A Proposal of a Methodology to Acquire Syntactic Rules Gradually by Inductive Logic Programming

Hiroyuki Kameda
School of Computer Science
Tokyo University of Technology
kameda@stf.teu.ac.jp

Saori Aida
School of Computer Science
Tokyo University of Technology
aidasor@stf.teu.ac.jp

Abstract

A new methodology to acquire syntactic rules gradually by applying deduction to grammatical sentence generation and induction to discover a more general and more compact set of syntactic rules. Induction process is implemented by Prolog interpreter system, and inductive process is implemented by an Inductive Logic Programming system Aleph. Some grammar acquisition experiments were done and considered feasibility, validity and limitation of the proposed methodology.

Key Words- Grammar acquisition, Inductive logic programming, Grammar acquisition experiments

1. Introduction

Every living life on Earth, in general, reacts and behaves as adaptively as possible to various kind of stimuli which come into and from their own body. Especially, human not only reacts and behaves adaptively, but also do much more intelligently even by recognizing and acquiring highly abstract concepts. As a result, human being has come to the evolutionary stage of using sophisticated languages, and now is thought as being at the top rank in the primates.

A natural language is one of important tools for us, human beings, to describe, keep, and transmit various information, concepts, to think, e.g., to infer logically, and then to communicate with each other.

From this point of view, elucidation of principles and mechanism is really important to understand our Human, i.e., the essence that makes human the human. At the same time, if a natural language is a tool that makes us social living things, then it is meaningful that robots and virtual agents are able to communicate like humans, so that they and human can communicate with each other smoothly.

On this belief, we have been challenging to elucidate the essence of language ("la langue" and "le language" in French). Author Kameda mainly is engaged in natural language processing from various sides of cognitive psychology, applied linguistics, and logics [1]. Another author Aida is one of young researchers who has been engaged in s3D (stereoscopic 3Dimensional) visual psychology, and has already found new phenomena of s3D [2]. Out of these studies, this short paper reports one result of our study on grammar acquisition.

More concretely to describe, we propose a syntactic rule acquisition methodology. At first, a small and simple set of syntactic rules is prepared as a seed, to creates a set of grammatically correct sentences. Then a new set of syntactic rules, as the next seed, is created with use of Inductive Logic Programming method. This procedure will be repeated again and again. As this procedure, it is expected that a more adequate and compact set of syntactic rules is created finally.

In this paper, we at first, in the chapter 2, describe related works historically, in chapter 3. technical terms are defined with some explanation for readers to understand this paper more easily. In the chapter 4, experiments are reported based on proposed methodology, and in the chapter 5. we conclude our study.

2. Related works (Historical review)

As readers would know, grammar acquisition has been studied for a really long time. One of famous research is the grammar research on Sanskrit language by an Indian linguist Panini found in the 4th century B.C. In modern age, Saussure's structuralism and Chomsky's generative grammar are well known. Especially Chomsky's theory of grammar established a rigid foundation of linguistic research [3]. Dependency theory by French linguist Tesnière and thoughts on language of Humboldt also gave an impact on grammar study to be done more precisely.

These studies, however, focused mainly on, for example, "what is language?" or "How can we describe language in more explicit way?" Indeed they contribute research on language, grammar and etc., but cannot clarify a methodology how we can grasp the essence of languages, and describe a system of a targeted language in concrete way.

On the other hand, as computer technology emerged in the 20th century develops, many programming languages

were studied and proposed. These studies gave much impacts on natural language research, and at the same time programming languages Snob, Lisp and Prolog and etc. gave also much impact on linguistic research. One of remarkable research which gave impact on natural language processing is Inductive Logic Programming proposed by Muggleton [4]. He implemented a new system called Prolog which has ability to infer inductively based on a background knowledge.

By this machine learning system, it may be feasible to discover a new rigid grammar based on background knowledge, i.e., the knowledge accumulated by researchers for a long time.

Another important point is that a natural language is a too complex system for human to grasp. This implies that systems engineering should be applied in the future.

By considering these facts, we have come to an idea of grammar acquisition methodology with use of inductive logic programming.

3. Our underlying basic ideas and definition of technical terms

In this chapter, we describe some technical terms and ideas to understand this paper more easily. Before we mention them, the research goal of this paper will be given first.

3.1. Our understanding of the present situation and our research goal

- Understanding of the present situation of grammar research: There are much amounts of fragments of knowledge on grammar. They are well-known and can be described explicitly. But a grammar is a too extremely complex system for us to understand it explicitly and holistically. Therefore, we have immense difficulty in design and implement knowledge on grammar systematically. If we could do that, it is unclear that acquired grammar is really optimized as a system from a view of systems engineering.
- Our research goal to reach: To get an optimized, i.e., compact & inconsistent system of grammar.

  In order to achieve our research goal, we propose a new methodology of acquiring a set of syntactic rules by inductive logic programming.

3.2. Definition of technical terms

- Chomsky's grammar: Grammar is a set of four things <Vn, Vt, s, P>;

  Vn: a set of non-terminal symbols,
  Vt: a set of terminal symbols, or vocabulary (a set of words)
  s: start symbol, from which sentences are derived or produced.
  P: a set of production rules or rewriting rules.

These terms are defined and used in general in choosy' school.

- abstract language: In Chomsky's grammar frame, Vt and s are defined concretely, but Vn and P are not yet defined concretely.
- concrete language: a set of all sentences which are grammatical correct from the view of a certain grammar system.
- seed grammar: an initial set of grammar. This is used as an initial value of gradual grammar acquisition method.

3.3. Our underlying basic ideas

(1) our research assumption:

Chomsky says:

- There exist words at the beginning (a belief).
- Sequence of words is called a sentence (a definition).
- sentences are grouped into two categories: grammatically correct ones and grammatically incorrect ones.
- There exists a set of rules which generate and accept grammatically correct sentences exactly (assumption).
- The set of rules can be described in the frame of Chomsky's well-known formal grammar (assumption).

Our comments:

- Language processing is done by human.
- Language processing ability is equal to symbol processing ability.
- This ability is implemented on computer systems.
- Computer system consists of hardware and software.
- Software consists of knowledge of a language system and that of procedure.
- These two knowledges can be described and implemented in the same expression form with use of programming languages Lisp and Prolog.
- That is, these two knowledges are expected to be learned and acquired by a certain method.

(2) Overview of our research idea:

In the following, we describe overview of our considering flow in the processing order.

[Step 1] To define word unit: Of course even though there may be many discussions and definitions, we defined in this paper that sequence of characters split by spaces is a word. Because this paper handles only English text.

[Step 2] To define part of speech (hereafter, "PoS") or category of words: Names of elements of PoS are arbitrary. But problems that "how many categories should be prepared?" or "There should be any relationship among these categories?" are still open in this paper. The problems are next research themes for us and *W-operator method* is one of the most feasible ones [5]. In this paper, we adopted a subset of a standard PoS depending on problem settings. A word has only properties of spelling, pronunciation, part of speech, and meaning.

[Step 3] To design grammar conceptually: At this point, vocabulary and a set of syntactic rules are defined conceptually, where the word "conceptually" means that the number of PoS and relationship among them are important, on the other hand, names of these are ignored at this step.

[Step 4] To describe conceptual grammar in the formal grammar frame: In this paper, Chomsky's frame was adopted to have basic discussion. A setting of grammar expression frame causes to drop out some details of grammar information by abstraction, but we ignored the phenomena in this research.

[Step 5] To implement grammar as a seed grammar in a programming language Prolog: An example of seed grammar is as follows;

--Syntactic rule:

```
sentence(A,[ ]) :- noun(A,B), verb(B,C),
                   adverb(C,[ ]).
```

--Vocabulary: noun([dogs|T],T). verb([ran|T],T).
adverb([fast|T],T).

[Step 6] To generate sentences, i.e., to generate a concrete language; All grammatical correct sentences are easily generated by Prolog system.

[Step 7] To discover some new grammar rules, i.e., some new more general but compact set of syntactic rules: To realize this, we adopt an inductive logic programming system Aleph (A Learning Engine for Proposing Hypotheses) [6].

[Step 8] To combine syntactic rules of a seed grammar and generated grammar together: The new combined rules is the next seed grammar.

[Step 9] To repeat procedures [Step 6]-[Step 8] until a good enough quality of grammar system, which means the grammar accepts almost all grammatical correct sentences, and almost all grammatical incorrect sentences: Details are discussed later in the chapter 4.

4. Experiments

4.1. Overview of the flow of experiment procedure

All Experiments in this paper were done according to the following flow of procedure in principle. Note that all examples hereafter are carefully selected ones, and type of grammar is restricted to context free, so that readers can understand the essence of my proposal clearly and easily.

[Step 1] To create a file "grammar_sync.b" manually by using a text editor, in which background knowledge of syntactic rules are written.

- Example: sentence(A,[ ]):-noun(A,B), verb(B,C), adverb(C,[ ]).

[Step 2] To create a file "grammar_voc.b" manually by using text editor, in which background knowledge of words are written. That is, "grammar_voc.b" contains data of vocabulary.

- Case of vocabulary 1:
  noun([dogs|T],T). verb([ran|T],T).
  adverb([fast|T],T).
- Case of vocabulary 2:
  noun([dogs|T],T). noun([cats|T],T).
  verb([ran|T],T). verb([walked|T],T).
  adverb([fast|T],T). adverb([slowly|T],T).

[Step 3] To create a file grammar_go.pl manually by using a text editor, in which a Prolog program producing all grammatical correct sentences one by one.

In this paper, the following one is used to produce a concrete language,. i.e., a set of grammatically correct sentences. Prolog program to generate a concrete language ***L*** used in this paper, cf. 3 in Appendix I:

```
:- g(A),fail.
```

[Step 4] To execute ILP system to discover a new more general and compact grammar: Procedure to execute program is as described below:

swipl aleph_2013_v2.pl
(To evoke SWI-Prolog interpreter system)
:- read_all(lang).
(To read in all information)
:- induce.
(This is a pre-install command of Aleph to discover new knowledge)

[Step 5] To create more general-and-compact abstract grammar by combining seed grammar and produced grammar in [STEP 4] above:

Examples:

- Case of vocabulary 1 :
  s(A,B) :- noun(A,C), verb(C,D), vp(D,B).
- Case of vocabulary 2 :
  s(A,B) :- noun(A,C), vp(C,B).

These procedures described above are executed in various settings in this paper.

4.2. A series of Experiments

[Preliminary Experiment 1]

(1) **Purpose:** To confirm validity of programming codes to produce a concrete language from an abstract grammar.

(2) **Programming code settings:** The following code in Prolog are written, and save it into a file "grammar_go.pl".
This code is used as a seed grammar through all experiments in this paper.
- **vocabulary:** Case of vocabulary 1 is set in a file grammar_voc.pl: noun([dogs|T],T). verb([ran|T],T). adverb([fast|T],T).
- **Syntactic rules:** sentence(A,[ ]) :- noun(A,B), verb(B,C), adverb(C,[ ]).

(3) **Results:** All generated sentences, in this case only one sentence *dogs ran fast.* are grammatical with no problems.

(4) **Considerations:** The Prolog program runs in a proper way, so with no problem expected output comes out.

[Experiment 2]
Does the proposed methodology work? (cf. Appendix I. and II.)

(1) **Purpose:** To confirm fundamental validity of the proposed experiment system in 4.1.

(2) **Programming codes settings:** In this experiment, Inductive Logic Programming system "Aleph" was adopted to discover a new set of syntactic rules from a concrete language, which was produced by the program in the preliminary experiment above.
- **Vocabulary:** noun([dogs|T],T). noun([cats|T],T). verb([ran|T],T). verb([walked|T],T). adverb([fast|T],T). adverb([slowly|T],T). 6 words in all.
- **Syntacitic rules:** same in [preliminary experiment], sentence(A,[ ]):-noun(A,B),verb(B,C),adverb(C,[ ]). One rule in all.
- **Background knowledge:**
  - ✓ vp(S1,S2) :- verb(S1,S3), adverb(S3,S2).
    one background knowledge in all.
    Note) In ILP, background knowledge file also contains parameter settings, mode declaration, data type definition. But explanation of these are omitted here in this paper. For more details, see references [4, 5].
- **Positive example dataset:** The dataset is created by SWI-Prolog system based on the vocabulary and syntactic rules described above. 8 sentences in all.
- **Negative example dataset:** The dataset we used was prepared by a human manually as followed;
  s([a,dog,walks,the],[ ]).
  s([a,man,walks,the],[ ]).
  s([a,man,walks,the,walks],[ ]).
  s([a,man,walks,the,house,a],[ ]).
  s([a,man,walks,the,dog,at],[ ]).
  s([the,man,walks,the,dog,to,the],[ ]).
  s([the,dog],[ ]).

(3) **Result:** ILP system Aleph acquired a syntactic rule such as s(A,B) :- noun(A,C), vp(C,B).

(4) **Considerations:** This experiments showed proposed methodology is fundamentally valid. But the size of seed grammar is small. Bigger size of seed grammar should be used to test the validity of the methodology.

[Experiment 3]
Methodology still works on the bigger size of seed grammar?

(1) **Purpose:** To check the validity of the proposed methodology on a bigger seed grammar. In this experiment, we try to use bigger vocabulary and bigger size of syntactic rules.

(2) Programming code settings:
- **Vocabulary:** noun([dogs|T],T). noun([cats|T],T). verb([ran|T],T). verb([walked|T],T). adverb([fast|T],T). adverb([slowly|T],T). det([the|T],T). adj([big|T],T).
- **Syntanctic rules:**
  sentence(A,[ ]):-np(A,B),verb(B,C),adverb(C,[ ]).
  np(A,B):-noun(A,B).
  np(A,C):-det(A,B),noun(B,C).
  np(A,C):-adj(A,B),noun(B,C).
  np(A,D):-det(A,B),adj(B,C),noun(C,D).
- **Background knowledge:**
  np(S1,S2) :- det(S1,S3), noun(S3,S2).
  np(S1,S2) :- det(S1,S3), adj(S3,S4), noun(S4,S2).
  vp(S1,S2) :- verb(S1,S3), adverb(S3,S2).
  Only three syntactic rules in all in this case.
  Note: In ILP, background knowledge is the knowledge we have already known so far.
- **Positive example dataset:** The dataset created by SWI-Prolog contains 32 sentences. No duplicated sentence was found in the dataset.
- **Negative example dataset:** same as in [Experiment 2].

(3) **Result:** Three syntactic rules were discovered.
  s(A,B) :-noun(A,C), vp(C,B).
  s(A,B) :-np(A,C), vp(C,B).

s(A,B) :-adj(A,C), noun(C,D), vp(D,B).

(4) **Considerations:** Everything works so far so good. But when positive example dataset might contain duplicated sentences, they should be eliminated, for example, by command "sort –u". And also in general, not all sentences produced by SWI-Prolog based on given background knowledge are positive. Sometimes negative examples may be generated. This fact is good to gather more negative example, but bad to create adequate grammar by this methodology. We need another software to eliminate positive and negative example from the generated sentences automatically.

[Experiment 4]
Methodology still works
on another bigger size of seed grammar?

(1) **Purpose:** To check the validity of the proposed methodology on another bigger seed grammar again. And to confirm there are in general both positive and negative examples in the generated sentences.

(2) **Programming code settings:**

- Vocabulary: A little bit bigger one such as
  noun([dogs|T],T). noun([ants|T],T).
  verb([ran|T],T). verb([walked|T],T).
  adverb([fast|T],T). adverb([slowly|T],T).
  det([the|T],T). det([a|T],T). det([an|T],T).
  adj([big|T],T). 10 words in all.
- **Syntactic rules:** same as in [Experiment 3].
- **Background knowledge:** same in [Experiment 3].
- **Positive example dataset:** The dataset was generated just in the same way in [Experiment 3] by SWI-Prolog based on the vocabulary, syntactic rules and background knowledge. In this case, the dataset has 64 non-duplicated sentences, but 32 of it, i.e., half of the data, were grammatically incorrect ones from the view point of standard English grammar. Examples are as followed: s([a,ants,ran,fast],[]). s([an,big,dogs,ran,fast],[]). These should be grouped into negative example, because number agreement between article and noun is not invalid in standard English.
- **Negative example dataset:** same in [Experiment 2].

(3) **Result:** same as in [Experiment 3].

(4) **Considerations:** As described before, Positive example dataset contains both positive and negative examples. But the Dataset was applied to ILP system to discover a new grammar. The result was luckily not so bad. This phenomenon suggests that proposed methodology with use of ILP should be robust against nosy data. This is one of the next research theme. At the same time, when we can write a software to eliminate positive examples and negative examples adequately, proposed methodology will be expected to work automatically and effectively. But the dataset is generated based on a seed gramma, and generated sentences are grammatically correct from the point of view in the given grammar. Therefore elimination should be done by human manually. But this task is too hard for human to do, because generated sentences, i.e., a concrete language is in general very large, and prescriptive grammar is also very complicate for us to handle. Systems Engineering methodology, e.g., MBSE (Model Based Systems Engineering) should be investigated if it is applicable to grammar acquisition. This one is also the next research theme.

5. Discussions

This paper proposed a methodology to acquire grammar gradually, in the first step by generating a concrete language, i.e., a set of grammatical correct sentences from a seed grammar, and in the second step by discovering an upgraded version of grammar with use of inductive logic programming system "Aleph". This methodology was checked by experiments, and fundamental validity was confirmed. Validity in the level of principle was indeed confirmed, but feasibility cannot become clarified. From the point of view of Systems Engineering, proposed methodology should be investigated further in terms of complexity and big data.

6. Conclusions

Grammar acquisition was tried by combining and mixing the two different types of inferences, i.e., deduction and induction. Deduction was implemented by SWI-Prolog interpreter system, and induction was implemented by inductive logic programming system Aleph. The latter system provides excellent faculty to infer inductively on background knowledge, which is already known and expressed in some language. For this reason, we proposed to apply the excellent faculty to grammar acquisition. We conclude that the proposed methodology was confirmed to work well in principle, but still should be investigated, and moreover estimated in terms of not only natural language processing but also systems engineering.

Acknowledgement

It is my pleasure to thank Prof. Chiaki Kubomura (Yamano College of Aesthetics). He has been working and contributed to studies on Thought and Language for a long time. This paper is also partially based on his research results by his hard efforts. Now he is, however, seriously sick. We hope he would recover his health soon, and come

back to our research activity. We really thank Prof. Kubomura again.

References

[1] Hiroyuki Kameda, Chiaki Kubomura, "Unknown Syntactic Rule Acquisition by Inductive Logic Programming -Towards Autonomously Evolutionary Intelligent Parser-," Proc. of the Fifth Japan-China International Workshop on Internet Technology and Control Applications (ITCA2006), 2006, pp.6-10.

[2] Saori Aida, Koichi Shimono, and Wa James Tam, "Magnitude of Perceived Depth of Multi Stereo-Transparent-Surfaces," Attention, Perception, & Psychophysics, vol.77, No.1, 2015, pp.190-206.

[3] Noam Chomsky, *Syntactic Structures*, Mouton, 2nd Edition, 1958.

[4] Stephen Muggleton, "Inverse Entailment and Progol," New Generation Computing, Ohmsya and Springer, Vol.13, 1995, pp.245-286.

[5] Shan-Hwei Nienhuys-Cheng, and Ronald de Wolf, Foundations of Inductive Logic Programming, Springer Verlag(1977).

[6] Ashwin Srinivasan, The Aleph Manual Version 4 and above, (http://www.cs.ox.ac.uk/activities/Machinelearning/Aleph/aleph, checked in 2017)

Appendix I. List of codes

1. Code in the file "grammar_voc.pl":

```
noun([dogs|T],T).    noun([cats|T],T).
verb([ran|T],T).     verb([walked|T],T).
adverb([fast|T],T).  adverb([slowly|T],T).
```

2. Code in the file "grammar_syn.pl":

```
sentence(A,[ ]) :- noun(A,B), verb(B,C), adverb(C,[ ]).
```

3. Code in the file "grammar_go.pl":

```
g(A) :- sentence(A,[ ]),
        tell('tmp.txt'), write(s(A,[ ])), write('.'), nl.
:- g(A),fail.
:- halt.
```

4. Code of Parameter setting:

```
:- set(clauselength,1000).
:- set(clauses,100).
:- set(i,10000).
:- set(depth,1000).
:- set(refine,false).
:- set(minpos,2).
:- set(nodes,500000).
:- set(bottom_construction,reduction).
```

5. Code of mode declaration:

```
:- modeh(1,s(+wlist,-wlist)).
:- modeb(1,det(+wlist,-wlist)).
:- modeb(1,prep(+wlist,-wlist)).
:- modeb(1,noun(+wlist,-wlist)).
:- modeb(1,verb(+wlist,-wlist)).
:- modeb(*,np(+wlist,-wlist)).
:- modeb(*,vp(+wlist,-wlist)).
:- modeb(1,adj(+wlist,-wlist)).
:- determination(s/2,det/2).
:- determination(s/2,prep/2).
:- determination(s/2,noun/2).
:- determination(s/2,verb/2).
:- determination(s/2,np/2).
:- determination(s/2,vp/2).
:- determination(s/2,adj/2).
:- determination(s/2,adverb/2).
```

6. Code of type definition:

```
wlist([]).  wlist([W|Ws]) :- word(W), wlist(Ws).
word(X):-atomic(X).
```

7. Code in the file "backgroundKnowledge.pl":

```
np(S1,S2) :- det(S1,S3), noun(S3,S2).
np(S1,S2) :- det(S1,S3), adj(S3,S4), noun(S4,S2).
det([a|S],S).  det([the|S],S).  det([every|S],S).
vp(S1,S2) :- verb(S1,S3), adverb(S3,S2).
noun([dogs|S],S).  noun([cats|S],S).  noun([cars|S],S).
noun([man|S],S). noun([dog|S],S).  noun([house|S],S).
noun([ball|S],S).  verb([ran|T],T).  prep([at|S],S).
prep([to|S],S).  prep([on|S],S).  prep([in|S],S).
prep([from|S],S).  adj([big|S],S).  adj([small|S],S).
adj([nice|S],S).  adj([happy|S],S).  adverb([fast|T],T).
adverb([slowly|T],T).  adverb([happily|T],T).
```

Appendix II. Output of the program

```
[theory]

[Rule 1] [Pos cover = 4 Neg cover = 0]
s(A,B) :-
  noun(A,C), vp(C,B).

[Training set performance]
        Actual
        +       -
   +    4       0       4
Pred
   -    4       7       11

        8       7       15

Accuracy = 0.7333333333333333
```