibit discourse structure discussed in [Iwańska, 1991]; Figure 2 shows one medium-length text from this set;

2. 1995 Time Magazine Almanac available on a CD-ROM that can be purchased in computer stores;

   We chose 104 articles, the entire content of the four January 1994 Time Magazine issues; Figure 3 shows one medium-length text from this set.

The subsequent sections contain our data collection instruction, show marked sample texts and the data collected from these texts.

2.3 Data Collection Instruction

Our instruction utilizes SGML (Standard Markup Language, ISO 8879)-like marks and attributes similar to the ones used by MUC-6 (Sixth Message Understanding Conference) [MUC, 1995]. A few sentences are taken more or less verbatim from the MUC-6 definite anaphora task description. The MUC-6 training data for this task were not useful for our purposes.

Referring and Non-Referring Definite Noun Phrases

Definite noun phrases start with the definite determiner "the" (we will ignore other definite expressions, including personal pronouns such as "she" and those definite noun phrases that start with the definite demonstrative pronouns such as "this", "these" and "those"). Such noun phrases often refer to the entities introduced explicitly in the text. For example, in the following five-sentence text[3]:

```
NISSEI BUILD KOGYO CO. SAID TUESDAY IT WILL LAUNCH
JOINT VENTURES IN ITALY, CANADA, SOUTH KOREA,
AND TAIWAN  TO CONSTRUCT HIGH-RISE PARKING
FACILITIES.
   THE KANAZAWA-BASED COMPANY SAID THE VENTURES
WILL BE EQUALLY OWNED, WITH NISSEI PROVIDING ELEVATORS,
CONTROL SYSTEMS, AND TURNTABLES. LOCAL PARTNERS WILL
 HANDLE PRODUCTION OF STEEL FRAMES, ASSEMBLY, AND
MAINTENANCE.
   THE COMPANY SAID ITS  JOINT VENTURE  IN CANADA
PLANS TO EXPORT FACILITIES TO SAN FRANCISCO AND OTHER
CITIES ON THE U.S. WEST COAST.
   THE COMPANY SAID THE NEW COMPANIES TO BE ESTABLISHED
WILL BE IMER NISSEI AND IMER INTERNATIONAL IN POGGIBONSI,
ITALY.

   THE OTHERS WILL BE THE BALSHINE FOUNDATION AND
GEORGE THE THIRD AND SON IN VANCOUVER, CANADA= SAMICK
```

[3]This text is the document number 0063 in the Tipster joint venture corpus.

```
<doc>
<DOCNO>  0249 </DOCNO>
<DD>     JULY 18, 1990, WEDNESDAY  </DD>
<SO>     Copyright (c) 1990 Kyodo News Service  </SO>
<TXT>
   KANSAI PAINT CO.,  JAPAN'S  LARGEST PAINT MAKER, ANNOUNCED WEDNESDAY IT WILL
TEAM UP WITH E.I. DU PONT DE NEMOURS AND CO., A LEADING U.S. CHEMICAL
MANUFACTURER, TO FORM A  JOINT VENTURE  IN NORTH AMERICA.

   THE VENTURE, DU PONT-KANSAI AUTOMOTIVE COATINGS CO., WILL MANUFACTURE, SELL,
AND SUPPLY TECHNICAL SERVICES FOR AUTOMOTIVE COATINGS AND RELATED PRODUCTS TO
 JAPANESE -AFFILIATED AUTOMOBILE MANUFACTURERS AND THEIR PARTS SUPPLIERS IN
NORTH AMERICA, COMPANY OFFICIALS SAID.

   KANSAI PAINT AND DU PONT HAVE COOPERATED IN THE DEVELOPMENT OF COATING
TECHNOLOGIES SINCE 1977, THEY SAID.

   THE OSAKA-BASED FIRM WISHES TO TAKE ADVANTAGE OF PLANS BY
 JAPANESE -AFFILIATED AUTOMOTIVE MANUFACTURERS TO PRODUCE 2.5 MILLION VEHICLES
PER YEAR IN NORTH AMERICA BY 1994, THEY SAID.

   KANSAI PAINT CURRENTLY HAS A 34 PERCENT SHARE OF THE MARKET SUPPLYING
COATINGS TO  JAPANESE -AFFILIATED AUTOMAKERS, AND HOPES, THROUGH THE
ESTABLISHMENT OF A  JOINT VENTURE,  TO BOOST THE SHARE TO 45 PERCENT, THEY SAID.

   THE DELAWARE-BASED E.I. DU PONT WILL BUY A 60 PERCENT STAKE IN THE VENTURE,
WHICH WILL BE CAPITALIZED AT 5 MILLION DOLLARS, WITH KANSAI PAINT PURCHASING THE

REMAINING 40 PERCENT,

   THE FIRM, BASED IN WILMINGTON, DELAWARE, WILL BE SET UP AND BEGIN OPERATIONS
SIMULTANEOUSLY AUGUST 1.

   THE FIRM'S OPERATING REGION WILL INCLUDE CANADA AND  MEXICO  AS WELL AS THE
U.S.

   PROJECTED SALES FOR THE FIRST YEAR ARE EXPECTED TO REACH ABOUT 6 BILLION YEN,
THE OFFICIALS SAID.
</TXT>
</doc>
```

Figure 2: Real-life English development and test data: a medium-length sample article from the Wall Street Journal and Tipster corpora

<DOC>
<SO> TIME--The Weekly Newsmagazine--1994,
Copyright (c) TIME Magazine, 1995 TIME Inc. Magazine Company;
(c) 1995 Compact Publishing, Inc. </SO>
<DD> Jan. 17, 1994, Genetics: The Future Is Now </DD>
<SE> TO OUR READERS, Page 12 </SE>
<HL> Elizabeth Valk Long
President </HL>
<TXT>
What do the high-powered pundits on CNN's Capital Gang talk about when the cameras aren't on? Health care and welfare reform? No. The subtleties of China policy? Not exactly. The Washington Redskins? "That's more like it," says White House correspondent Margaret Carlson, who became a regular member of the gang last fall. "Every Saturday afternoon before the show there's this towel-snapping get-together with three guys who are always talking about football when I get there."

Carlson and Elaine Shannon, who covers the guns, spies and wiretaps beat for TIME, are among Washington's newest talking heads. For several months Shannon has been on the weekly news-analysis series To the Contrary. The two join congressional correspondent Julie Johnson, who is often seen on PBS's Washington Week in Review and ABC's This Week with David Brinkley; national-security correspondent Bruce van Voorst, a contributor to the MacNeil/Lehrer News-Hour; and chief political correspondent Michael Kramer, who holds forth periodically on the Charlie Rose Show on PBS.

A veteran print reporter, Shannon initially found the transition to TV daunting. "TV forces you to get right to the point. It helps if you're simpleminded," says the Georgia native, deadpan.

Carlson seldom has time to catch her breath on the war of words that is CNN's Capital Gang. She spends Saturday evenings struggling to be heard above the din raised by Al Hunt, Mark Shields and Robert Novak, who says Carlson's work on the White House beat makes her a valuable addition to the program. "She's not a stuffed-chair pundit. And I'm sure Margaret won't like me saying this, but more than any of the women we've had on the show, she's one of the boys."

"One of the boys?" responds Carlson, incredulous. "Well, I guess that's more p.c. than calling me one of the girls." Watch out, Novak. That comment might just have earned you Margaret's "Outrage of the Week" on next week's show.
</TXT>
</DOC>

Figure 3: Real-life English development and test data: a medium-length sample article from the Time Magazine corpus

```
<doc>
<DOCNO>  0249 </DOCNO>
<DD>     JULY 18, 1990, WEDNESDAY  </DD>
<SO>     Copyright (c) 1990 Kyodo News Service  </SO>
<TXT>
   <RD ID="1">KANSAI PAINT CO.</RD>,  JAPAN'S  LARGEST PAINT MAKER, ANNOUNCED WEDNESDAY IT WILL
TEAM UP WITH <RD ID="3">E.I. DU PONT DE NEMOURS AND CO.</RD>, A LEADING U.S. CHEMICAL
MANUFACTURER, TO FORM <RD ID="2">A  JOINT VENTURE  IN NORTH AMERICA</RD>.

   <RD ID="501" REF="2" RT="D">THE VENTURE</RD>, DU PONT-KANSAI AUTOMOTIVE COATINGS CO.,
WILL MANUFACTURE, SELL, AND SUPPLY TECHNICAL SERVICES FOR AUTOMOTIVE COATINGS AND RELATED
PRODUCTS TO JAPANESE -AFFILIATED AUTOMOBILE MANUFACTURERS AND THEIR PARTS SUPPLIERS IN
NORTH AMERICA, <RD ID="6">COMPANY OFFICIALS</RD> SAID.

   KANSAI PAINT AND DU PONT HAVE COOPERATED IN <ND ID="502">THE DEVELOPMENT OF COATING
TECHNOLOGIES</ND> SINCE 1977, THEY SAID.

   <RD ID="503" REF="1" RT="I">THE OSAKA-BASED FIRM</RD> WISHES TO TAKE ADVANTAGE OF PLANS BY
 JAPANESE -AFFILIATED AUTOMOTIVE MANUFACTURERS TO PRODUCE 2.5 MILLION VEHICLES
PER YEAR IN NORTH AMERICA BY 1994, THEY SAID.

   KANSAI PAINT CURRENTLY HAS <RD ID=4">A 34 PERCENT SHARE OF <ND ID="504">THE MARKET SUPPLYING
COATINGS TO  JAPANESE -AFFILIATED AUTOMAKERS</ND></RD>, AND HOPES, THROUGH <ND ID="505">THE
ESTABLISHMENT OF A  JOINT VENTURE</ND>,  TO BOOST <RD ID="506" REF="4" RT="D">THE SHARE</RD>
TO 45 PERCENT, THEY SAID.

   <RD ID="507" REF="3" RT="D">THE DELAWARE-BASED E.I. DU PONT</RD> WILL BUY <RD ID="5">A 60
PERCENT STAKE IN <RD ID="508" REF="1" RT="D">THE VENTURE</RD></RD>,
WHICH WILL BE CAPITALIZED AT 5 MILLION DOLLARS, WITH KANSAI PAINT PURCHASING
 <RD ID="509" REF="5" RT="D">THE REMAINING 40 PERCENT</RD>,

   <RD ID="510" REF="503">THE FIRM</RD>, BASED IN WILMINGTON, DELAWARE, WILL BE SET UP AND
 BEGIN OPERATIONS SIMULTANEOUSLY AUGUST 1.

   <RD ID="511" REF="503">THE FIRM'S OPERATING REGION</RD> WILL INCLUDE CANADA AND  MEXICO
AS WELL AS <ND ID="512">THE U.S.</ND>

   PROJECTED SALES FOR <ND ID="513">THE FIRST YEAR</ND> ARE EXPECTED TO REACH ABOUT 6 BILLION YEN,
<RD ID="514" REF="6" RT="D">THE OFFICIALS</RD> SAID.
</TXT>
</doc>
```

Figure 4: Collection of the real-life English development and test data: marked sample article from the Wall Street Journal and Tipster corpora

```
<DOC>
<SO> TIME--The Weekly Newsmagazine--1994,
     Copyright (c) TIME Magazine, 1995 TIME Inc. Magazine Company;
     (c) 1995 Compact Publishing, Inc.  </SO>
<DD> Jan. 17, 1994, Genetics: <RND ID="" REF="">The  Future Is Now </DD>
<SE> TO OUR READERS, Page 12 </SE>
<HL> Elizabeth Valk Long
     President </HL>
<TXT>
     What do <ND ID="501">the high-powered pundits on <RD ID="1">CNN's Capital Gang</RD></ND>
talk about when <ND ID="502">the cameras</ND> aren't on? Health care and welfare reform?
No. <ND ID="503">The  subtleties of China policy</ND>? Not exactly.
<ND ID="504">The  Washington Redskins</ND>? "That's more like it," says White
House correspondent Margaret Carlson, who became a regular member of
<RD ID="505" REF="1" RT="D">the gang</RD> last fall. "Every Saturday afternoon before
<RD ID="506" REF="1" RT="I">the show</RD> there's this towel-snapping get-together with
 three guys who are always talking about football when I get there."

     <RD ID="2">Carlson and Elaine Shannon</RD>, who covers <ND ID="507">the guns,
spies and wiretaps beat</ND> for TIME, are among Washington's
<RD ID="3">newest talking heads</RD>. For several months Shannon has been on
<ND ID="508">the weekly news-analysis series To <ND ID="509">the Contrary</ND></ND>.
<RD ID="510" REF="2" RT="I">The two</ND> join congressional correspondent Julie Johnson,
who is often seen on PBS's Washington Week in Review and ABC's This Week with David Brinkley;
national-security correspondent Bruce van Voorst, a contributor to <ND ID="511">the
MacNeil/Lehrer News-Hour</ND>; and chief political correspondent Michael Kramer, who
holds forth periodically on <ND ID="512">the Charlie Rose Show on PBS</ND>.

     A veteran print reporter, <RD ID="4">Shannon</RD> initially found
<RD ID="513" REF="3" RT="I">the transition to TV</ND> daunting.
"TV forces you to get right to <ND ID="514">the point</ND>. It helps if you're
simpleminded," says <RD ID="515" REF="4" RT="I">the Georgia native</RD>, deadpan.

     Carlson seldom has time to catch her breath on <ND ID="516">the war of words</ND> that is
CNN's Capital Gang. She spends Saturday evenings struggling to be heard above
<ND ID="517">the din raised by Al Hunt, Mark Shields and Robert Novak</ND>, who says Carlson's
work on <ND ID="518">the White House beat</ND> makes her a valuable addition to
<RD ID="519" REF=">the program</ND>.
"She's not a stuffed-chair pundit. And I'm sure Margaret won't like me saying
this, but more than any of <ND ID="520">the women we've had on
<RD ID="521" REF="506" RT="D">the show</RD></ND>, she's one of <ND ID="522">the boys</ND>."

     "One of <RD ID="523" REF="522" RT="D">the boys</RD>?" responds Carlson, incredulous.
"Well, I guess that's more p.c. than calling me one of <ND ID="524">the girls</ND>."
Watch out, Novak. That comment might just have earned you Margaret's "Outrage of
<ND ID="525">the Week</ND>" on next week's show.
</TXT>
</DOC>
```

Figure 5: Collection of the real-life English development and test data: marked sample article from the Time Magazine corpus

```
NISSEI CO. AND WOOSUNG HEAVY INDUSTRIAL CO. IN INCHO
SOUTH KOREA= AND TAIWAN  NISSEI, WHICH WILL BE FORMED
BY NISSEI AND A GROUP OF MEDIUM AND SMALL ENTERPRISES
 IN TAIWAN.
```

the definite noun phrase "THE KANAZAWA-BASED COMPANY" in the second sentence refers to the entity introduced in the first sentence "NISSEI BUILD KOGYO CO."; the definite noun phrase "THE VENTURES" in the second sentence refers to the entities introduced in the first sentence "JOINT VENTURES IN ITALY, CANADA, SOUTH KOREA, AND TAIWAN". We call such noun phrases referring definite noun phrases (RD). The relation between a referring expression and its antecedent is called a coreference relation.

The coreference relation may be direct in that the connection between the referring expression and its antecedent is established by the simple occurence of the same word, most often the head noun. The relation between the expression "THE VENTURES" and the expression "JOINT VENTURES IN ITALY, CANADA, SOUTH KOREA, AND TAIWAN" is an example of a direct coreference, with the noun "ventures" being the head noun.

The coreference relation may be also be indirect in that the connection between the referring expression and its antecedent must be established by some kind of reasoning, and cannot be computed by simple match between two words or their morphological variants. The relation between the expression "THE KANAZAWA-BASED COMPANY" and the expression "NISSEI BUILD KOGYO CO." is an example of an indirect coreference; the reasoning required to make the connection involves understanding that "CO." and "company" are related, and in this case, that the relation is abbreviation. Understanding the coreference relation between the expressions "CNN's Capital Gang" and "the show" may require a three-step inference:

1. "CNN is a network";
2. "Networks have shows";
3. "Capital Gang is a show".

Some other examples of such indirect relations include:

Synonymy
An entity introduced by the phrase "a joint venture" may be referred to as "the company"; understanding this reference may require an apriori knowledge that these expressions are synonymous;

Set/superset relation
An entity introduced by the phrase "a dog" may be referred to as "the animal; understanding this reference may require an apriori knowledge that dogs are animals (i.e., that the set of animals includes the set of dogs);

Part/subpart relation
After introducing an entity by the phrase "a house" one of its parts may be referred to by the expression "the door"; understanding this reference may require an apriori knowledge that houses have doors (i.e., that a door is a subpart of a house).

Not all definite noun phrases refer to the entities introduced explicitly in the text. For example, the noun phrase "THE BALSHINE FOUNDATION" in the fifth sentence does not refer to any entity introduced before or after it. We call such noun phrases non-referring definite noun phrases (ND).

SGML-like Tagging

The object is to collect positive and negative examples of referring definite noun phrases (referring definite noun phrases and their antecedents, and non-referring definite noun phrases) by tagging English texts with SGML-like (Standard Markup Language, ISO 8879) marks. These marks are partially described by the following grammar:

```
SGML_Tag      ->   Start_Mark End_Mark
Start_Mark    ->   < Tag_Name Attributes >
End_Mark      ->   </ Tag_Name >
Tag_Name      ->   RD |
                   ND
Attributes    ->   ID="Integer" |
                   REF="Integer" |
                   RT="Value" |
                   empty-string
Value         ->   "D" |
                   "I"
```

The direct and indirect coreference relations will be distinguished by the value of the "RT" (reference type) attribute: the value "D" will mark a direct coreference relation, and the value "I" will mark an indirect coreference relation.

The two coreference relations above will be encoded by inserting SGML-like tags into the original text as follows:

```
<RD ID="1">NISSEI BUILD KOGYO CO.</RD> SAID
TUESDAY IT WILL LAUNCH  <RD ID="2">JOINT
VENTURES IN ITALY, CANADA, SOUTH KOREA, AND
TAIWAN</RD> TO CONSTRUCT HIGH-RISE PARKING
FACILITIES.

<RD ID="3" REF="1" RT="I">THE KANAZAWA-BASED
COMPANY</RD> SAID <RD ID="4" REF="2" RT="I">THE
VENTURES</RD> WILL BE EQUALLY OWNED, WITH NISSEI
PROVIDING ELEVATORS, CONTROL SYSTEMS, AND
TURNTABLES. LOCAL PARTNERS WILL HANDLE PRODUCTION
OF STEEL FRAMES, ASSEMBLY, AND MAINTENANCE.
```

There is one markup per string. Other links can be inferred from the explicit links. We assume that the coreference relation is symmetric and transitive, so if phrase A is marked as coreferential with B (indicated by a REF pointer from A to B), we can infer that B is coreferential with A; if A is coreferential

with B, and B is coreferential with C, we can infer that A is coreferential with C.

The non-referring definite noun phrase "THE BALSHINE FOUNDATION" will be marked as follows

```
THE OTHERS WILL BE <ND ID="5">THE BALSHINE
FOUNDATION</ND> AND GEORGE THE THIRD AND SON
IN VANCOUVER, CANADA= SAMICK NISSEI CO. AND
WOOSUNG HEAVY INDUSTRIAL CO. IN INCHON,
SOUTH KOREA= AND TAIWAN  NISSEI, WHICH WILL BE
FORMED BY NISSEI AND A GROUP OF MEDIUM AND SMALL
ENTERPRISES IN TAIWAN.
```

The "ID" and "REF" attributes are used to indicate that there is a coreference link between two strings; lack of the REF attribute means the definite noun phrase is non-referring, and therefore marked with the "ND" tag name. The "ID" is an arbitrary integer uniquely assigned to the string during markup. The "REF" attribute uses that "ID" number to indicate the coreference link. We assume the following convention:

- Numbers 1-499 are reserved for the IDs of the expressions to which referring noun phrases refer; such expressions have "RD" tag names;
- Numbers 501-999 are reserved for the IDs of the definite noun phrases.

So the above two coreference relations should be marked as follows:

```
<RD ID="1">NISSEI BUILD KOGYO CO.</RD>
SAID TUESDAY IT WILL LAUNCH <RD ID="2">JOINT
VENTURES IN ITALY, CANADA, SOUTH KOREA, AND
TAIWAN</RD> TO CONSTRUCT HIGH-RISE PARKING
FACILITIES.

<RD ID="501" REF="1" RT="I">THE KANAZAWA-BASED
COMPANY</RD> SAID <RD ID="502" REF="2" RT="D">
THE VENTURES</RD> WILL BE EQUALLY OWNED, WITH
NISSEI PROVIDING ELEVATORS, CONTROL SYSTEMS, AND
TURNTABLES. LOCAL PARTNERS WILL HANDLE PRODUCTION
OF STEEL FRAMES, ASSEMBLY, AND MAINTENANCE.
```

Another departure from the MUC-6 data collection methodology is that tags (marks) may overlap, as in the sentence marked below:

```
KANSAI PAINT CURRENTLY HAS <RD ID=3">A 34
PERCENT SHARE OF <ND ID="504">THE MARKET
SUPPLYING COATINGS TO JAPANESE-AFFILIATED
AUTOMAKERS </ND> </ND>, AND HOPES, THROUGH
<ND ID="505"> THE ESTABLISHMENT OF A JOINT
VENTURE</ND>, TO BOOST <RD ID="506" REF="3">
THE SHARE</RD> TO 45 PERCENT, THEY SAID.
```

Only the $< TXT >$ portion of the article should be annotated. You may speed up the data collection process if you first mark all the definite noun phrases, and then make the referring/non-referring distinction, marking accordingly the referred to expressions.

The set of positive examples, $E+$, consists of all the expressions marked with the "RD" tag names and containing the "REF" attribute; the set of negative examples, $E-$, consists of all the expressions marked with the "ND" tag names and containing no "REF" attribute.

2.4 Data from Sample Texts

Figures 6 and 7 show data collected from the two sample texts.

```
Ed+ = { THE VENTURE
        THE SHARE
        THE DELAWARE-BASED E.I. DU PONT
        THE VENTURE
        THE REMAINING 40 PERCENT
        THE FIRM
        THE FIRM'S OPERATING REGION
        THE OFFICIALS }

Ei+ = { THE OSAKA-BASED FIRM }

E-  = { THE DEVELOPMENT OF COATING TECHNOLOGIES
        THE MARKET SUPPLYING COATINGS TO JAPANESE-AFFILIATED AUTOMAKERS
        THE ESTABLISHMENT OF A JOINT VENTURE
        THE U.S.
        THE FIRST YEAR }
```

Figure 6: Collected real-life English development and test data; $E+ = Ed+ \cup Ei+$ is the set of positive examples of referring definite noun phrases found in the sample article from the Wall Street Journal and Tipster corpora; $Ed+$ contains directly referring noun phrases; $Ei+$ contains indirectly referring noun phrases; $E-$ is the set of negative examples of referring definite noun phrase found in the sample article.

```
Ed+ = { the gang
        the show }

Ei+ = { the show
        the two
        the transition to TV
        the Georgia native
        the boys }

E- = { the high-powered pundits on CNN's Capital Gang
       the cameras
       The subtleties of China policy
       The Washington Redskins
       the guns, spies and wiretaps beat
       the weekly news-analysis series
       the Contrary
       the MacNeil/Lehrer News-Hour
       the Charlie Rose Show on PBS
       the point
       the war of words
       the din raised by Al Hunt, Mark Shields and Robert Novak
       the program
       the women we've had on the show
       the boys
       the Week }
```

Figure 7: Collected real-life English development and test data; $E+ = Ed+ \cup Ei+$ is the set of positive examples of referring definite noun phrases found in the sample article from the Wall Street Journal and Tipster corpora: $Ed+$ contains directly referring noun phrases; $Ei+$ contains indirectly referring noun phrases; $E-$ is the set of negative examples of referring definite noun phrase found in the sample article.

3 Grammatical Inference

3.1 General Context-Freeness

First, we experimented with context-freeness. We assumed that the problem of distinguishing referring from non-referring definite noun phrases is recursive and that a context-free grammar (CFG) can capture or, at least, reasonably well approximate the problem. We approached the problem as a pure grammatical inference, i.e., learning a CFG from positive and negative examples, and considered a number of general, not tailored to natural language, algorithms. Definition 1 below formally states this learning problem.

3.2 Formal Problem Statement

Definition 1 (Grammatical Inference)

Let Σ be an alphabet and L some unknown language over Σ. The language L may or may not be context-free. Design an algorithm that given

1. *$E+$, a set of positive examples of L, some subset of L,*
2. *$E-$, a set of negative examples of L, some subset of $\neg L$, the complement of L*

produces a context-free grammar G that generates the best approximation of L, i.e., such that
$L(G) \approx L$.

For example, one such algorithm, given the sets
$E+ = \{\ aaaaa,\ aa,\ a\ \}$
$E- = \{\ b,\ baa\ \}$
may produce the grammar $S \rightarrow a \mid aS$.

3.3 Input

In case of the artificial language data, input consists of two finite sets $E+$ and $E-$; Figure 1 shows positive and negative examples of a sample artificial language to be learned.

In case of the natural language English data, input to the learning algorithms consists of two finite sets $E+$ and $E-$ obtained from multiple texts. The $E+/E-$ pairs of sets from individual textual documents are correspondingly unioned to produce one collective pair of the $E+$ and $E-$ sets; Figures 6 and 7 show such data obtained from the sample English texts.

3.4 Generic Steps of Learning Algorithms

The following are generic steps of algorithms considered:

Input

Positive examples
$E+ = \{e_1^+, \ldots, e_n^+\}$;
Negative examples
$E- = \{e_1^-, \ldots, e_m^-\}$;
Character alphabet
$\Sigma = \{a_1, \ldots, a_N\}$
Lexical categories
$V^{lex} = \{V_1^{lex}, \ldots V_M^{lex}\}$
Lexical productions
For character-based languages,
$P^{lex} =$
$\{V_1^{lex} \rightarrow a_1,$
$\ldots$
$V_M^{lex} \rightarrow a_M\}$
Lexical category recognizers
For word-based languages,
$R^{lex} = \{R_1^{lex}, \ldots, R_M^{lex}\}$, where each R_i^{lex} is a function mapping a string over the alphabet Σ onto some subset of lexical categories, i.e.,
$R_i^{lex} : \Sigma^* \rightarrow V^{lex*}$

Step 0: Initialize grammar :
Trivial initialization:

1. $\Sigma = \emptyset$, or Σ is some standard alphabet such as English alphabet
   $\Sigma^{English} = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, x, y, z\}$
2. $V = \{S\} \cup V^{lex}$, where S is a start nonterminal symbol, V^{lex} is a set of lexical categories
3. $P = P^{lex}$, a set of lexical productions;

Nontrivial initialization: start with a given grammar

Step 1: Select examples to learn from:

1. Remove some elements from $E+$ or $E-$
2. Add some additionally generated elements to $E+$ or $E-$

Step 2: Select subsequent example e_i to process:

If $e_i = w_1 \ldots w_j$ is a positive example
Check if e_i is generated by the current intermediate grammar G_i: If generated, then go to Step 2 (process a subsequent example); If not generated, then update G_i to generate e_i: add a new production $V_j \rightarrow w_1 \ldots w_j$ to P; generalize/simplify the resulting new intermediate grammar; verify that no previously seen negative examples are covered:

Abbreviated, Full Name		Open ?	Examples, Explanations
adj	adjective	closed	alive, happy, ugly
adv	adverb	closed	almost, fast, very
det	determiner	closed	an, many, the
number	number	closed	three, twenty, fiftieth
prep	preposition	closed	after, below, onto
pron	pronoun	closed	mine, they, who
interj	interjection	closed	aah, gosh, whoops
conj	conjunction	closed	however, if, likewise
and_	and	closed	Boolean "and"
or_	or	closed	Boolean "or"
not_	not	closed	Boolean "not"
neither_	neither	closed	Boolean neither "(not (X or Y))"
nor_	nor	closed	Boolean "(not or)"
cue	cue phrase	closed	furthermore, meanwhile moreover
noun	noun	open	book, January, year
verb	verb	open	were, may, read
abbr	abbreviation	open	"N" for "North" "Ct" for "Court" "Bldg" for building
name	name	open	Agatha, Detroit, June

Table 1: Lexical categories in the dictionary of the UNO natural language processing system. Further subcategorizations are possible via features encoded in the dictionary entries.

If $e_i = w_1 \ldots w_j$ is a negative example
Check if generated by the current intermediate grammar G_i: If generated, then update G_i to not generate e_i: If not generated, then go to Step 2;

Individual algorithms differe primarily in the following three aspects:

1. Initialization of a start grammar;
2. Selection of examples utilized for learning;
3. Type and aggressiveness of the generalization strategy.

3.5 Issues, Idealizations, Assumptions

In the discussion below, L is the language to be inferred from positive and negative examples given.

3.5.1 Perfect Coverage Versus Approximation

One of the most important issues is the question of how well the grammar produced by the algorithm approximates L. The algorithms we are interested in produce grammars that can approximate, and not necessarily completely cover, the class of referring definite expressions. As a practical quantitative measure, we use the ratio of the correctly classified examples to the overall number of examples from both the development and test sets.

N- total number of examples
N_c^+ number of correctly labeled positive examples
N_c^- number of correctly labeled negative examples
$Q = (N_c^+ + N_c^-)/N$

The goodness of the degree of closeness between the language generated by the inferred grammar and L depends, in part, on the future uses of such grammars. For us, the two main purposes for such grammars are:

1. To facilitate automatic knowledge acquisition,
2. To reduce computational cost of definite anaphora resolution.

For the purpose of automatic knowledge acquisition, a perfect coverage of the grammar is not necessary at all, largely because errors appear to be smoothed by the statistically meaningful thresholds preventing the system from acquiring garbage. It appears that even the 10%-level quantitative performance of a grammar is sufficient because this very imperfect coverage can be made up by a larger number of texts to learn from. Roughly, in order to acquire the same amount of knowledge, a grammar that exhibits an 80%-level performance would require 8 times fewer texts to learn from than a grammar with 10%-level performance.

In contrast, for the purpose of computing referents of referring definite expressions without so-called in-depth processing a high degree of accuracy is necessary. Anything close to or below 50%-level quanti-

tative performance is basically useless[4].

The two usages of the acquired grammars that we plan for are therefore not entirely compatible. In other words, the same inferred grammar can be used for both purposes only provided its coverage is very good.

3.5.2 Deductive Or Inductive Inference

Deductive inference truly captures data seen, without really generalizing the results. Precisely because of this, such algorithms do not learn much; the inferred grammars are basically long disjunctions of the input examples. Inductive algorithms, on the other hand, make generalizations that may or may not be correct. If their gambling strategy happens to be correct, the produce grammars that allow one to predict well future instances.

For example, for the input sets $E+ = \{aaaaa, aa, a\}$ and $E- = \emptyset$, a deductive inference may produce a CFG like $S \rightarrow aaaaa \mid aa \mid a$, and an inductive inference– a CFG like $S \rightarrow aS \mid a$.

3.5.3 Generalization Strategies

We considered the following generalization strategies:

Generalization of terminal to nonterminal

This is a constant to variable kind of generalization whose effect is moving a lower node higher in the grammar graph. For example, a grammar

$S \rightarrow a$
$A \rightarrow a$

becomes

$S \rightarrow A$
$A \rightarrow a$

A grammar

$S \rightarrow she$
$Pron \rightarrow she$

becomes

$S \rightarrow Pron$
$Pron \rightarrow she$

Generalization of lower nonterminal to dominating it nonterminal

This generalization has an effect of attaching a lower node higher in the grammar graph. It may introduce cycles in the grammar graph. For example, a grammar

$A \rightarrow a$
$X \rightarrow A$
$Y \rightarrow X$

becomes

$A \rightarrow a$
$X \rightarrow Y$
$Y \rightarrow X$

A grammar

$Pron \rightarrow she$
$Noun \rightarrow dog$
$I \rightarrow Pron \mid Noun$
$Y \rightarrow she$

becomes

$Pron \rightarrow she$
$Noun \rightarrow dog$
$I \rightarrow Pron \mid Noun$
$Y \rightarrow Pron$

3.5.4 Learning Simple Versus Complex Languages

The same algorithm may or may not work equally well for primitive and complex (derived via some operations) languages. Examples are:

- Primitive language
  $L_{primitive} = a^*$
- Complex language
  $L_{complex} = \{a^n b^n \mid n > 0\} \cup a^*b$

3.5.5 Convergence

Another important issue is whether an algorithm guarantees to converge given the appropriate amount or type of input. It may matter whether the number of examples is finite or infinite. Things could be different for finite and infinite languages to be inferred.

3.5.6 Approximation of Formal Power

If L is a truly unknown formal language, the question arises to which extent can an algorithm approximate L.

3.5.7 Inference with Structural Information

The problem appears to be much easier if positive examples were given along with the structural information such as parse trees or derivations. But is it indeed easier ? And does it make the inference problem trivial ?

3.5.8 Inference With Or Without Negative Examples

Our learning framework is similar to the learning in the limit framework [Gold, 1978], for which it was shown that no infinite language can be learned without negative examples. So the presence of the negative examples makes it possible to learn.

The question arises, however, whether a reasonable approximation can be learned from positive examples only.

3.5.9 Tolerance for Noise, False Positive and Negative Examples

Sensitivity to noise, e.g., the presence of false positive and false negative examples, is another major consideration. We assume that both false positive and false negative examples may occur. In case of

[4] Instead of doing any kind of computation, one may just as well flip a coin and obtain comparable level of performance.

natural language, we are slightly biased toward minimizing the number of false positives because non-referring definite noun phrases may be in an arbitrary relation to the neighboring concepts, computation of such relations may involve a number of expensive inference steps, and as a result, the system may run a danger of acquiring garbage.

3.5.10 Incrementality

An algorithm may or may not be incremental. With more input, it may generate a new grammar by refining the previously learned one, or it may produce an entirely new grammar from scratch.

3.5.11 Order-Dependent Or Order-Independent

The issue of the order-dependency or its lack is closely related to the incrementality issue.

3.5.12 Considering Some, All and Additional Examples

Both the computational cost of inference as well as its quality may be different in case when some input examples are disregarded or when additional examples are obtained. Learning may become less computationally expensive as well as guarantee a better grammar if certain examples are disregarded. These examples to be disregarded may represent situations that are difficult to learn from. It is equally plausible to expect that with a larger number of examples, learning improves. One such source of additional positive examples that are not present in the input are examples that can be generated from an intermediate inferred grammar by utilizing Pumping Lemma for context-free languages [Hopcroft and Ullman, 1979].

3.5.13 All Negative Examples Are Not Equal

For artificial data, we make no assumption about the goodness of negative examples. For natural language data, we are somewhat biased toward more informative negative examples that to some extent represent systematic near-misses.

3.5.14 Character-Based Versus Word-Based Languages

We assume that the same algorithm should learn both character-based and word-based languages.

In case of artificial, character-based languages, each character has a unique corresponding nonterminal symbol which serves as its lexical category.

In case of natural languages such as English, we make an assumption that each lexical item can be labeled (tagged) by one or more nonterminal symbols that represent a finite set of lexical categories. Table 1 gives a complete list of lexical categories utilized in our NLP system. Further subcategorizations of lexical items are possible via features encoded in the dictionary entries. One example of such a subcategorization is the "common" versus "proper" noun distinction. Another example is the "cardinal" versus "ordinal" number. Most of our lexical categories are rather standard[5]. We do not assume the availability of a complete dictionary of the open-class lexical items (nouns, verbs, abbreviations and names); this would be very unrealistic. We do, however, assume the availability of a complete or close to complete dictionary of the closed-class lexical items (all the other lexical categories). This assumption is quite realistic: our own NLP system has such a rather complete dictionary, there are a number of taggers available at various research and development NLP groups.

3.6 Algorithms

Figure 8 contains an artificial context-free grammar that was used to generate test data shown in Figure 1 . It provides a basis for comparison among the outputted grammars.

3.6.1 Algorithm I: General-Purpose

This algorithm assumes that elements of $E+$ are correct (no noise); no assumption is made about false negatives, i.e., mislabeled positive examples. No assumptions are drawn about strings which belong to both $E+$ and $E-$.

The algorithm applies the following generalization strategy: a single occurence of a character is generalized to allow for this character to appear an arbitrary number of times. This aggressive generalization may produce very permissive, too broadly covering grammars.

This strategy of updating a current grammar is biased toward producing grammars with deep, but not bushy trees.

The algorithm is order-sensitive in that the order of occurences of different characters matters.

This algorithm is inductive– a single occurrence of a character leads to an inductive conclusion that any number of these characters is allowed. Its deductive aspect lies in its inability to predict other complex new patterns such as equality of certain types of symbols. This algorithm has a generic flavor to it, but it appears to be blind to those other complex patterns beyond fixed partial order of certain lexical items. As a consequence, if distinguishing $E+$ and $E-$ depends on some sort of complex pattern, then this algorithm cannot produce very good approximations of the language to be learned

In **Step 0**, a trivial grammar initialization is performed;

In **Step 1**, all examples are kept and no additional examples are provided. However, the algorithm does not make any attempt to modify the resulting grammar based upon the elements of $E-$. The negative

[5]Although lexical item classification may or may not be standard.

$R \rightarrow Det\ (Pre\_Modfs)\ Head\ (Post\_Modfs)$
$Det \rightarrow a \mid b \mid c$
$Pre\_Modfs \rightarrow e \mid f \mid Pre\_Modfse \mid Pre\_Modfs\ f$
$Head \rightarrow h$
$Post\_Modfs \rightarrow Prep\ NP$
$Prep \rightarrow m \mid n \mid o$
$NP \rightarrow e \mid f \mid NP\ e \mid NP\ f$

Figure 8: An artificial context-free grammar that was used to generate test data for generic grammatical inference algorithms.

$V = \{S, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$
S start symbol
$\Sigma = \{a, e, h, b, f, m, n, o\}$
$P = \{S \rightarrow P_1 \mid P_4$
$P_1 \rightarrow a \mid aP_1 \mid aP_2,\ P_2 \rightarrow e \mid eP_1 \mid eP_3,\ P_3 \rightarrow h \mid hP_3$
$P_4 \rightarrow b \mid bP_4 \mid bP_5,\ P_5 \rightarrow f \mid fP_5 \mid h \mid hP_5 \mid e \mid eP_5 \mid hP_6,\ P_6 \rightarrow m \mid mP_6 \mid e \mid eP_6 \mid h \mid hP_6 \mid hP_7,\ P_7 \rightarrow n \mid nP_7 \mid h \mid hP_7 \mid hP_8,\ P_8 \rightarrow o \mid oP_8 \mid f \mid fP_8 \mid e \mid eP_8 \mid h \mid hP_8\}$

Figure 9: A context-free grammar G_I^1 produced by Algorithm I for the artificial, character-based language whose positive and negative examples are shown in Figure 1.

examples are used only to evaluate the quantitative performance of the grammar.

In **Step 2**, as each character a_i of the currently considered example e_i is read, the algorithm checks if a_i is already in the set of the terminals Σ:

if a_i is in Σ, **then if** a_i is the first character in e_i, **then** the algorithm checks **if** there is a derivation from the start symbol $S \rightarrow a_i$; **if** there is, do nothing; else two new productions are created:

$S \rightarrow P_{i+1}$

The production whose left-hand-side is P_{i+1} is updated to include one more right-hand-side a_iP_{i+1}

if a_i is not the first character in e_i

then the algorithm examines the productions whose left-hand-side is P_i:

if P_i derives a_i, **then** do nothing

if P_i does not derive a_i, **then** a new production

$P_{i+1} \rightarrow a_i \mid a_iP_{i+1}$ is created;

if $a_i \notin \Sigma$, **then** a_i is added to Σ:

$\Sigma = \Sigma \cup \{a_i\}$;

a new production $P_{i+1} \rightarrow a_i \mid a_iP_{i+1}$ is added to P;

if a_i is the first character in e_i, **then** a new production $S \rightarrow P_{i+1}$ is added to P; else the productions with P_i as their left-hand-sides are updated to include the right-hand-side $a_{i-1}P_{i+1}$

if a_i is not the first character in e_i, **then** a new production $P_{i+1} \rightarrow a_i \mid a_iP_{i+1}$ is added to P.

Example 1

Let us illustrate the algorithm for the artificial language shown in Figure 1 for the first two positive examples $e_1 = aeh$ and $e_2 = bhhf$. We begin with the following initial grammar G_0:

$V = \{S\}$, $\Sigma = \emptyset$, $P = \emptyset$

The counter i, used for subscripting productions, is initialized to 0.

The first character $a_1 = a$. We first check **if** $a_1 \in \Sigma$; it is not, so the character "a" is added to Σ:

$\Sigma = \Sigma \cup \{a\}$;

Two productions $S \rightarrow P_1$ and $P_1 \rightarrow a \mid aP_1$ are added to P; $i = i + 1$;

The current grammar G_i is:

$V = \{S, P_1\}$

$\Sigma = \{a\}$

$P = \{S \rightarrow P_1,\ P_1 \rightarrow a \mid aP_1\}$

The next character in $e_1 = aeh$, $a_2 = e$. The algorithm first checks if $e \in \Sigma$; it is not, so this character updates Σ:

$\Sigma = \Sigma \cup \{e\} = \{a, e\}$;

Two new productions are added to P:

$V = \{S, P_1, P_2\}$

$\Sigma = \{a, e\}$

$P = \{S \rightarrow P_1,\ P_1 \rightarrow a \mid aP_1\} \cup \{P_2 \rightarrow e \mid eP_2,\ P_1 \rightarrow aP_2\} =$

$\{S \rightarrow P_1,\ P_1 \rightarrow a \mid aP_1 \mid aP_2,\ P_2 \rightarrow e \mid eP_2\}$

After processing the third character $a_3 = h$, the current grammar G_1 is:

$V = \{S, P_1, P_2, P_3\}$

$\Sigma = \{a, e, h\}$

$P = \{S \rightarrow P_1,\ P_1 \rightarrow a \mid aP_1 \mid aP_2,\ P_2 \rightarrow e \mid eP_2 \mid eP_3,\ P_3 \rightarrow h \mid hP_3\}$

The second positive example is $e_2 = bffh$. Its first character $a_1 = b$. The algorithm first checks if $b \in \Sigma$;

it is not, so $\Sigma = \Sigma \cup \{b\}$;

Three new productions $S \rightarrow P_4$ and $P_4 \rightarrow b \mid bP_4$ are added to P:

$V = \{S, P_1, P_2, P_3, P_4\}$

$\Sigma = \{a, e, h, b\}$
$P = \{S \rightarrow P_1 \mid P_4,\ P_1 \rightarrow a \mid aP_1 \mid aP_2,\ P_2 \rightarrow e \mid eP_2 \mid eP_3,\ P_3 \rightarrow h \mid hP_3,\ P_4 \rightarrow b \mid bP_4\}$

The next character $a_2 = f$ is not in Σ, so it gets updated to $\Sigma = \{a, e, h, b, f\}$;
Two new productions $P_5 \rightarrow h \mid hP_5$ are added to P:
$V = \{S, P_1, P_2, P_3, P_4, P_5\}$
$P = \{S \rightarrow P_1 \mid P_4,\ P_1 \rightarrow a \mid aP_1 \mid aP_2,\ P_2 \rightarrow e \mid eP_2 \mid eP_3,\ P_3 \rightarrow h \mid hP_3,\ P_4 \rightarrow b \mid bP_4 \mid bP_5,\ P_5 \rightarrow f \mid fP_5\}$

The next character $a_3 = f$ and Σ does not change. P also does not change because a_3 can be derived from the most recently introduced nonterminal P_5;

Finally, the last character $a_4 = h$ is processed; again, Σ does not change, but P does because the most recently introduced nonterminal P_5 does not derive $a_4 = h$:
$V = \{S, P_1, P_2, P_3, P_4, P_5, P_6\}$
$P = \{S \rightarrow P_1 \mid P_4,\ P_1 \rightarrow a \mid aP_1 \mid aP_2,\ P_2 \rightarrow e \mid eP_2 \mid eP_3,\ P_3 \rightarrow h \mid hP_3,\ P_4 \rightarrow b \mid bP_4 \mid bP_5,\ P_5 \rightarrow f \mid fP_5 \mid fP_6,\ P_6 \rightarrow h \mid hP_6\}$

Example 2

Let us illustrate the algorithm for the following
$E+$ = {(*the man in black*), ,(*the green vase*), (*the tall man*)}
Our initial grammar G_0 is:
$V = \{S\}$, $\Sigma = \emptyset$, $P = \emptyset$.
The first positive example is
$e_1 =$ (*the man in black*).
We examine its first element (a word and not a character as in the artificial language example above) $a_1 = the$. We first examine Σ to see if $a_1 \in \Sigma$; it is not and $\Sigma = \Sigma \cup \{the\}$.
Three new productions $S \rightarrow P_1$ and $P_1 \rightarrow the \mid theP_1$ are added to P:
The next word in e_1 is $a_2 = man$. The algorithm first checks if $man \in \Sigma$; it is not, and this word updates Σ:
$\Sigma = \Sigma \cup \{man\} = \{the,\ man\}$.
Three new productions are added to P:
$V = \{S, P_1, P_2\}$
$\Sigma = \{the, man\}$
$P = \{S \rightarrow P_1,\ P_1 \rightarrow the \mid the\ P_1\ \} \cup \{P_2 \rightarrow man \mid man\ P_2,\ P_1 \rightarrow the\ P_2\} =$
$\{S \rightarrow P_1,\ P_1 \rightarrow the \mid the\ P_1 \mid the\ P_2,\ P_2 \rightarrow man \mid man\ P_2\}$

The next word $a_3 = in$. The algorithm first checks if $in \in \Sigma$; it is not, so this element updates Σ:
$\Sigma = \Sigma \cup \{in\} = \{the, man, in\}$.
Three new productions are added to P:
$V = \{S, P_1, P_2, P_3\}$
$\Sigma = \{the, man, in\}$
$P = \{S \rightarrow P_1,\ P_1 \rightarrow the \mid the\ P_1 \mid the\ P_2,\ P_2 \rightarrow man \mid man\ P_2\} \cup \{P_3 \rightarrow in \mid in\ P_3,\ P_2 \rightarrow man\ P_3,\ \} =$
$\{S \rightarrow P_1,\ P_1 \rightarrow the \mid the\ P_1 \mid the\ P_2,\ P_2 \rightarrow man \mid man\ P_2 \mid man\ P_3,\ P_3 \rightarrow in \mid in\ P_3\}$

Similarly, for $a_4 = black$, the algorithm first checks if $black \in \Sigma$; it is not, so this character updates Σ:
$\Sigma = \Sigma \cup \{black\} = \{the,\ man,\ in,\ black\}$.
Three new productions are added to P and the new non-terminal P_4 updates V:
$V = V \cup \{P_4\} = \{S, P_1, P_2, P_3, P_4\}$:
$V = \{S, P_1, P_2, P_3, P_4\}$
$\Sigma = \{the,\ man,\ in,\ black\}$
$P = \{S \rightarrow P_1,\ P_1 \rightarrow the \mid the\ P_1 \mid the\ P_2,\ P_2 \rightarrow man \mid man\ P_2,\ P_3 \rightarrow in \mid in\ P_3\} \cup \{P_4 \rightarrow black \mid black\ P_4,\ P_3 \rightarrow inP_4$
$\} =$
$\{S \rightarrow P_1,$
$P_1 \rightarrow the \mid the\ P_1 \mid the\ P_2,\ P_2 \rightarrow man \mid man\ P_2 \mid man\ P_3,\ P_3 \rightarrow in \mid in\ P_3 \mid P_4,\ P_4 \rightarrow black \mid black\ P_4\}$

After examining $e_2 =$ (*the green vase*), our grammar is:
$V = \{S, P_1, P_2, P_3, P_4, P_5, P_6\}$
$\Sigma = \{the,\ man,\ in,\ black,\ green,\ vase\}$
$P = \{S \rightarrow P_1,\ P_1 \rightarrow the \mid the\ P_1 \mid the\ P_2 \mid the\ P_5,\ P_2 \rightarrow man \mid man\ P_2 \mid man\ P_3,\ P_3 \rightarrow in \mid in\ P_3 \mid in\ P_4,\ P_4 \rightarrow black \mid black\ P_4,\ P_5 \rightarrow green \mid green\ P_5 \mid green\ P_6,\ P_6 \rightarrow vase \mid vase\ P_6\}$

Upon examination of $e_3 =$ *the tall man*, $a_2 = tall$ is handled as described previously. We find that, while $a_3 = man$ already exists in Σ, it is not directly reachable from the current production P_6 at the time it is encountered, so two new productions
$P_7 \rightarrow man$
and
$P_7 \rightarrow man\ P_7$
are created and added to P:
$V = \{S, P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$
$\Sigma = \{the, man, in, black, green, vase, tall\}$
$P = \{S \rightarrow P_1 \mid P_4 \mid P_6,\ P_1 \rightarrow the \mid the\ P_1 \mid the\ P_2 \mid the\ P_5,\ P_2 \rightarrow man \mid man\ P_2 \mid man\ P_3,\ P_3 \rightarrow in \mid in,\ P_4 \rightarrow black\ black\ P_4,\ P_5 \rightarrow green \mid green\ P_5 \mid green\ P_6,\ P_6 \rightarrow vase \mid vase\ P_6,\ P_7 \rightarrow tall \mid tall\ P_6 \mid man \mid man\ P_7\}$

For the first example in $E+$ in Figure 7, examples collected from the sample article in Time Magazine, Figure 3, this algorithm produces the following grammar:
$V = \{S, P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}\}$
Σ = {*the, gang, show, two, transition, to, TV, Georgia, native, boys*}
$P = \{S \rightarrow P_1,\ P_1 \rightarrow the \mid the\ P_2 \mid the\ P_3 \mid\ P_4 \mid the\ P_5 \mid the\ P_8 \mid the\ P_{10},\ P_2 \rightarrow gang \mid gang\ P_2,\ P_3 \rightarrow show \mid show\ P_3,\ P_4 \rightarrow two \mid two\ P_4,\ P_5 \rightarrow transition \mid transition\ P_5\ transition\ P_6,\ P_6 \rightarrow to \mid to\ P_6 \mid to\ P_7,\ P_7 \rightarrow TV \mid TV\ P_7,\ P_8 \rightarrow Georgia \mid Georgia\ P_8 \mid Georgia\ P_9,\ P_9 \rightarrow native \mid native\ P_9,\ P_{10} \rightarrow boys\}$

3.6.2 Algorithm II

As in the case of Algorithm I, this algorithm does not utilize negative examples for learning. We produce a context-free grammar representing a given language by examining only the elements of $E+$.

G^1_{II} (from $Ed+\cup Ei+$):
Σ = {*the, venture, share, Delaware – based, E.I., DuPont remaining, 40, percent, firm, firm's, operating, region, officials, Osaka – based*}
V = {*S, DET, PREMOD, HEAD, POSTMOD*}
P = {
$S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$HEAD \rightarrow venture \mid share \mid DuPont \mid percent \mid firm \mid region \mid officials$
$PREMOD \rightarrow Delaware - based\ E.I. \mid remaining\ 40 \mid firm's\ operating \mid Osaka - based$
$POSTMOD \rightarrow \epsilon$
}

Figure 10: A context-free grammar G^1_{II} produced by Algorithm II for the positive and negative examples obtained first the first sample text shown in Figure 2.

G^2_{II} (from $Ed+\cup Ei+$):
Σ = {*the, gang, show, two, transition, to, TV, Georgia, native, boys*}
V = {*S, DET, PREMOD, HEAD, POSTMOD*}
P = {
$S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$HEAD \rightarrow gang \mid show \mid two \mid transition \mid native \mid boys$
$PREMOD \rightarrow Georgia$
$POSTMOD \rightarrow to\ TV$
}

Figure 11: A context-free grammar G^2_{II} produced by Algorithm II for the positive and negative examples obtained from the second sample text shown in Figure 3.

An inherent bias consists of a particular skeleton grammar G^{ske} shown below that the algorithm begins with. This bias means that the algorithm does not learn from scratch, but instead learning is refinement of this initial grammar.

Σ = {*the*}
V = {*S, DET, NP, HEAD, PREMOD, POSTMOD*}
P = {
$S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$PREMOD \rightarrow \epsilon$
$POSTMOD \rightarrow \epsilon$
}

The only "knowledge" of the English language this algorithm requires is the ability to recognize nouns, in order to identify the determiner and the head h_i; the other elements of V_i are lumped together as either premodifiers or postmodifiers with no further breakdown.

In **Step 0**, a trivial grammar initialization is performed; the G_0 becomes G^{ske};

In **Step 1**, all examples are kept, no additional examples provided. Only positive examples drive the learning process.

In **Step 2**, for the currently considered e_i, after reading the determiner "the", the algorithm searches for the first word identifiable as a noun, which it will identify as the head h_i; the string of all words between the determiner "the" and h_i are held in a buffer, the contents of which will be called pre_i and will be treated as a single unit which is a premodifier; similarly all words in e_i subsequent to h_i to the end of the string will be treated as a single postmodifier $post_i$, as described later.

If h_i is not in the set of terminals Σ, **then** h_i is added to Σ and a new production $HEAD \rightarrow h_i$ is created and added to P. **If** $h_i \in \Sigma$, **then** the algorithm checks to see if h_i is reachable in the production with $HEAD$ as its left-hand-side; (h_i may appear in a production other than the one with $HEAD$ on the left-hand-side); **if** it is, **then** do nothing, **else** a new production $HEAD \rightarrow h_i$ is created and added to P.

We now examine the buffer holding all words between the determiner and h_i which we will call pre_i. The algorithm first checks each a_i in pre_i is examined to determine $a_i \in \Sigma$. **If** $a_i \in \Sigma$, **then** do nothing, **else** a_i updates Σ. **If** pre_i is not empty, **then** a new production $PREMOD \rightarrow pre_i$ added to P.

We now examine the string of words which follow h_i to the end of e_i, which we will call $post_i$. The

algorithm first checks if each a_i in $post_i$ is also in Σ: **if** $a_i \in \Sigma$. **then** do nothing, else a_i updates Σ: **If** $post_i$ rewrites to ϵ, **then** a new production $POSTMOD \rightarrow post_i$ is added to P.

Example 3

Suppose $e_1 \in E+$ is *the lovely green vase*. We begin with the initial grammar $G_0 = G^{ske}$:
$\Sigma = \{the\}$
V =
$\{S,\ DET,\ NP,\ HEAD,\ PREMOD,\ POSTMOD\}$
$P = \{S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$PREMOD \rightarrow \epsilon$
$POSTMOD \rightarrow \epsilon$
$\}$

The algorithm will read the determiner "the", **then** search for the first noun which will be designated as the head. In this case $h_i = vase$. We **then** check to see if $vase \in \Sigma$. It is not, so Σ will be updated: $\Sigma \cup \{vase\} = \{the, vase\}$, A new production $HEAD \rightarrow vase$ will be added to P:
$\Sigma = \{the, vase\}$
V =
$\{S,\ DET,\ NP,\ HEAD,\ PREMOD,\ POSTMOD\}$
$P = \{S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$HEAD \rightarrow \epsilon$
$PREMOD \rightarrow \epsilon$
$POSTMOD \rightarrow \epsilon$
$\}$
$\cup$
$\{HEAD \rightarrow vase$
$\} =$
$\{S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$HEAD \rightarrow vase$
$PREMOD \rightarrow \epsilon$
$POSTMOD \rightarrow \epsilon$
$\}$
All words encountered between "the" and "vase" are held in a buffer. The contents of that buffer will now be examined as the string $pre_1 = lovely\ green$. First, the algorithm will check Σ for the each a_i in pre_1, updating Σ: $a_1 = lovely$ is not in Σ, so Σ will be updated: $\Sigma \cup \{lovely\} = \{the, vase, lovely\}$. Next, the algorithm will check to see if $a_2 = green$ is in Σ, it is not, so Σ will be updated: $\Sigma \cup \{green\} = \{the,\ vase,\ lovely,\ green\}$, A new production
$PREMOD \rightarrow lovely\ green$
is added to P:
$\Sigma = \{the,\ vase,\ lovely,\ green\}$
V =
$\{S,\ DET,\ NP,\ HEAD,\ PREMOD,\ POSTMOD\}$
$P = \{S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$HEAD \rightarrow vase$
$PREMOD \rightarrow \epsilon$
$POSTMOD \rightarrow \epsilon$
$\}$
$\cup$
$\{PREMOD \rightarrow lovely\ green$
$\} =$
$\{DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$HEAD \rightarrow vase$
$PREMOD \rightarrow lovely\ green$
$POSTMOD \rightarrow \epsilon$
$\}$
As e_1 ends with the head, i.e., there are no postmodifiers, the grammar does not change.

Suppose e_2 in $E+$ is *the man who came to dinner*. Again, the algorithm will read the determiner, **then** search for the first noun h_2, which is *man*. We **then** check to see if $man \in \Sigma$. It is not, so Σ will be updated: $\Sigma \cup \{man\} = \{the, vase, lovely, green, man\}$, A new production $HEAD \rightarrow man$ will be added to P:
$\Sigma = \{the,\ vase,\ lovely,\ green,\ man\}$
V =
$\{S,\ DET,\ NP,\ HEAD,\ PREMOD,\ POSTMOD\}$
$P = \{S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$HEAD \rightarrow vase$
$PREMOD \rightarrow lovely\ green$
$POSTMOD \rightarrow \epsilon$
$\}$
$\cup$
$\{HEAD \rightarrow man$
$\} =$
$\{DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ (POSTMOD)$
$HEAD \rightarrow vase \mid man$
$PREMOD \rightarrow lovely\ green$
$POSTMOD \rightarrow \epsilon$
$\}$

As h_2 was encountered immediately after the determiner, the string pre_2 rewrites to ϵ, so no modifications are made. There are, however, postmodifiers, words following h_2: all words encountered between h_2 and the end of e_2 will be treated as a single string that the nonterminal $POSTMOD$ can generate.

As with the premodifier, the algorithm checks if $a_3 = who \in \Sigma$. It is not, so a_3 updates Σ:
$\Sigma \cup \{who\} =$
$\{the,\ vase,\ lovely,\ green,\ man,\ who\}$.
Similarly the remaining words to the end of the string are checked and Σ updated as needed. After each word in the postmodifier has been checked, a new production
$POSTMOD \rightarrow$ *who came to dinner*

is added to P:
Σ =
{*the, vase, lovely, green, man, who, came, to, dinner*}
$V = \{S,\ DET,\ PREMOD,\ HEAD,\ POSTMOD\}$
$P = \{S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ POSTMOD$
$HEAD \rightarrow vase \mid man$
$PREMOD \rightarrow lovely\ green$
$POSTMOD \rightarrow \epsilon$
}
∪
{$POSTMOD \rightarrow$ *who came to dinner*
} = {$S \rightarrow DET\ NP$
$DET \rightarrow the$
$NP \rightarrow (PREMOD)\ HEAD\ POSTMOD$
$HEAD \rightarrow vase \mid man$
$PREMOD \rightarrow lovely\ green$
$POSTMOD \rightarrow$ *who came to dinner*
}

This algorithm does not generalize. If there is some sort of underlying pattern which typifies the grammar for $E+$, this algorithm would not be able identify it because it considers a particular class of grammars.

3.6.3 Algorithm III: Natural Language-Specific

Where the algorithm which produces G_I^2 from the previous section relies on a pre-specified shell, this next algorithm is more inductive, starting with an initial production which give the most abstract form of a definite noun phrase: $S \rightarrow NP$.

The rest of the grammar is built inductively, using a lexical recognizer which recognizes the lexical categories shown in Table 1.

In **Step 0**, a trivial grammar initialization is performed.

In **Step 1**, all examples are kept, no additional examples provided. Only positive examples drive the learning process.

In **Step 2**, for the currently considered e_i, the algorithm examines each succeeding a_i as encountered. As in the previous algorithm, a_i is a word rather than a character.

If $a_i \in \Sigma$, **then** do nothing; **else** $\Sigma = \Sigma \cup \{a_i\}$.

Next the lexical category V_i^{lex} of a_i is determined using the lexical recognizers R_i^{lex}.

If $V_i^{lex} \in V$ **then** do nothing; **else** $V = V \cup \{V_i^{lex}\}$

A new production $V_i^{lex} \rightarrow a_i$ is added to P.

If $i = 1$, i.e., a_i is the first word in e_i, **then** a new production $S \rightarrow V^{lex}$ is added to P; **else** a new production $V_{i-1}^{lex} \rightarrow V_i^{lex}$ is added to P.

Example 4

Let $E+$ = {(*the lovely green vase*), (*the man who came to dinner*)}

In **Step 0**, a trivial grammar initialization is performed;

In **Step 1**, all examples are kept, no additional examples are provided. Only positive examples drive the learning.

In **Step 3**, we examine the first positive example $e_1 =$ *the lovely green vase*. The algorithm first checks if $a_1 = the \in \Sigma$. It is not, so a_1 updates Σ: $\Sigma \cup \{the\} = \{the\}$. **Then** the lexical recognizer determines that *the* is of the lexical category *determiner*, V^{det}. The algorithm **then** checks if $V^{det} \in V$; it is not, so $V \cup \{V^{det}\} = \{S, V^{det}\}$. A new production $V^{det} \rightarrow the$ is added to P.

Since a_1 is the first word in e_1, a second new production $S \rightarrow V^{det}$ is added to P as well:
$\Sigma = \{the\}$
$V = \{S, V^{det}\}$
$P = \emptyset \cup \{V^{det} \rightarrow the,\ S \rightarrow V^{det}\} =$
$\{S \rightarrow V^{det},\ V^{det} \rightarrow the\}$

Next the algorithm checks if $a_2 = lovely \in \Sigma$. It is not, so a_2 updates Σ: $\Sigma \cup \{lovely\} = \{the,\ lovely\}$. **Then** the lexical recognizer determines that *lovely* is of the lexical type *adjective*, V^{adj}. The algorithm **then** check to see if $V^{adj} \in V$; it is not, so $V \cup \{V^{adj}\} = \{S, V^{det}, V^{adj}\}$. Two new productions:
$V^{adj} \rightarrow lovely$
$V^{det} \rightarrow V^{adj}$
are added to P: $\Sigma = \{the, lovely\}$
$V = \{S, V^{det}, V^{adj}\}$
$P = \{S \rightarrow V^{det},\ V^{det} \rightarrow the\} \cup \{V^{adj} \rightarrow lovely,\ V^{det} \rightarrow V^{adj}\} =$
$\{S \rightarrow V^{det},\ V^{det} \rightarrow the\ V^{adj},\ V^{adj} \rightarrow lovely\}$

Similarly the algorithm checks if $a_3 \in \Sigma$. It is not, so a_3 updates Σ: $\Sigma \cup \{green\} = \{the,\ lovely,\ green\}$. **Then** the lexical recognizer determines that *green* is of the lexical type *adjective*, V^{adj}. The algorithm **then** check to see if $V^{adj} \in V$; it is, so V is not updated. A new production $V^{adj} \rightarrow green$ is added to P: $\Sigma = \{the, lovely, green\}$
$V = \{S, V^{det}, V^{adj}\}$
$P = \{S \rightarrow V^{det},\ V^{det} \rightarrow the,\ V^{adj} \rightarrow lovely\} \cup \{V^{adj} \rightarrow green\} =$
$\{S \rightarrow V^{det},\ V^{det} \rightarrow the,\ V^{adj} \rightarrow lovely \mid green\}$

Lastly the algorithm checks if $a_4 = vase \in \Sigma$. It is not, so a_4 updates Σ: $\Sigma \cup \{vase\} = \{the,\ lovely,\ green,\ vase\}$. **Then** the lexical recognizer determines that *vase* is of the lexical type *noun*, V^{noun}. The algorithm **then** check to see if $V^{noun} \in V$; it is not, so V is updated: $V \cup \{V^{noun}\} = \{V^{det},\ V^{adj},\ V^{noun}\}$. two new productions:
$V^{noun} \rightarrow vase$
$V^{adj} \rightarrow V^{noun}$
are added to P:
$\Sigma = \{the, lovely, green, vase\}$
$V = \{S, V^{det}, V^{adj}, V^{noun}\}$
$P = \{S \rightarrow V^{det},\ V^{det} \rightarrow the,\ V^{adj} \rightarrow lovely \mid green\}$
$\{V^{noun} \rightarrow vase,\ V^{adj} \rightarrow V^{noun}\} =$

Σ = {*the*, *venture*, *share*, *Delaware* – *based*, *E.*, *I.*, *DuPont*, *remaining*, 40, *percent*, *firm*, *firm's*, *operating*, *region*, *officials*, *Osaka* – *based*}
$V = \{S, V^{det}, V^{noun}, V^{adj}, V^{abbr}, V^{number}\}$
$P = \{S \rightarrow V^{d}et \mid V^{noun} \mid V^{adj}$
$V^{det} \rightarrow the \mid V^{noun} \mid V^{adj}$
$V^{noun} \rightarrow venture \mid share \mid DuPont \mid percent \mid firm \mid region \mid officials$
$V^{adj} \rightarrow Delaware - based \mid V^{abbr} \mid remaining \mid V^{number} \mid firm's \mid operating$
$V^{abbr} \rightarrow E. \mid I. \mid V^{noun} V^{number} \rightarrow 40 \mid V^{noun}$
}

Figure 12: A context-free grammar G^1_{III} produced by Algorithm III for the positive and negative examples obtained from the first sample text shown in Figure 3.

$\Sigma = \{the, gang, show, two, transition, to, TV, Georgia, native, boys\}$
$V = \{S, V^{det}, V^{noun}, V^{number}, V^{prep}\}$
$P = \{S \rightarrow V^{det}$
$V^{det} \rightarrow the \mid V^{noun} \mid V^{adj}$
$V^{noun} \rightarrow gang \mid show \mid V^{number} \mid transition \mid V^{prep} \mid TV \mid Georgia \mid native \mid boys$
$V^{number} \rightarrow two$
$V^{prep} \rightarrow to \mid V^{noun}$
}

Figure 13: A context-free grammar G^2_{III} produced by Algorithm III for the positive and negative examples obtained from the second sample text shown in Figure 3.

$\{S \rightarrow V^{det}$, $V^{det} \rightarrow the$, $V^{adj} \rightarrow lovely \mid green \mid V^{noun}$, $V^{noun} \rightarrow vase\}$

The second string in $E+$ = e_2 = *the man who came to dinner* is processed in a similar fashion. When finished, G_0 is: Σ = {*the*, *lovely*, *green*, *vase*, *man*, *who*, *came*, *to*, *dinner*}
$V = \{S, V^{det}, V^{adj}, V^{noun}, V^{pron}, V^{verb}, V^{prep}\}$
$P = \{S \rightarrow NP$, $V^{det} \rightarrow the \mid V^{adj} \mid V^{noun}$, $V^{adj} \rightarrow lovely \mid green$, $V^{noun} \rightarrow vase \mid man \mid V^{pron} \mid dinner$, $V^{pron} \rightarrow who \mid V^{verb}$, $V^{verb} \rightarrow came \mid V^{prep}$, $V^{prep} \rightarrow to \mid V^{noun}\}$

References

[MUC, 1995] (1995). *Proceedings of MUC-6, Sixth Message Understanding Conference.* Morgan Kaufmann Publishers, Inc.

[Gold, 1978] Gold, M. (1978). Complexity of Automaton Identification from Given Data. *Information and Control*, 37:302–320.

[Guha and Lenat, 1990] Guha, R. V. and Lenat, D. B. (1990). CYC: A Mid-Term Report. MCC Technical Report ACT-CYC-134-90, Microelectronics and Computer Technology Corporation, ACT Program, Artificial Intelligence Lab.

[Hobbs et al., 1988] Hobbs, J., Stickel, M., Martin, P., and Edwards, D. (1988). Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of the ACL*, pages 95–103.

[Hobbs, 1979] Hobbs, J. R. (1979). Coherence and coreference. *Cognitive Science*, 3:67–90.

[Hopcroft and Ullman, 1979] Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation.* The Addison-Wesley Publishing Company.

[Iwańska, 1991] Iwańska, Ł. (1991). Discourse Processing Module. Technical Report 91CRD180, Artificial Intelligence Laboratory, GE Research and Development Center.

[Schank and Riesbeck, 1981] Schank, R. C. and Riesbeck, C. K. (1981). *Inside computer understanding.* Lawrence Erlbaum Associates.