2013 International Conference on Advances in ICT for Emerging Regions (ICTer): 137 - 143

Learning a Stochastic Part of Speech Tagger for Sinhala

M. Jayasuriya[#1], A. R. Weerasinghe [*2]
[#]*Virtusa (Pvt) Ltd, Sri Lanka*
[*]*Language Technology Research Lab, University of Colombo School of Computing, Sri Lanka*

[1]maheeka.j@gmail.com, [2]arw@ucsc.cmb.ac.lk

Abstract*— This paper presents the results of developing a part of speech (POS) tagger for Sinhala. The tagger is able to handle lexical items with multiple POS tags while also predicting POS tags of previously unseen words. A stochastic approach, Hidden Markov Model (HMM) with tri-gram probabilities was used as the training and tagging model. Linear Interpolation is used to smoothen the tri-gram probabilities while the Viterbi algorithm is used to decode the results of the HMM to decide on the best POS tags for each word. The tagger learns the lexical items (words and their possible POS tags) and the tri-gram probabilities using a POS tag annotated corpus. The tagger achieved an overall accuracy of 62%. Approximately 24% of the errors were for words whose POS tags have been unknown in the corpus. The lack of a Named Entity recognizer has also contributed to 10% of the overall error.

***Keywords*— Part of speech tagging, Hidden Markov Model, Viterbi algorithm, Linear Interpolation, Sinhala language**

I. INTRODUCTION

Part of Speech (POS) can be defined as the grammatical type of a word. POS tagging is the process of marking up the words in a sentence to its corresponding parts of speech, based on both a word's identity and its context. A POS tagger is a prerequisite for many Natural Language Processing (NLP) tasks.

As far as we are aware, to date there is a single reported result on Sinhala POS tagging [4]. The lack of an accurate and reliable POS tagger is one of the reasons behind the lack of NLP applications such as parsers and translators for Sinhala.

II. BACKGROUND

Sinhala is a morphologically rich agglutinative language of the Indic family. While this makes it harder to build an accurate tagger without a good morphological pre-processor, it provides a compelling reason for attempting to build a POS tagger for Sinhala.

To construct a robust tagger we need to undertake two main processes. The first is to tag a text corpus and the second, to train a model using this tagged corpus. This paper focuses on the research carried out with regards to both these aspects.

A stochastic POS tagger was previously proposed for Sinhala, based on a HMM using bi-gram probabilities resulting in an accuracy of approximately 60% [3]. This research is an attempt to develop a learning POS tagger with improved accuracy by using a better defined tag set and a well balanced corpus for training.

III. LANGUAGE BACKGROUND

There are different versions of the language that is in use. These vary based on geographical regions and the generation of people. However in general, Sinhala language has two main categories – written and spoken.

Written Sinhala has a structured approach and always follow the language grammar and vocabulary standards. Spoken colloquial Sinhala is highly varying and depends on geographical areas, generation, ethnic group, etc. It also keeps changing rapidly due to coining of new words and follows a completely different grammar structure. Therefore, it is difficult to construct a model to grasp the colloquial spoken Sinhala concepts. When it comes to computer linguistics, the need of processing colloquial Sinhala is also less. Hence, this research only focuses on written Sinhala.

The linguistic research to be carried out includes the following components.

A. Tokenization

This is the knowledge on the language's boundary marking symbols or punctuations. For Sinhala, the identified boundary markers are as follows:

- Sentence boundary marker punctuations – period (.), question mark (?), exclamation mark (!)
- Other punctuation marks - quotation marks (" "), single quotation marks(' '), hyphen (-), apostrophe ('), slash (/), brackets (( ), { }, [ ]), colon (:), comma (,), semi-colon (;)

B. Tag Set

Tag set defines the parts of speech in a language. When it comes to POS tagging, the tag set might not have to represent all of the tags in the language. It depends on the requirement and the level of research that is being done. Defining the size and content of a tag set is a very tedious linguistic oriented task [2] that involves in depth linguistic research and analysis. The tricky part is that, it keeps changing as the language evolves.

Tag set composes of the best suited POS tags for a language. A descriptive tag set will enable encoding of more knowledge, which is especially suitable for morphologically rich languages But too descriptive will result in ambiguity and difficulty in distinguishing among tags. It should also not be

978-1-4799-1276-6/13/$31.00 ©2013 IEEE

too abstract since then most of the language features could go unnoticed. It is sufficient if the tag set captures the core of the language.

For Sinhala , the primary POS tags are, *Nama* (Nouns), *NamaVisheshana* (Adjectives), *Kriya*(Verbs), *KriyaVisheshana* (Adverbs), *Nipatha* (Prepositions, Conjunctions, Articles), *Krudhantha* (Verbal Nouns) [3]. As mentioned above, this is insufficient. Therefore these tags have been further categorized by the Language Technology Resource Laboratory (LTRL) of University of Colombo School of Computing, and have defined a more elaborated POS tag set for Sinhala language which includes 22 tags [7]. This tag set will be used in this research.

C. Word Classes

Part of speech can be of two categories – closed class and open class. Closed classes have a relatively fixed membership. For example "*nipatha*"(prepositions) in Sinhala would be a closed class. There is a fixed set of *nipatha* and there is minor chance for the number to increase. Open classes on the other hand are for example "*nama*" (nouns) and "*kriya*" (verbs) where the words continuously change and increase in size due to coining of new words.

The advantage of closed class parts of speech in POS tagging is that, those words do not need to be referred explicitly. They can be referenced in a fixed reference. This can be advantageous when it comes to increasing querying efficiency.

With relevance to the above chosen tag set of 22, we have identified 6 closed class and 16 open class tags.

D. Lexicon

A lexicon is a repository of words [6]. Computational lexicons are usually structured with a list of each of the stems and affixes of the language together with a representation of the morph tactics to tell how they can fit together.

Lexical resources can be of two types; dictionary based and corpus based. There is little research done on creation of lexical databases for Sinhala.

Lexicon that will be used for this project will need to hold only the parts of speech annotations as opposed to saving word forms which is essential for morphologically rich languages since morphology is not addressed in this research.

Due to evolution of languages that introduce new words to a language, a dictionary based lexicon would not be successful in capturing these words. The reason for it is that they are limited in terms of reliance on introspection and linguistic exposure to human compilers [5]. Corpora based lexicon is a less expensive, less time consuming and robust alternative approach for less studied languages like Sinhala. Therefore, a corpus based lexicon is developed specifically for this system.

E. Corpus

A corpus can be defined as a collection of machine readable authentic texts that is sampled to be representative of a particular natural language or language variety [12]. He further mentions that the size and the material of the corpus depend on its intended usage. However, the most important feature of a corpus is being balanced or representative. Representativeness is achieved by including full range of variations of a language.

The corpus developed by LTRL with 10 million words out of which 1 million is already annotated manually is used in this research. An annotated corpus is the most important resource for a data driven POS tagging approach [3]. Therefore this corpus is used in this research.

IV. POS TAGGING

Input to a tagger is a string of words and a tag set. The output is a single best tag for each word in the input [6]. However, the identification of the best fit tag requires addressing two problems [6][2].

The first is the problem of ambiguous words. This is when a word occurs for which more than one POS tag is legally possible (e.g.: *can* – noun, auxiliary).

The second is the problem of unknown words, which are words unseen in the training data. It is the capability to address these two problems that decides the accuracy of the tagger.

A statistical POS tagger uses a statistical model to decide on the best suited tag for a particular word.

V. TAGGER

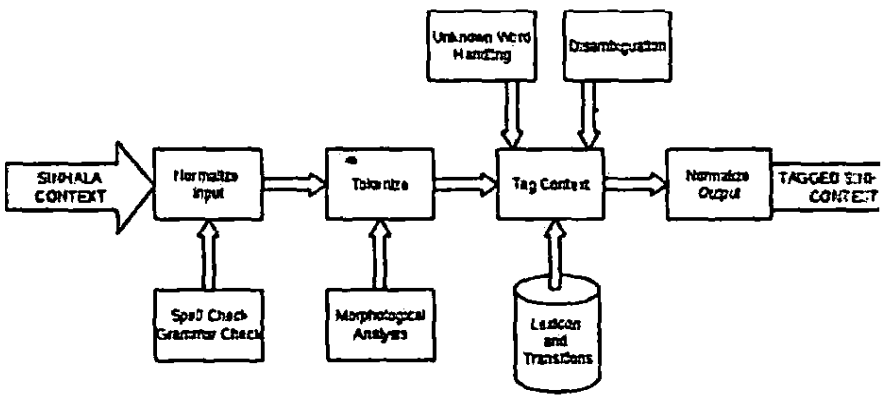

Figure 1: Tagger Architecture

As shown in Figure 1, at the outset, the tagger normaliz the input given to it. Normalizing includes aspects of spellir and grammar checking. Subsequently the text is tokenize During tokenization, a morphological analyser is used overcome the word structure problems in tokenizing. Then t tokens are passed for tagging. The tagging process retriev lexical and transition probabilities from the trained model. T lexical data will decide whether a word has a single ta multiple tags or whether the word is unknown. For unknov words and those with multiple tags, the transition probabiliti are used to predict the tag. Thereafter, the tagged data normalized and returned as output.

A. Normalizing

Since the corpus published by the LTRL was used, th required a certain level of pre-processing to address the cor encoding and formatting to be used as input to the tagger.

This process also requires spelling and grammar checki although it was not addressed in this research. Further, mechanism for dealing with acronyms and abbreviations also not implemented.

In most of the taggers, in the resultant, POS tag follo the word. This is the format for achieving the optimum res

for the tagger since it is easy to refer when it comes to large corpus and querying. The LTRL corpus tagging has used an underscore for the delimiter between the word and the tag. Formatting the results of the output in the same format as the corpus is done in output normalization phase.

B. Tokenization

Tokens may correspond to words, numbers, punctuation marks or even proper names. Text segmentation is the process of converting a well defined text corpus into its component words and sentences; and word segmentation breaks up the sequence of characters in a text by locating the word boundaries [11]. These words are referred to as tokens and thus the process is called tokenization.

Tokenization is done in three phases – sentence segmentation, phrases segmentation and word segmentation. Since Sinhala is a space-delimited language, word boundaries are indicated with white space. Therefore, white space was considered as the word delimiter for segmentation. Periods, question marks and exclamation marks were considered as sentence delimiters. Comma, colon, semi-colon, hyphen, parenthesis and quotation marks were considered as phrase delimiters.

There are three types of fundamental word structures that tokenization has to address [1].

- Isolating – words do not divide to smaller units. E.g.: විශ්ව,විද්‍යාලය
- Agglutinative – words divide to smaller units (morphemes) with clear boundaries among morphemes. E.g.: ගියේය, බලබලා
- Inflectional – boundaries between morphemes are not clear and morphemes can express more than one grammatical meaning. E.g.: උස්සනයි, විසිනි

To address these problems, a morphological analyser has to be used.

Sentence and word segmentation are dependent on one another. Abbreviations are one such instance in which the period symbol does not act as a sentence boundary [1]. Therefore there needs to be a component to exactly identify what the relevant delimiter or punctuation mark mean at any given time.

C. Tagging

This process contains two phases. The first is the lexicon lookup, which is to look up the lexicon for corresponding POS for a text. The query will produce possible tags for each word that can be classified as – single POS, multiple POS, unknown POS. The result of the POS tagger should always be a single POS tag per word. For single known POS tags the determination is straightforward. For multiple known POS tags, a disambiguation module is required and for unknown POS tags, a guesser module is required. This is the second phase.

To further elaborate, let $W = w_1w_2.....w_n$ be a sentence having n words. The task of POS tagging is to find the set of tags $T = t_1t_2.....t_n$ where t_i corresponds to the POS tag of w_i, $1 \leq i \leq n$, with as the highest accuracy possible [2].

D. Use of Rules

Apart from the statistical model that is being used for the tagging, rules were also implemented although on a minor scale. Rules are applied for the words before applying the statistical model for disambiguation. Therefore the statistical model follows any rule applications and sees them as single POS words thereby avoiding any interference to the model. The rules used for this system are as follows:

- Numeric words–Numbers and numeric values are distinguished by Arabic numbers appearing in a word.
- Foreign words– Foreign words more specifically English words are identified by English letters in the word.
- Punctuations – If the word is a punctuation mark, the corresponding tag is also the same punctuation mark.

VI. Statistical Modelling

Statistical models are applied for three tasks of the POS tagger – disambiguation, decoding and smoothing for handling of unknown words.

A. Disambiguation - Hidden Markov Model (HMM)

Disambiguation is the sequence prediction model. HMM is the chosen model for this research. This model performs the task of counting cases and making a table of probabilities for a certain sequence. This probability will be obtained from previous knowledge of occurrences of similar POS tag sequences from an annotated corpus. In this model words are observables and tags are hidden labels. The considered tag sequence is a triplet in this research. Therefore, the Hidden Markov Model is said to be using tri-gram probabilities. When several ambiguous words occur together, the number of possible tag combinations (in sequences) increases drastically. The most suitable sequence is chosen based on previous probabilities of sequences.

There are two schemes in finding the optimal tag for a given sentence [2].

- Sentence level tagging – maximizing over a possible tag sequence for a sentence.
- Word level tagging – maximizing over the possible tags for each word.

Of these, sentence level tagging is the most researched and reliable approach and is used in this research.

In Markov model tagging, the sequence of tags in a text is considered as a Markov chain and has two properties – limited horizon and time invariant. Limited horizon is to assume that a word tag depends only on a considered number of previous tags. Time invariant is that the above dependency does not change over time [8].

These properties are only approximately accurate. It omits long distance relationships among words thus reducing accuracy of the time invariance.

The HMM calculates transition and lexical probabilities as below [1]:

Unigrams: $\hat{P}(t_i) = f(t_i)/N$

Bigrams: $\hat{P}(t_i|t_{i-1}) = f(t_{i-1},t_i)/f(t_{i-1})$

Trigrams: $\hat{P}(t_i|t_{i-2},t_{i-1}) = f(t_{i-2},t_{i-1},t_i)/ f(t_{i-2},t_{i-1})$

Lexical: $\hat{P}(w_i|t_i) = f(w_i,t_i)/f(t_i)$

The above calculations are made for all t_i, in the tag set and w_i in the lexicon. These are captured using the LTRL corpus in this research.

B. *Decoding - Viterbi Algorithm*

Decoding is when a model and a sequence of observations is provided, determining the most likely state sequence that produces the observations [10].

In POS tagging, the sequence of observations is the word sequence and the state sequence is the related tag sequence. Therefore, decoding produces the most likely tag sequence for a given word sequence.

The Viterbi model has three phases. They are initialization, induction and termination. At the end of this process the path read out is the result of the algorithm. This model calculates two functions. One is $\delta_i(j)$ which gives the probability of being in state (tag) j at word i, and $\Psi_{i+1}(j)$ which gives the most likely tag at tag i, given that we are in state j at word i+1[2].

C. *Smoothing - Linear Interpolation*

The sparse data problem is addressed using smoothing. This is when particular transition probabilities for tri-grams may not have occurred in the training corpus. The probability for such a trigram cannot be assigned zero since it will affect the probability calculation.

A linear Interpolation model is used as the smoothing technique in this research. Following is the formula used for smoothing an unknown transition probability of tags t_1, t_2, t_3 occurring in sequence [1].

$P(t_3|t_1,t_2) = \lambda_1 \hat{P}(t_3) + \lambda_2 \hat{P}(t_3|t_2) + \lambda_3 \hat{P}(t_3|t_1,t_2)$

P is the maximum likelihood estimates of the probabilities and λ are the weights which also have the property [1],

$\lambda_1 + \lambda_2 + \lambda_3 = 1$

A context independent variant of linear interpolation is used where the values of λ do not depend on the tri-gram concerned. Although a context-dependent variant might provide better accuracy, it cannot be used due to the sparse data problem for each tri-gram [1].

D. *Handling Unknown Words*

To tag an unseen word, the tag which provides maximum of transition probabilities in combination with the previous two tags is selected.

VII. LEARNING

This is to decide the automation level for training the tagger. In machine learning, a model is induced automatically for some domain, given some data and other information [6].

There are two approaches to machine learning – supervised, unsupervised. In supervised learning, the computer learns a classification system the developer has created. Where as in unsupervised learning, the system by itself learns what needs to be done. Supervised learning is dependent on the pre-determined classifications and their accuracy. Unsupervised learning will require an indication of system success and it is a highly time consuming task since it has to completely learn by trial and error.

Supervised learning is to be mainly concerned with predicting missing information based on observed information [13]. In POS tagging this means predicting missing information of a tag sequence or generation of rules based on a corpus.

We decided to use supervised learning approach for this system. The reason is to keep track of the processing since we are in an initial stage for POS tagging for Sinhala. This learning process will be helpful to analyse when further improving the tagger in future.

VIII. TRAINER

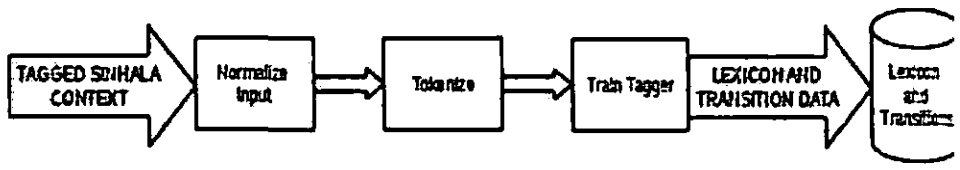

Figure 2: Trainer Architecture

As shown in Figure 2 above, the trainer takes in tagged Sinhala text as input. This input is taken from a pre-tagged corpus. After this, the text is normalized and tokenized. The lexical and transition data are then extracted from the text, and is used to update the database with new entries and frequencies of occurrence.

A. *Normalizing*

During normalization similar to in tagging, it will address the corpus encoding and formatting. This also includes excluding of corpus meta-data for the training input.

B. *Tokenizing*

During tokenization for training data the delimiter considered is only white space. The corpus text is tokenized by space which results in tokens <word>_<tag> format. The delimiter between words and tags is an underscore as per LTRL corpus standard. These tokens are then passed to training tagger.

C. *Train Tagger*

This phase contains the process of reading the tokens produced in the earlier stage to update the lexicon and transition probabilities.

D. *Lexicon*

```
<lexiconelem>
  <word>ඉඳිය</word>
  <tag>
    <freq>1</freq>
    DET
  </tag>
  <tag>
    <freq>45</freq>
    PRP
  </tag>
</lexiconelem>
```

Figure 3: Sample lexicon data

Above Figure 3 shows how a lexicon item is stored in an XML file. For each word, their possible tags and their frequencies of occurrence is stored in the above format.

By referring the lexicon, words are categorized as single POS, multiple POS or unknown POS. The frequencies obtained from this are then used in Markov model to calculate lexical probabilities.

E. *Transition Probabilities*

```
<gram>
  <tag>VP</tag>
  <freq>5938.0</freq>
</gram>
```

Figure 4: Sample Transition – Unigram

```
<gram>
  <tag>NNPI</tag>
  <tag>VFM</tag>
  <freq>17.0</freq>
</gram>
```

Figure 5: Sample Transition – Bigram

```
<gram>
  <tag>PRP</tag>
  <tag>POST</tag>
  <tag>VNP</tag>
  <freq>9.0</freq>
</gram>
```

Figure 6: Sample Transition – Trigram

Above Figure 4, Figure 5 and Figure 6 shows how transition frequencies are stored in xml files. All the tag sequences extracted from the annotated corpus and their frequencies of occurrences are stored as above. These are used to calculate unigram, bigram and trigram probabilities for the Markov model.

Thus, calculated unigram, bigram, trigram and lexical probabilities together make up the complete tri-gram Markov model for tagging.

IX. Tests and Evaluations

A corpus was used for testing and training the tagger. This corpus from LTRL included around 100,000 pre-tagged words. Approximately 80% of these data was used for training the tagger. Another 10% was set out for validation and the remaining 10% was used for testing the tagger. According to these proportions, the corpus was divided for different tasks using a stratified sampling technique.

The tagger was trained and tagged in five phases. Performing the tagging in five phases was to evaluate its accuracy with the increase of training data. A summary of each training phase with the number of tokens and the tagger accuracy is shown below in Table 1 and Figure 7.

Accuracy	Phase				
	1	2	3	4	5
Total Accuracy	67.97%	60.58%	59.14%	58.78%	60.66%
# Trained Tokens	12374	31338	50679	66620	71918
# Unique Words	4304	8537	11505	13829	14338
# Unique Tokens	4730	9926	13593	16385	16963

Table 1: Accuracy with Training Phase

Here, unique words refer to one lexical entry which composes of word and its corresponding tags. Unique tokens refer to such unique word and tag combinations.

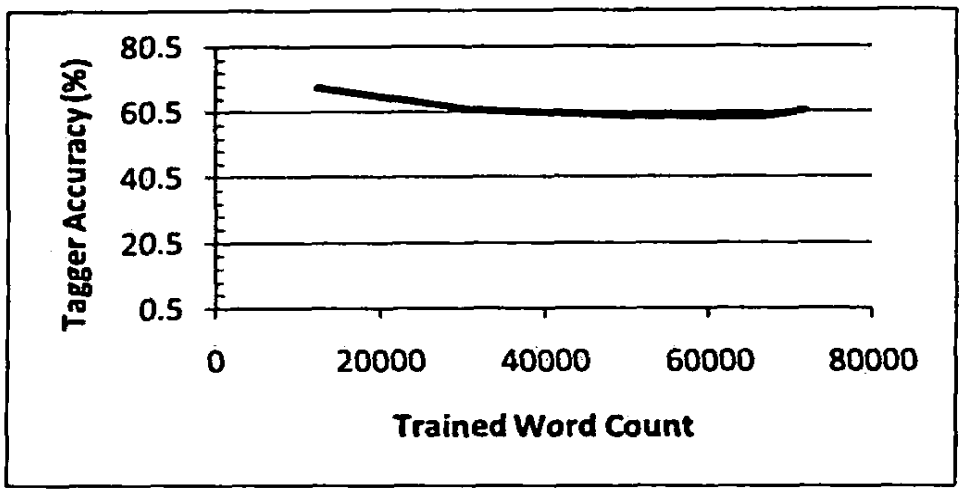

Figure 7: Accuracy with Training Phase

According to the above graph and data, there is no major improvement in the accuracy of the tagger with the increase of training data.

Further testing was carried out after reaching 80% of corpus data for training the tagger. This testing led to the results shown in Table 2.

Total Counts				
# of Tokens	9459			
# Correctly tagged	5638			
# Wrongly tagged	3380			
# Undecided	441			
Total Tagged		Total Wrong	Wrong Percentage	Contribution to Total Error
Single Tag	3232	206	6.37 %	**6.09%**
Multiple Tag	4858	2355	48.47%	**69.67%**
Unseen Word	1369	819	59.82 %	**24.23%**
Final Accuracy	**62.51%**			

Table 2: Final Test Results

According to these figures the overall accuracy of the tagger is 62.51%. Further, 59.82%, 48.47% and 6.37% of unseen words, multiple tagged words and single tagged words respectively have been tagged wrong.

X. Error Analysis

An error analysis of the results is given in below Table 3.

Phase	Error Percentage (%)			
	Single Tag	Multiple Tags	Unseen Words	Total Error
1	36.54	41.07	12.9	32.02
2	10.18	48.35	64.36	39.03
3	9.25	50.84	65.8	40.85
4	8.69	51.04	66.79	41.21
5	8.31	49.26	66.33	39.33

Table 3: Percentage Errors with Their Tag Appearance

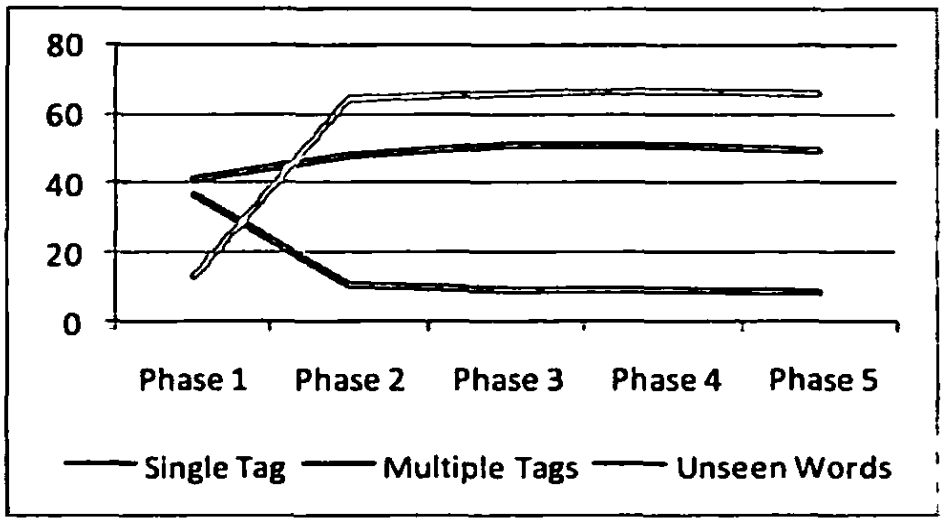

Figure 8: Percentage Errors with Their Tag Appearance

When a word is tagged wrong by the tagger, the appearance of that word in the lexicon is as shown above in Table 3 and Figure 8Percentage Errors with Their Tag Appearance. As mentioned above, there is no major improvement to the tagger accuracy with increased training data. Thus, although the overall error percentage is somewhat in similar range, the error percentages categorisation based on the word's appearance in lexicon has changed.

Single tag, multiple tags and unseen words refer to the availability of a particular token's tag in the lexicon. Words that have the potential to have multiple tags, and words not seen in the training data are tagged by estimating their most likely tag using the Viterbi algorithm.

When referring their percentages, the errors due to single tags have reduced with the amount of training. Errors due to multiple tags have remained moderate in the same zone. The highest increase in errors is due to unseen words. This is further emphasised in the final test results presented in Table 2.

XI. Conclusion

The motivation for this project was to develop a POS tagger for Sinhala. It requires several intermediate text processing tasks, such as normalization and tokenization.

The tagger achieved an overall accuracy of 62%. Some 24% of these errors are due to the incorrect prediction of tags for unseen words.

10.17% of the total tagged words are named entities and 30.97% of these have been tagged incorrectly. Therefore, it is also a significant area to consider when building a better tagger.

Our tagger achieved the accuracy of 62% for a limited genre. Robustness is an important feature that has to be addressed. Therefore, although the present system provides a reasonable accuracy level, it requires a lot of future work before any other NLP application could use it.

A. Limitations

Following are a few limitations of the tagger developed in this research.

- This research was limited to written Sinhala and more specifically the "Newspaper" genre. Therefore there might be a loss of accuracy when tested with other genres.
- Transitional probabilities are limited to two prior tags. Long distance relationships are not considered.

B. Future Enhancements

The following are suggested as future enhancements for the tagger.

- Implementation of a verification of tags on the basis of being open or closed classes.
- Incorporation of a morphological analyser, a spell checker, a grammar checker and a named entity recognizer to improve accuracy of the system.
- A Morphological analyser will improve the accuracy of tokenizing and will support identification of forms of words when defining words with exception to exact matches.
- A Named Entity recognizer will support in tagging named entities to avoid being tagged wrongly and thereby minimizing wrong impact for neighbouring words.
- Implementation of rules for some of the tags would improve accuracy.
- Tokenization should not be limited to delimiters only. It impacts incorrectly in situations such as abbreviations and large numbers separated by commas.

References

[1] Brants T. (2000), TnT – A Statistical Part-of-Speech Tagger, In 6th Applied Natural Language Processing Conference, Seattle, WA., April 29th – May 3rd 2000

[2] Güngör T., (2010), Part of Speech Tagging, In: Handbook of Natural Language Processing.2nded. New York, USA: Chapman & Hall/CRC

[3] Herath D.L. , Weerasinghe A.R. (2004), A Stochastic Part of Speech Tagger for Sinhala, Language Technology Research Laboratory, University of Colombo School of Computing, In 2nd International Conference on E-Governance, IITC, Sri Lanka, 29th November – 1st December 2004.

[4] Herath D.L., Weerasinghe A.R. (2010), Research Report on Sinhala WordNet, Language Technology Research Laboratory, University of Colombo School of Computing, In 11th International Mother Language Day, UNESCO, 21st February 2010

[5] Herath D.L., Weerasinghe A.R. (2009), Corpus-based Sinhala Lexicon , Language Technology Research Laboratory, University of Colombo School of Computing, In 7th Workshop on Asian Language Resources, Singapore, 6th -7th August 2009

[6] Jurafsky D., Martin J. H., (2008), Speech and Language Processing. 2ndEd., New Jersey: Prentice Hall

[7] Language Technology Resource Laboratory (2005). Part of Speech Tagset for Sinhala, PAN Localization Project 2005.

[8] Manning C.D.,Schiitze H. (2000), Statistical Natural Language Processing. 2nd Ed., Massachusetts Institute of Technology, USA: Lucida Bright

[9] Manning C.D.,Schiitze H., (1999) Foundations of Statistical Natural Language Processing, London, England: The MIT Press

[10] Méndez L.A.T. (2000), Viterbi Algorithm in Text Recognition, [online]. Available: http://www.cim.mcgill.ca/~latorres/Viterbi/va_main.html, [Accessed 1st April 2011]

[11] Palmer D.D., (2010), Text Preprocessing, In: Handbook of Natural Language Processing.2nded. New York, USA: Chapman & Hall/CRC

[12] Xiao R., (2010), Corpus Creation, In: Handbook of Natural Language Processing.2nd ed. New York, USA: Chapman & Hall/CRC

[13] Zhang T., (2010), Fundamental Statistical Techniques, In: Handbook of Natural Language Processing.2nd ed. New York, USA: Chapman & Hall/CRC