Rule Based Classification of Tamil Poems

K. V. Madhavan, S. Nagarajan, and Rajeswari Sridhar, *Member, IACSIT*

***Abstract*—Natural language processing is a technique where any language can be analyzed for its pure class of grammar. In this paper, we have presented the classification of Tamil poetry to the 'paa' to which it belongs to by using a rule based technique. The rules are constructed using Context Free Grammar by referring to the literature of Tamil. The input is parsed, converted to an intermediate representation and then the classification is performed. A classification accuracy of more than 90% was achieved.**

***Index Terms*—Venpa, tamil poem, context free grammar.**

I. Introduction

The implementation of this paper is based on Context Free Grammar [1] - a notation that has been used extensively for defining the syntax of programming languages. The set of rules which is nothing but the CFG [1] ,for analyzing the Tamil poetry was given by *Tholkappiyar*[2], who is believed to be lived in 5th century B.C.

II. Context Free Grammar

A. CFG – An Introduction

Chomsky proposed the notion of Context Free Grammar (CFG) [1] as a model for describing natural languages. It was successful in defining smaller sentence in English Language. Later, Backus and Naur proposed the Backus Naur Form (BNF) [3] for representing the syntax of the programming Languages. But BNF was equivalent to CFG with some minor changes in the notion and format.

B. Parts of Context Free Grammar:

Terminal: Terminals define the basic symbols of which strings in the language are composed.

Non-Terminal: Non-Terminals are special symbols that denote the set of strings of the language. Non-terminals are described recursively in terms of each other and terminals.

Productions: Productions are rules that define the ways in which non-terminals may be built from one another and from terminals. Production rules are represented as follows:

$$A \rightarrow \alpha \tag{1}$$

where A is a non-terminal and α is a string of terminals and non-terminals.

Start Symbol: Start Symbol is a special non-terminal from which all other strings are derived. It signifies the language being defined.

A CFG can be represented in the form of quadruple (N, T, P, S), where N and T are finite sets of non-terminals and terminals respectively. P is the finite set of productions; each production is of the form $A \rightarrow \alpha$, where A belongs to N and α is a string of symbols from (N U T); S is a Starting symbol. An Example of CFG is given below as,

$$S \rightarrow aAb \tag{2}$$

$$A \rightarrow ab \mid aAb \tag{3}$$

where S and A are Non-terminals and S is the starting symbol. The remaining symbols are terminals which constitute the string.

III. Tamil pa Detection Basics

Asai (அசை): An Asai is the most basic unit. It is essentially a syllable consisting of one or two Kuril and Nedil. There are two types of Asai.

- NerAsai (நேரசை)
- NiraiAsai (நிரைஅசை)

TABLE I: ASAI AND RULES

Asai	Rules
NerAsai	Single kuril and kuril ottru
	Double kuril and Double kuril ottru
NiraiAsai	Single Nedil and Nedil ottru
	Kuril Nedil and Kuril Nedil ottru

Asai can be classified into two major components namely seer and thalai.

SEER (சீர்): Seer is very different from the concept of word. A Seer is a string of one to four Asais. Two, Three are the most common Seers. IyarSeer and Venseer are the two possible sets of seers that are identified according to the rules discussed in the next section.

THALAI (தளை): Thalai is simply a juxtaposition of seers. Thalai is a fundamental construct that determines the musical aspects of a pa. A thalai is specified by which asai succeeds which type of seer. The rules are mentioned later. The Pa vakai mainly depends on the maximum count of a particular thalai in a poem.

IV. Need for Venpa Detection Algorithm

Grammar is one of the most important component of any language across the globe. The richness and usage of a particular language would help in the analysis of the language. Tamil language is blessed with its most rich

Manuscript received April 15, 2012; revised April 29, 2012.

K. V. Madhavan and S. Nagarajan is with the College of Engineering, Anna University Chennai (e-mail: madhavan.varadan@gmail.com, naga4042@gmail.com).

Mrs. Rajeswari Sridhar is with the Department of Computer Science and Engineering at Anna University, Chennai, India (e-mail: rajeswari@cs.annauniv.edu).

grammar rules which makes it unique than any other Dravidian languages. Tamil language has a well established and compiled way of writing poems which is given by Tollkapiyar[2]. Tollkapiyar classifies tamil poems to fall under four important protocols (a set of rules) as explained in Table 2.

TABLE II: PAA TYPE AND THEIR CORRESPONDING THALAI CLASSIFICATION

Pa Type	Thalai
VENPA	Iyarseerventalai , Venseerventalai
ASRIYAPA	Asiriyathalai
KALIPA	Kalithalai
VANJIPA	Vanjithalai

Analyzing these rules would be of great help in knowing the way how good Tamil poems are organized and would encourage people to work more on Tamil poems and learning the richness of the language. Also given any Tamil poem it helps to classify it according to the four Pa class mentioned. This type of classification helps us to differentiate between prose and poetry. Current innovations [4] in this field include the identification of Venpa using Context free grammar[1] .Here we have extended the work by optimizing the 3 parses made by a scanner and a parser [5] to 2.

V. PROPOSED ALGORITHMS

The application is named as Tamil Yappilakanakandigai (தமிழ் யாப்பிலக்கணகண்டிகை).The application identifies the seer asai and thalai of a Tamil song and concludes graph is an "inset", not an "insert".

- Tokenization
- ADI analyzing

} Parse 1

- ASAI analyzing
- THALAI analyzing

} Parse 2

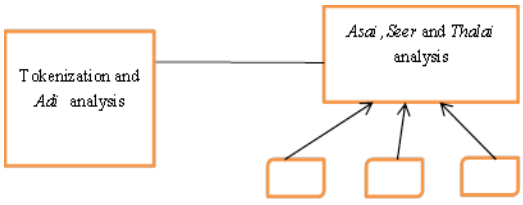

Fig. 1. The phases of the application

A. Tokenization Procedure

The CFG for the Kuril, Nedil and Ottru are given below. (The CFG is only theory based. Implementation is not done using this CFG).

S:=<அ>|<இ>|<உ>|<எ>|<ஒ>|A<space>|A<u0BBF>|A<u0BC1>|A<u0BC6>|A<u0BCA>

A:=<க> | <ங> | <ச> |<ஞ> | <ட> | <ண> | <த> | <ந> | <ப> | <ம> |<ய>| <ர> | <ல>| <வ> |<ழ> |<ள>| <ற> | <ன>

S:=<ஆ>|<ஈ>|<ஊ>|<ஏ>|<ஐ>|<ஓ>|<ஔ>|A<u0BBE>|A<u0BC0>|A<u0BC2>|A<u0BC7>|A<u0BC8>|A<u0BCB>|A<u0BCC>

A:=<க> | <ங> | <ச> |<ஞ> | <ட> | <ண> | <த> | <ந> | <ப> | <ம> |<ய>| <ர> | <ல>| <வ> |<ழ> |<ள>| <ற> | <ன>

S:=A<u0BCD>

A:=<க> | <ங> | <ச> |<ஞ> | <ட> | <ண> | <த> | <ந> | <ப> | <ம> |<ய>| <ர> | <ல>| <வ> |<ழ> |<ள>| <ற> | <ன>

Fig. 2. CFG for kuril, nedil and ottru

1) Adi Analyzer

Adi analysis of Venpa Parser follows the following rules,

- Each line (ADI) of the venpa poem should have four words (SEER) .
- Last line of the venpa poem should have three words.

This analyzes is implemented by taking the pattern of spaces and carriage return with line feed taken into account .For example, let us analyze a Thirukural [6] for ADI pattern,

வெள்ளத் தனைய மலர்நீட்டம் மாந்தர்தம்
உள்ளத் தனைய துயர்வு

Fig. 3. Sample thirukural

The above Kural gives the ADI pattern of 111211, where 1 denotes the <space> and 2 denotes the <carriage return><line feed>. These patterns are stored in a String Buffer instead of external file. Thereby retrieval of ADI pattern from external memory is eliminated and it decreases the time complexity of parsing the given poem.

2) Intermediate Code

In this second step, the input is the Tamil characters given by the user. From the given input, each Tamil character are analyzed and are converted to intermediate language composed of English characters. The intermediate English language composed of three letters denoting Kuril (k), Nedil(n) and Ottru(o). Consider an example,

மலர்நீட்டம்→kkonoko (Intermediate form)

Fig. 4. Intermediate generation of tamil word

3) Segmentation Based on Asai ,Seer and Thalai

From the output of parse 1 (intermediate code), the segmentation based on asai seer and thalai is done. The rules are stored in a file which is loaded as a hash table at the time of usage. The intermediate results are stored in a buffer and processed. The processing is done most efficiently. Venpa can have only Iyarseerventalai and Venseerventalai. Hence based on the thalai result obtained, the conclusion on whether the input poem is venpa or not is made.

4) Rules For Pa Identification

The asai, seer and thalai rules are used to finally identify the Paa.

A. Asai rules - Figure 5 is the automata corresponding to Nerasai and Niraisai.

k,n,ki,ni (*intermediate* form) – Nerasai

kk,kn,kki,*kni* (intermediate form) – Niraiasai

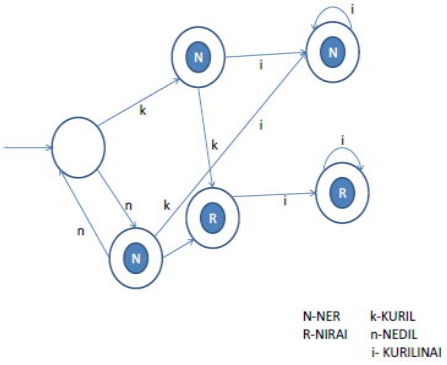

Fig. 5. Automata for nerasai and niraiasai

B. Seer Rules

Ner: Nal
Nirai: Kasu
NerNer: Thema
NiraiNer: Pulima
NiraiNirai: Karuvilam
NerNirai: Kuvilam
NerNerNer: Themangai
NiraiNerNer: Pulimangai
NiraiNiraiNer: Karuvilamkai
NerNiraiNer: Kuvilamkai

C. Thalai Rules

Ma mun Nirai,Vila mun Ner –IyarseerVentalai.
Ma mun Ner,Vila mun Ner --VenseerVentalai.

VI. Mapping of Paa Identification Algorithm with Phases Of The Compiler

The Tokenizer: The tokenizer that we have built is exactly the same as used in any complier. With this tokenizer it would be easy for future development and adding more features .The tokenizer can also be used as a separate component for further researches also.

The Intermediate Code: The intermediate code generated as the output of the first parse is analogous to the intermediate code generated in any compiler. The intermediate code has the following advantages:

- The intermediate code is in English language and hence analyzing it is much easier.
- The intermediate code could also be used for other analysis as it is extremely portable.

VII. Implementation and Results

Input Test Cases: The application was found to give exact results to THIRUKURAL [6] which is profound venpa. The application is designed in a manner that it could accept as many lines of a given poem as is input and would display the result. The results are given in Table 3.

TABLE III: Results Obtained for Thirukkural Test Cases

S.no	Input song	Thalai	Result
1.	நன்றி மறப்பது நன்றன்று நன்றல்லது அன்றே மறப்பது நன்று	Iyarseerventalai Iyarseerventalai venseerventalai Iyarseerventalai Iyarseerventalai	Venpa
2.	யாகாவார் ஆயினும் நாகாக்க காவாக்கால் சோகாப்பர் சொல்இழுக்கு பட்டு	Venseerventalai Iyarseerventalai venseerventalai venseerventalai venseerventalai venseerventalai	Venpa

A. Complexity

The application is designed in a way that is more efficient than the previous implementation. All reference files are implemented as Hash files and hence reference to them does not increase to the complexity of the application.

VIII. Conclusion and Future

The present implementation is to identify venpa only. But by extending the rules the other entire three pa's can be easily identified. The past implementation [4] had three passes which is reduced to 2 passes by us. This could be further reduced to a single pass (including tokenization).

References

[1] N. Chomsky, "Three models for the description of language," *IRE Trans.Info Theory* vol. 2, no. 3, pp. 113-124, 1956
[2] Tolkappiyam written by Tolkappiyar. [Online]. Available: http://pm.tamil.net/pub/pm0100/tolkap.pdf
[3] Bakus Norm Form–Description and example. [Online]. Available: http://otal.umd.edu/drweb/c++tutorial/lessons/BNF.HTM
[4] L. S. Raman, S. Ishwar, and S. K. Ravindranath. Context Free Grammar for Natural Language Constructs, an implementation for Venpa class of Tamil Poetry. Bala. [Online]. Available:
[5] http://www.infitt.org/ti2003/papers/19_raman.pdf
[6] Basics of scanner and parser in Natural Language Processing. [Online]. Available: http://www.boost.org/doc/libs/1_32_0/libs/spirit/doc/basic_concepts.html
[7] Thirukural written by Thiruvalluvar. [Online]. Available: http://pm.tamil.net/pub/pm0001/trkrl1.pdf
[8] Rules for Venpa Identification. [Online]. Available: http://forumhub.com/tlit/venpa.txt
[9] The phases of a compiler unicode representation of Tamil Words. [Online]. Available: http://www.angelfire.com/empire/thamizh/2/aanGilam/

K. V. Madhavan is a Computer Science student from college of engineering ,Anna University . His interests include Signal Processing, Compiler theory, Theoretical Computer Science, Language Technologies.

S.Nagarajan is a Computer Science student from college of engineering ,Anna University . His interests include Signal Processing, Java technologies, Language Processing.

Mrs.Rajeswari Sridhar is a Sr. Lecturer in the Department of Computer Science and Engineering at Anna University, Chennai, India. She is currently doing her Ph.D. in the area of Carnatic music signal processing. She has nearly 10 publications to her credit and is interested in the areas of Signal Processing, Theoretical Computer Science, Language Technologies.