Maharawana

Towards a Grammar Checker for Sinahala

N. Katugampala, P. Ambegoda, A. Gunarathna , D. Bandara, G. Dias , S. Jayasena
Department of Computer Science and Engineering,
University of Moratuwa,
Moratuwa, Sri Lanka

Abstract— **Sinhala is the main language used in Sri Lanka. With the increased use of Information Technology, the typical Sri Lankans tend to use software up to a significant extent in their day to day operations. A major problem that has occurred was the lack of software support for Sinhala language. There is no proper Sinhala Grammar checker implemented yet. In order to build a grammar checker it is mandatory to have a Parts Of Speech tagger which would assign tags for the given words. In this paper, we present a Parts Of Speech tagging approach and grammar checking algorithm we developed for Sinhala.**

Keywords- POS tag (Parts of Speech tag), Sinhala Grammar Checker, POS tag set for Sinhala

I. INTRODUCTION

Part-of-speech tagging (POS tagging) [1] is assigning proper POS tags to each word based on both its definition as well as its context. A POS tag means the grammatical category of a given word. This assignment can be done manually or automatically. Different POS tags can be assigned to the same word depending on the context [2]. As an example consider the word "දුවදුව".In the sentence "දුව ගෙදර ගියාය" , the tag for the word "දු දුව" is NFS, which is a Noun Feminine Singular. But in the sentence "ඔහු දුව දුවා ආවේය", the tag should be "VNF1", which stands for Verb Non Finite. But in a defined context, it can only be assigned with a definite POS tag. As an example, in this sentence: "අම්මා බත් උයයි.", the POS tagger must be able to perform the following: "අම්මා - NFS, බත් - NNN, උයයි - VFM".

Why POS tagging comes first? You may come across that question. Humans are very clever decoders who can sense of the characters on the page and can gain extraordinary amount of inferred knowledge. When compared to human, computer lacks this knowledge. To take full advantage of the query potential of a machine readable text we must make it explicit in at least some of the basics of readably knowledge. If we do so, we can perform many operations quickly and accurately which will be difficult or not practicable for human readers to do. We cannot extract only a list of nouns (or other parts of speech); we can identify syntactic fragments, such as the sequence of three adjectives. A variety of enormous opportunities for inquiry open up with a POS-tagged text. As an example grammar checking is easier with POS tagging. We just need to write grammar rules using proper POS tags [3].

We identified several methodologies for grammar checking process and we analyzed the feasibility of those methods and selected the most appropriate method for implementing our Sinhala grammar checker. We analyzed various grammar-checking approaches such as Syntax based approach, Statistical based approach, Rule based approach and Sliding window based approach [4]. Sliding window based approach was followed when developing the grammar checker.

II. LITERATURE REVIEW

A. Natural Language Processing

Natural Language processing [5] is an emerging discipline which combines computer science and linguistics. NLP helps to develop spell checkers, grammar checkers, translators and etc. Here we use NLP techniques to develop a grammar checker for Sinhala.

We conducted a fair amount of research on Natural Language Processing. There we noted that we need to develop a Part Of Speech tagger before going in to grammar checking. Parts Of Speech category of a word means whether the word is a verb, adjective, noun etc. First of all we had the challenge of coming up with a POS tag set for Sinhala. The way we achieved it is explained under "Research on Sinhala Language". Parts Of Speech tagger assigns POS tags to words in the text. Then we need to define grammar rules using POS tags in such a way that grammar patterns using POS tags would be matched with the text in order to identify the deviations.

B. POS tag set

We went through numerous sources of Sinhala in order to get an idea about designing a POS tag set. The design requires sound grammar knowledge of Sinhala language. We basically used the books "සිංහල වියරණය" [6] and "සිංහල භාෂා ව්‍යාකරණය" [7]. We designed a draft POS tag set and got the feedback from Prof. Rohini Paranawithana of University Of Colombo, Department of Sinhala. She advised us to use Vibhakthi (case) approach when designing the tag set. Later it was further improved with the feedback from Prof. Gihan Dias, University Of Moratuwa.

C. POS tagging

We did our research on POS tagging. Initially we found that there are many types of POS tagging algorithms.

1. Rule based [8]: Rule-based taggers use hand-written rules to distinguish the tag ambiguity.
2. Statistics based [9]: Stochastic taggers are either HMM based, choosing the tag sequence which maximizes the product of word likelihood and tag sequence probability, or cue-based, using decision trees or maximum entropy models to combine probabilistic features.
3. Pattern based [10] – POS tags are assigned based on the patterns
4. Manual [11] - We can use a look up table to store words and tags.

Since our initial task was to develop a POS tagger we searched on many existing taggers. One alternative was to use the POS tagger in Language tool; however due to its complexity of Language tool we had to go for an alternative. Then we searched about Stanford POS tagger. It is a statistics based POS tagger. With the limited time, we felt that it is infeasible to develop a statistics based tagger. Since our attempt was to develop a manual tagger, finally we decided to develop a tagger of our own.

D. Research on the database

We investigated several options for developing a dictionary to store Sinhala dictionary with their corresponding POS tags. Since we have so many words, we decided to use SQLite [12], which is a light weight database. It can be embedded to the application so that user can use the application directly without worrying much about the database. It has Java Database Connectivity support, so we can use Java which has the support for Open Office extensions along with SQLite.

SQLite is different from most of the other SQL database engines in its primary design goal of being simple:

- Simple to administer
- Simple to operate
- Simple to embed in a larger program
- Simple to maintain and customize

Many people like SQLite because it is small and fast. Users also find that SQLite is very reliable which a consequence of its simplicity is. With less complexity, there is less to go wrong. So, yes, SQLite is small, fast, and reliable, but first and foremost, SQLite strives to be simple.

Depending on your requirement, simplicity in a database engine can be either a strength or a weakness. In order to achieve simplicity, SQLite has had to sacrifice other characteristics that some users require, such as high concurrency, fine-grained access control, a rich set of built-in functions, stored procedures, esoteric SQL language features, XML and/or Java extensions, tera- or peta-byte scalability, and so on. If you need some of these features and do not mind the added complexity that they bring, then SQLite is probably not the database for you. SQLite is not intended to be an enterprise database engine. It is not designed to compete with Oracle or PostgreSQL.

The basic rule of thumb to identify the occasions where it is appropriate to use SQLite is as belows: Use SQLite in situations where simplicity of administration, implementation, and maintenance are more important than the countless complex features that enterprise database engines provide. As it turns out, situations where simplicity is the better choice are more common than many people realize.

III. METHODOLOGY

A. Design of the POS tag set

When designing the tags, we tried our level best to obtain meaningful tags which would summarize the grammatical meaning of the word. As an example the tag NFS stands for Common Noun Feminine Singular. Simultaneously we tried to adhere to conventional tags defined in other POS tag sets. As an example we used JJ for adjectives which are generally used in tag sets for other languages [13].

The designed POS tag set is confronted under the section of Results and Analysis.

B. Manual POS tagging approach.

Our first approach was to develop a manual POS tagger. We decided to use a look up table which consists of words and their tags. If the word has multiple tags, then those tags are also defined in the table.

Table 1: POS tags look-up table

Word	**Tag**
අම්මා	NFS
දුව	NFS
දුව	VNF1
ගස	NNS

When the sentence is given as the input, the tagger would tokenize it and assign tags based on the look up table. If a word has multiple tags, all the multiple tags are assigned.

Sentence: ළමයා බත් කයි

Tagged sentence: START S[stt], ළමයා [NNM], බත් [NNN], කයි [VFM], END S[end]

stt and end stand for Start and End tags respectively in order to indicate the start and end of sentences.

C. Selection of a data base

We decided to use SQLite as our database to store the tags. The reason for our decision is that SQLite is a light weight data base which can be embedded to our product.

D. Automate POS tagging

We automated POS tagging by developing a tool. We need to give the base word (or stem), then the tagger would generate all the derivations and assign POS tags automatically and insert to the database. We analyzed common patterns of nouns and verbs and implemented the tool using Java.

We analyzed Sinhala grammar and came up with the following plan.

1. Develop a noun tagger - Assigning tags for the different derivations of the same noun, this can be adopted with the knowledge on grammatical cases in Sinhala language. We analyzed different cases and came up with automated noun tagger.
2. Develop a verb tagger - Here we analyzed verb patterns, and identified that there are patterns when conjugating verbs. Present tense verbs conjugate differently than past tense verbs. So we implemented two algorithms for each. And the stem, which is given as input to the tagger also differs according to the tense.
3. Any word tagger - If we want to tag any word with any tag, this approach can be used.
4. Tag patterns- This is where we can input tag patterns easily to the data base, we just have to input a single text file which contains tag patterns and it would update the database of tag patterns.

E. Grammar Checking Algorithm

The major problem of conventional syntax based approach is that they use the user input text directly, without using their POS tags. But after some researches we were able to implement a model for Sinhala grammar checking process using POS tags.

In this model first we assign POS tags for every word in the sentence. For the words that have multiple tags, all the tags will be assigned. Following diagrams depict how the words are assigned with tags.

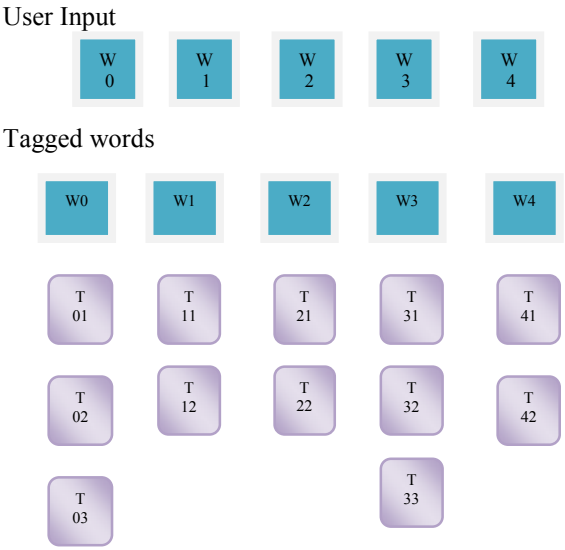

Figure 1: Tagging process

After this process is done, we retrieve all the possible input tag sequences as the next step. Some of the possible tag sequences from this example can be listed as follows.

Tag 01, Tag 11, Tag 21, Tag 31, Tag41
Tag 01, Tag 12, Tag 21, Tag 31, Tag41
Tag 01, Tag 12, Tag 22, Tag 31, Tag41
The total number of sequences can be obtained: 3*2*2*3*2= 72

The next step is to check whether these sequences are grammatically correct or not. In order to do that we should have pre defined POS tag graphs. Let's assume that we have a graph as below.

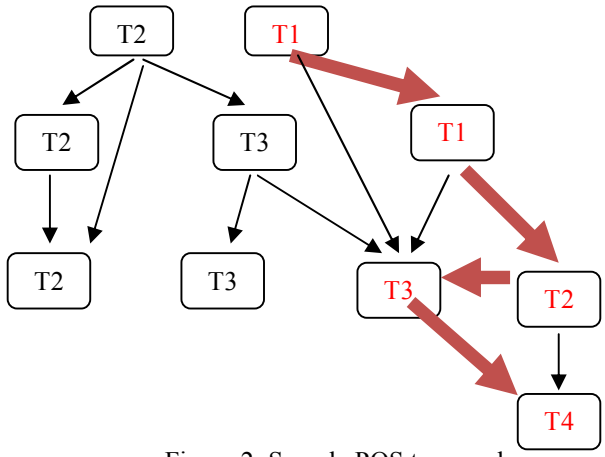

Figure 2: Sample POS tag graph

According to the above graph we can observe that the tag sequence "Tag 01, Tag 11, Tag 21, Tag 31, Tag 41" is grammatically correct. In that case we can suggest that the given sentence is grammatically correct. If we cannot observe even a single traversable sequence out of all the possible sequences, the grammar of the given sentence would be incorrect.

Our approach is a combination of rule based and sliding window approach and the main steps of the algorithm are given below.

Step 1: First input sentence will be tokenized in to words.

Step 2: Tokenized words are tagged with corresponding POS tags. Tags are retrieved from the database.

Step 3: All the Input tag sequences are obtained. Every sequence contains 3 tags since the grammar rules are defined by considering 3 consecutive words that can appear in a sentence.

Tag 01, Tag 11, Tag 21
Tag 11, Tag 21, Tag 31
Tag 21, Tag 31, Tag 41, etc.

Step 4: After matching the possible tag sequences with defined sequences in the database, impossible sequences are

eliminated. This example is chosen in way that it does not have a correct sequence. This sentence is incorrect according to the rules defined in the database.

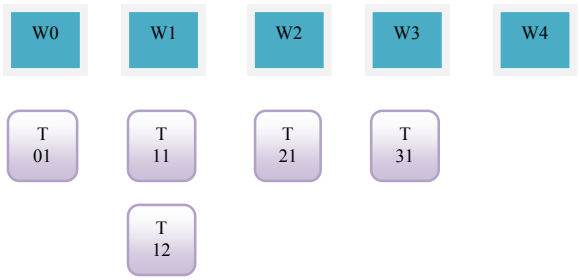

Figure 3: Tag sequences in the database that match with the sentence

Step 5: Possible suggestion will be produced as mentioned below,

1. Retrieve all the tags that match with stem of the "Word 4"
2. Retrieve all the possible sequences match with "Tag21", "Tag31", Tag_s1 / Tag_s2/ Tag_s3
3. Retrieve the words which have,
   tags: "Tag_s1 or Tag_s2"
   Stem: stem of Word4
4. Suggesting the above words as suggestions for Word4.

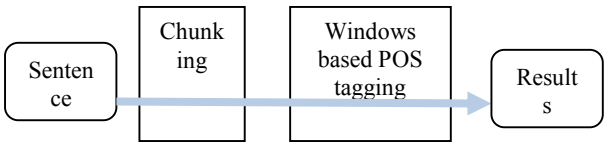

Figure 4: Grammar checking process

As we began developing our system the first requirement was to develop the Part of Speech tagger. From the knowledge we got from our research the best approach found was the sliding window POS tagging algorithm [4]. This approach is based on probability. There it takes two tags and assign the probability of occurrence of some other tag after that. These probabilities have been found by studying tagged sequences for about million of sentences tagged manually. And due to the massive amount of works that should be done it has taken many years to be completed. As our project expands only about two semesters even though this approach is good we had to make it possible to do it on time. (More details of the original sliding window based grammar checking is provided bellow under Sliding window with probability based POS tagging)

The difference that we made was removing the probability. The system will have data regarding three word sequences that can occur inside a sentence by doing so we have change the approach from a probability based approach in to a rule based approach. And a sentence will be checked in the same process that is the sliding window based approach.

For this approach to be effective we added two words a start word and an end word to the beginning and to the end of the sentence. The starting word has the string of "START" and the tag "stt". And the end word has the string "END" and the tag "end".

E.g.: The sentence "මම බත් කමි" will be converted to "START මම බත් කමි END". By doing so we can check the sequence "START මම බත්" and the sequence "බත් කමි END" that would not have been checked otherwise.

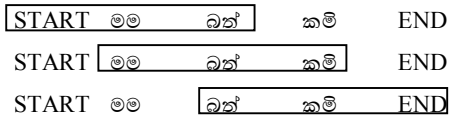

Figure 1: Assigning tag sequences

If all tag sequences are found in this approach then the sentence is said to be grammatically correct.

Then from this approach we found that this can be enhanced in to grammar checking. The main purpose of building the POS tagger was to remove irrelevant tags and to tag the sentence with relevant tags for the pattern.

But as no patterns are found for incorrect sentences we could mark those sentences as incorrect sentences.

E.g.: For the incorrect sentence "මම බත් කමු"

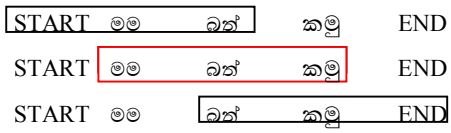

Figure 6: Assigning tag sequences

The first pattern will be found in the database. But the next pattern will not be there in the database tag sequences. But the next sequence is not there in the database. As the first seaquake is there the sequence of first two tags for the given words will be considered to be correct. Thereby we can come to the assumption that the last word of the incorrect tag sequence is incorrect. With this approach we can give error suggestions from another approach.

IV. MAHARAVANA ARCHITECTURE

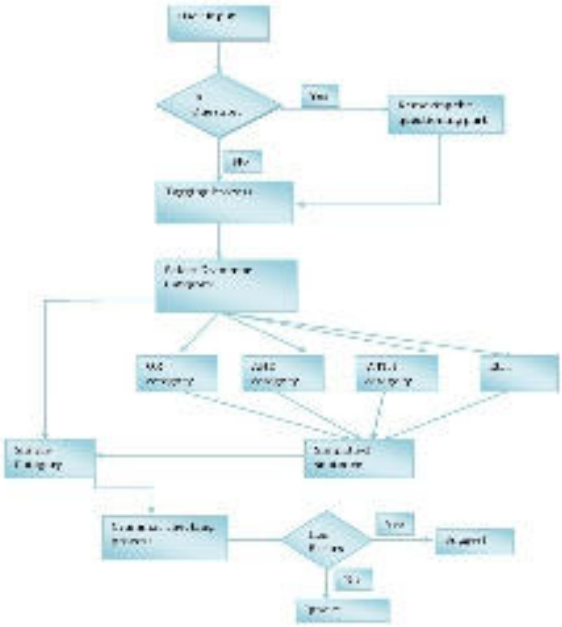

Figure 7: Maharawana Architecture

V. RESULTS AND ANALYSIS

We have come up with a POS tag set for Sinhala and a look up table based POS tagger for Sinhala.

A. POS tag set for Sinhala

The suggested POS tag set is as follows.

Table 2: POS tags for grammar cases nouns

විභක්තිය -Case	Singular	Plural
Nominative Case ප්‍රථමා විභක්තිය	NFS- Noun Feminine Singular NMS- Noun Masculine Singular NNS - Noun Neuter Singular-ගස	NFP- Noun Feminine Plural NMP- Noun Masculine Plural NNP- Noun Neuter Plural-ගස්
Accusative Case-කර්මවිභක්තිය	NFSO- Common Noun Feminine Singular Objective NMSO- Common Noun Masculine Singular Objective NNSO- Common Noun Neuter Singular Objective	NMPO- Common Noun Masculine Plural Objective-මිනිසුන් NFPO- Common Noun Feminine Plural Objective-නිලියන් NNPO- Noun Neuter Plural Objective
කතෘ විභක්තිය	NSI – Common Noun Singular Instrumental-බල්ලා විසින්	NPI – Common Noun Plural Instrumental-බල්ලා විසින්
Dative Case - සම්ප්‍රදාන විභක්තිය	NSD – Common Noun Singular Dative නිලියට	NPD- Common Noun Plural Dative නිලියන්ට
Ablative Case - අවධි විභක්තිය	NSA- Common Noun Singular Ablative ගසෙන්	NPA- Common Noun Plural Ablative-ගස්වලින්
Genitive Case - සම්බන්ධ විභක්තිය	NSG – Common Noun Singular Genitive-බල්ලාගේ	NPG – Common Noun Plural Genitive-බල්ලන්ගේ
Locative Case - ආධාර විභක්තිය	NSL – Common Noun Singular Locative- බල්ලා කෙරෙහි	NPL – Common Noun Plural Locative- බල්ලන් කෙරෙහි
Instrumental case- කරණ විභක්තිය-Same as Common Noun Ablative	NSA – Noun Singular Ablative පොරොවෙන්	NPA - පොරෝවලින්
ආලපන විභක්තිය-Vocative Case	NSV- Noun Singular Vocative- ළමයා	NPV- Noun Plural Vocative-ළමයි

Table 3: POS tags for verbs

Present Tense	**Singular**	**Plural**
Masculine	VSMP – Verb Singular Masculine Present – කයි,කන්නේය	VPP- Verb Plural Present-කති,කන්නෝය,කන්නාහ
Feminine	VSFP -Verb Singular Feminine Present -කයි ,කන්නීය	VPP -Verb Plural Present කති,කන්නෝය
Past Tense	**Singular**	**Plural**
Masculine	VSMPT- Verb Singular Masculine Past-කෑවේය	VPPT -Verb Plural Past-කෑවෝය
Feminine	VSFPT - Verb Singular Feminine Past - කෑවාය	VPPT- Verb Plural Past - කෑවෝය

First Person	Singular	Plural
Present Tense	VFSP – කමි, කන්නෙමි	VFPP- කමු ,කන්නෙමු
Past Tense	VFSPT-කෑවෙමි	VFPPT - කෑවෙමු
Second Person	**Singular**	**Plural**
Present Tense	VSSP - කන්නෙහි	VSPP-කන්නෙහු
Past Tense	VSSPT-කෑවෙහි	VSPPT-කෑවෙහු

Table 4: POS tags for remaining word categories

NR	Common Noun Root - නාම ප්‍රකෘති	මිනිස්, බලු
PRFS	Pronoun First person Singular - උත්තම පුරුෂ	මම
PRFP	Pronoun First person Plural - උත්තම පුරුෂ	අපි
PRSS	Pronoun Second person Singular -මධ්‍යම පුරුෂ	නුඹ
PRSP	Pronoun Second person Plural- මධ්‍යම පුරුෂ	නුඹලා
QFNUM	Number Quantifier	එක, දෙවෙනි, පලවෙනි
DET	Determiner	මේ, ඒ, අර, බොහෝ, සියලු
JJ	Adjective - නාම විශේෂණ	සුමුදු
RB	Adverb- ක්‍රියා විශේෂණ	හයියෙන්
RP1	Particle	ද, ත්
VNF1	Verb Non Finite 1	බල, නට , පැන
VNF2	Verb Non Finite 2	බලා
VNF3	Verb Non Finite 3	බලමින්
VNF4	Verb Non Finite 4	බලද්දී
VNF5	Verb Non Finite 5	බලනොත්
VP1	Verb Participle 1	බලන බලන, බැලූ
VP3	Verb Participle 3	බලන්නේ, බැලුවේ
VNN	Verbal Non Finite Noun	බැලීම්, බැලිලි, බැලුම්
POST	Postpositions	ගැන, ලෙස, සඳහා
CC1	Conjunctions1	සහ, සමඟ
CC2	Conjunctions2	හෝ
NVB	Noun in Verb	"නැටුම්" නටනවා
JVB	Adjective in Verb	"නපුරු" වෙනවා
UH	Interjection	අහෝ, චික්
FW	Foreign Word	Computer
NC	Not Classified	A4

B. Comparison with existing POS tag sets

We found that University Of Colombo School Of Computing also has designed a POS tag set for Sinhala [14]. We felt that we can further improve it by adding more tags.

The first thing that we noticed was they have not assigned tags based on the case of a noun. We were guided by Professor Rohini Paranawithana on designing the tag set. With her input, we decided to define tags for all nine cases in Sinhala language. When analyzing the cases, we noticed that Instrumental case (කරණ විභක්තිය) is similar to ablative case (අවධි විභක්තිය) in the usage, so we assigned a one tag for both.

Then we noticed that there are no separate tags for verb conjugations. Then we added many tags for different conjugations of verbs based on their tense, gender and singular or plural.

Then we had the problem of identifying the tags at the first sight. The tags did not resemble their grammatical properties directly. As an example the tag we have assigned for singular, neuter (E.g.: ගස) is NNN. We wanted to make tags more meaningful. So we changed the tag set in a manner which would resemble the grammatical properties. As an example the tag for a singular, neuter is changed to NNS which stands for Noun Neuter Singular.

C. Automated Tagger

We developed an automated POS tagger where we can derive all the POS tags for a given base word. The tool automatically inserts all the words to the database. So adding a new word to the database is not difficult as manual tagging.

Our POS tagging tool consist of four main components.

- Noun tagger – When a noun stem is given, it derives nouns based on 9 cases and assigns the respective tag accordingly.
- Verb conjugation - When a verb stem is given, it derives all related verbs and assigns the respective tag accordingly.
- Tag patterns – We can input a text file consisting of tag patterns of three POS tags or enter tag patterns manually and add it in to the database. This is important in adding new rules to the grammar checker. If the rule is simple, such that it can be represented through a set of tag sequences, then it is easy to insert it to the database.
- Other – this helps to enter individual words with respective tags into the database. If it is neither a verb nor a noun we can assign the tag manually and insert it to the database.

POS tagging tool helps to enter Sinhala words with their respective POS tags into database by just entering stem only. One stem would help the user to add many words to the database. It is more comfortable than adding word by word with respective tags at a time.

D. Noun Tagger

A noun can appear in several forms according to the 9 cases ("Vibhakthi") which are defined in Sinhala grammar [7]. Almost all the nouns in Sinhala language can be converted in to cases by following a similar pattern. All the cases of a particular noun can be derived using the stem of that noun and is the key point of our automation of noun tagging process.

Table 5: How nouns are derived according to nine cases

Cases	Example
ප්‍රථමා	මිනිසා
කර්ම	මිනිසා
කතෘ	මිනිසා විසින්
සම්ප්‍රදාන	මිනිසාට
අවධි	මිනිසාගෙන්
සම්බන්ධ	මිනිසාගේ
ආධාර	මිනිසා කෙරෙහි
කරණ	මිනිසාගෙන්
ආලපන	මිනිසා

For a single case, a noun can again be categorized as below,

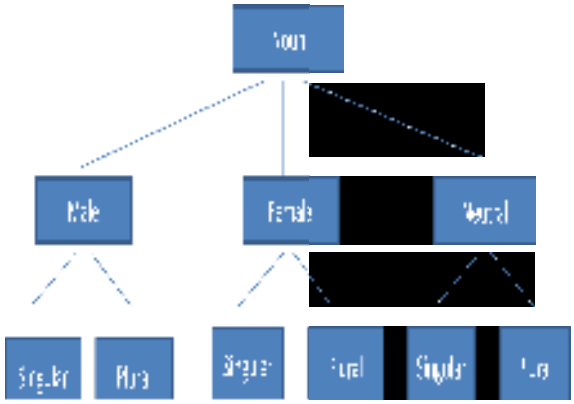

Figure 7: Classification of a noun

After analyzing nouns we understood that we can automate the tagging process by adding several suffixes at the end of a noun. These suffixes that need to be added at the end of the noun do not changed depending on the noun, but the suitable form of the base word will have to be decided manually. The example below shows how a singular noun gets converted in to nine cases. To get nine cases of the plural words of those, we cannot use the same stem. In that case we have to use "මිනිස්සු" and "බල්ලෝ".

To derive the 9 cases of these plural nouns, we cannot find a common stem. To handle that we have to enter the proper form of the noun manually.

Table 6: Derivation of noun "මිනිසා" into nine cases

Base	Suffix	POS Tag
මිනිසා		NFS
මිනිසා		NMSO
මිනිසා	විසින්	NSI
මිනිසා	ට	NSD
මිනිසා	ගෙන්	NSA
මිනිසා	ගේ	NSG
මිනිසා	කෙරෙහි	NSL
මිනිසා	ගෙන්	NSA
මිනිසා		NSV

Table 7: Derivation of noun "බල්ලා" into nine cases

Base	Suffix	POS Tag
බල්ලා		NFS
බල්ලා		NFS
බල්ලා	විසින්	NSI
බල්ලා	ට	NSD
බල්ලා	ගෙන්	NSA
බල්ලා	ගේ	NSG
බල්ලා	කෙරෙහි	NSL
බල්ලා	ගෙන්	NSA
බල්ලා		NSV

Since we cannot find a common stem for all the forms of nouns to derive 9 cases, the tagging process cannot be fully

automated. We need to analyze some factors of the noun manually,

- Gender of the noun (masculine /feminine /neutral)
- Singular/Plural
- Correct form of the base word

E. Verb Conjugation

When it comes to verbs in Sinhala also they are derivations of particular stem according to 9 cases [7, 15]. (We could understand that though there are 9 cases, when verbs are considered only 7 cases come into play) When the stem is given the relevant suffix is appended to the stem considering tense, gender and Singular/plural and relevant POS tag is assigned automatically. When tense is considered stem which we have to enter may be different. As shown below to derive the present and past tense verbs "ක" and "කෑ" should be given separately as stems.

Table 8: Verb Conjugation in Present Tense

Verb	POS Tag
කමි	VFSP
කමු	VFPP
කන්නෙහි	VSSP
කන්නෙහු	VSPP
කන්නේය	VSMP
කන්නීය	VSFP
කයි	VSFP
කයි	VSFP
කති	VPP
කන	VP1
කමින්	VNF3
කද්දී	VNF4
කතොත්	VNF5
කන්නාහ	VPP

Table 9: Verb Conjugation in Past Tense

Verb	POS Tag
කෑවෙමි	VFSPT
කෑවෙමු	VFPPT
කෑවෙහි	VSSPT
කෑවෙහු	VSPPT
කෑවාය	VSFPT
කෑවේය	VSMPT
කෑවෝය	VPPT

F. Grammar Suggestions

We discovered the possibility of giving error suggestions. In order to approach giving suggestions we have to make the assumption that the user has entered a grammatically inappropriate form of word that he wanted to enter.

E.g.: The user wanted to enter the sentence which will be giving the meaning that he eats rice. That is "මම බත් කමි".
But he did not know or missed the rule that when the subject been "මම" the verb should be "කමි" and entered the sentence "මම බත් කමු". This sentence will be detected as an incorrect sentence from the sliding window based algorithm explained above.

Now if we can derive the correct word that could come for the incorrect word based on that word and the three word seaquake we can give error suggestions.
The approach we took to find the incorrect word from the incorrect, inappropriate word is based on the base word of any word. For the first two words of the tag sequence we can find sequences that are there with those two tags and some other tag after that.
E.g.: For the tag sequence PRFS, NNPO, VFPP and for the sentence "මම බත් කමු" as we did not find it in the database sequence table it checks for any sequence with starting tags PRFS and NNPO. And from that it gives] PRFS]]NNPO]]VFSP] is a correct tag sequence. Now we have to find derived word in VFSP format from the base word of "කමු".

The approach we took is defining a base word for each word in the database. While we are entering words to the database we derive from base word. That is a stem. Thereby for a given word in the database there exist the "word" column to keep the string format of the word, the "tag" column to keep the appropriate tags for the relevant word and the "stem" which keeps the stem of that word.

G. Problems with Sliding Window Grammar Checking

The existing grammar checking algorithm, which is the shift window algorithm, will not check some of the sentences even in the simple sentence format. As in a simple sentence for which we should check grammar there can be several adjectives and adverbs. In such scenarios the subject and the verb will not be matched as we are checking with three word sequences.
E.g.: මම දුවමින් සෙමින් කමි
The grammar will be checked for Start මම දුවමින්, මම දුවමින් සෙමින්, දුවමින් සෙමින් කමි and for සෙමින් කමි end. Thereby the object මම and the verb කමි will not be matched. There by it want show every error.

H. The Solutions for the Problems with Sliding Window Grammar Checking

One option for this problem is to remove all the words with adjective and adverb format and then check grammar. This was our first approach but by doing so there can be other problems. Adjectives come before nouns and adverbs come before verbs. But with removing all of them we are not considering the possession of usage of those adjectives and adverbs.
E.g.: මම සෙමින් කලු කමි will not be identified as an incorrect sentence. As "කලු" which is in adjective format has been used as an adverb or a noun, as it has come before "කමි" which is in verb format.

The best approach for this problem is to use finite automata that will chunk adjectives and adverbs with their relevant nouns and verbs.

The approach taken will also cover some other grammar rules. The sentence as usual will be tagged for each words and tags sequences will be generated when there are multiple tags assigned to the same word. For an example if there exists a word with three tags and another word with two tags then there exists three into two, that is six tag sequences for that particular sentence.

E.g.: 1: සුදු මම රතු බත් සෙමින් කමු.The word බත් has two tags NNP and NNPO. Thereby there are two tag sequences.
E.g. 2: Tag sequences for අම්මා බත් **කන්නිය** are:
NFS NNP VSFP, NFSO NNPO VSFP, NSV NNP VSFP, NFS NNPO VSFP, NFSO NNP VSFP, NSV NNPO VSFP

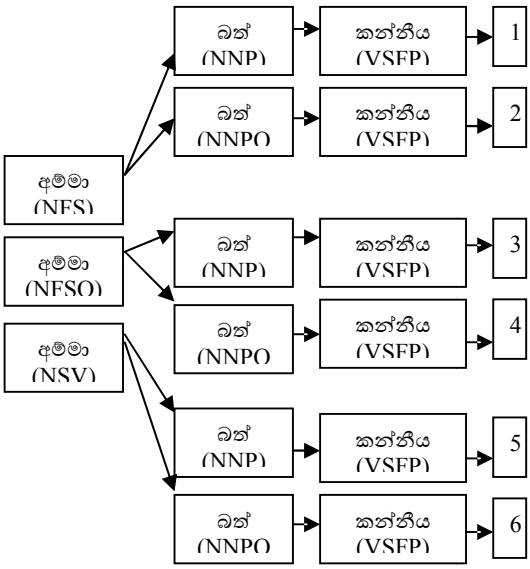

Figure 8: Tag assigning

Then each tag sequence is taken in reversed order. This is done because in Sinhala language we have to match from the verb for the relevant noun and the verb exists at the end of any sentence. If a verb is not found then it will directly go to the error state from state 1 (figure 1). The system is checking whether it is a verb by keeping the tags that could come for verbs in the database. Some of the tags that come for verbs are VSMP, VSFP and VPP, etc.

E.g.:"සුදු මම රතු බත්" will be an incorrect sentence as the system will identify "බත්" is not in verb format. And if it is a verb then it goes to the second state. Then before verbs there can be an adverb. For each adverb found, it will remain in the same state. At the state two if the word is not a noun or not an adverb at the start of a sentence it will go to the error state.

E.g.: In "යනවා කමී", "යනවා" is not a noun or not an adverb. In "සෙමින් කමී" in state two it will get the next input as the start of the sentence. Tags for adverbs are RB and VNF3.

From state 2 if it is a noun then it will go to the state 3. Then if a noun or an adjective is found then it will remain in the state 3. Tags for nouns are NFS, NMS and NNS, etc. The tag for adjective is JJ. Then if the start of the sentence is reached, it goes to the final state. E.g.: සුදු මම රතු බත් සෙමින් කමී.

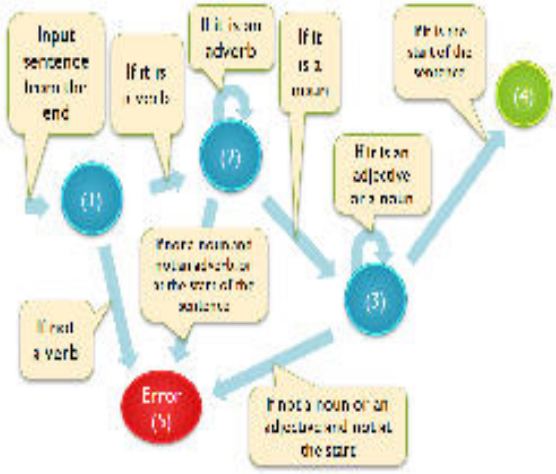

Figure 9: Chuncking Process

At the state 3 the system has read the word "බත්" then it checks for "රතු", and as it is an adjective it remains in the state 3. Then the word "මම" comes which is a noun. And its adjective "සුදු" comes. Now as it is at the start of the sentence it goes to the final state.

During this process of checking the sentence when a verb or a noun is found the system triggers a flag. Then it will remain until the next noun or the adjective is been found. During that time all the words are adjectives or adverbs found for the earlier found verb or the noun. Those adjectives and adverbs will be chuncked with them.
E.g.: The sentence "සුදු මම රතු බත් සෙමින් කමු" will be chunked as shown bellow.
START[stt], {සුදු]JJ], }මම]PRFS], {රතු]JJ], }බත්]NNP], {සෙමින්]RB], }කමු]VFPP], END[end],
Words in curly brackets are chuncked for the word after the curly brackets. "සෙමින්" is chuncked for "කමු" and is the adverb of that word. These adjectives and adverbs are been kept inside the noun and adverb word object.

As there can be several tag sequences there can be several incorrect sequences. For then they will be eliminated giving the reasons for the error in the console. If there exists one or more tag sequences found then those sequences will be sent to be checked by the pattern matching.

E.g.: When checking for "අම්මා කමු" the chunked sequences will be.

1. START>stt, අම්මා>[{NFS}NFS], කමු>[{VFPP}VFPP], END>end,
2. START>stt, අම්මා>[{NFSO}NFSO], කමු>[{VFPP}VFPP], END>end,
3. START>stt, අම්මා>{error pos <this should be a noun>}NSV, කමු>[{VFPP}VFPP], END>end,

The first two tag sequences are correct with respective to the chunking algorithm as the rules are been satisfied by those sequences. But in the third sequence is incorrect with respect to the algorithm.

Thereby only the first two tag sequences will be sent to the window based grammar checking algorithm.

If no correct tag sequence is found for particular sentence then it implies that the sentence is completely wrong in the chunking level. These errors are due to the early mentioned errors that are been checked for during the chunking process.

E.g.: for the sentence "සුදු මම රතු බත් සෙමින්":

It will give the error "!!!! Last word of the chunk doesn't have a verb tag"

Then it will not proceed in to the window base grammar checking.

I. How Chuncker uses the Database

For the chunking to keep which tags are nouns, verbs, adjectives and which tags are adverbs we have used a table in the database. It shows the tag and for which it should be chunked for.

E.g.: The stem of "කමු" and "කමි" is "ක".

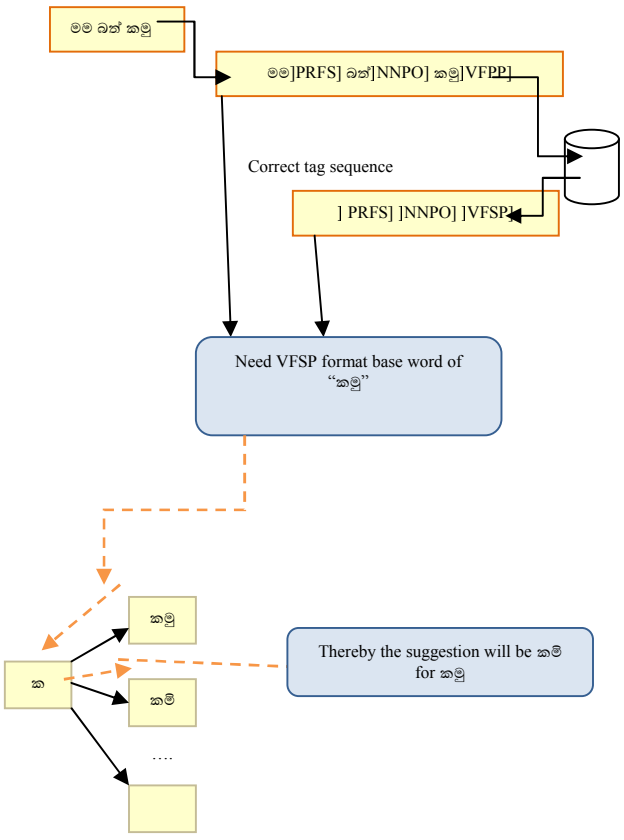

Figure 10: Suggestion Mechanism

Thereby we check the stem of "කමු" and we can find "ක" is the base word. Now as we want VFSP we search for the word with the tag VFSP and with the base word "ක". The resulting one will be "කමි".

For this approach to be effectively used the words should be added to the database to reflect this format. And there should be tag sequences in the database for every pattern that can come. A word stemmer will be ok with English language. But for Sinhala language with more complex form of word derivations this approach is more efficient.

This approach is a brute force approach which uses a lookup table that maps form the existing infected word to the stem word. The Suffix stripping algorithm does not use a look up table like in brute force algorithm. Instead, a typically smaller list of "rules" is stored which provide a path for the algorithm, given an input word form, to find its root form. The inappropriateness of this approach has been mentioned above.

J. Future Improvements

We are currently working on Chunking. Chunking is where two or many POS tags can be assigned a separate tag. Chunks are normally taken to be a non recursive correlated group of words. Sinhala has a complex morphological and syntactical structure. It is a relatively free word order language but in the phrasal and clausal construction, it behaves like a fixed word order language. Then it is easy and less complicated to analyze [16]. There are more grammar rules that can be implemented further. When implementing more grammar rules, new POS tags can be introduced.

CONCLUSION

At present there is no grammar checker for Sinhala language. Our goal was to develop a basic grammar checker for Sinhala which would remove the barrier of producing a grammar checker for Sinhala. The methods which we used in this project can be applied when developing grammar checkers even for other languages. The best example is the pattern matching mechanism. The algorithm which we developed for error suggestions can be applied to any language. But the rule specific algorithms which we developed for advanced grammar rules can be used only for Sinhala. But the logic can be used by other developers. We did not focus on producing a complete grammar checker. Instead we selected a certain set of rules and focused on them with the intention that future developers would contribute to the project Maharavana.

ACKNOWLEDGMENT

We are thankful to the project coordinator Dr.Shantha Fernando for his guidance and constant supervision. We would also like to express our gratitude towards all the staff and the support staff of CSE for all the support they have extended towards us.

REFERENCES

[1] "Part of speech - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Part_of_speech. [Accessed: 04-Sep

[2] Sandipan Dandapat, "Part-of-Speech Tagging for Bengali", Department of Computer Science and Engineering Indian Institute of Technology, Kharagpur, January 2009.

[3] Martin Mueller, "A part of speech tag set for written English, from Chaucer to the present", November 2009.

[4] Daniel Naber, "A Rule-Based Style and Grammar Checker", October 2003.

[5] "Natural language processing - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Natural_language_processing. [Accessed: 04-Sep-2011].

[6] Set සිංහල වියරණය, 2nd ed. Wasana Publishers, 2009.

[7] W.S.Karunathilaka, සිංහල භාෂා වාහකරණය. Gunasena, 2011.

[8] "Rule Based POS Tagging," NATURAL LANGUAGE PROCESSING. [Online]. Available: http://language.worldofcomputing.net/pos-tagging -base -p -tagging . [Accessed: 04-Sep-2011].

[9] L. Altunyurt, Z. Orhan, and T. Gungor, "Towards combining rule-base and statistical part of speech tagging in agglutinative languages," Computer Engineering, vol. 1, no. 1, pp. 66–69, 2007.

[10] Geunbae Lee, Jeongwon Cha, and Jong Hyeok Lee, "Syllable pattern-base
of korean," Computational Linguistics, vol. 28, no. 1, Mar. 2002.

[11] M. Mieskes and M. Strube, "Part-of-speec
speech," in Pr , 2006, pp. 935–938.

[12] "About SQLite." [Online]. Available: http://www.sqlite.org/about.html. [Accessed: 04-Sep-2011].

[13] Dinn Dien, Hoang Kiem, "POS-Tagger for English-Vietnamese Bilingual Corpus", vol. 3, pp. 88-95, 2003.

[14] University of Colombo School of Computing, "A Part of Speech Tagset for Sinhala."

[15] J. B. Dissanayake, ක්‍රියා පදය. Godage, 2008.

[16] Dhanalakshmi , Anand kumar , Rajendran , Soman , "POS Tagger and Chunker for Tamil Language"