Context-free Grammar Learning from Text Document using Sequential Pattern

Ramesh Thakur
International Institute of
Professional Studies
Devi Ahilya University
Indore, India

ABSTRACT

The World-Wide-Web and information system has gained significant achievements over the last two decades as expressed their dominance in various business and scientific applications. As estimated by Blumberg and Atre more than 85% of all business information exists in the form of unstructured and semi-structured document, typically formatted for human viewing, not for system processing. Extracting information from these document are challenging task. Extracting grammar rules from these documents is interesting idea. Grammar rules can be used to create structural descriptions of text documents. In this paper I propose grammatical inference using sequential pattern to infer formal language (context free grammar), which describes the given sample set.

Keywords

Information Extraction, Grammatical Inference, Sequential Pattern.

1. INTRODUCTION

The computer and information systems have gained significant achievements over the last two decades as expressed by their dominance in various business and scientific applications. The management of text document is recognized as one of the major unanswered problem in information technology due to unavailability of suitable tools and techniques to transform it for business intelligence. As estimated [1, 2] more than 85% of all business information exists in the form of unstructured and semi-structured data, it is commonly available in the form of text documents and web pages. These documents are intended for human viewing, and not for the application to process it. The text document is without a well defined schema or data model *i.e.* do not have global schema. Grishman and Sundheim [3] described Information Extraction as “The identification and extraction of instances of a particular class of events or relationships in a natural language text and their transformation into a structured representation *e.g.* database”.

The objective of the grammatical inference to infer a formal language, such as context-free grammar, which describes the given sample set. These grammar rules will be used to create structural descriptions of the text documents. In automated grammar learning, the task is to infer grammar rules from given information about the target language. Information Extraction from textual data has various applications, such as semantic search [4]. If the sentences confirm to a language described by a known grammar, several techniques exist to generate the syntactic structure of these sentences. Parsing [5, 6] is one of such technique that rely on knowledge of grammar.

In automated grammar learning, the task is to infer grammar rules from given information about the target language. The sentences (or strings of alphabet) are given as examples for such learning. If the example belongs to the target language, it is called as a positive example. Otherwise, it is called as a negative example. A language that can be inferred by looking at a finite number of positive examples only said to be identifiable in the limit [7, 8].

In this paper I propose a Grammatical (context-free grammar) inference methodology using discovery of sequential pattern form text document. The documents are treated as sequence of string over a fixed alphabet set. The algorithm selects the constituents sequentially.

2. SEQUENCE AND SUBSEQUENCE

Finding sequential pattern in large transaction database is an important data mining problem. The problem of mining sequential pattern and the support-confidence work ware originally proposed by Agrawal and Srikant [9, 10].

Let $I = \{i_1, i_2, \ldots\ldots, i_n\}$ be a set of items in text. We call a subset $X \subseteq I$ an itemset and we call $|X|$ the size of X. A sequence $S=(s_1, s_2, \ldots., s_m)$ is an ordered list of itemsets where $s_i = (s_1, s_2, \ldots\ldots, s_m)$ is an ordered list of items where $s_i \subseteq I, i \in \{1, \ldots, m\}$. The size m of a sequence is the number itemset in the sequence *i.e.* $|s|$ the length of sequence $s=(s_1, s_2, \ldots. s_m)$ is defined as:

$$l =^{def} \sum_{i=1}^{m} |\mathrm{s_i}|$$

A sequence with length l is called l of sequence. A sequence $s_a=(a_1, a_2, \ldots. a_n)$ is contained in another sequence $s_b=(b_1, b_2, \ldots. b_n)$ if there exist integers $1 \leq i_1 < i_2 < \cdots < i_n \leq m$ such that $a_1 \subseteq b_{i1}, a_2 \subseteq b_{i2}, \ldots., a_{in} \subseteq b_{in}$. If sequence S_a is contained in sequence S_b, then we call S_a a subsequence of S_b and S_b is called supersequence of S_a. For example {*bbobbb*}, are sequence and {*bob*} are subsequence. A dataset D is a set of strings in the document and X is the item set such that $X \subseteq I$.

2.1 Frequent Sequential Pattern

A datasheet D is a set of strings, where each string represents the one listing in corpus, and X is an item set in corpus such that $X \subseteq I$ *i.e.* X represents a single line of string. Each x_i $1 \leq i \leq m$ (where m is the no of string in corpus) representing the individual item in D, we refer this representation of D as its sequence representation.

The absolute support of a sequence s_a in the sequence representation of a datasheet D is defined as the number of sequence $s \in D$ that contains s_a, and the relative support is defined as the percentage of sequence $s \in D$ that contain s_a. Given a support threshold "$minSup$", a sequence s_a is called a frequent sequential pattern on D if $supD(s_a) \geq minSup$. The problem of finding sequential pattern is to find all frequent sequential patterns for a datasheet *D*, given a support threshold sup [11].

2.2 Support Factor

The support Factor (SF_β) for sub-sequences in corpora C

$$SF_\beta = \sum_{i=1}^{N} \frac{\text{count of } \beta \text{ in sentences} \times \text{length of } \beta}{\text{length of sentences}}$$

Where *N*= number of sentences in Corpora C and β is a candidate sub-sequence for replacement.

3. SEQUENCE MINING ALGORITHM

I propose a grammar inference methodology to automate the construction of grammar rules from text document. I used finding frequent sequential pattern based on support factor to generate context-free grammar rules from text document.

Our algorithm that infers a sequential pattern from a sequence sentences from the input corpus. The basic insight is that sub-string is selected on the basis of high support factor by taking entire sentences into account. Which appears more frequently in string can be replaced by a grammatical rule $X \rightarrow \alpha$ where α is sub-sequence that generate the new strings, and this process is repeated many times, producing a single length rules of the sequence. The result is strictly a context-free grammar rule, which provide a compact summary of corpora that aids understanding of its properties.

I split the problem of grammatical inference into following phases:

1. Codification of string: I transformed the data into suitable format for processing simplicity. The algorithm expects a set of positive sequence of symbols from a finite alphabet set. So the strings (sentences) of input data sets are codified based on their syntactic categories.
2. Discovery of sequential pattern: Searching of repeated sub-sequences is performed and the sub-sequence having highest support factor is selected for replacement ($X \rightarrow \alpha$ where α is sub-sequence). Repeated sub-sequence is replaced by a non-terminal symbol with proper grammar.
3. Replacement rules of discovered sequence are stored as grammar rule.
4. Rule Simplification: Right part of production rule having similar sequences is compacted by a single rule.

3.1 Proposed Algorithm

```
    Input: corpora $C$ of Flat sentences (codified text String)
           Max-Length (sub-string)
    Output: Set of CFG rules $R$
    Begin
        Initialize rule set $R$=ϕ
    // Calculate sub-sequence upto given Max-length
    $\Sigma$= sub-sequence($c$, Max-length)
    while for every $\alpha$ in $C$ and length of $\alpha$ >1 do
            for each sub-sequence $\beta \in \Sigma$ do
                // calculate support factor for sub-string $\beta$
                $SF_\beta$=support-factor($\beta$)
    end //for
    γ=select $\beta$ of highest score $SF_\beta$
    $N$= select next non terminal symbol
    // add new rule to rule set $R$
    if length of $\gamma$ >1 then
    $R=R\cup\{N\rightarrow\gamma\}$
    // apply replacement rule for each string in the corpora
    update($c$, $N\rightarrow\gamma$)
    else
    $R=R\cup\{N\rightarrow\gamma+\}$
    Update($C$, $N\rightarrow\gamma+$)
    end if
    end while
end // SEQPD
```

Fig 1: Proposed Algorithm for CFG extraction.

```
    Procedure support-factor ($u$)
    // This procedure return support factor for the sub string u in the corpora.
    Begin
    score=0.0
    for $\forall \alpha \in C$, $|\alpha|$>1 do
        score=score+(count_of_subsequence_u_in_α* $|u|/|\alpha|$)
    end for
    return(score)
    end
```

Fig 2: Procedure support factor.

4. EXPERIMENTAL RESULTS

The proposed algorithm has been implemented using a code written in C programming language. The main data structure used is the array of string, which holds the text document. It uses support factor in deciding the replacement rules in the corpora. Some of the experimental results obtained are shown bellow.

Arithmetic expression based on terminal *T= {0, 1, +, -, (,)}*

Table 1. Arithmetic expression

1+1
10+1
1+(1+0-10)
(1+10)+1+0
1+0+11+101+0+(1+1)
10+101+1+1
1+0+(10+1+100)
11+1+(1+1+01)
(1+0+10)-(11+101)
(11+0+1+111+((1+1+0))
…
……

4.1 Rules for Codification

b→0/1, o→+/-, a→(, c→)

Table 2. Coded Arithmetic expression

bob
bbob
boabobobbc
abobbcobob
bobobbobbboboabobc
bbobbbobob
boboabbobobbbc
bboboabobobbc
abobobbcoabbobbbc
abbobobobbboaabobobcc
…
……

4.2 Result (Grammar Inference)

Start→A/B/J/N/R/P/K/L/I/M

A→bob, C→b+ , B→CA, E→aAoCc, F→o+ , D→BF, G→Cc, H→Fa, De, J→CFE, K→BDC, L→AHDG, M→EHBG, N→aAGFA, O→DACHF, P→Oc, Q→AFBDa, R→Qac.

I have applied the algorithm on arithmetic expression for grammatical inference.. The algorithm is applied repeatedly to the text document till all the token are replaced by non-terminal symbol of a context-free grammar. In the each iteration of the algorithm, the support factor for each tokens are calculated and the tokens with highest support factor is replaced by a non-terminal symbol.

5. EVALUATION OF ALGORITHM

The evaluation of Information Extraction using grammatical inference problem has different approaches. Generally, the evaluation of grammar inference algorithm is carried out by giving input to the algorithm a set of unstructured data and evaluating its output (grammar rules). Three principal evaluation strategies usually applied for evaluating grammar inference algorithm [12].

a. Looks-Good-to-me,
b. Compare Against Treebank,
c. Rebuilding Known Grammars.

In Looks-Good-to-me strategy grammar inference algorithm is applied to a piece of unstructured text and the resulting grammar is qualitatively evaluated on the base of the linguistic intuitions of the evaluator, that highlights the grammatical structures which look "good". This kind of evaluation is mainly conducted by experts who have specific knowledge of the syntax of the language.

In *Compare against Treebank* evaluation strategy consists of applying the grammar inference algorithm to a set of plain unstructured sentences that are extracted from an annotated treebank, which is selected as a "gold standard". The structured sentences generated by the algorithm are then compared against the original structured sentences from the Treebank and the *recall* and *precision* are computed.

The *Rebuilding Known Grammars* approach is another evaluation strategy. This method, starting from a pre-defined (simple) grammar, generates a set of example sentences, which are given as input to the grammar inference algorithm and the resulting grammar is compared manually to the original grammar. If the inferred grammar is similar or equal to the original grammar then the learning system is considered good. The following metrics have been used to compare the grammar learned by the proposed algorithms.

Precision, which measures the number of correctly learned constituents as a percentage of the number of all learned constituents. The higher the precision, the better the algorithm is at ensuring that what has been learned is correct.

$$\text{Precision} = \frac{\sum \text{Correctly Learned Constituentes}}{\sum \text{Learned Constituentes}}$$

Recall, which measures the number of correctly learned constituents as a percentage of the total number of correct constituents. The higher the recall, the better the algorithm is at not missing correct constituents.

$$Recall = \frac{\sum Correctly\ Learned\ \text{Constituentes}}{\sum possible\ correct\ \text{Constituentes}}$$

I have used the *Rebuilding Known Grammars* evaluation strategy for the evaluation of our proposed algorithms. The following metrics have been used to compare the grammar learned by the proposed algorithms. I have prepared two different data sets. The first set (sample one) contains the data which follow the uniform rules for generation of document. The second set (sample two) that contains the data which do not follow uniform rules for generation of document.

Firstly, manually annotated results were stored, and then the same content was supplied to the proposed algorithm for automatic grammar extraction. Then the results were compared with the human annotated results. The results of evaluation are as follows:

Table 3. Results of SEQPD algorithm

Data Set	Corpus size (Sentences)	Precision %	Recall %
Sample set one	744	75.4	76.2
Sample set two	878	55.6	42.8
Average	---	**65.5**	**59.5**

6. CONCLUSION

I have introduced a new algorithm for mining sequential pattern form semi- structured document based on grammar inference. It finds possible constituents and afterwards select it based on high support factor by taking entire sentences into account. The output of the algorithm is context-free grammar rules, which provide a compact summary of corpora that aids understanding of its properties.

7. REFERENCES

[1] Blumberg, R., & Atre, S. "The problem with unstructured data." DM REVIEW, 13, pp 42-49 2003.

[2] James, S., Mark, D. Roger, F., Melliyal, A., Jean, I., & Xavier, L. "Managing Unstructured Data with Oracle Database 11g". An Oracle White Paper , pp. 1-9 Feb 2009.

[3] B. M. Sundheim, "Overview of the third message understanding evaluation and conference," In Proceedings of the Third Message Understanding Conference (MUC-3), pp. 3–16, San Diego, CA, 1991.

[4] P. Palaga, L. Nguyen, U. Leser, and J. Hakenberg, "High-performance information extraction with AliBaba ," In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT '09 ACM New York pp 1140–1143, 2009.

[5] Allen J., "Natural Language Understanding," The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, USA. Second Edition, 1995.

[6] N. A. Chinchor, Overview of MUC-7/MET-2 1998.

[7] E. M. GOLD, "Language identification in the limit," Inform Control. vol.10, no.5, pp 447–474,1967.

[8] E M. Gold, "Complexity of automaton identification from given data," Inform. Control, vol. 37, pp 302–320, 1978.

[9] Agrawal and R. Srikant. "Mining Sequential Patterns." In Proceedings of the International Conference on Data Engineering (ICDE), Taipei, Taiwan, 1995.

[10] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements", In Proceedings of the 5th International Conference on Extending Database Technology (EDBT), Avignon, France, March 1996.

[11] Suresh Jain, Ramesh Thakur and N.S. Chaudhari. "Discovery of Sequential Pattern from Text Document". In Proceedings of the National Conference on Intelligent Information Retrieval & Processing (NCIIRP - 2006), CSI Surat Chapter Bardoli, Surat , April, 2006.

[12] D'Ulizia, Arianna, Fernando Ferri, and Patrizia Grifoni. "A survey of grammatical inference methods for natural language learning." Artificial Intelligence Review 36, No. 1 pp 1-27, 2011.