Kannada Morphological Analyser and Generator Using Trie

Shambhavi. B. R[*], Dr. Ramakanth Kumar P[#], Srividya K[†], Jyothi B J[††], Spoorti Kundargi[*], Varsha Shastri G[†††]

[*]Department of CSE, R V College of Engineering, Bangalore
[#]Department of ISE, R V College of Engineering, Bangalore
† Yahoo! India Software Development Center, Bangalore
†† Oracle India Pvt Ltd, Bangalore
††† National Instruments India, Bangalore

Summary-
Morphological Analyser and Generator tool is an essential component of any NLP application. This paper presents the morphological analyser and generation tool for the South Indian language of Kannada language using paradigm approach. The application uses trie as the datastructure for the storage of suffixes and root words. Though there have been attempts of building a Morphological Analyser for Kannada in the recent past, no full fledged analyser for Kannada has been built. The performance demonstrated by our application has been really encouraging.

Keywords-
Natural Language Processing, Morphological Analyser, Paradigm, Trie

1. Introduction

Morphology deals with the study of internal structure of words A Morphological Analyser takes a complete word form as input and produces the structure, the syntactic and morphological properties of the word. A generator, on the other hand, does the reverse of an analyser. It generates a word given a root and its features (affixes) . It includes two modules namely noun generator and verb generator. In the Noun generator, the inputs are a root noun, oblique form, plural marker, case marker and postpositions. In the verb generator, a root verb, tense marker, number, person, gender marker, relative participle suffix and verbal participle suffix are given as inputs.

Morphological analysis is a vital step for languages with complex morphology like Kannada. Kannada is one of the major Dravidian Languages of India. There are around 40 million speakers and is the 27th spoken language in the world. The language exhibits a very rich and complex system of morphology. It is a diglossic language .A lot of distinctions exist between the written and the spoken forms of the language. There are around 20 dialects. But the written form is more or less the same. The tool developed here is restricted to only the literary variety.

The paper is organized as follows. The next section discusses the current state of the art of morphological analysis for various languages. Section 3 will cover the morphological details of Kannada. Design of paradigms and Datastructure used in the application developed are detailed in section 4 and 5 respectively. Implementation and other features are explained in the next section followed by a brief description of results obtained. The final section concludes the paper along with future work.

2. Previous work

Morphological analysis work has been successfully done for languages like English, Chinese, Arabic and European languages. The history of morphological analysis dates back to the ancient Indian linguist Panini, who formulated the 3,959 rules of Sanskrit morphology. One of the breakthrough works in English was done by John Carroll [1] and his team. They developed a morphological analyzer in English ‘morpha’ which was based on finite-state techniques. Various other techniques have been experimented for morphological analysis since then.

In the last decade, extensive work has also been done for Indian languages with respect to grammar analysis. IIIT Hyderabad has developed analysers for Hindi, Marathi, Telugu, Kannada and Punjabi. It is freely available for download from LTRC website for Linux operating system. A paradigm based Hindi analyser and generator has been discussed in [2]. Jisha. P Jayan et al. [3] have compared three methods to morphological analysis namely paradigm approach, suffix stripping and hybrid methods for Malayalam language. An open-source analyser using finite state technology for Bengali has been experimented in [4]. Initial work on Tamil morphological analysis was taken up by Anusaraka group. Finite automata based approach was adopted in [5]. Machine learning approach based on SVM [6] along with sequence labelling for Tamil gave accuracy of 95.45%.

Not much has been done in this research area for Kannada language. MORPH- A network and process model for Kannada morphological analysis/ generation was developed by K. Narayana Murthy as part of the project entitled "Machine Aided Translation from English to Kannada"[7]. Vikram and Shalini [8] have attempted to

Manuscript received January 5, 2011
Manuscript revised January 20, 2011

build a prototype Kannada analyser based on Finite State Machines and can handle only 500 noun and verb stems. Our tool is based on the paradigm based algorithm developed by Akshar Bharati et al. [9] for morphological analysis of Indian languages. This approach has proved to be effective for inflectionally rich Indian languages.

3. Kannada Morphology

Kannada is a morphologically rich language in which morphemes combine with the root words in the form of suffixes. Kannada grammarians divide the words of the language into three categories namely i) Declinable words (namapada) ii) Conjugable words or Verbs (kriyapada) and iii) Uninflected words (avyaya). Declinable words are inflected to mark differences of case, number and gender. Nouns, Pronouns and Adjectives are included in this class. Verbs are inflected to mark differences of person, gender, number, aspect, mood and tense. The class of uninflected words include unchangeable words.

3.1 Declinable words

Morphology of declinable words, as in many Dravidian languages is fairly simple compared to verbs. Kannada words are of three genders- masculine, feminine and neutral. Declinable and conjugable words have two numbers- singular and plural. Declinable words have seven cases as depicted in Table 1. The various inflections of the noun stem ಉದ್ಯಾನ *(udyana- 'park')* and its meaning with different suffixes is shown in Table 2.

Table 1: Different Cases and their corresponding Characteristic Suffixes for Nouns

Kannada Name	English Name	Characteristic Suffix
Prathama	Nominative	*0 (nu/ ru/ vu/ yu)*
Dwitiya	Accusative	*annu/ vannu/ rannu*
Tritiya	Instrumental	*iMda/ niMda/ riMda*
Chaturthi	Dative	*ge/ ige/ kke*
Pachami	Ablative	*deseyiMda*
Shashti	Genitive	*a/ ra/ da/ na*
Saptami	Locative	*alli/ nalli/ dalli/ valli*
Sambhodana	Vocative	*ee*

Table 2: Inflections of a noun stem and its corresponding meanings

Inflected Nouns	Meaning in English	Type of Inflection (Number, Case)
ಉದ್ಯಾನ – ವು	Garden	Singular+Nominative
ಉದ್ಯಾನ – ವನ್ನು	The garden	Singular+ Accusative
ಉದ್ಯಾನ – ದಿಂದ	From the garden	Singular+Instrumental
ಉದ್ಯಾನ – ಕ್ಕೆ	To the garden	Singular + Dative
ಉದ್ಯಾನ - ದೆಸೆಯಿಂದ	Because of garden	Singular + Ablative
ಉದ್ಯಾನ – ದ	Of the garden	Singular + Genitive
ಉದ್ಯಾನ – ದಲ್ಲಿ	In the garden	Singular + Locative
ಉದ್ಯಾನ – ಗಳು	Gardens	Plural + Nominative
ಉದ್ಯಾನ – ಗಳನ್ನು	The gardens	Plural + Accusative
ಉದ್ಯಾನ – ಗಳಿಂದ	From the gardens	Plural+ Instrumental
ಉದ್ಯಾನ – ಗಳಿಗೆ	To the gardens	Plural + Dative
ಉದ್ಯಾನ - ಗಳದೆಸೆಯಿಂದ	Because of gardens	Plural + Ablative
ಉದ್ಯಾನ – ಗಳ	Of the gardens	Plural + Genitive
ಉದ್ಯಾನ – ಗಳಲ್ಲಿ	In the gardens	Plural + Locative

3.2 Verbs

The verb is much more complex than the nouns. There are three persons namely first, second and third person. Tense of verbs is past, present or future. Aspect may be simple, continuous or perfect. Verbs occur as the last constituent of the sentence. They can be broadly divided into finite or non-finite forms. Finite verbs have nothing added to them and are found in the last position of a sentence. They are marked for tense with Person-Number-Gender (PNG) markers. Non-finite verbs, on the other hand cannot stand alone. They are always marked for tense without PNG marker. The following table summarises some of the inflections of the verb stem ಮಾಡು.

Table 3: Few inflections of a verb stem and its corresponding meanings

Inflected Verb	Meaning in English	Tense	Aspect	PNG
ಮಾಡುವನು	He will do.	Future	Simple	3SM
ಮಾಡುತ್ತಿದ್ದಾನೆ	He is doing.	Present	Continuous	3SM
ಮಾಡಿರುವಳು	She has done.	Future	Perfect	3SF

ಮಾಡುತ್ತಿದ್ದಳು	She was doing.	Past	Continuous	3SF
ಮಾಡಿದಿರಿ	You did.	Past	Simple	2P-
ಮಾಡುತ್ತೇನೆ	I will do.	Future	Simple	1S-
ಮಾಡಿದ್ದರು	They did.	Past	Perfect	3P-
ಮಾಡಿರುತ್ತದೆ	It did.	Present	Perfect	3SN

3.3 Uninflected words

Uninflected words may be classified as adverbs, postpositions, conjunctions and interjections. Some of the example words of this class are *haage, mele, tanaka, alli, bagge, anthu* etc.

3.4 Morphophonemics

In Kannada, adjacent words are often joined and pronounced as one word. Such word combinations occur in two ways- *Sandhi* and *Samasa*. *Sandhi* (Morphophonemics) deals with changes that occur when two words or separate morphemes come together to form a new word. Few *sandhi* types are native to Kannada and few are borrowed from Sanskrit. We in our tool have handled only Kannada *sandhi*. However we do not handle *samasa*.

Table 4: Sandhi types and examples

Complex word	Simple/inflected words	Sandhi type
ಚೆಂಡಾಟ	ಚೆಂಡು + ಆಟ	ಲೋಪ ಸಂಧಿ
ಸುಂದರವಾದ	ಸುಂದರ + ಆದ	ಆಗಮ ಸಂಧಿ
ಕೈದೋಟ	ಕೈ + ತೋಟ	ಆದೇಶ ಸಂಧಿ

Kannada sandhi is of three types - lopa, agama and adesha sandhi. While lopa and agama take place both in compound words and in the junction of the crude forms of words and suffixes, adesha sandhi occurs only in compound words. Detailed description of sandhi types can be found in [10]. Table 4 gives examples for the various sandhi types handled.

4. Design of paradigms

Our application aims to accomplish the process using a Kannada root words dictionary and rules for analysis in the form of paradigms. The linguist or the language expert is asked to provide different tables of word forms covering the words in a language. All the words in a word class either declinable or conjugable need not share the same paradigm. For example, all verbs would not follow the same paradigm or have the same inflectional pattern. In Kannada language, it can be seen that most of the words ending with ಅ and whose gender is neutral, the suffixes take the form as *vu*, v*annu*, *iMda*, etc. Similarly, those ending with ಅ and whose gender is masculine, the suffixes take the form as *nu*, *annu*, *niMda*, etc. This may not hold good for all words. There are exceptional cases as well which have been handled appropriately. For example, the words *kaadu* and *pashu* are neutral gender and both end with *u*. But both have different suffix rules. Continuing the analysis this way, declinable words were classified into 21 paradigms. Most of the Kannada verbs fall under the identified 6 paradigms. Each verb paradigm handles 135 inflections. Paradigms are not required for uninflected words and a dictionary would suffice. Our tool handles around 3250 declinable stems, 350 verb stems and 100 uninflected words, totalling to around 3700 stems.
Each paradigm table would consist of a set of suffixes for the corresponding word form and a number indicating the number of characters to be removed from the root before appending the suffix to it. For e.g., consider a word ಕಾಡು. If we want to append ನ್ನು to obtain the accusative case, the character ು should be removed from the root ಕಾಡು. Hence the number corresponding to this table for accusative case is 1. By following this approach we can substantially reduce the space required to store all forms of words. We have seen that most of the words in the language will fit in one of these categories. However there may be exceptional cases for which we can write separate set of rules and update the tables.

5. Datastructure used

A trie, or prefix tree, is an ordered tree data structure that is used to store an associative array where the keys are usually strings. The trie data structure is used for searching a string in a large database of strings. For the storage of roots, trie is build from left to right rather than backwards. A separate trie is constructed to handle suffixes corresponding to each paradigm class. This trie is linked to all the roots which map to the same paradigm class.

Figure 1 depicts the trie data structure built for four root words- *raama*, *ramaa*, *kavana* and *kavi*. Here every node contains a Unicode character corresponding to Kannada script. Trie enhances the searching speed.

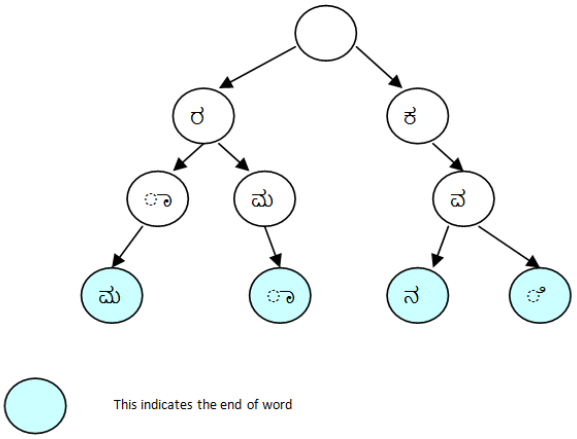

Fig. 1 Example of Trie Datastructure

6. Implementation and other Features

The application starts off with the initialization process which creates the trie structure of roots and paradigm tables. The user then makes a choice of either the analyser module or the generator module. In the analyser module, input is obtained from the user who types the Kannada sentences in a text field provided. The segmentation routine tokenizes the input sentence into its constituent words delimited by white space. These words are then given to the word analyser. In the word analyser, the input word is searched for in the indeclinable words dictionary and root dictionary. It is then given to noun and verb analyser. If the word can be analysed by the noun and verb analyser, the information corresponding to the word is collected from the paradigm table and is displayed. Else it is seen that the word may be a complex word and is given to the *sandhi* analyser. Here we match the longest suffix with the existing corpus, strip that suffix off the word once it matches and continue this process until all characters in the word are stripped off.

A similar approach is used for the generator. The generator has two forms, namely the noun generator and verb generator. It takes the root word and suffix information as its input. It then searches for the root in the dictionary of roots and if it matches, it retrieves the suffix for the corresponding root. It then appends the corresponding suffix to the root and displays it to the user. For example, if root word is ರಾಮ and if we select ಪುಲ್ಲಿಂಗ, ಏಕವಚನ, then the output of the generator will be ರಾಮನು.

Exceptional cases are handled by hard coded rules. For example, in the noun generator module, trying to generate singular nominative case of ಜನ (*jana,* 'people') would be an illegal combination. We have tried to handle many such exceptions efficiently.

We have used the Unicode format to store the Kannada characters in the trie structures. Also, the input is taken in the Unicode format. We do not make use of any external database. However text files are used for storage. The tool used to type the Kannada characters is a virtual keyboard which supports all Kannada characters.

7. Results

When we tested the analyser and generator components of the application, we obtained the following results.

Morphological Analyser

- Input: ನರಿಯ
- Output: [Root] ನರಿ, [Type] Noun, [Gender] N, [Case] Genitive, [Number] S

- Input: ಬರೆದಿರುತ್ತಾರೆ
- Output: [Root] ಬರೆ, [Type] Verb, [PNG] 3P-, [Tense] Present, [Aspect] Perfect

- Input: ಚೆಂಡಾಟವಾದ್ದರಿಂದೆಂದು
- Output: ಚೆಂಡಾಟವಾದ್ದರಿಂದ + ಎಂದು→ *lopa sandhi*
  ಚೆಂಡಾಟ + ಆದ್ದರಿಂದ →*agama sandhi*
  ಚೆಂಡು + ಆಟ → *lopa sandhi*

Morphological Noun Generator

- Input: root- ಕಾಡು, number- plural, case- Genitive.
- Output: ಕಾಡುಗಳ

Morphological Verb Generator

- Input: root- ಬರೆ, PNG- 3SF, Tense- past, Aspect-simple.
- Output: ಬರೆದಳು

8. Conclusion & Future Work

Development of a morph analyser and generator for Kannada language is a challenging task as the language is both agglutinative and morphologically very rich. Our application is rule based with paradigm approach and in order to permit rapid matching of suffixes and root words, trie Datastructure has been adopted. The only disadvantage of trie is that it consumes more memory as each node can have at most 'y' children, where y is the alphabet count of the language. The developed tool can handle up to 3700 root words and around 88K inflected words.

Morphological analysis forms the pre-processing activity of various NLP tools like POS tagger, spell checker, stemmer, machine translation etc. In future we plan to extend this work to increase the root word count to 30,000 and later develop a morpheme component based POS tagger.

References

[1] J. Carroll, Minnen G. and D. Pearce, "Applied morphological processing of English", Natural Language Engineering. 2001.

[2] Vishal Goyal, Gurpreet Singh Lehal, "Hindi Morphological Analyzer and Generator", Proceedings of First International Conference on Emerging Trends in Engineering and Technology, 2008, IEEE, pp 1156-1159

[3] Jisha. P. Jayan, Rajeev R. R, S. Rajendran, "Morphological Analyser for Malayalam- A Comparison of Different Approaches", International Journal of Computer Science and Information Technology, Vol. 2,No. 2, Dec 2009, pp 155-160.

[4] Abu Zaher Md. Faridee, Francis M. Tyers, "Development of a morphological analyser for Bengali", Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation, Spain, November 2009, pp 43-50.

[5] Anandan. P, Ranjani Parthasarathy, Geetha T.V.,"Morphological Analyzer for Tamil", ICON 2002, RCILTS-Tamil, Anna University, India.

[6] Dhanalakshmi V, Anandkumar M,Rekha R U, Arunkumar C, Soman K P, Rajendran S, "Morphological Analyzer for Agglutinative Languages Using Machine Learning Approaches", Proceedings of International Conference on Advances in Recent Technologies in Communication and Computing, 2009 IEEE, pp 433-435.

[7] K. Narayana Murthy, "A Network and Process Model for Kannada morphological Analysis/ Generation", Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, INDIA.

[8] T.N. Vikram and Shalini R. Urs, "Development of Prototype Morphological Analyzer for the South Indian Language of Kannada".

[9] Bharati, Akshar, Vineet Chaitanya and Rajeev Sangal. (1995). Natural Language Processing: A Paninian Perspective, Prentice-Hall of India, New Delhi.

[10] http://ccat.sas.upenn.edu/plc/kannada/

Shambhavi B R received BE degree from Visvesvaraya Technological University in 2003 and MS from BITS, Pilani in 2007. She is currently pursuing Ph.D under VTU in the field of Natural Language Processing. She is presently working as Lecturer in the Department of CSE, RVCE, Bangalore. She is a life member of Indian Society for Technical Education (ISTE)

Dr. Ramakanth Kumar P completed his Ph.D. from Mangalore University in the area of Pattern Recognition. He has experience of around 16 years in Academics and Industry. His areas of interest are Image Processing, Pattern Recognition and Natural Language Processing. He has to his credits 03 National Journals, 15 International Journals, 20 Conferences. He is a member of the Computer Society of India (CSI) and a life member of Indian Society for Technical Education (ISTE). He has completed number of research and consultancy projects for DRDO.

Srividya K received BE degree from Visvesvaraya Technological University in 2010. She studied Computer Science at RV College of Engineering, Bangalore. She is currently working as a Software Engineer at Yahoo! India Software Development Center.

Jyothi B J completed her B.E in Computer Science from R.V College of Engineering under Visvesvaraya Technological University in 2010. She has done her internship in HP-STSD in Bangalore and is currently working as a Software Engineer in Oracle India(Server Technology).

Spoorti Kundargi received B.E degree from Vishvesvaraya Technological Institue in 2010. She graduated with distinction from Computer Science Department of R V College of Enginering, Bangalore. She successfully completed her project "TDM to MySQL tool" during her Internship at National Instruments.

Varsha Shastri G recieved B.E.degree at R.V College of Engineering under Visvesvaraya Technological University in 2010. She interned at National Instruments India R&D at Bangalore and is currently working there as a Software Engineer.