Chapter 8

Discussions

This chapter recalls the summary of the dissertation's main contributions. In this thesis, we propose different pre-processing strategies for text processing in a series of principled approaches and evaluate their effectiveness using standard measures. Primarily, we study the effect of stopword removal, stemming and decompounding techniques in the Indian language IR.

8.1 Observations

We summarize and highlight here the main findings and contributions of the thesis in light of the research questions broadly guiding our research goals stated in Chapter 1. The critical discussion of the entire work will also reveal the underlying connection across the preceding chapters.

Issue 1. Why are most of the Indian languages low-resourced? What are the challenges in building text collection in Indian languages?

In an NLP application, the primary requirement is a large amount of data. People across the world speak thousands of languages, but only a few languages have a text corpus of millions of words. English has the largest amount of data on the web, followed by Chinese, Spanish, and Japanese. In terms of the number of native speakers, Hindi is the third largest spoken language in the world. On the other hand, most of the languages spoken in Asia and Africa lack the training data required to build NLP and IR systems. These languages are called low-resource languages. In other words, a language lacking large monolingual or parallel corpora, linguistic and NLP tools is considered a low-resource language.

Many test collections were built by TREC [1] for English NLP and IR. Similarly, CLEF [2], NTCIR [3] and FIRE [4] provide test collections for European and Asian languages. However, in low-resource Indian languages, few corpora, linguistic and NLP tools are available on the web. Evidently, attempts need to be made to improve this situation. So, we built a small news corpus in Sanskrit, primarily for IR tasks that were hitherto missing in the FIRE collection. We also explore different stemming strategies in Sanskrit. We believe our work directly addresses this lacuna in Sanskrit NLP and IR domains.

Sanskrit is a heritage language with limited linguistic resources and is thus less studied computationally. The scarcity of digitized text and the presence of very few news publishers are two major issues in Sanskrit text collection building. Moreover, there is a lack of trained professionals in the field of Sanskrit computational linguistics who can work on developing language resources for NLP and IR tasks. Although a few datasets are available for different NLP tasks [68], they are not suitable for text search and retrieval. The digitized Sanskrit text 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. Additionally, the existing Sanskrit texts suffer from poor quality, such as spelling, grammar, and syntax errors. All of these make it challenging to use the available Sanskrit texts for NLP and IR tasks.

We built a text collection in Sanskrit of the news genre and presented the morphological difficulties in Sanskrit text processing (Chapter 6). The corpus comprises daily or periodical newspapers. The corpus is built by extracting the news documents from 'All India Radio Sanskrit' and 'Samprati Vartah'. The statistic of the corpus is shown in Table 6.4. The text collection is publicly available on Github [5]. Text collection can be used in other text processing tasks like text categorization, text summarization, named entity recognition, and other NLP tasks.

Issue 2. What pre-processing steps should be applied in different Indian languages text processing?

Pre-processing is an essential step for NLP tasks in Indian languages, as it helps improve the models' effectiveness. Extensive research has been conducted on the preprocessing strategies

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

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

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

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

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

in European languages, such as tokenization, stopword removal, stemming, lemmatization, POS tagging and named entity recognition, but it is less explored in the low-resource Indian languages. The fundamental reason behind is less availability of document collection in electronic format and lack of technically skilled manpower, coupled with a lack of annotated corpora, lexicons, and NLP tools available for low-resource Indian languages. In recent years, multilingual data has grown substantially on the web, which has a fair share of non-English low-resource languages, including Indian languages. Hence, different researchers recently proposed a few linguistic tools such as named entity recognizer, spell checker, pos-tagger, stemmer and morphological analyzer for low-resource Indian languages.

This thesis focuses on three pre-processing strategies: stopword removal, stemming method and decompounding techniques in Indian language IR. First, we evaluated the impact of different non-corpus-based and corpus-based stopword removal in the Indian language IR. We downloaded the non-corpus-based stopword list from the web and proposed different corpus-based ones by applying different statistical approaches and evaluating their effectiveness in the IR domain. Second, we proposed different stemming strategies and evaluated their effectiveness in the Sanskrit text analysis domain. Finally, we investigated different corpus-based, hybrid machine learning-based and deep learning-based decompounding models in different Indian languages and evaluated their effectiveness from an IR perspective.

Issue 3. What are the effects of different pre-processing strategies on Indian language IR in general?

We explored three different pre-processing techniques, namely stopword removal, stemming and word decompounding in Indian languages and examined their effectiveness in text processing tasks.

First, we evaluated the effect of stopword removal using a non-corpus-based stopword list in the Indian language IR. Examination in different Indian languages (Tables 4.1, 4.2, 4.3 and 4.5) shows that stopword removal using the non-corpus-based stopword list enhances the effectiveness of an IR system. We also explored the relationship between stopwords and average document length. (Tables 4.6, 4.7, 4.8, and 4.10) reveal the relationship between stopwords and average document length. We observed that the effect of stopword removal is quite low in short documents compared to long ones. We also found that the different retrieval models prefer long documents over their shorter counterparts. Although stopword removal using non-corpus-based stopword lists improves MAP scores in different Indian languages, only a few of the stopwords from the list are actually found in a given collection, while many are absent. We also noticed that stopwords are collection-specific.

Hence, we proposed different corpus-based stopword lists in Indian languages and evaluated the effectiveness of their removal in the IR task. We compared the MAP scores after stopword removal using the corpus-based stopword list with those using the non-corpus-based stopword list. Experiments on different Indian languages (Tables 5.1, 5.2, 5.3, 5.4 and 5.5) show that stopword removal using either non-corpus-based or corpus-based stopword list improves retrieval effectiveness of an IR system. Using corpus-based stopword lists, however, outperforms retrieval using non-corpus-based lists. The effect of stopwords varies from one language to another. Table 5.6 shows that using a smaller length of a corpus-based stopword list outperforms using a larger length of a non-corpus-based stopword list, as far as IR effectiveness is concerned.

We also proposed two stemmers, i.e., 'light' and 'aggressive', for the Sanskrit text processing task and evaluated their effectiveness from NLP and IR perspectives. The 'light' stemmer strips the inflectional suffixes, and the 'aggressive' stemmer strips both inflectional and derivational suffixes and improves the effectiveness of an IR system. Table 6.10 shows that stemmers are effective from an NLP point of view. Similarly, Table 6.11 shows that the stemming technique improves the effectiveness of an IR system. Examination of Table 6.11 reveals that the 'aggressive' stemmer offers the best MAP scores and outperforms the 'verb', 'light', 'GRAS' and Trunc-n based indexing/stemming approaches.

Finally, we studied different decompounding models, such as corpus-based, hybrid machine learning-based and deep learning-based and their effects on improving retrieval effectiveness in Indian language IR. Table 7.2, 7.3, and 7.4 show the effect of corpus-based decompounding models in Indian languages IR. Similarly, the impact of hybrid machine learning-based and deep learning-based decompounding models in Indian languages IR are shown in Tables 7.5, 7.6, 7.7, 7.8, 7.9 and 7.10. Examining different tables shows that the corpus-based decompounding models are ineffective, and attention-based deep learning models perform best in Indian language IR.

Issue 4. How are the chosen pre-processing steps interrelated? Do their combinations add up to the overall improvement of retrieval effectiveness? If yes, to what combinations and to what extent?

The combinations of pre-processing steps can significantly impact the retrieval effectiveness of an IR system. The effectiveness of a pre-processing combination depends on several factors, such as the language, the genre of the text, and the task at hand. We studied the effectiveness of three common pre-processing steps in different Indian languages: stopword removal, stemming,

and decompounding techniques. Stopword removal eliminates some commonly used words that are unlikely to be useful for relevant document retrieval. Stemming reduces the morphological variants of a word to their base or root form by stripping the suffixes and prefixes from an inflected word. Decompounding technique splits the compound words into their original form, such as the term 'notebook' being split into 'note' and 'book' potentially resulting in more hits with the query and improving the system's document ranking. The order in which these steps are performed also affects the overall effectiveness of a retrieval system. For example, stemming before stopword removal may result in some keywords being incorrectly removed if they are modified by stemming. The specific algorithms used for each pre-processing step can also impact the overall effectiveness. For example, different stopword removal methods, stemming algorithms and decompounding techniques may produce different results and may be more or less appropriate for a specific language or a genre of text. In this thesis, we used a corpus-based stopword list (IDF-based approach), language-independent stemming technique, i.e., GRAS [92] and Bidirectional-LSTM with attention-based decompounding model respectively that provide the best retrieval effectiveness in Marathi. The effect of different pre-processing strategies is shown in Table 8.1. For Marathi, the stopword removal improves MAP score by 2.32%. The combination of stopword removal and stemming technique improves the MAP score by 30.48%. The combination of stopword removal, stemming and decompounding techniques improved the MAP score by 41.3%. We believe that the same trend can be found in other Indian languages as well.

Table 8.1: MAP scores before and after pre-processing steps in Marathi retrieval

Techniques	BM25	TF-IDF	InL2	InexpC2	BB2	LM
Baseline (No stopword removal, stemming and decompounding)	0.2833	0.2840	0.2504	0.2155	0.2533	0.2692
Stopword removal	0.2882	0.2888	0.2563	0.2205	0.2579	0.2686
Stemming	0.3171	0.3202	0.2900	0.2760	0.2964	0.3039
Decompounding	0.2921	0.2918	0.2736	0.2671	0.2766	0.2763
Stopword removal + Stemming	0.3179	0.3212	0.2945	0.2812	0.2989	0.3022
Stopword removal + Decompounding	0.2955	0.2958	0.2775	0.2679	0.2804	0.2756
Stemming + Decompounding	0.3189	0.3245	0.2961	0.2840	0.3012	0.3115
Stopword removal + Stemming + Decompounding	0.3196	0.3260	0.2969	0.3043	0.3027	0.3133

In summary, the interrelation among pre-processing steps impacts retrieval effectiveness, and the specific combinations of steps used will depend on the data and application. Experimentation and evaluation of different combinations of pre-processing steps can help to determine the most effective approach for a given task. For example, using stopword removal, stemming, and

decompounding together improves retrieval effectiveness compared to using just one or two of these techniques alone. However, the extent to which these combinations improve the system's effectiveness will depend on the specific language and dataset. A pre-processing combination that is effective for one task or domain may not be effective for another task or domain. However, notwithstanding these nuances, it can be said in general that pre-processing steps are interdependent, and how they are combined can significantly impact the overall retrieval effectiveness. Stopword removal, stemming and decompounding techniques uniquely affect IR tasks. We recommend that they must be applied in the given sequence for the best results, especially for Indian languages, as shown in Table 8.1.

8.2 Limitations

In this Section, we outline the few limitations of our study.

This dissertation used Marathi, Bengali, Gujarati, Hindi and English language topics built during FIRE [6] evaluation campaign. We used only 50 topics for evaluation. In Sanskrit, we created the collection (documents + topics + relevance judgments), but due to the small document collection size, we could not build more than 50 topics. The dissertation's observation may not be scaled up to a larger collection, particularly for Sanskrit.

While building the Sanskrit test collection, we built the pool using a limited number of retrieval models. It is likely that a new system completely different from the systems we considered may bring some new relevant documents that are outside the pool. Hence, the system may be under-evaluated using the pool. The problem can be overcome by enriching the pool with mechanisms like active learning [103], interactive search and judgements, etc.

[6] http://fire.irsi.res.in/fire/static/data