IMPACT: International Journal of Research in Engineering & Technology (IMPACT: IJRET)
ISSN(E): 2321-8843; ISSN(P): 2347-4599
Vol. 2, Issue 5, May 2014, 97-110
© Impact Journals

HIERARCHICAL ANNOTATOR SYSTEM FOR KANNADA LANGUAGE

BHUVANESHWARI C. MELINAMATH

Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, Andhra Pradesh, India

ABSTRACT

We have developed a wide coverage Hierarchical morpho-syntactic annotator system for Kannada language using hierarchical tag set. Developing annotator using hierarchical tag set is another new attempt, as there are no such systems exists for Kannada or any Indian Language. Attempts tried so far are using flat tag set. Our annotating system relay on five resources 1) Tag set 2) Dictionary 3) Morphological system 4) Named entity recognizer. Morphological system is developed using well defined saMdhi rules and using finite state transducer (FST) transition file shows the order of suffixation. The architecture is general and can be adopted for other language families just by replacing morph relevant information files. There is no hard coding. The system takes a Kannada sentence as input and gives POS tag/tags for each word of the sentence as output. There is not much work is done in automatic processing of Kannada language.

The major types of morphological process like inflection, derivation, and compounding are handled in this system. The goal of our work is to create a new computational model within the framework of finite state technology that will account for word formation processes in Kannada language. Annotation plays an important role in Natural Language Processing applications such as Parsing, Information Extraction, Information Retrieval and Machine Translation. Results are encouraging with respect to noun as compared to verbs. More than 90% results are obtained for nouns and around 85% results are obtained for verbs.

KEYWORDS: Part of Speech (POS), Natural Language Processing (NLP), Finite State Transducers (FST)

INTRODUCTION

Natural Language Processing (NLP) is an area which is concerned with the computational aspects of the human language. The goal of the NLP is to analyze and understand natural languages used by humans and to encode linguistic knowledge into rules or other forms of representation. Statistical and machine learning algorithms have taken a lead over complex linguistic grammar. There has been a great progress in natural language processing, through the use of statistical methods trained on large tagged corpora. Statistical tagging approaches are able to tackle inherent ambiguity problem by assigning a probability to each word. One more drawback of statistically based algorithms is that they require large quantities of annotated data for training.

Annotation refers to task of assigning an appropriate POS tag for each word in a sentence. Annotation plays an important role in Natural Language Processing applications such as Parsing, Information Extraction, Information Retrieval and Machine Translation. The main aim of the system is to perform annotation for Kannada. Annotators are not yet available for many Indian languages including Kannada. Suitable POS tagsets are also not available.

Our Hierarchical annotator includes development of morphological analyzer, development of 30000 words dictionary using the hierarchical tag set and named entity recognizer to tag proper nouns in the incoming text.

We have also developed hierarchical tagset at the early stage of our research. Developing any system using hierarchical tag set is effortful job as compared to those developed using flat tag set. Nouns in English has maximum of 2 inflection cases for any noun for example boy has boy, boys as its forms. But the same noun in Kannada has around 250 different inflections forms one can observe the complexity of morphology in Kannada Language. Similarly any verb in English has around 5 inflection forms for verb maaDu (do) as does, doing, done did etc. But in Kannada there are more than 3000 forms for the same verb do, "maaDu". Developing a morphological system for Languages like Kannada is challenging. The morph module deals with word formation process. The morph module incorporates morphotactics and saMdhi rules which are generally referred as orthographic rules. In Kannada Suffix gets added to the root word to form the complex stem. The order of attaching the suffixes is rule governed.

LITERATURE SURVEY

Looking at the scenario of Indian Languages. We have several organizations within and outside the country seriously engaged in research and development in computational linguistics and NLP.

In general there are several approaches attempted for developing morphological analyzer and Generator worldwide [3] [4] [5] [6]. In 1983 Kimmo Koskenniemi developed a two-level morphology approach, where he tested this formalism for Finnish language [1]. In his two level representation. In 1996, Beesley developed an Arabic finite state transducer for MA using Xerox finite state transducer (XFST) [2]. In case of Indian languages, AU-KBC Research Centre of Anna University developed a finite state automata based morphological analyzer for Tamil language [20]. Previous works shows that very little work is done in computational aspect of Kannada Prof. Kavi Narayan Murthy “Network and Process Model” [18] which handles only inflectional morphology, IIIT Hyderabad and Dept. of Calts have done some work in respect of Kannada Morphology, but the approach is paradigm i.e. suffix list based. Performance of the system is limited by size of the dictionaries and derivational morphology is not handled. Finite state technique is vital tool in the implementation of natural language morphology[7][8]. Part of Speech tagger for Indian languages using machine learning approaches in [9][10][11].

An idea regarding hierarchical tag set is proposed for the first time by Leech and Wilson [21], we have adopted this standard guidelines of EAGLES with few modifications for the development of our hierarchical tag set for Kannada. The morphological analyzers rely on two sources of information: a dictionary of valid lemmas of the language and a set of rules for inflection handling. To cover aspect of proper noun, we have developed a rule based NER as part of our annotator system. NER has drawn more attention from NLP researchers since the last decade 1998. [15],[14][17][18]. Asif exbal[12] proposed machine learning approaches for ner, Shilpi[16] proposes different combination of machine learning approaches for Hindi.

Preliminary studies show that the dictionaries and morphological analyzers are incomplete and imperfect and need substantial improvements to develop a good Annotation system. Very little work is done in computational aspect of Kannada Prof. Kavi narayan Murthy "Network and Process Model" [19], IIIT Hyderabad and Dept. of calts have done some work in respect of Kannada Morphology. Our annotation system consists of five major modules like tag set, dictionary, morphological analyzer, and NER. Kannada language is not computationally explored. This situation is also motivated us to move in this direction.

PROPOSED ANNOTATOR ARCHITECTURE

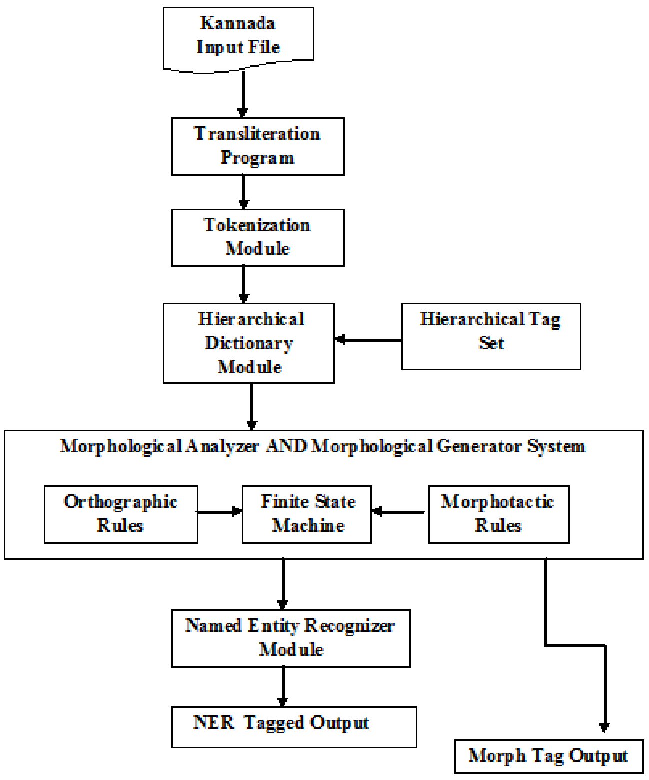

Figure 1: Proposed Architecture of Hierarchical Annotator

The architecture of proposed Hierarchical annotator system is as shown in figure 1. The system takes an input a word or it may be file. The input file may be in Unicode or Iscii. Initially it is converted in to Romanization form (transliterated form) using the convertor program ir.PL or Ur.pL which developed at university of Hyderabad by us. The using tokenization module the input is tokenized. Tokenization module performs preprocessing like removing of any unwanted characters or blank lines etc. The tokens are searched for their existence in the dictionary. If the word is found in the dictionary the dictionary tag is assigned to the word. Otherwise it is passed to morphological system. The morph module checks for its inflections or derivational features and analyzes the word and assigns the tag using the hierarchical tag set. The tag assigned by dictionary and morph module are hierarchical in nature, they give detail analysis of the given word. If the input word is not analyzed by morph module then it may be proper noun or non Kannada word, if it is proper noun then it is analyzed by Named entity recognizer module.

Morphological Generator->Input

root/stem + affix(es)

Output: Word-form (valid)

Morphological Analyzer ->Inflected Input stem

Output: Tagged output

Morphotactic Rules: Gives the order of suffixation. The model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes during word formation. The rule that Kannada plural morpheme follows the noun stem rather than preceding it and case suffix comes before clitic suffix are governed by the grammar of the language.

Root + Number+Case+Clitic1+Clitic2

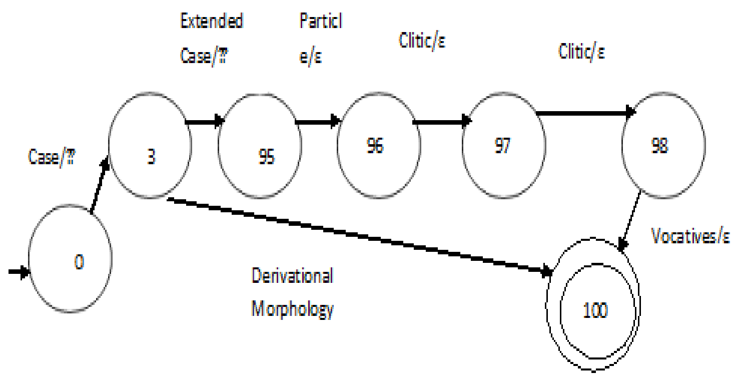

Figure 2: FSM for Noun Morphotactic

Orthographic Rules: These are glide insertion rules used to model the changes that occur usually when two morphemes combine. For example, insert “y” as glide if the lexical word ends with "e" (or i,). The following examples illustrate the glides insertion in accusative case. There are 7 cases identified and 9 case like suffixes are also identified. The rule vary with the cases.

huDuga + annu => huDuga + n + annu➔huDuganannu

mara + annu => mara + v + annu➔maravannu

taMde + annu => taMde + y + annu➔taMdeyannu

nore + annu => nore + y + annu➔noreyannu

shatru + annu => shatru + y + annu➔shatruvannu

hasu + annu => hasu + v + annu➔hasuvannu

niiru + annu => niir + ε + annu➔niirannu: - nothing is inserted here because ‘u’ is enunciative

kavi + annu => kavi + y + annu ➔kaviyannu

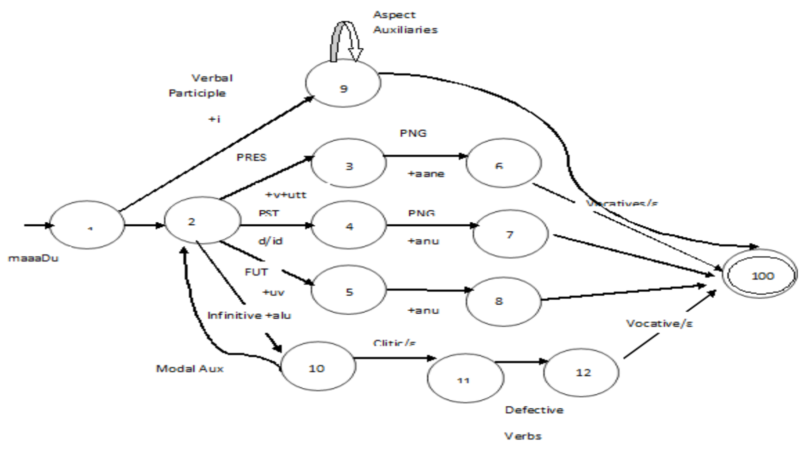

Figure 3: Sample FSM Trastion Governing Orthographic Rule

Consider the input word boy "huDuganuu", "boy too", our annotator system tags the word as shown below, input may be even text file.

huDuganuu 1: [huDuga: N-COM-COU-MS-NOM]: singular - nominative -inclusive - null.

Here the word is in inflected form with inclusive clitic. Given any file the file tokenized and each token is searched in the morph related lexicon, if found then it tagged with dictionay tag and given as output, otherwise it is passed to morphological analyzer to analyze its inflections, the morph system makes use of FST transitions rules, morphotactics, orthographic rules, hierarchical module, and also dictionary module for its task, to produce the tagged output.

It the input is proper noun then it passed to NER module for tagging.

Finite-State Transducers

Finite State Automaton (FSA) is a model of computation consisting of a finite set of states, a start state, an input alphabet, and a transition function that maps input symbol and current state to next state. A state transition usually has some rules associated with it that govern when the transition may occur. The result given by a FSA is limited: either the string is accepted, or it is rejected. Finite state technology has some important advantages, making it most appealing for implementing natural language morphology. However, it is usually very easy to build a finite state machine to describe a specific morphological phenomenon. Finite state techniques are a vital tool in the implementation of natural language morphology.

The transducer is defined as T = (Q, L,δ, qI, F,) where Q is a finite set of states, L a set of transition labels, qI Є Q the initial state, F ⊆ Q the set of final states, and $\delta: Q \times L \rightarrow 2^Q$ the transition function (where 2^Q represents the set of all finite sets of states). The set of transition labels is $L = (\Sigma \cup \{ \varepsilon \} \times (\Gamma \cup \{ \varepsilon \})$ where Σ is the alphabet of input symbols, Γ the alphabet of output symbols, and ε represents the empty symbol. According to this definition, state transition labels may therefore be of four kinds: (σ : γ), meaning that symbol σ Є Σ is read and symbol γ Є Γ is written (σ : ε), meaning that a symbol is read but nothing is written; (ε : γ), meaning that nothing is read but a symbol is written; and (ε : ε) means that a state transition occurs without reading or writing. The last kind of transitions are not necessary neither convenient in final FSTs, but may be useful during construction. It is customary to represent the empty symbol ε with a zero (“0"). A letter transducer is said to be deterministic when δ : Q × L→ Q. Note that a letter transducer which is deterministic with respect to the alphabet $L = (\Sigma\cup\{ \varepsilon \} \times (\Gamma \cup\{ \varepsilon \})$ may still be nondeterministic with respect to the input Σ.

A string w ' Є Γ * is considered to be a transduction of an input string w Є Γ * if there is at least one path from the initial state qI to a final state in F whose transition labels form the pair w : w ' when concatenated. There may in principle be more than one of such paths for a given transduction; this should be avoided, and is partially eliminated by determinization. On the other hand, there may be more than one valid transduction for a string w (in analysis, this would correspond to lexical ambiguity; in generation, this should be avoided). In analysis, the symbols in Σ are those found in texts, and the symbols in Γ are those necessary to form the lemmas and special symbols representing morphological information, such as <noun>, <feminine>, <first person p1. second person p2. Third person p3> for pronouns, etc. In generation, Σ and Γ are exchanged.

The general definition of letter transducers is completely parallel to that of non-deterministic finite automata

(NFA) and that of deterministic letter transducers, parallel to that of DFA; accordingly, letter transducers may be determinized and minimized (with respect to the alphabet L) using the existing algorithms for NFA and DFA (Hopcroft & Ullman 1979; Salomaa 1973; van de Snepscheut1993). Transitions labeled (ε : ε) may be eliminated during determinization using a technique parallel to ε closure. For all one writes if and only if *q* can be reached from *p* by going along zero or more ε arrows. For any, the set of states that can be reached from *p* is called the **epsilon-closure** or ε **-closure** of *p*.

For any subset, define the ε -closure of *P* as for any P⊂Q subset, define the ε -closure of *P* as Which allows a transformation to a new state without consuming any input symbols. For example, if it is in state 1, with the next input symbol an *a*, it can move to state 2 without consuming any input symbols, and thus there is an ambiguity: is the system in state 1, or state 2, before consuming the letter *a*.

Because of this ambiguity, it is more convenient to talk of the set of possible states the system may be in. Thus, before consuming letter *a*, the NFA-epsilon may be in any one of the states out of the set {1,2}. Equivalently, one may imagine that the NFA is in state 1 and 2 'at the same time': and this gives an informal hint of the power set construction 2^Q.

Hierarchical Tag Set

NLP application like parsing, desires that, all linguistically useful information is to be encoded in the tag. So hierarchical tag set is desirable as it includes all the detailed aspects of the word. Parsing needs all kind of morphosyntactic information, and also semantic information. Flat tagsets can not capture finer language specific properties because of their rigid nature so they cannot be used for annotating across a group of languages. Most of English tagsets like Penn, Brown C5, C7 and existing tagsets are flat tagsets. But in hierarchical tagset tag elements are structured relative to one another. Hierarchical tagset will contain a small number of categories because of tree like extensible structure.

The EAGLES guidelines was based on this concept of hierarchical tagsets. The morphosyntactic details are encoded in the separate layers of hierarchy. major categories at the top level progressing down to cover morphosyntactic features structure is category-type-feature. Major word classes should be top in the hierarchy, followed by sub classifications and lastly morphological features. An issue of general concern is that in an effort to reduce the number of tags we should not miss out on crucial information related to grammatical and other relevant linguistic knowledge which is encoded in a word, particularly in agglutinating languages like Kannada. It is better to encode all features in word. We have used 136 atomic tag elements to tag the categories.

Dictionary Module

Developing a knowledge base of legal words is an important task in computational linguistics. Electronic dictionary are assets for computational linguistics. Using semiautomatic methods we have developed electronic dictionary using DOE CILL Corpus. Simple techniques such as heuristic pattern matching using regular expressions, frequency vs. rank kind of knowledge, stemming and filtering technique are used are used in building our dictionary. Around 30000 plus words dictionary with hierarchical tag set is designed. Sample of the dictionary is shown below figure 4. Department of electronics (DoE) CIIL corpus is used for developing this dictionary. A total of 112 suffixes patterns are used for extraction of words from corpus.

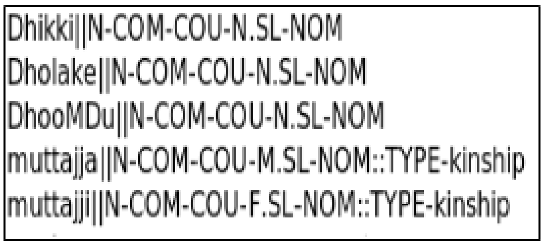
Dhikki||N-COM-COU-N.SL-NOM
Dholake||N-COM-COU-N.SL-NOM
DhooMDu||N-COM-COU-N.SL-NOM
muttajja||N-COM-COU-M.SL-NOM::TYPE-kinship
muttajji||N-COM-COU-F.SL-NOM::TYPE-kinship

Figure 4: Sample Dictionary Entries

Named Entity Recognizer Module

Named Entity Recognition (NER) is an important task in many Natural Language Processing (NLP) applications. We have developed a rule based NER system for Kannada. Named entity recognition is the task of identifying proper names in the text. Rule based approach gives better results than machine learning. Since machine learning techniques require huge amount of tagged We have proposed NER algorithm which can recognize three classes of NEs: In recognition phase we have used pre-defined rules generated from the language knowledge. And manually developed proper noun dictionary of 5000 words for this task. Our's is the first attempt in this regard for Kannada. Our approach for NER is language independent, as there is no hard coding in the program, it can be used for any of the Indian Languages. Main focus here is to recognize and classify the NEs as person name, organization name, place name.

We have used suffix, prefix, gazetteer’s information for classification of the tokens. In English language, lots of work has been done in NER. Here we emphasis on identification of Kannada names which is still open in Kannada. The lack of capitalization is, however, an intrinsic feature of Kannada, thus the NER task in Kannada becomes immediately harder than in English. The ambiguity caused by this feature is moreover increased, as most Kannada proper nouns are indistinguishable from forms that are common nouns and adjectives. In addition, Kannada is a highly morphological language, thus posing more challenges for the NER task. Our results are encouraging precision is around 86%. The NER problem has received much attention, as NER forms the basic building block of any Information Extraction system.

Challenges in Kannada Language

The following features in Kannada makes the named entity recognition a difficult task.

- Kannada lacks capitalization this information, which plays a very important role in identifying Nes,
- Non-availability of large gazetteer, lack of standardization and spelling,
- Number of frequently used words (common nouns) which can also be used as proper names are very large. “Also the frequency with which they can be used as common noun as against person name is more or less unpredictable.
- Lack of labeled data makes named entity more complex,
- Scarcity of resources and tools: There is Scarcity of resources and tool in Kannada like other Indian languages, Kannada is also a resource poor language. Annotated corpora, name dictionaries, POS taggers etc. are not yet available in the required quantity and quality.

- Ambiguity makes difficult to decide the sense, there can be semantic ambiguities. Ambiguities of NEs for example hubbaLLi is place name and also surname representing person. All the above features makes NER task in Kannada a difficult task.

We have developed separate named entity tag set, for person identification, location, organization identification, numbers and time etc. To evaluate the NER system we randomly downloaded corpus from famous Kannada News paper PrajavaaNi website around 20 files of varying length and ran our NER. We observed that around 79.52% NEs are identified correctly, 12.34% are identified wrong and around 8.56% are left unrecognized.

RESULTS

To evaluate the developed annotator methodology, we selected top 10000 words of CILL Corpus. Gold Standard data of 14400 pronouns with all possible inflected word types and 3119 gold standard data for verb maaDu (do). We have used three parameters, namely, precision, recall and F-measure. The expressions precision, recall and F-measure are given in (1), (2) and (3) respectively. Precision is the percentage of correct positive recognitions obtained by the methodologies. It is computed as the ratio between the number of words correctly identified by the methodology. True Positives (TP) and the total number of words returned by the system. The precision is calculated by dividing TP by the sum of TP and false positives (FP).

$$precision = \frac{Tp}{Tp+Fp} \cdots \tag{1}$$

$$\mathrm{Re}call = \frac{TP}{TP+FN} \tag{2}$$

Recall indicates the percentage of positive cases recognized by the system. It is computed as the ratio between the number of words correctly identified by the system (TP) and the number of words that the system was expected to recognize. Thus, Recall is the number of (TP) divided by the sum of (TP) and false negatives (FN).

$$F = \frac{(\beta + 1) * \mathrm{Pr}ecision * \mathrm{Re}call}{\beta(\mathrm{Pr}ecision + \mathrm{Re}call)} \tag{3}$$

F-measure is the common weighted harmonic mean between Precision and Recall, where beta is the weighting factor. We carried out the experiments by varying the testing data and observe that, our precision and recall are good, and having nearer values**.**

Table 1: Showing Recognition Efficiency

Data for Testing	Size of Corpus Words	Not Recognized Words	Recognized Words
Pronoun gold Standard data	14400	68	14332
Verb gold standard data	20000	3009	16991
Kvk raw Corpus	2744	1400	1344
Top raw CIIL Words	10000	8998	1002
Noun gold Standard data	2500	250	2550

Here Gold standard data is data manually generated by us, consideriong all possible inflections. We took all types of pronouns their case and clitic inflections generated valid forms. Similarly for nouns also we selected two words ending with a, e, i, o, u, one representing human and other non human and manually generated inflections (Only valid forms) for testing purpose.

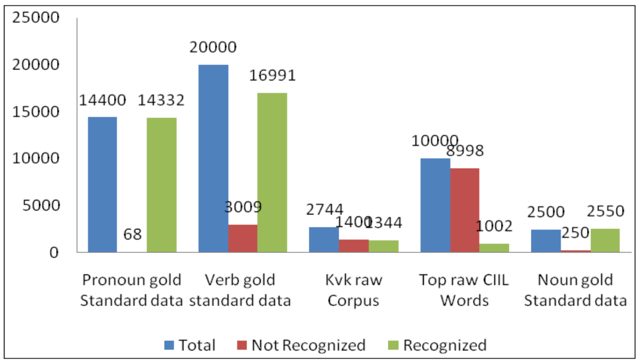

Figure 5: Recognition of Word Types for Different Corpus

A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix. The following table shows the confusion matrix.

Table 2: Confusion Matrix

Corpus	TP	TN	FP	FN	Precision	Recall	F-Measure 2pr/p+r
14400	14332	0	68	0	99.5	100%	99.5%
2744	1344	300	400	300	77%	81%	78.5%
2500	2250	0	250	0	90%	100%	94.5%
10000	8019	1350	300	400	93%	95%	93%
20000	17000	0	3000	0	85%	100%	91 %

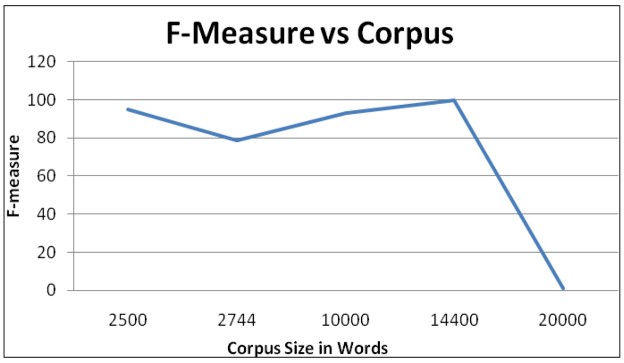

Figure 6: Corpus Size verses F-Measure

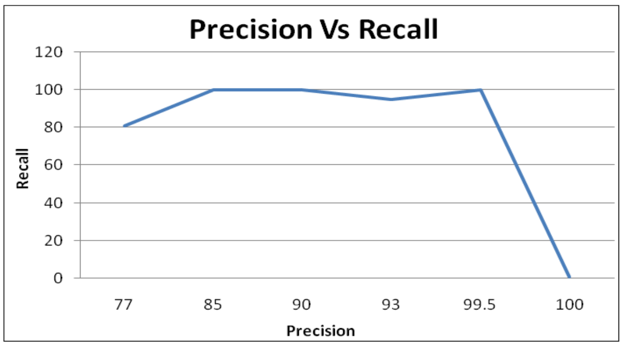

Figure 7: Precision versus Recall

Precision versus Recall graph shows that our precision and recall values are not differing much. In a recognition task, the precision for a class is the *number of true positives* (i.e. the *number of items correctly labeled as belonging to the positive class*) *divided by the total number of elements labeled as belonging to the positive class* (i.e. the sum of true positives and false positives which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the *number of true positives divided by the total number of elements that actually belong to the positive class* (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been).

A Recognition graph is a plot with F-measure on the Y axis and the corpus size in words on the X axis. **F-measure**, which is the *weighted harmonic mean of precision and recall* from figure 7. We observe that there is good recognition and there is increase in recognition rate as we increase the corpus size in no of words, but however this is not always the case, it depends on the corpus we are choosing.

The first 10000 most frequent Kannada words of Department of electronics (DoE), Central Institute of Indian Languages (CIIL) corpus words are selected which included all the nominal, pronominal, adjectival and verbal inflections of the Kannada language. Around 80 % words were analyzed correctly, manually checked and verified, remaining 20% words were a mixture of spelling variations, dialect variations, compound words, hence such words in the raw corpus selected for testing are not analyzed. This system is first of its kind for Kannada using FST and hierarchical tag set.

A sample output of morphological analyzer for noun "boy". huDuga.

A sample output of morphological analyser for noun "boy". huDuga.

1: huDuganu 1: [huDuga: N-COM-COU-MS-NOM] : singular - nominative -null -

2: huDuganuu 1: [huDuga: N-COM-COU-MS-NOM] : singular - nominative -inclusive - null

3: huDuganee 1: [huDuga: N-COM-COU-MS-NOM] : singular - nominative -emphatic - null

4: huDuganoo 1: [huDuga: N-COM-COU-MS-NOM] : singular - nominative -indefinite - null

5: huDuganaa 1: [huDuga: N-COM-COU-MS-NOM] : singular - nominative -interrogative -

6: huDuganuunaa 1: [huDuga: N-COM-COU-MS-NOM] : singular - nominative- inclusive - interrogative

7: huDuganeenaa 1: [huDuga: N-COM-COU-MS-NOM] : singular - nominative- emphatic - interrogative

658: hasugaLigiMtaluu

1: [hasu: N-COM-COU-NS-NOM-REALU] : plural - comparative - inclusive - null

659: hasugaLigiMtalee

1: [hasu: N-COM-COU-NS-NOM-REALU] : plural - comparative - emphatic - null

660: hasugaLigiMtaloo

1: [hasu: N-COM-COU-NS-NOM-REALU] : plural - comparative - indefinite - null

661: hasugaLigiMtalaa

1: [hasu: N-COM-COU-NS-NOM-REALU] : plural - comparative - interrogative -

1: [maaDu: V-] : 15 17 +uvudu= gerund 4 +(u)= nominative 91 +uu= inclusive_clitic 95 98 100

3486: maaDuvuvaadaruu (i): (ii) = maaDuvu + [aadaruu : CONJ-SUB] ???

3487: maaDuvuvallade (i): (ii) = maaDuvu + [allade : CONJ-COOR] ???

3488: maaDuvuveMbadannu (i): ???

A Sample morph output for the verb 'Do' maaDu and its ambiguity cases.

1: maaDide Ex: a) adu maaDide;'It has Done'. b) naanu maaDide; 'I have done. c) niinu maaDide. 'you have done it'.

We have got 3 analysis for this word maaDide all 3 analysis are correct. However this ambiguity can be resolved by seeing person agreement at higher levels.

Morph output: 1: [maaDu: V-] : 15 +i_u= cjp 30 +ide=perfective_present_p3_n_singular 91 98 100 2: [maaDu: V-] : 15

17 +id= past 20 +e= p2_mfn_singular 91 98 100 3: [maaDu: V-] :

15 17 +id= past 20 +e(nu)= p1_mfn_singular 91 98 100

Disambiguation rule: By seeing person agreement, and subject gender.

2: maaDaballa

- avanu maaDaballa
- adu maaDaballa kelasa.

1:[maaDu: V-]: 15 17 +al(u)= infinitive 25 26 +balla= rp_capabilitative100

2: [maaDu: V-]: 15 17 +al(u)= infinitive 25 26 +ball(a)=capablitive 100

Disambiguation rule: rp is precedes a noun and it will not come in final position. Where as capablitive comes in verb final position and subject agreement is avanu.

3119 words seen. Overall Performance: (87%)

1 word(s) found in dictionary: (0 %)

2717 word(s) analyzed by Morph: (87 %)

401 word(s) could not be analyzed: (13 %)

624 word(s) had more than one analyses. (20 %)

Average Ambiguity per Word: 1.30

Maximum number of analyses: 9

Not analyzed words were a mixture of spelling variations, dialect variations, compound words, hence such words in the raw corpus selected for testing are not analyzed, regarding ambiguity error is very less.14000 different inflected words forms of pronoun are generated manually and ran through morph and all the word forms are analyzed correctly. 100%correct results are obtained here. It is shown here that each noun in Kannada has about 250 word forms for single noun. One can observe the complexity in morphology as compared to English, which has only three forms like say boy, boys, boy's. Similarly in English maximum inflections for a verb 'go' are 5 in number like, go, gone, went, going, goes. But in Kannada it is proved that more than 3000 different forms exists. One can understand the complexity of Kannada. Developing tools like morphological analyzer or generator is quite challenging.

REFERENCES

1. K. Koskenniemi, “Two-level morphology: A general computational model for word form Recognition and production”. Master’s thesis, Department of Genera Linguistics, Helsinki University, 1983.
2. Kenneth R Beesley, “Arabic morphology using finite state operations” ph.D Thesis, 1983.
3. John Chen et al. “Towards automatic generation of natural language generation system,". *Proc. of the 18th International Conference on Computational Linguistics,* New York City, USA, 2000.
4. Goldsmith and Gaussier, “Unsupervised learning of derivational morphology from inflection lexicons”. *Proc. of ACL Workshop proceedings,* pp. 24–30, 1999.
5. Goldsmith, “Unsupervised learning of the morphology of a natural language". *Computational Linguistics*, pp. 153–193, 2001.
6. Creutz M, “Unsupervised segmentation of words using prior distributions of morph length and frequency”. *Proc. of the Association for Computations Languages (ACL03),* pp. 280–287, Sapporo, Japan, 2003.
7. E Roche and Y Schabes, “Introduction finite state language processing”. MIT Press, 1997.
8. Hopcroft J.E and J D Ullaman., “Introduction to automata theory languages and Computation”. Addition Wesley, 1979.
9. Sivaji Bandyopadhyay, Asif Ekbal, and Debasish Halder, “Hmm based pos tagger and rule-based chunker for Bengali”. *Proc. of NLPAI Machine Learning Contest* 2006.
10. Sandipan Dandapat and Sudeshna Sarkar, “Part of speech tagging for Bengali with Hidden Markova model”. *Proc of NLPAI Machine Learning Contest* 2006.
11. Aniket Dalal, Kumar Nagaraj, Uma Sawant, and Sandeep Shelke, “Hindi part-of speech tagging and chunking: A maximum entropy approach”. *Proc. of* NLPAI *Machine Learning Contest*, IIIT, Hyderabad, India, 2006
12. Ekbal and S. Bandyopadhyay. “Named entity recognition in Bengali: A Conditional random field”. *Proc. of ICON*, pp 123–128, 2008.
13. Michael Fleischman. “Automated sub categorization of named entities”. *Proc. of Conference of the European Chapter of Association for Computational Linguistic*, pp 25–30, 2001.

14. Grishman. “The nyu system for muc-6”. *Proc. of Sixth Message Understanding Conference (MUC-6)*, pp 167–195, Fairfax, Virginia. 1995.

15. YUNGWEI DING HSINHSI CHEN and SHIHCHUNG TSAI. “Named entity extraction for information retrieval”. *Proc. of HLT-NAACL*, pp 8-15, 2003.

16. Mukund Sangalikar, Shilpi Srivatsava and D.C. Kothari. “Named entity recognition System for Hindi language”. *International journal of Computational Linguistics* vol. (2), pp 10–23, 2011.

17. Kashif Riaz. “Named Entities Workshop”. *Proc. of Association for Computational Linguistics ACL*, Uppsala, Sweden., pp. 126–135, 16 July 2010.

18. Asahara, Masayuki and Matsumoto. “Japanese Named Entity Extraction with Redundant Morphological Analysis”. *Proc. of Human Language Technology conference* – North American chapter of the Association for Computational Linguistics, 2003.

19. Kavi Narayana Murthy. A network and process model for morphological analysis/generation. In Proc. International Conference on South Asian Languages, Punjabi University, Patiala, India, 9-11 January, 1999.

20. Vijay and Shobha. Tamil morphological analyzer. http://nrcfosshelpline.in/smedia/images /downloads/Tamil Tagset-opensource.odt

21. Leech G and Wilson, “A. Recommendations for morph syntactic annotation of corpora”. EAGLES Technical report, 1996.