Chapter 6

Effect of stemming in Sanskrit IR

6.1 Introduction

Stemming is an important pre-processing step in text-processing tasks such as text mining, text summarization and IR. The IR issues are primarily investigated from three viewpoints. First, what are the challenges in building a Sanskrit text collection for IR? What are the morphological barriers in Sanskrit text processing? How to get around them? Does stemming improve the overall effectiveness of Sanskrit? If yes, to what extent? Sanskrit is found to be morphologically rich. Hence, what could be the most suitable stemmer for Sanskrit IR? We propose two stemmers: a light stemmer and an aggressive stemmer. We compare their effectiveness with another language-independent stemmer. For Sanskrit IR, should we use a light stemmer, an aggressive stemmer, or a language-independent one during indexing? In this study, we build a Sanskrit text collection and explore different Sanskrit indexing, stemming and searching strategies. We also evaluate the effectiveness of each activity not only from an IR perspective but also in a general framework of text analysis.

In particular, the effectiveness of the stemmers is evaluated in two ways: a *direct* and an *indirect* IR-based evaluation. In direct evaluation, we find that the stemmers effectively find the roots of the words or not. In indirect evaluation, we apply different retrieval models such as BM25, TF-IDF, Divergence from Randomness (DFR) and language models. The proposed stemmers are compared with three different stemming approaches: GRAS, language-independent indexing (trunc-n) and no-stemming baseline. Among different stemming methods, aggressive stemming performs the best. Among the

retrieval models, the language model outperforms others we experimented with.

6.2 Sanskrit Morphology

Sanskrit is the oldest language of civilization, belonging to the Indo-European language family [117]. Some widely spoken languages in this family are English, Hindi, Bengali and Marathi. Sanskrit is primarily written in the Devanagari script and a few other variants, e.g., the Brahmic script, and uses 47 primary characters. The character set comprises 14 vowels, 33 consonants, and frequently used nasal symbol anusvar ('ं') and visarga ('ः'). There is no distinction between uppercase and lowercase letters.

Sanskrit is a highly inflectional language having rich nominal declension and uses extensive compound nouns [59]. According to Sanskrit grammar rules, almost all nouns and adjectives follow identical inflections. Adjectives follow the inflections based on gender, case, and the number of nouns. Sanskrit nouns have three possible genders, i.e., masculine, feminine, and neuter. Feminine nouns are generally derived from masculine nouns by two common suffixes, i.e.,ā –(आ) and – ī– (ई). (e.g., छात्र(male student), छात्रा (female student)). The masculine and neuter are differentiated by inflections. Nominal inflections in Sanskrit are carried out by 24 morphological sup suffixes. The suffixes are *'su, au, jas, am, au, śas, ā, bhyām, bhis, e, bhyām, bhyas, asi, bhyām, bhyas, as, os, ām, i, os, sup, su, au, jas'*. These suffixes are divided into eight sets (nominative, accusative, instrumental, dative, ablative, genetive, locative and vocative). Each set comprises three numbers, i.e., singular एकवचनम् (*ekavachanam*), dual द्विबचनम् (*dvivachanam*), and plural बहुवचनम् (*bahuvachanam*). The suffix sets are: *(su, au, jas), (am, au, śas), (ā, bhyām, bhis), (e, bhyām, bhyas), (asi, bhyām, bhyas), (as, os, ām), (i, os, sup), (su, au, jas).* Note that each gender-case combination does not require a distinct suffix. For example, the dual of the nominative and accusative cases are identical. Similarly, the dual of instrumental, dative and ablative cases remain the same as shown in Table 6.1. An inflection of a noun depends upon number, case, and gender, which makes the Sanskrit language one of the most complex morphological structures similar to another Indian language Marathi [37].

Unlike many Indian languages, Sanskrit possesses a rich derivational morphology. The derivational morphology in Sanskrit is achieved in different ways. In the primary derivation, the consonants or vowels are modified in the word endings (e.g., सर्ज् (sarj)→

Table 6.1: Examples of Sanskrit masculine noun (man) declensions

Case	**Singular**	**Dual**	**Plural**
Nominative	नरः (*narah*)	नरौ (*narau*)	नराः (*naraah*)
Accusative	नरम् (*naram*)	नरौ (*narau*)	नरान् (*naran*)
Instrumental	नरेण (*narena*)	नराभ्याम् (*narabhyam*)	नरैः (*naraih*)
Dative	नराय (*naraya*)	नराभ्याम् (*narabhyam*)	नरेभ्यः (*narebhyah*)
Ablative	नरात् (*narat*)	नराभ्याम् (*narabhyam*)	नरेभ्यः (*narebhyah*)
Genitive	नरस्य (*narasya*)	नरयोः (*narayoh*)	नराणाम् (*naranam*)
Locative	नरे (*nare*)	नरयोः (*narayoh*)	नरेषु (*nareshu*)
Vocative	नर (*nara*)	नरौ (*narau*)	नराः (*naraah*)

सर्ग (sarg)) . In the secondary derivation, the prefixation (e.g., स्मरति (smarati)→ विस्मरति (vismarati)) or suffixation (e.g., कृष्ण (krushna)→ कृष्णत्व (krushnatva)) are added to the root word. In general, part of the speech of a word changes when a suffix is added to the root word. In this study, we consider only the suffixes because prefix removal often changes the original meaning of a word (e.g., precaution vs caution). In Sanskrit, derivational suffixes are added before the case endings. We design and implement an aggressive stemmer that strips inflectional and frequently used derivational suffixes. Finally, we remove the frequently occurring terms called stopwords to improve the effectiveness of an IR system.

Compound words with unmarked word boundaries or a closed compound word, e.g., यथा (yatha) + शक्ति (sakti) → यथाशक्ति (*yathasakti*), जगत् (jagat) + ईश (isha) → जगदीश (*jagadish*) is another morphological attribute in Sanskrit that causes major bottleneck during retrieval. The constituent words are present only in either query (or documents) but not in the counterpart, leading to poor effectiveness during retrieval. In Sanskrit, compound words are generated by concatenation of two words or by using *sandhi*[1] rules and is also found in other Indian languages like Bengali (e.g., *mulyabriddhi* (price hike), *upanagar* (town)), Marathi (e.g., *sadasukhi* (eternally happy), *saatrasta* (seven roads)). European languages like German generate compound words using glue characters (e.g., actionsplan, here 's' is a glue character) or a compound word without glue character ('Kon-

[1]The word sandhi means compounding

trollpersonal'). Many compound words are also generated in other European languages, such as French, Finnish and Russian. Braschler and Ripplinger [17] show that compound splitting improves MAP score in German text retrieval. However, compound splitting sometimes generates semantically unrelated words that reduce the effectiveness of a system. For example, 'basketball' split into 'basket' and 'ball', generating two semantically unrelated words from the original meaning.

Sanskrit possesses a strong vocabulary. Different Indian and European languages borrowed many words from Sanskrit. (E.g., '*Neem*' is an English word derived from the Sanskrit word 'Nimbah') [1]. Similarly, Hindi word (उद्यम (udyama) means 'effort') derived from Sanskrit word उद्यम् (udyam) and Marathi word साप (sapa)(means 'snake') derived from Sanskrit word सर्प (sarpa) . Sanskrit also possesses a very rich synonym set, i.e., many distinct words with the same or similar meanings (e.g., 'Hasti' and 'Gaja' are two different variants of the word 'elephant'). These morphological attributes of Sanskrit also affect the retrieval effectiveness of a system.

6.3 Contribution

The contributions made in this chapter are centered around the following research questions (as introduced in Sec. 1.6) and exploring answers to them.

> [**RQ 2**] *What are the challenges in building a Sanskrit text collection for IR? What are the morphological barriers in Sanskrit text processing? How to get around them?*
>
> We discussed the morphological difficulties in Sanskrit text processing and describe below (Section 6.4) the process of Sanskrit text building.
>
> [**RQ 3**] *Does stemming improve the overall effectiveness of the Sanskrit retrieval system? If yes, to what extent?*
>
> We propose two stemming techniques for Sanskrit text processing and evaluate different stemming strategies in the text analysis domains in the following sections.
>
> [**RQ 4**] *For Sanskrit IR, what is the most suitable stemmer? Should we use a light stemmer, an aggressive stemmer, or a language-independent one during indexing?*

Based on our series of experiments, we observe that the aggressive stemmer performs best for Sanskrit for different tasks in both NLP and IR domains.

According to Savoy [37], an effective stemmer is built by stripping the suffixes from nouns and adjectives and ignoring the verbs. Implementing a common stemming approach to verb forms causes more harm than benefits. We propose a ***light*** and an ***aggressive*** stemmer for Sanskrit in the same way as implemented in other Indian languages in the past few years [37]. Our proposed stemmers remove only inflectional and derivational suffixes but not prefixes.

6.3.1 Light Stemmer

The light stemmer proposed removes inflectional suffixes (*e.g.*, *actions* $\rightarrow$ *action*) from nouns and adjectives by applying 51 language-specific rules. The example of a few inflectional suffixes handled by light stemmer is shown in Table 6.2 with ITRANS[2] notations. These suffixes are identified by observing the word inflections in adjectives, nouns and three genders, i.e., masculine, feminine and neuter. The suffixes are selected by two experts (doctoral students of Sanskrit with Master's degrees in Sanskrit). Generally, a root word or stem is obtained after removing the suffix from an inflected word. But, sometimes, a few words require deletion and substitution of a few characters to get the root word. E.g., the word *'naran'* generates the word *'nar'* after stemming, but we added the suffix *'a'* to generate the root word *'nara'*. Technically, this may not be called a stem but is more acceptable as a root word from the linguistic point of view. An example of different suffixes handled by light stemmer is shown in Table 6.2. The complete list of suffixes stripped by light stemmer is shown in Table 1. In the light stemmer, we considered suffix lengths up to four. The suffixes are truncated in descending order from the longest possible suffix to the shortest suffix. If there is no suffix match in the system, then the system returns the same word as the root word. For stemming, the minimum word length considered was 5 characters to reduce cases of over-stemming. Often, short words (length < 5 chars) that should not be stemmed like (अहम् (aham), त्वं (twam)) can be stemmed otherwise.

[2]Indian languages TRANSliteration, https://en.wikipedia.org/wiki/ITRANS

Table 6.2: Examples of root word generation by the light stemmer in Sanskrit

Suffix Type	**Devanagari Notation**	**ITRANS Notation**
Nominative	नरः, नरौ, नराः→ नर	nara[h], nara[u] → nara nar[ah]
Accusative	नरम् , नरान्→ नर	nara[m], nara[n] → nara
Instrumental	नरेण, नरैः → नर	nare[na], nar[aih] → nara
Dative	नराय → नर	nara[ya] → nara
Ablative	नरात् → नर	nara[t] → nara
Genitive	नरस्य, नरयोः → नर	nara[sya], nara[yoh] →nara
Locative	नरे, नरेषु → नर	nar[e], nar[eshu] → nara

6.3.2 Aggressive Stemmer

The aggressive stemmer removes different derivational suffixes (*e.g.*, *'educational'* → *'educat'*) by applying an additional set of 20 rules to the light stemmer version. The aggressive stemmer removes inflectional and derivational suffixes observed during word inflections in nouns, adjectives and different genders, i.e., masculine, feminine and neuter. A few examples of derivational suffixes handled by aggressive stemmers are shown in Table 6.3. The complete list of additional derivational suffixes stripped by the aggressive stemmer is shown in Table 2 (Appendix).

Table 6.3: Examples of root word generation by the aggressive stemmer in Sanskrit

Suffix Type	**Devanagari Notation**	**ITRANS Notation**
Accusative	दातारम् → दाता	data[ram] → *data*
Instrumental	नराभ्याम् → नर , शत्रुणा→ शत्रु	nar[abhyam] → nara, satru[na] →satru
Dative	नरेभ्यः → नर	nar[ebhyah] → nara
Genitive	नराणाम् → नर, बारिणाम् → वारि	nar[anam] → nara, vari[nam] → vari bhu[nam] → bhu
Locative	सेनायां, सेनायाम् → सेना	sena[yam] → sena

6.3.3 Algorithm for stemming

The basic text processing steps towards generating the root words (stems) are as follows. We first describe the light-stemming approach followed by the aggressive-stemming approach.

1. Read a set of input tokens from the user.
2. Remove the stopwords from input tokens.
   for **Light Stemmer**
3. Check the length of a word in each token. If the token length is less than five (5), then EXIT.
4. For each suffix in the *inflectional suffix list* (Table 1) (L_{inf}), check whether the suffix is there in the given token or not. If yes, remove the suffix from the word and treat the remaining word of the input as the root word (stem). Otherwise, leave the entire input word as stem only.
   for **Aggressive Stemmer**
5. Check the length of the word in each token. If the token length is less than seven (7), then EXIT.
6. For each suffix in the *derivational suffix list* (Table 2) (L_{der}), check whether the suffix is there in the given token or not. If yes, remove the suffix from the word and treat the remaining word of the input as the root word (stem). Otherwise, leave the entire input word as stem only.

A more detailed description is shown in algorithm 1.

In direct evaluation, we manually check the root word by domain experts. However, we use the Terrier retrieval system in indirect evaluation. Our proposed stemmer is publicly available on Github [3].

[3]https://github.com/cse-iitbhu/Sanskrit-Stemmer

Algorithm 1: Sanskrit suffix stripper

Input: $W \leftarrow\{w_1, w_2, ..., w_n\}$; //list of tokens

$Y \leftarrow L_{inf}$ or L_{der};

Z ←list of stopwords

Output: W^* ←list of stemmed words

Drop the stop words from input words, i.e., $W' = W - Z$;
for *each word(w_i) in W'* **do**
    w_i^{suffix} ←word_suffix(w_i);
    for **Light Stemmer**
    if *(length < 5)* **then**
        w_i^* ←w_i;
        W^* ←add $\{w_i^*\}$;
        continue;
    for *each y_j in L_{inf}* **do**
        if $y_j == w_i^{suffix}$ **then**
            length[w_i*] ← len[w$_i$] - len[w_i^{suffix}];
            W^* ←add $\{w_i^*\}$;
        else
            continue;
    end
    for **Aggressive Stemmer**
    if *(length < 7)* **then**
        w_i* ←w$_i$;
        W^* ←add $\{w_i^*\}$;
    for *each y_j in L_{der}* **do**
        if $y_j == w_i{}^{suffix}$ **then**
            length[w_i*] ←len[w_i] - len[w_i^{suffix}];
            w_i^* ←w_i;
            W^* ←add $\{w_i^*\}$;
        else
            continue;
    end
    return W^* ;
end

6.4 Test Collection

IR is an experimental science, typically based on the Cranfield style of evaluation popularized by the Text Retrieval Conference (TREC). TREC [4] provides test collections for different languages so that different retrieval models can be tested in laboratory settings. The ideology of TREC is followed by other text processing consortia like CLEF [5], NTCIR [6] and FIRE [7]. They also developed different text collections for European and Asian languages.

Since 2000, CLEF has been working on European languages in multilingual and multimodal information access evaluation involving monolingual as well as cross-lingual IR systems. The NTCIR workshops started in 1998 focus on East-Asian languages like Japanese, Korean, Thai and Chinese. Since 2008, FIRE has also been working on different IR issues to provide a testbed of various South Asian languages for performing experiments on text processing. The testbeds for Indian languages such as Hindi, Gujarati, Bengali, Marathi, Odia and English have been built. Test collections created by the different consortiums for different languages primarily focus on newswire data that are freely available. The corpus comprises different categories of news like politics, business, entertainment, health, sports, technology, and culture. Even though different test collections are created for a number of Indian languages, no such corpus exists for IR experimentation in Sanskrit.

The scarcity of digitized text and the presence of very few news publishers are two major issues in Sanskrit text collection building. Although a few datasets are available for some NLP tasks [68], they are not suitable for text search and retrieval in particular. The digitized text in Sanskrit is mostly either not fit for text-processing tasks (available in an image or pdf format) or from literature and/or religion genres containing overwhelmingly domain-specific terms. In this study, we build a Sanskrit IR text collection of news genres. The fundamental reason for considering news documents is that news contains content from diverse backgrounds, such as politics, business, sports, science, and other current affairs, covering almost all walks of life. The words used in the news genre are contemporary and of general purposes that can work for a standard domain-independent IR system. Besides IR, the collection can also be used for other text-processing tasks such as text categorization, summarization, named entity recognition, and so on.

[4] https://trec.nist.gov/

[5] https://clef2020.clef-initiative.eu/

[6] http://research.nii.ac.jp/ntcir/index-en.html

[7] http://fire.irsi.res.in/fire/2020/home

6.4.1 Data Collection

We build a text collection in Sanskrit comprising daily or periodical newspapers. The corpus was built by extracting the news documents from 'All India Radio Sanskrit News' and 'Samprati Vartah News' from 2015 to 2019. Since the data were extracted from multiple sources with different formats, text extraction required handling different images and advertisements. The original text data is non-standard, so we transcoded it into a common format, UTF-8. The collection consists of 7,057 documents. The longest document in the collection is 61 KB, whereas the shortest is 156 bytes. The mean number of indexing terms per document is 56.04. We also create a set of information needs (called *topics*) for the document collection by the domain experts. We chose 50 topics. The topics cover information needs of national and international importance, such as politics, economics, technology, health, sports, etc. The relevance judgments are independently made by two human assessors who are experts in the language and formulated the queries. The statistic of the corpus is shown in Table 6.4. The text collection is publicly available on Github [8].

Table 6.4: Statistics of Test Collection

	Sanskrit
Size (in MB)	11
# of documents	7057
Number of indexing terms per document	
Mean	56.04
Median	44
Standard deviation	90.34
Maximum	2788
Minimum	12
Number of tokens in the longest document	3250
Number of tokens in the shortest document	14
Number of topics	50
No. of words extracted for direct and gold standard based evaluation	1240

[8]https://github.com/cse-iitbhu/Sanskrit-Text-Collection

6.4.2 Data Processing

Since the data comes from multiple sources with different forms and formats, extracting text involves cleaning data, removing formatting tags, and handling images and advertisements. Some text data are originally in image form; we apply OCR techniques to convert them into textual format. There are some errors in the OCR output. The OCR output is checked by experts in Sanskrit, and errors are resolved manually. We also assign each news document a unique doc−id and convert it into TREC format. Based on the TREC model, a typical document starts with a unique document number <DOCNO> followed by <HEAD> and the content of the document is described inside the body (<BODY>). Figure 6.1 shows an example of a document.

<DOC>
<DOCNO>10319</DOCNO> <HEAD> कावेरीजलं दास्यतीति कर्णाटकम् । </HEAD>
<BODY>
कावेरीनदीतटे वर्तमानेभ्यः कृषकेभ्यः कार्षिकवृत्यर्थं नदीजलं दातुं सन्नद्धमिति कर्णाटकराज्येन निश्चितम् । एत−दनुकूलं निश्चयपत्रं [प्रमेयः] कर्णाटकस्य द्वाभ्यां विधानसभाभ्यामपि अङ्गीकृतम् । कावेरीजलं तमिलनाडु राज्याय निश्चयेन देयमिति सर्वोच्चन्यायालयस्य कर्कशादेशस्य आधारे एवायं निश्चयः । विधानसभायाम् अवतारिते निश्चयपत्रे यद्यपि तमिलनाडु राज्यस्य नाम नोल्लिखितं तथापि तत्रत्यानां कृषकाणामपि जलं लभ्येत । किन्तु दीयनानस्य जलस्य परिमाणमधिकृत्य निर्णयः नाभवत् ।
</BODY>
</DOC>

Figure 6.1 : An example of a document in Sanskrit

6.4.3 Topic Creation

An information need is a sequence of terms called a query on a topic the user wants to know. Using random combinations of query terms as an information need is generally not a good idea because they do not resemble the actual information needs. To properly evaluate a system, test information needs must be designed by domain experts and judged for relevance in the test document collection. Initially, we designed 55 queries by domain experts through manual observation of the document collection. We discarded a few queries because our retrieval system could not retrieve any relevant documents for them. Finally, a set of 50 queries is chosen for evaluation. Following the TREC model, each topic is organized into three logical sections: a brief title (under the <TITLE> section) comprising two to four words, followed by a description

<DOC>
<DOCNO>10319</DOCNO>
<HEAD> Water of Kaveri given to Karnataka </HEAD>
<BODY>
The Karnataka government has given the protesting farmers water from the Kaveri River. Both houses of the Karnataka government have approved the document about the water issue. According to the recommendation of the Supreme Court, the water of the Kaveri River should also be allocated to the Tamilnadu. Although the resolution tabled in the Assembly does not mention the name of the state of Tamil Nadu, farmers there will also get water. But there was no decision on the amount of water to be given.
</BODY>
</DOC>

Figure 6.2: English translation of the above document in Sanskrit

(under the <DESC> section) comprising a one-sentence user's information need, and a narrative (under the <NARR> section) that describes the relevant assessment criteria. An example of a topic is shown in Fig 6.3. The average length of TITLEs for the 50 topics is 2.92 words, the minimum query length is 2, and the maximum is 4. Like other languages, in Sanskrit, most users express information needs in 2 to 3 terms. The topic covers national events (e.g., 'Decision of Supreme Court of India for women entries in the Sabarimala temple', 'Renaming Allahabad to Prayagraj', etc.) and international events such as (e.g., '10th BRICS summit', 'Donald Trump becomes president of U.S.A.', etc.). In the given experiments, we consider only a query's title (T) section.

<TOPICS>

...

<TOP>

<NUM>3</NUM>

<TITLE> दक्षिण–अफ्रीकायाः दशम–ब्रिक्स–सम्मेलनम् </TITLE>

<DESC> दक्षिण–अफ्रीकायाः जोहान्सबर्गे दशम–ब्रिक्स–सम्मेलनं भविष्यति । </DESC>

<NARR> दक्षिण–अफ्रीकायाः जोहान्सबर्गे पञ्चानां ब्रिक्सराष्ट्रप्रमुखाणाम् अध्यक्षतायाम् आयोजितस्य दशम–ब्रिक्ससम्मेलनस्य सम्बन्धिनः विषयाः अत्र भवेयुः । भारतस्य सुदृढ–पारस्परिक–सम्बन्धार्थम् एतत् सम्मेलनम् अति–महत्वपूर्णं विद्यते । अन्यत् किमपि राष्ट्रियम् अन्ताराष्ट्रियं वा सम्मेलनम् अत्र प्रासङ्गिकं नास्ति । </NARR>

</TOP>

...

</TOPICS>

Figure 6.3: An example of a topic in Sanskrit within a topic-set

<TOP>

<NUM>3</NUM>

<TITLE> 10th BRICS summit at South Africa </TITLE>

<DESC> 10th BRICS summit will be held in Johannesburg of South Africa </DESC>

<NARR> Relevant documents should outline the 10th BRICS summit in Johannesburg, South Africa. Discussion about other international summits is irrelevant. </NARR>

</TOP>

Figure 6.4: English translation of the above Sanskrit topic

6.4.4 Pool Creation

To evaluate the effectiveness of any IR system, on top of a document collection and a set of queries, we need ground truths or relevance judgments for each query: i.e., for every query, what are the documents from the collection that are relevant to the query? In other words,

we need to have ground truths as to whether a given document is relevant for a given query or not. If the collection is vast (say 1 million documents), for 50 queries, we thus need 50 $\times$ 1 million = $50M$ judgments, which is a time-consuming and expensive process. Hence, we follow the TREC-style Cranfield method of experiments [109], where a top sampling-based pooling strategy is adopted. We take a good number of diverse systems collectively assumed to retrieve all the relevant documents for a given query within top-k (for some reasonable k) ranks from the entire collection. This set of documents is called a pool for the query. We create pools for 50 topics from the top 100 documents retrieved by different retrieval models such as BB2, BM25, Hiemstra_LM, IFB2, In_expB2, In_expC2, InL2, and TF-IDF supported by Terrier retrieval system [9]. The pooling process generates 5719 documents, including 427 relevant documents for 50 topics. The average number of pooled and relevant documents per query is 114.38 and 8.54, respectively. The pool statistic is shown in Table 6.5.

Table 6.5: Statistics of pooled documents

No. of topics	50
Pool-depth	100
No. of pooled docs	5719
No. of relevant docs	427
Average pooled docs per query	114.38
Average relevant docs per query	8.54

6.4.5 Relevance Judgments

For decades, relevance has served as a key criterion for evaluating the effectiveness of IR systems. Relevance from the users' perspective is characterized as multidimensional, situational, cognitive, emotional, and dynamic. It is thus difficult and time-consuming and also essentially subjective in nature. However, for objective evaluation, we consider user-based binary relevance judgments for simplicity, which is in line with standard practice at other traditional IR evaluation forums like TREC, CLEF, NTCIR and FIRE. For each query in the query set, all the documents in the pool are checked by two domain experts in Sanskrit (doing PhD in Sanskrit after their Master's degree in the language) independently to label them as relevant or not.

[9] http://terrier.org/

6.5 Experimental Setup

We build a Sanskrit text collection, especially for IR tasks. We then propose and evaluate different stemming strategies on the collection from NLP and IR perspectives. We also compare the MAP scores of different stemming approaches vis-a-vis the no-stemming technique (None) as the baseline. The proposed stemming technique is also compared with existing verb-based stemmer [90]. Nair et al. [90] proposed a rule-based suffix stripping technique that only strip the verb suffixes in Sanskrit. The verb-based stemming technique is extracted from GitHub [10].

6.6 Evaluation

As already introduced earlier, the proposed stemmers are evaluated in two ways: direct and indirect. We describe them in the following.

6.6.1 Direct Evaluation

As discussed in Sec. 3.3.1, direct evaluation is done based on the ground truths generated by our linguistic experts. We experiment with four different stemming approaches: two of our language-based approaches, light and aggressive stemmers, and two state-of-the-art techniques: verb-based (language-specific) and GRAS (language-independent) [92].

The confusion matrices for verb, light, aggressive and GRAS stemmers are shown in Table 6.6, 6.7, 6.8 and 6.9 respectively. Table 6.10 shows the effectiveness of verb, light, aggressive and GRAS stemmers in direct evaluation. Overstemming, understemming, precision, recall, F_1 measures are computed based on formulae provided in Sec. 3.4.1 from the Table 6.6 - 6.9. The number of actually stemmed words (s) are 219, 891, 875 and 425, respectively, for verb-based, light, aggressive and GRAS stemmers. ICF calculations are thus shown below. The results show that the verb-based stemming technique offers the poorest effectiveness among all, much lower than the light and aggressive stemming approaches. The primary reason is that the verb-based stemmer only strips the suffixes of verbs and ignores the suffixes of nouns and adjectives. In Sanskrit, many morphological inflections are found in nouns and adjectives. The stemmer does not generate root words for those suffixes, which reduces its effectiveness. We also observe that the verb-based stemmer produces high over-stemming and under-stemming errors. Our proposed stemmers are also showing better performances than the state-of-the-art language-independent stemmer 'GRAS'. Paik et al. [92] proposed a graph-based stemmer called 'GRAS'. The stem-

[10] https://github.com/s00rajsnair/sanstem

ming technique detect the suffixes statistically and generate root word. They found that the stemming method improves retrieval effectiveness in highly inflectional languages. Among the different stemming techniques, aggressive stemmer is a clear winner in all respects of direct evaluation.

Table 6.6: Result-summary of the verb-based stemmer

	Actually Stemmed	Not Stemmed
Words to be stemmed	82	715
Words not to be stemmed	137	306

Table 6.7: Result-summary of the light stemmer

	Actually Stemmed	Not Stemmed
Words to be stemmed	796	84
Words not to be stemmed	95	265

Table 6.8: Result-summary of the aggressive stemmer

	Actually Stem	Not Stemmed
Words to be stemmed	820	90
Words not to be stemmed	55	275

Table 6.9: Result-summary of the GRAS stemmer

	Actually Stem	Not Stemmed
Words to be stemmed	336	574
Words not to be stemmed	89	241

Table 6.10: Result summary of direct-based evaluation

Stemmers	No. of words	Over stemming	Under stemming	ICF	Precision	Recall	F_1 measure	Accuracy
Verb	1240	62.55%	89.71%	32.26%	37.44%	10.28%	16.13%	31.29%
Light	1240	10.66%	9.54%	76.3%	89.34%	90.45%	89.89%	85.5%
Aggressive	1240	6.28%	9.89%	76.7%	93.71%	90.11%	91.87%	88.3%
GRAS	1240	20.94%	63.07%	67.33%	79.05%	36.92%	50.33%	46.53%

$$ICF_{\text{verb}} = (\frac{1240 - 840}{1240}) \times 100 = 32.26\% \tag{6.1}$$

$$ICF_{\text{light}} = (\frac{1240 - 293}{1240}) \times 100 = 76.3\% \tag{6.2}$$

$$ICF_{\text{aggressive}} = (\frac{1240 - 288}{1240}) \times 100 = 76.7\% \tag{6.3}$$

$$ICF_{\text{GRAS}} = (\frac{1240 - 405}{1240}) \times 100 = 67.33\% \tag{6.4}$$

In English, Porter stemmer [98] achieves an ICF of 33% by considering 10000 English words, while our aggressive stemmer achieves an ICF of 76%, the best among all four stemmers we considered. The fundamental reason behind this is that Sanskrit is a more morphologically rich language than other languages like English.

Ramanathan and Rao [105] show that the rule-based stemmer provides an under-stemming and over-stemming error of 4.68% and 13.84%, respectively. Similarly, Suba et al. [136] observe that hybrid stemmer offers an accuracy of 90.7% and reduces index size by 52%. The light and aggressive-based stemmer give comparable effectiveness to rule-based Indian language stemmers ([105], [136]). However, our proposed stemmers suffer from some limitations. In light stemmer, we observe that over-stemming of words like अङ्कित (Ankit) (something that is drawn/painted or the name of a male person) and अङ्किता (Ankita) (a feminine name) are conflated to the same stem अङ्कित (Ankit), and under-stemming of words like अंगीकृतवान् (angikrutaban) to only → अंगीकृतव (angikrutab) reduces the overall effectiveness of the stemmer. Similarly, in aggressive stemmer, over-stemming of words like अंगीकृतः (angikrutah)→ अंगीकृ (angikru) and under-stemming of words like अग्रेसरतवं (agresaratwam)→ अग्रेसरत्व (agresaratwa) reduces the effectiveness of the stemmer. We also observe that certain words like नराणाम् (naranam) → नर (nara), वारिण्याम् (barinyam)→ वारि (bari) are not stemmed by the light stemmer leading to poor effectiveness of the light

stemmer compared to that of the aggressive one. The proposed stemmers also could not stem the compound words like अंगीकृतमस्ति (*angikrutamasti*). The primary reason is Sanskrit compound words are generated by rules of *sandhi*. In *sandhi*-ed compound words, the word suffixes are changed, and new words are produced that are difficult to handle by a rule-based stemmer. Sanskrit possesses diverse compound words where the stemming technique could not find the root word.

This kind of evaluation is adequate and accurate but certainly time-consuming. Hence, even though it can be comfortably attempted in the case of small-sized data, this evaluation is not feasible for large-scale data.

We also conduct a comparative evaluation of the results of verb, light, aggressive, and GRAS stemmers. Like the comparison made by Pande et al. [95], using Levenshtein distance [77], we implement the same method as a base measure. The Levenshtein distance is evaluated by the number of deletions, insertions, or substitutions required to transform the source string into the target string. The Levenshtein distance (LD) represents how many units a word has been stripped by a stemmer. We hypothesize that the light, aggressive, and GRAS stemmer is as efficient as that of the verb stemmer, and then the distributions of Levenshtein distances between a word and its stripped stem tend to be the same. For a given word, we have four treatments in hand: Verb, Light, Aggressive, and GRAS stemmer. Thus, for a given word, we have a pair of LDs. For comparison, we set the null hypothesis H_0 that no differences exist between the Verb, Light, Aggressive, and GRAS stemmers. We implement a statistical test for paired samples. In particular, a non-parametric statistical test, i.e., Wilcoxon signed-rank test with Bonferroni correction [128], is performed because there was no evidence about the distribution of these distances. On conducting the Wilcoxon test over sample data, we get the p-value less than $2.2e^{-16}$ between Verb-Inflectional (light) and Verb-Derivational (aggressive) stemmer. Similarly, Verb-GRAS stemmers provide p-value= 0.0134. Thus, we have strong evidence that the light, aggressive and GRAS stemmers provide significant differences to the Verb stemmer at a 95% level of confidence.

6.6.2 Indirect Evaluation

In indirect evaluation, we evaluate the stemming techniques using different retrieval models supported by the Terrier retrieval system. We use retrieval models such as BM25, TF-IDF, and DFR-based retrieval models like BB2, IFB2, INL2 and the Hiemstra language model. Table 6.11 shows the MAP scores for six retrieval models and four different stemming strategies. In Table 6.11, the 'none' column denotes the MAP scores of different retrieval models with-

out any stemming approach. We observe that the Verb stemmer performs poorly compared to other stemming approaches. The fundamental reason is that Sanskrit comprises many inflectional, derivational suffixes in nouns, adjectives and different cases (masculine, feminine and neuter), but the Verb stemmer ignores those suffixes that reduce the stemmer effectiveness. We also find that the proposed Light and Aggressive stemmers give better MAP scores than the no-stemming approach. It has been observed that the language-independent stemmers provide comparable effectiveness to the rule-based stemming approach in European and Indian languages; hence, we evaluate the language-independent stemmer as GRAS [92]. We took GRAS as the language-independent stemmer because it provides the best MAP score among different language-independent stemming techniques [92]. The last column in Table 6.11 shows another language-independent stemming-cum-indexing strategy called 'Trunc-6' [37]. In this approach, the words are truncated to their first six characters (e.g., 'education' produces 'educat'). We consider $n = 6$ because it provides the best MAP scores. The mean average precision for different trunc-n values is shown in Fig 6.5. The mean value indicates the average of six retrieval models and gives an overview of each stemming approach. Among the different retrieval models, the language model gives the best MAP score, and the best MAP score of the different stemming approaches is shown in bold.

Table 6.11: Mean average precision (MAP) of IR models vs. stemmers in Title-only (T) queries

	Mean Average Precision (MAP)					
Retrieval Model	None	Verb	Light	Aggress	GRAS	Trunc-6
BM25	0.4208	0.4244	0.4342	0.44	0.4351	**0.4382**
TF-IDF	0.4218	0.4241	0.4282	0.434	0.4299	0.4366
BB2	0.4079	0.4118	0.4232	0.4257	0.4132	0.4313
IFB2	0.411	0.4179	0.4314	0.4352	0.418	0.4317
InL2	0.4025	0.4056	0.4215	0.4253	0.4169	0.4293
Hiem_LM	**0.4339**	**0.4276**	**0.4431**	**0.4454**	**0.4389**	0.4275
Mean	0.4163	0.4186	0.4303	0.4343	0.4253	0.4324
% change		+0.57%	+3.35%	+4.31%	+2.16%	+3.87%

We observe that the stemming method improves the MAP scores compared to the baseline

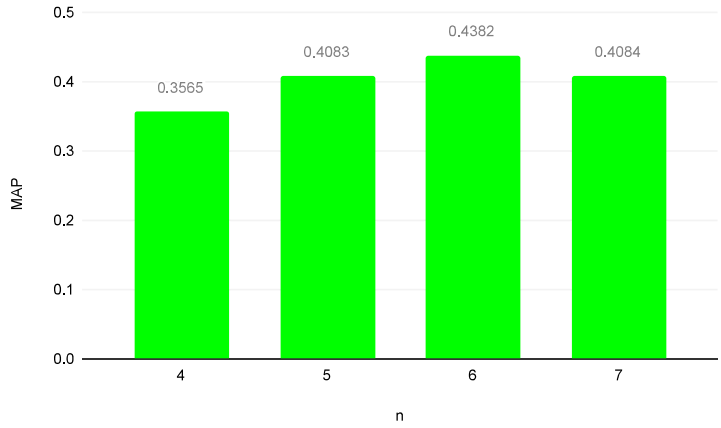

Figure 6.5: MAP for different trunc 'n'

approach. When the proposed stemming method is compared with no stemming technique, it is observed that the AP score of the Light stemmer is enhanced by 3.35% that of the Aggressive stemmer by 4.31% the GRAS stemmer by 2.16% and of 'trunc-n' indexing approach by 3.87%. These improvements in MAP score are quite comparable with that of a few European languages such as (+4.1% for Dutch, +4% for English). However, the proposed stemming method provides poor MAP score than different Indian languages such as (28% for Hindi, 42% for Marathi, and 18% for Bengali) [37] and European languages (+9% French, +15% Italian, +19% German, +29% Swedish, or +40% Finnish) [140]. The effectiveness of these stemmers is relatively poor. It can be attributed to the fact that we experimented with and evaluated the stemmers on a small dataset compared to other European and Indian languages. Evaluating the stemmer in a larger dataset is therefore an important future task. We also need to mention that the proposed stemming technique considers only the suffix information, not the prefix, which could be another factor behind the poor effectiveness [138].

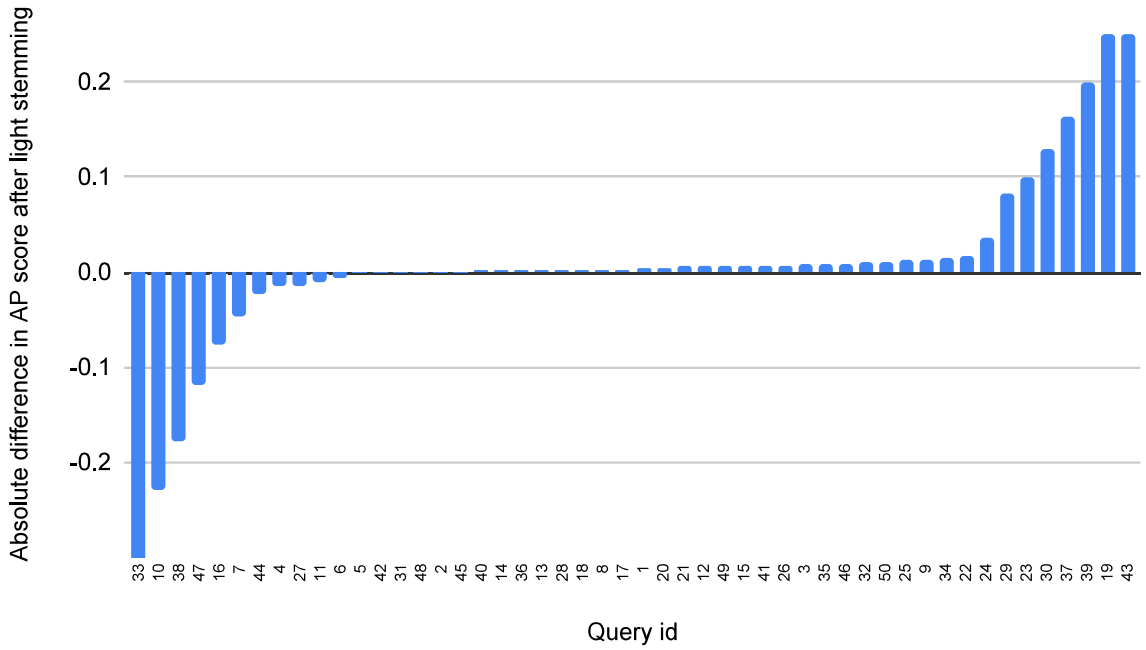

Figure 6.6: A query by query evaluation in light stemming by Hiemstra_language Model

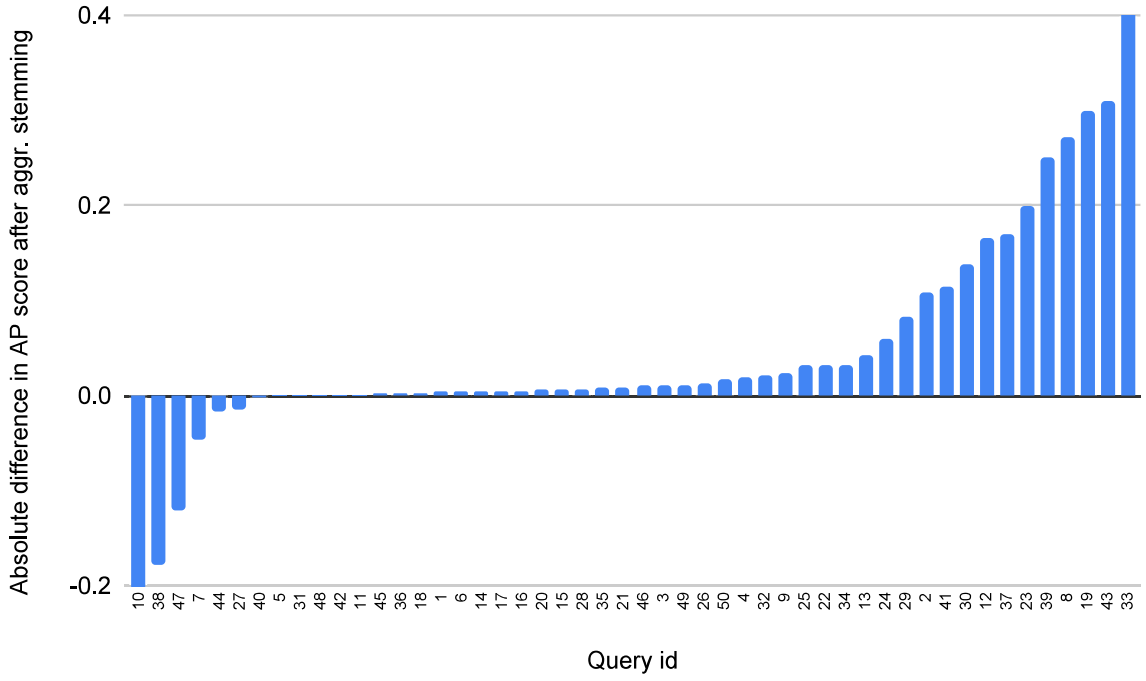

Figure 6.7: A query by query evaluation in aggressive stemming by Hiemstra_language Model

We also perform a query-by-query analysis. Here, we consider only the Hiemstra language model, one of the best retrieval models we experimented with. On closer examination, we find that the stemmers improved AP scores in more than 30 topics (35 with Light and 39 with Aggressive) compared to the stemmer without the stemming approach. The stemmer reduces the AP score in 11 topics for Light stemming and 7 topics for Aggressive stemming approach. The AP score of each query for light stemming and aggressive stemming approach is shown in Fig 6.6 and Fig 6.7 respectively. In Topic #24 भारते वस्तुसेवाकरः (Goods and Services Taxes in India), we observe that both Light and Aggressive stemmers improve AP score substantially compared to no stemming approach. The AP without stemming is 0.0627, and with stemming is 0.1243 (98.25% improvement). The improvements can be explained as, भारते (In India) is a proper noun which is found in different relevant documents in different forms (nominative, accusative and ablative), after stemming the root word is matched with many documents and enhances the effectiveness of the system. A similar observation is found in Topic #32 भारत–अमेरिकयोः आतङ्कवादसाहमत्यम् (India-America agreement on terrorism) where stemming improve system effectiveness significantly. In a few queries, we observe that the stemming technique reduces the effectiveness of the IR system. E.g., in Topic #10 जम्मूकश्मीरे आतङ्कवादसंघर्षः (Jammu Kashmir terrorist fight) the term जम्मूकश्मीरे (Jammukashmire) stemmed into जम्मूकश्मीर (Jammukashmir) stem is found in different documents; hence several documents are retrieved. But those documents do not discuss the incident of the 'Jammu Kashmir terrorist fight' and, therefore, are irrelevant. We also observe that compound terms like आतङ्कवादसंघर्षः (fight against terrorism) is a single word in a query whereas in the relevant documents, are in split form. A mismatch of query compound terms in documents reduces the effectiveness of an IR system. Similarly, in Topic #31 पाकिस्थाने विस्फोटनेन मृताः व्रणिताश्च (bomb-blast causes people's death in Pakistan). In the query, there are only four terms. None of the relevant documents contains all four terms. After stemming the terms like मृताः (mrutah) stemmed into मृत (mruta), and विसफोटनेन (bisfotanena) stemmed into विस्फोटन (bisfotana) find a lot of matches in non-relevant documents with higher term weights lowering the rank of relevant documents. The ranks of relevant documents are lowered after stemming compared to the no-stem baseline. We also observe that the query contains terms like पाकिस्थान (Pakisthan) where the relevant documents contain the term पाकिस्तान (Pakistan), which causes term mismatches in relevant documents and reduces the effectiveness of an IR system.

6.7 Summary

Sanskrit is a heritage language that is less studied computationally. In this study, we built a text collection for Sanskrit and presented the morphological difficulties in Sanskrit text processing. We also proposed two rule-based stemmers, where one (called Light) strips the inflectional suffixes, and another (called Aggressive) strips both inflectional and derivational suffixes. To evaluate different stemming strategies, we applied direct and indirect evaluation. In direct evaluation, we observed that the stemming methods significantly enhance system effectiveness, and our proposed stemmers outperform some state-of-the-art language-specific and language-independent stemmers. In indirect evaluation, we consider several retrieval models like BM25, TF-IDF, DFR-based and language models. Among them, Hiemstra_LM gives the best MAP scores. The aggressive stemming technique performs best among all stemming methods experimented with. Although the stemming strategy improves the average precision score in most queries, it causes a drop in a few queries due to a high number of compound words. The presence of compound words is a serious challenge in Sanskrit text retrieval that we need to address and is attempted in the following chapter.