Technical Report
IIITB-KNJ-2011-01-01

Issues in Information Retrieval for the Kannada Language

Kanaja
(An Online Kannada Knowledge base)

Sumant Kulkarni, S. Rajagopalan, Srinath Srinivasa
{sumant, raj, sri}@iiitb.ac.in

International Institute of Information Technology, Bangalore
# Table of Contents

1. **Introduction**.......................................................................................................................... 3  
   1.1 Scope of the document................................................................................................................... 3  
   1.2 Acronyms.................................................................................................................................... 3  

2. **Encoding in Kannada**............................................................................................................. 4  
   2.1 Old proprietary encoding................................................................................................................ 4  
   2.1.1 Shree Lipi encoding....................................................................................................................... 4  
   2.1.2 Nudi encoding.............................................................................................................................. 4  
   2.1.3 Baraha encoding.......................................................................................................................... 5  
   2.2 ISCII encoding............................................................................................................................... 5  
   2.3 Unicode encoding.......................................................................................................................... 5  
   2.4 Open issues..................................................................................................................................... 6  

3. **Text Segmentation and Ambiguous Spellings**..................................................................... 7  
   3.1 Text Segmentation.......................................................................................................................... 7  
   3.1.1 Word segmentation....................................................................................................................... 7  
   3.1.2 Sentence segmentation................................................................................................................. 7  
   3.1.3 Topic Segmentation....................................................................................................................... 7  
   3.1.3.1 Previous work in topic level segmentation in other languages............................................. 7  
   3.1.3.2 Topic level segmentation in Kannada...................................................................................... 8  
   3.1.4 Open Issues in Kannada Text Segmentation............................................................................. 8  
   3.2 Ambiguous Spellings of the Same Word.................................................................................... 9  
   3.2.1. The flexibility of writing the same spelling in different ways................................................. 9  
   3.2.1.1 Usage of *arkavattu*............................................................................................................... 9  
   3.2.1.2 Usage of *anuswara*............................................................................................................. 10  
   3.2.2 The flexibility of writing same word in different related spellings........................................ 10  
   3.2.3 Influence of other languages.................................................................................................... 10  
   3.2.4 Open Issues............................................................................................................................... 11  

4. **Stop words in Kannada**.......................................................................................................... 12  
   4.1 Common Stop-words...................................................................................................................... 12  
   4.2 Kannada Stop-words.................................................................................................................... 12  
   4.3 Stop-word distribution.................................................................................................................. 12  
   4.4 Open Issues................................................................................................................................. 13  

5. **Stemming**............................................................................................................................... 14  
   5.1 Lovins Algorithm.......................................................................................................................... 14  
   5.1.1 Example of a successful case of Lovins’ stemming................................................................. 15  
   5.1.2 Example of a failure case of Lovins’ stemming................................................................. 15  
   5.1.3 Short comings of Lovins’ algorithm......................................................................................... 15  
   5.2 Porter’s stemming algorithm....................................................................................................... 15  
   5.2.1 Why Porter’s algorithm is important..................................................................................... 16  
   5.3 The Paice and Husk stemming algorithm................................................................................... 16  
   5.3.1 Checking applicability of the rule............................................................................................ 16  

---

---
5.3.2 Example of rules and their meanings ................................................................. 16
5.3.3 Algorithm steps ................................................................................................. 17
5.4 Other main stemming algorithms ................................................................. 17
5.4.1 Dowson’s algorithm ...................................................................................... 17
5.4.2 Statistical algorithms .................................................................................... 17
5.5 Stemming in Kannada Language ................................................................. 17
5.5.1 Kannada Noun forms ................................................................................... 18
5.5.2 Kannada Verb forms ..................................................................................... 18
5.6 Open Issues ........................................................................................................ 18

6. Summary .................................................................................................................. 20

7. References .............................................................................................................. 21
1. Introduction

Today, millions of people use the Information Retrieval to efficiently access and use the web content. Major part of today's Web content is in English [2]. Hence, the majority of the search (Information Retrieval) mechanisms on the Web work well efficiently only with English. However, there has been a trend seen showing the increase of non-English content on the Web in last decade. Also, the local literature in available in many different languages is being digitized. Hence, it has become important to develop Information Retrieval mechanism for all the languages for which the content is available.

Every language has its own grammar, syntax and morphologies [3]. So, the Information Retrieval techniques and algorithms used to search English content can’t be used for every other languages.

Kannada is an Indian language being spoken by around 44 million people predominantly in the south Indian state of Karnataka [3]. There has been considerable increase in the web content in the Kannada language in last one decade. But, there has been very limited study done on the information retrieval for Kannada content. Hence, the Information Retrieval in Kannada posts many fresh and interesting challenges. There have been few search engines supporting Kannada language search. But they are not complete search engines. It has been observed that, the issues like stemming are not addressed efficiently in them.

The important issues related to the search (information retrieval) for Kannada script are discussed in the following sections.

1.1 Scope of the document

This document is the technical report covering the state of the art of Information Retrieval for the south Indian language “Kannada”. This report covers the following topics:
1. Encodings being used to represent Kannada script.
2. Segmenting the Kannada text into logical parts.
3. Ambiguous representation of same words in Kannada and their impact on Information Retrieval.
4. Stop words in Kannada. And generation of stop-word list in Kannada.
5. Stemming algorithms in Kannada and how they are different from popular stemming algorithms of English.

1.2 Acronyms

IR – Information Retrieval.
NLP – Natural Language Processing
UTF – UCS Transformation Format (UCS - Universal Character Set)
UTF-8 – UCS Transformation Format - 8
UTF-16 – UCS Transformation Format - 16
2. Encoding in Kannada

A common character encoding system consists of a codes which pair each character from a given script with a sequence of natural numbers, to facilitate the storage and transmission and viewing of the data over electronic medium [4]. Through character encoding, we can display any language script into human readable format on electronic devices.

Kannada language has its own script. Though Kannada literature is very rich, due to the lack of proper encoding mechanisms till last decade, there has been very less Kannada content on the web.

The content available in Kannada on the Web (or in offline digital formats) is available either in few old proprietary encodings or in the universal encoding format Unicode. The major old proprietary encodings are developed using the applications like Shree Lipi [6], Nudi [9] and Baraha (before Baraha version 5) [5]. These applications use few old proprietary encodings, which are still widely used by Kannada users. Most people had sticked to the old proprietary encoding due to the absence of a good conversion utility that would help them switch once for all. But, now after the availability of Kannada Unicode character encoding, Baraha, Shree Lipi and Nudi applications have come up with the tools to convert the proprietary encodings into Unicode. They allow the users to save the files in Unicode formats directly as well.

The availability of conversion tools would help to make the system unified to use Unicode. Hence, this can increase the number of users using Unicode Kannada to search, blog and converse in Kannada on the Internet.

The following sub-sections discuss the different proprietary encoding formats as well as the universal encoding format Unicode.

2.1 Old proprietary encodings

There are many software systems being used to encode Kannada script into electronically storable format. Each one of them uses one or more encoding standards.

2.1.1 Shree Lipi encoding

Shree Lipi is a proprietary product from Modular InfoTech Pvt. Ltd to create the Kannada text documents. Earlier it supported only proprietary fonts fonts (like ShreeLipi-Ex and ShreeLipi-7) and used proprietary encodings. However, In recent days, it also has started supporting ISCII and Unicode encoding standards [6]. There are tools available to convert the Shree Lipi proprietary encodings into Unicode.

2.1.2 Nudi encoding

Nudi is a computer program and font-encoding standard used for managing and displaying the Kannada script [9]. Nudi is a free-ware. The Karnataka state government owns Nudi. It was
developed and being managed by the Kannada Ganaka Parishat (KGP), a non-profit organization. Majority of the font available with Nudi can be used for dynamic font embedding purposes. Also, they can be used in other situations like database management as the fonts and software are available freely. It has its own a font-encoding based standard which uses ASCII values to store glyphs. However, Nudi editor also provides feature of saving the data in Unicode and ISCII. Nudi engine also allows inputting the data in Unicode.

2.1.3 Baraha encoding

Baraha is a word processing application for creating documents in many Indian languages. At present it supports Kannada and Devanagari scripts [10]. Baraha can be used to send the emails and to create web content in the supported languages (like Kannada, Marathi, Hindi, Sanskrit etc). Before version 5, Baraha used only ANSI format. But, Baraha 5 and above support both ANSI and Unicode formats.

Baraha comes with a few ‘True Type Fonts’ (TTF) which can be used to format the ANSI text. These Baraha (BRH) fonts are not Unicode based and hence, they cannot be applied over the Unicode text [11].

2.2 ISCII encoding

The Indian Standard Code for Information Interchange (ISCII) is a open coding scheme for representing different scripts (writing systems) of India [7] [8]. It encodes the main Indic scripts and a Roman transliteration. It supports the Assamese, Bengali, Devanagari, Gujarati, Gurumukhi, Kannada, Malayalam, Odiya, Tamil and Telugu scripts.

The Brahmi-derived writing systems are mostly rather similar in structure, but have different letter shapes [9]. So ISCII makes use of this relatedness of the different Indic scripts and encodes the letters with the same phonetics value at the same code point regardless of the script. For example, the ISCII code \texttt{0xB3 0xDB} represents \texttt{ki} in all the writing systems (scripts) supported by ISCII and having a phonetic value \texttt{ki}. The writing system can be selected in rich text by markup or in plain text by means of the ATR code.

The Shree Lipi and the Nudi support the ISCII encoding. ISCII has not been widely used outside of certain government institutions.

2.3 Unicode encoding

Unicode is a single universal standard for the consistent encoding, representation and handling of text expressed in most of the world's language scripts (writing system) [12]. Unicode is an open encoding standard in which, every basic character of the language script is given a unique value. It has been developed in cooperation with the Universal Character Set standard. The latest version of Unicode supports more than 109,000 characters of 93 scripts.
Unicode has three major types of encodings, UTF-8, UTF-16, UTF-32, for the character encoding. But, the applications like Nudi support only UTF-8 and UTF-16 for Kannada language.

UTF-8 standard encoding uses sequence of 8 bit blocks to represent the characters. In this encoding, each character can be of size 1 to 4 blocks. It is backward compatible with the ASCII encoding and hence every valid ASCII character is a valid Unicode character in this encoding. UTF-8 avoids the complications of endianness and bit order mapping (BOM) as it is represented in blocks of single bytes.

UTF-16 standard encodes the characters in blocks of two bytes. Every character is represented as one or more blocks called surrogate pairs. UTF-16 makes use of byte order mapping (BOM) to recognize the byte order to aid parsing in machines with different endianness.

The major differences between the UTF-8 and UTF-16 standards are:

1. UTF-8 is compatible with the ASCII. So, an UTF-8 file having only ASCII characters is equivalent to an ASCII file. But, UTF-16 is not compatible with the ASCII encoding.
2. UTF-8 has no dependency on byte orientation. But, UTF-16 has the dependency on the byte orientation.
3. Some UTF-8 characters can be represented in a single byte. But, UTF-16 uses minimum 16 bytes to represent a character. Also, UTF-8 can be represented in multiples of single byte block. But, UTF-16 characters are represented as series of two byte blocks. So, there is no way we can represent a character in odd number of bytes in UTF-16.
4. The UTF-16 can represent many Asian languages characters (like Kannada) in only two bytes. But UTF-8 needs 3 bytes to represent each characters in the same languages.

Kannada Script has the Unicode support. The Kannada Unicode characters are between the values 0x0C80 and 0x0CFF (represented in UTF-16). The Kannada content can be generated in Unicode using the applications like Baraha, Nudi and Shree Lipi. Also, Nudi and Baraha provide conversion tools to convert the proprietary encoding to Unicode.

### 2.4 Open issues

1. As per best of our knowledge, few of the proprietary encoding standards are not publicly available. This makes it complicated to retrieve and manage such proprietary encoded information.

2. The Unicode needs to be made the single standard of creating and storing the Kannada data on Web. This helps to overcome many proprietary issues related to encoding. But, how it has to done remains open. The UTF-8 takes 3 bytes to represent each character in Kannada script while UTF-16 needs only 2 bytes. In the meanwhile, UTF-16 needs 2 bytes to store many common ASCII characters like space, while UTF-8 needs only one byte. So, a study has to be done to understand the efficiencies of both encodings to select one.

---

1. [http://unicode.org/faq/utf_bom.html](http://unicode.org/faq/utf_bom.html)
3. Text Segmentation and Ambiguous Spellings

3.1 Text Segmentation

Text Segmentation process divides the written text into different physical and/or logical units [13]. The division produces words, sentences by tokenization and topics by the topic segments.

3.1.1 Word segmentation

This is the physical process of dividing the given text into individual words. This is done on the bases that the words are separated by some known delimiters. This helps the string search depending on individual words. Sometimes delimiter based word segmentation becomes complicated in languages like Chinese, Japanese where the words are not delimited, and in languages like German and Kannada where words are added together to form a long string. Even though we can't separate/split every word using the delimiters in Kannada, we can segment the words which are pronounced separately while reading. Kannada script has a few distinct ways to recognize the end of words in these cases. Here, the delimiter would be any one of the \textit{space, comma, full stop, exclamation mark, bracket or new line} characters.

3.1.2 Sentence segmentation

Sentence segmentation is the physical segmentation method of dividing the given text into sentences. This is sometimes helpful in semantic parsing, document summarization etc [38]. The sentence segmentation becomes tougher when the same characters are used both as sentence terminators as well as to signify something special which is not the end of sentence. For example, in case of English and Kannada, dot (.) is used as sentence terminator as well as to represent end of an abbreviation.

3.1.3 Topic Segmentation

In full-text environment, topic can be defined as the logically and semantically related text segment which can be used for information retrieval and summarization [14]. Hence, topic segmentation is the process of dividing the data into chunks, depending upon the subject of discussion [39]. Many times, a passage can represent a topic, though it is not always true.

In English, there have been many publications discussing the segmentation a full text environment into topics.

3.1.3.1 Previous work in topic level segmentation in other languages

In 1994, Hearst [40] proposed an idea of using only the similarity of word distributions in a given text to segment the text in the paper. This is a generic approach and can be applied to any kind of text. She claimed that higher frequency of certain words in a certain area of the document is an indication of the topic in that area [40]. Also she claimed that when the frequencies of the words differ then they indicate different topics. She proposed an algorithm to find out the topic in a document and called it as TextTile.
Ponte and Croft [14] argued that it may not be true that paragraphs are the building blocks of a topic; and there can be text without paragraphs discussing more than one topic. They also argued that the topic may not always contain huge number of sentences. Some time, a topic can also be made up of one or just a couple sentences. They use Local Context Analysis (LCA), Pair wise Sentence Similarity, segment ranking and dynamic programming to find word and phrases related to sentence and from them the topics.

Sultan and Singhal [15] have also discussed about decomposing the text into segments. This segment is semantically equivalent to the topic. Here, the segmentation starts at paragraph level and different paragraphs are added to make a topic. This process makes use of text relation map to decide whether two paragraphs are on the same topic or not.

Utiyama M. and Isahara H. [17] proposed a statistical model to Domain Independent Text Segmentation. The proposed model finds the maximum probability segmentation index.

Dias, et al. [18] proposed to use the advantage of word co-occurrence for the language independent topic segmentation. It avoids to existing linguistic resources and does not rely on lexical repetition.

Choi [41] came up with the domain independent linear text segmentation method. He claimed that the segmentation method was twice as accurate and seven times as fast as contemporary methods analyzed in Reynar’s state-of-the-art. His algorithm took sentences as inputs [41] [42]. In his algorithm, makes use of the similarity matrix, ranking, clustering and speed omission to identify a topic. He named this algorithm as C99. He experimented Textiling, DotPot and Segmenter, and found that C99 (his algorithm) to be more accurate [41].

According to Dias, et al, their experiments have shown that Choi’s [41] C99 algorithm and Hearst’s [15] TextTiling algorithms shows improved results both with and without the identification of multiword units [18].

3.1.3.2 Topic level segmentation in Kannada

Kannada script has a specific way to represent passage (paragraph). However, as mentioned above a paragraph always may not be made up of only one topic; rather there can be multiple topics embedded. Every couple of lines can be for a different topic. Also, sometimes on the other hand, only one paragraph may not make a complete topic and many paragraphs can come together to form it. Hence, topic segmentation is a very important issue in Kannada IR systems. But as per our best knowledge, there hasn’t been any published work on the segmentation in Kannada.

3.1.4 Open Issues in Kannada Text Segmentation

1. As per best of our knowledge, no segmentation work has been done in Kannada language. Hence, there has to be a detailed study on the applicability of different types of
segmentation on Kannada script. Especially, it has to be studied that how topic level segmentation can be achieved for Kannada.

2. The NLP methods for Kannada language differ from the NLP methods for the English language. Hence if we directly can not use NLP methods of English (or other languages) for Kannada script. So, the evaluation of segmentation techniques designed for English language have to be evaluated for Indic languages.

3. Kannada has a complex script. The words take different forms depending on the tense, case and whether the used as singular or plural [3]. This makes it tough to keep track of the frequency of a word which is present in different morphological forms. Hence, the algorithms which depend on word frequency to identify a topic may not work for Kannada. For example, a simple noun like kaaDu can take the morphological forms like kaaDinalli, kaadugaLu, kaaDina etc. lexically it is very tough to determine that the root of these morphological forms is kaaDu.

3.2. Ambiguous Spellings of the Same Word

We have observed that Kannada has variations in its spellings while representing the same word. This variation can be majorly due to the following reasons:

3.2.1. The flexibility of writing the same spelling in different ways

In Kannada, there are ways to represent the same pronunciations in two different ways (two different spellings, which are pronounced in the same way). We face these ambiguities majorly while using arkavattu and anusvara. Even though, both of representations are correct, they have different encodings. These differences make the mapping between two representations complex.

3.2.1.1 Usage of arkavattu

This is a very common example. Kannada has two different ways of writing the consonant clusters with ‘ra’ (Unicode value 0C80) as the first consonant [19]. They are given below. There are no publications which say that Unicode support this kind of variation. There have been no evidences found to say that Unicode supports the first way of writing. Only the ‘Nudi’ encoding has been found supporting both these spellings.

The ‘nudi’ encoding of the above spellings is given below:

<table>
<thead>
<tr>
<th>ಕಾದು</th>
<th>ಕಾದುಳಿ</th>
<th>ಕಾದುಗಳೂ</th>
<th>ಕಾದುಣೇ</th>
<th>ಕಾದುಳಿ</th>
</tr>
</thead>
<tbody>
<tr>
<td>\ä\d\0\4</td>
<td>\ä\6\d\0\d\4\fe</td>
<td>\ä\4\d\0\4</td>
<td>\8\1\ä\6\d\0\d\4\fe</td>
<td>\ä\4\d\0\4</td>
</tr>
</tbody>
</table>
3.2.1.2 Usage of anuswara

Similarly in Kannada, a word using Anuswara can also be written in a way not using Anuswara. The example is given below. All the encoding types in Kannada support these variations in spelling. Also, as the spellings differ, the Unicode value changes for the same word represented in these two different ways.

Even though the latter case is very rare, we can not completely deny the second case. The Unicode representations of these two terms are given below:

มวลರಾಣು → u3246?㉗?㉀?㉷?㈶
มวลರಾಣು → u3246?㉗?㈂?㈶

3.2.2 The flexibility of writing same word in different related spellings

In Kannada, there are cases where a word has two pronunciations which slightly differ and has the same meaning. So, when one spelling is used in IR, it is expected that other spelling is also searched for. One of the simplest examples of this case is the word meaning Story. We can pronounce it as Kathe as well as Kate. Their Kannada scripts are as below:

ಕಥೆ ಕತೆ

Here, as the letters used in these spellings are different, the Unicode (and others too) differ for these two types of representations as below.

ಕಥೆ → u3221?㈷?㉰
ಕತೆ → u3221?㈶?㉰

3.2.3 Influence of other languages

In recent time, Kannada has been influenced by much other language vocabulary (at least blogs). Many bloggers and authors have started writing in a way they speak to give a natural feeling, and use many English terms while writing in Kannada. There English words are transliterated into Kannada and used. Hence, it is possible that the transliteration can be done in different ways.

For example, the word QUIZ can be transliterated as two forms given below:

ಕವಝ ಕವಜ
Here, the letters used in these spellings are different. Hence the Unicode representations differ for these two types of representations and are given below:

\[
\begin{align*}
\text{ಕವಝ} & \rightarrow \text{u3221}\text{u3277}\text{u3253}\text{u3263}\text{u3229}\text{u3277} \\
\text{ಕವಜ} & \rightarrow \text{u3221}\text{u3277}\text{u3253}\text{u3263}\text{u3228}\text{u3277}
\end{align*}
\]

3.2.4 Open Issues

As discussed above, there are many issues specific to Kannada script and language.

1. The section 3.2.1 talks about very common ambiguity in Kannada script. More commonly, the ambiguity in section 3.2.1.1 is seen very often in literature. So, the issue remains as “How to address the ambiguous spellings in section 4.1 are to be addressed in IR?”. There has to be some exploratory work to be done to understand the applicability of the phonetic hashing to address this issue.

2. The issue mentioned in 3.2.2 is having many examples though they are specific. So, the issue of handling such cases in IR is to be addressed. The possibility of using phonetic hashing has to be evaluated in this case.

3. An exploratory survey of usage of transliterated words in Kannada script has to be done. Then the severity of the issue mentioned in section 4.4 has to be analyzed. Then if severe, ways to solve the problem has to be found. Intuitively it looks like phonetic hashing solves this problem. But, more exploration has to be done on this.

4. There are no phonetic hashing algorithms for Kannada. So, a study has to be done on coming up with a phonetic hashing algorithm. The possibilities of the algorithm and the results have to be evaluated.
4. **Stop words in Kannada**

The Stop words are very commonly used words in a language which are to be neglected by the information retrieval system (search engines) while indexing [20]. From an IR perspective, these stop word don’t add any value to the information. Instead they can complicate the IR process. Hence, in order to increase the performance of the search, to make the search efficient as well as to save disk space, it is preferable to neglect these stop-words while processing the text [21]. A stop-word-list is a list which contains all these stop-words of a language. Hence, a stop word list makes an IR system more efficient.

Due to the importance, research is going on to generate the stop-word-list in various languages. The languages like English, Hindi already do have stop word lists [24] [25].

4.1 **Common Stop-words**

From the definition of the stop word and from the stop word lists of Hindi and English, we claim that the in any language, the following contribute to the stop word list:

1. All articles (e.g. a, an, the etc)
2. All pronouns (e.g. I, you, he etc)
3. All auxiliary verbs (e.g. be, is, was etc)
4. All prepositions (e.g. on, in, to etc)
5. All conjunctions (e.g. and, but, because etc)
6. Some adjectives (e.g. quantifiers like many, few, multiple etc)
7. Some Adverbs (e.g. fast, suddenly etc)
8. The numbers expressed as quantifiers (e.g. one, two, couple)

However, any word which may not be belonging to the above mentioned categories also can be a stop word. For example, see is a verb in English but it is a stop word [21].

4.2 **Kannada Stop-words**

Following are the examples for the stop words in Kannada language.

ಅಲಲ, ಅವನು, ಅವರ, ಅವಳ, ಅದರ, ಅದು, ಅಂದು, ಅವನ, ಅವಳ, ಅಥವಾ, ಅವಲಂಬಸದೆ, ಅಂದರೆ, ಅಂತಮ, ಇಂದ, ಬಗೆಗ,ದೆೊಡಡ, ಮುಖಯ,ತನನ,ಆದರೆ,ಮೇಲೆ,ನಂತರ,ಇತರ,ಹಾಗೆ etc.

4.3 **Stop-word distribution**

Zipf showed that in a huge document corpus (for most languages), if we rank the words by their frequency, the word which has repeated the most being ranked 1 and next is 2 so on, then the frequency (f) of a word is inversely proportional to its rank (r). In other words, the product of a word’s frequency f(w) and its rank r(w) is a constant [23].

This same Zipf’s law has been used to identify that the most repeating term in a corpus are the **stop words**. In the near future, we propose to verify this theory on the Kannada Wikipedia corpus.
4.4 Open Issues

1. In many Indic languages like Kannada, the words get combined to form compound words. Hence, it is not always possible to separate a stop word from another word (whether the latter is stop word or not does not matter) when they have come together to form a compound word. Hence, the issue of separating the stop word from the compound word exists in Kannada language. Examples of compound word:
   a. *HAgAdare* (ಹಾಗಾದರೆ) is made up of *HAge* (ಹಾಗೆ) and *Adare* (ಆದರೆ), which are both stop words.
   b. *bharataveMdare* (ಭಾರತವೆಂದರೆ) is made up of *bharata* (ಭಾರತ) and *andare* (ಅಂದರೆ), in which first one is not a stop word while latter is a stop word.

2. Unlike English where usually the case makers do not change the spelling of the noun, in Kannada, the case makers often change the spellings of the noun or pronouns to which they get added [3].
5. Stemming

In many languages, the morphological variations of a word have the same semantic interpretations as of their root [26]. Hence, in case of IR, they can be considered equivalent to the original word (root) from which they are derived. This original word is called the stem of all the morphological forms. Hence, we can define the process of rediscovering the stem from the morphological variations as *stemming*.

Consider the morphological forms - *refined, refinement, refining* and *refines*. Here we have defined many morphological forms of a word. By looking at the morphological forms, any human being who knows English language (read and speak) can easily understand that these all are originated from the verb ‘*refine*’. So ‘*refine*’ is the stem of all of the above morphological forms. And our brain did the stemming on these morphological forms to discover that the stem ‘*refine*’. It looks very easy for us to do. But, for machines, it is a very big challenge. Hence, stemming is one of the challenging problems of NLP.

There has been much work done in English stemming algorithms [27]. There are many methods proposed and are being used for stemming. But, none of these methods are fool proof. The stemming methods can be of two types and they are explained below.

**Dictionary based stemming:** These stemming methods directly look into the dictionary of words for the stem after applying very few rules to the original word. Chopping last few letter of the word can be one of the very few rules associated with these kinds of stemmers.

**Algorithm based stemming:** Algorithmic stemmer first apply a set of rules to crop and/or modify the original word to derive its stem. These number of rules can vary from few to hundreds depending on the algorithm being used.

The algorithmic stemmers are considered more efficient than dictionary based stemmers [27]. It has also been observed that algorithm based stemmers can become more efficient by adding dictionary based approach in them. In algorithm based stemmers, the dictionary will be very small and usually contains only the exception cases. The following sections talk about few popular stemming algorithms (algorithm based stemmers).

5.1 Lovins Algorithm

Lovins’ algorithm was the first ever published stemming algorithm [28]. It was published by Lovins JB in 1968. He targeted both computer linguistics and IR area through this algorithm. Lovins worked in technical environment and hence we see his algorithm to be majorly influenced by the technical vocabulary.

The algorithm has 294 endings, 29 conditions and 35 transformation rules. Hence it is bigger than the Porter’s algorithm. But, this algorithm is faster than Porter’s algorithm. When a word is to be stemmed, the following steps are followed.
1. The word's longest ending is matched with available 294 endings.
2. Then condition related is verified (from 29 conditions). If the condition returns true, then
   the appropriate transformation (step 3) rule is applied.
3. This transformation rule (one out of 35) either removes or replaces the selected ending.

The transformation rules handle features like doubled consonants (sitting -> sitt -> sit [29]),
irregular plurals (matrix and matrices), and English morphological oddities caused by the behavior
of Latin verbs of the second conjugation (assume/assumption, commit/commission etc) [28].

5.1.1 Example of a successful case of Lovins’ stemming

Consider the word sleeplessness. This word has a ending lessness which is associated with the
condition A. Condition A is ‘no restriction on stem length’. Hence lessness is removed to retain
sleep. Sleep is the correct stem.

5.1.2 Example of a failure case of Lovins’ stemming

The word nationally has the ending ationally. This ending is associated condition B, which is
‘minimum stem length = 3’. But removing ationally would leave a stem of length 1. So, this
condition hence this ending is rejected. Now we look for other ending. We see that this word also
has ending ionally. This ending is associated with the condition A. Condition A is ‘no restriction
on stem length’; so ionally is removed, leaving nat. But nat is the incorrect stem. Instead, we were
expecting nation as the stem.

5.1.3 Short comings of Lovins’ algorithm:

1. The algorithm even though was targeted for both IR and Computer Linguistics; it didn’t
even excel in both or either of them [30].
2. Even after targeting so many endings, this algorithm had left out many endings which were
   very common [28].
3. It is a single pass algorithm. Hence, it removes a maximum of one suffix from a word.
   However, if there are multiple suffixes attached to the same word, this algorithm can’t
   handle them.
4. As explained in 3.1.2, the stemming sometimes returns wrong results. Hence the algorithm
   is not completely reliable.

5.2 Porter’s stemming algorithm

Porter’s stemming algorithms was first presented in 1981 [31], and revised in 2001 and 2002 [32].
This is a context sensitive algorithm and removes only suffixes from a word. The implementation
of the algorithm is available in many languages and also it is most widely used stemming
algorithm [31]. The stemmer makes use of 5 to 6 linear steps to produce the final stem from its
morphological form. Within each phase there are various conventions to select rules; for example,
selecting the rule from group of rules that applies to the longest suffix [1].
An english word can be represented as \([C](VC)m[V]\), where \(C\) represents a consonant, \(V\)
represents a vowel and square brackets represent that it is optional [31]. \((VC)m\) represents that
there can be \( m \) repetitions of \( VC \). The value \( m \) is called measure of the word. And it should be greater than or equal to zero.

The Porter’s algorithm was having two distinct differences from Lovins algorithm [33]:

1. The algorithm is fairly simple and has 60 suffixes, two recording rules and ultimately only one context sensitive rule. This context sensitive rule will determine whether the suffix should be removed or not. As the number of suffixes and rules reduce, the complexity associated with them also reduces.

2. Lovins’ algorithm many times does over-stemming. Unlike Lovins’ algorithm, the Porters’ algorithm uses single unified approach to handle the context. It doesn’t depend up on the rule based system; rather it depends minimal length based on the number of consonant-vowel-consonant strings (the measure) remaining after removal of a suffix.

The Porters’ original algorithm has been implemented to cover Romance (French, Italian, Portuguese and Spanish), Germanic (Dutch and German) and Scandinavian languages (Danish, Norwegian and Swedish), as well as Finnish and Russian (Porter, 2006) [33].

5.2.1 Why Porter’s algorithm is important [33]:

1. It provides a simple approach of combinations. It looks to work in many languages especially European languages.

2. It made stemming as the main stream research area rather than having it as a part of IR.

5.3 The Paice and Husk stemming algorithm

Paice and Husk proposed this stemming algorithm in Lancaster University in late 1980s [34]. This is an iterative algorithm and has only one table with 125 rules [29] [35]. Each rule will specify either deletion or replacement of an ending. Hence, this avoids the recoding stage which is needed in Lovins’ algorithm. These rules grouped into sections. Each rule section can be accessed just by looking at the last letter of the suffix [35]. The ordering of the rules within the section also matters. Also, few rules apply only if the if the suffix is intact (i.e. no rule is applied earlier on the word).

5.3.1 Checking applicability of the rule

When a word is to be processed, the stemmer takes the last letter of the word suffix and the corresponding rule is found [34]. This rule is executed only when it paces the applicability test (e.g. testing whether it is intact). In applicability test, some special condition may be tested which are specific to the rule. If the word passed the applicability test, then only the rule can be applied.

5.3.2 Example of rules and their meanings

Consider the rule \( e1 > \{-e-\} \). It says that if the last letter is \( e \), then remove the \( e \) (only one letter). Hence it turns out to be an \( e \) removal rule [35].
5.3.3 Algorithm steps

1. Select relevant section.
2. Check applicability of rule as explained above.
3. Confirm that applicable rule leaves correct stem. If so, goto step 3, else goto step 4 [35].
4. Apply the rule.
5. Look for another rule.

This algorithm continues iteratively till there is a rule available for the last letter of the suffix [29]. Otherwise, it ends. Also, if the word starts with a vowel and has only tow letters or if the word starts with a consonant and has only 3 letters left, then also this algorithm ends.

As compared to Porter’s algorithm, it is found that the Paice/Husk algorithm has more over stemming tendency.

5.4 Other main stemming algorithms

5.4.1 Dowson’s algorithm

This is considered as the improvement of the Lovins’ algorithm [29]. It also has a very long table of suffixes (around 1200). It stores the suffixes in the table in the reverses order. It replaces the suffix which matches with the stored index using the set of rules defined. It was found that the algorithm was complex and may be because of this, the algorithm didn’t become popular.

5.4.2 Statistical algorithms

There were many statistical algorithms proposed and being used in stemming. The N-gram algorithm analyses the distribution of N-grams in a document [29]. The Inverse Statistical Frequency (IDE) is used to find the stem of variants of a stem. This algorithm is tested successfully for 8 European languages. Hidden Morkov Model is also used to stemming [29].

5.5 Stemming in Kannada Language

Indic languages like Kannada are grammatically very different from English. Hence, the English stemming algorithms can’t be directly applied on Kannada script.

There hasn’t been much work done in stemming in Kannada language. Text categorization, summarization and retrieval have not been achieved in most of the Asian languages due to the lack of the essential stemming algorithms which are language specific [3].

The only available Paper in Kannada script stemming is by Urs and Vikram [3]. This paper speaks about the formation of different generic morphological forms of Kannada nouns and verbs. Also it speaks about the need of the stemming algorithms for Kannada script.

In a Kannada sentence, derivations of words are usually formed by either by adding two or more different distinct words or by adding affixes to distinct words. Also, it is observed that while these derivations are formed, the spellings of the words/affixes involved change. Hence, it is a
challenge to separate affixes and words added to a stem. The process of adding two or more words (including affixes) to create a single conjunction of word is called Sandhi. Let us see some examples, how the Sandhis create morphological forms for verbs and nouns.

5.5.1 Kannada Noun forms

Urs and Vikram [3], have some generic way of defining nouns. A morphological form of a noun is influenced by its gender, case and whether it is used in singular context or a plural one [3]. Consider the example:

\[ \text{KAD}i\text{Nalli} (ಕಾಡನಲಲ) - meaning ‘in the forest’) is the morphological forms of the noun KADu (ಕಾಡು - meaning ‘the forest’). This morphological form is created by a noun form KAdu (ಕಾಡು), and the locative case maker (Saptami vibhaki) Alli (ಅಲಲ). As, this noun is in the singular context, no need to add affix to make it a plural. We see here that the letter Na (ನ) has come in between these word and case maker to join them.

Hence, Kannada noun form = Nouns Stem + Plural Marker + Case Marker [3].

5.5.2 Kannada Verb forms

Similarly, the verbal forms of Kannada language are influenced by its PNG (i.e. Person-Number-Gender) and tense. PNG tells which person the verb is used in (first, second or third), whether the verb is for singular noun or for plural and the gender of the noun. Consider the example below:

\[ \text{ODidavu} (ರೇಡದವು - meaning ‘(they) ran’) is the morphological form of the verb ODu (ರೇಡ -meaning ‘to run’). This morphological form is created by adding the proper verb ODu (ರೇಡ) and the past-plural-third person maker -davu (ದವು). We can see that there has been the removal of the vowel u(ು) and addition of vowel i (ಇ). These kinds of removal and additions of letters change the spellings of the word.

Hence, Kannada verb form = Verb Stem + Tense Marker + PNG Marker [3].

5.6 Open Issues

1. Many morphological forms of a word are formed by Sandhis. Stemming the morphological forms of verbs and nouns is great challenges as there are many instances of Sandhis which modify the spellings.

2. In Kannada Sandhis, two words (none of them is an affix) can come together and form a conjunction of words. Here the stemming algorithm split this conjunction and generate two meaningful stems. Unlike suffix removal, we can not afford to loose a word here.

3. There is no notion of Capital Phrase in Kannada. Hence, it is tough to recognize the proper nouns in Kannada. So, we have to come up with the algorithms to recognize the proper nouns and stop stemming them.
4. Even though Urs and Vikram [3] speak about stemming verbs and nouns, there hasn’t been any algorithm to tag a word as a verb or noun. So, recognizing the type of the words (parts of speech tagging) in a Kannada sentence is still an open issue.

5. English has only 52 letters; 26 in lower case and 26 in upper case. Anything we write in English can have only these 56 letters (or symbols). But, in Kannada we can have around 80,000 symbols of different types [3], generated due to its diacritics, compound characters. Hence, any stemming algorithm to be developed has to take care that it recognizes each character and its roots. A broad feasibility study has to be done about the stemming algorithms for Kannada language. Only after getting some positive results from this study, we can work towards developing stemming algorithms for Kannada.
6. Summary

As per the study done, the IR system in Kannada has many fresh and interesting challenges. The old proprietary types of encoding specification for Kannada script are not publically available. Hence, developing a search (IR) mechanism which works for all the encoding formats of Kannada script is still an open challenge. Also, there has not been much work done on the text segmentation methods in Kannada. Hence, the segmentation methodologies have to be invented or adapted from other languages as per the requirements. The IR system faces interesting challenges from ambiguous spellings in Kannada script. There ways to address these problems have to be found out yet. Though there has been some work done on stemming algorithms on Kannada script, there is lot of work yet to be done. The stemming methods to address the problems like Sandhis have to be looked into.

In total, the Kannada IR system is still very immature and there is a big scope for lot of improvement and new methodologies to be developed.
7. References

5. Kannada/ToDo. Available at: http://wiki.sampada.net/Kannada/ToDo (last accessed on Jan 11, 2011)
17. Utiyama , M. and Isahara, H. A Statistical Model for Domain-Independent Text Segmentation
20. Stop word. Available at: http://searchsoa.techtarget.com/definition/stop-word (last accessed on Jan 11, 2011)


25. Hindi Stop words. Available at: http://mail.sarai.net/pipermail/prc/Week-of-Mon-20080204/001656.html (last accessed on Jan 11, 2011)

26. What is Stemming. Available at: http://www.comp.lancs.ac.uk/computing/research/stemming/general/ (last accessed on Jan 11, 2011)


30. The Lovins Stemmer. Available at: http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm (last accessed on Jan 11, 2011)

31. The Lancaster Stemming Algorithm. Available at: http://www.comp.lancs.ac.uk/computing/research/stemming/Links/porter.htm

32. The English (Porter2) stemming algorithm. Available at: http://snowball.tartarus.org/algorithms/english/stemmer.html (last accessed on Jan 11, 2011)


34. The Paice/Husk Stemmer. Available at: http://www.comp.lancs.ac.uk/computing/research/stemming/general/paice.htm (last accessed on Jan 11, 2011)


37. About Baraha. Available at: http://www.baraha.com/about.htm (last accessed on Jan 11, 2011)

38. Sentence Segmentation. Lothar Lemnitzer. Available at: http://www.cl.uni-heidelberg.de/courses/archiv/ss06/texttech/slidesSentSeg.pdf (last accessed on Jan 11, 2011)

