Chapter 1

Introduction

This chapter serves as the starting point of the thesis and establishes the key concepts and vocabulary used in the rest of the document. We begin with a general introduction to information retrieval in Section 1.1. Different components of an information retrieval system are described in Section 1.2. Different steps of an IR system are described in Section 1.3. Section 1.4 provides the motivation for our work. Section 1.5 provides an overview of the dissertation. Section 1.6 highlights the research goals by articulating different research questions. We summarize the main contributions of the thesis in Section 1.7. Finally, Section 1.8 presents the layout of the rest of the thesis.

1.1 Information Retrieval

Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) [84]. The term 'unstructured data' means the data does not have a clear, semantically overt, easy-to-understand (for a computer) structure. It is the opposite of structured data. An example of structured data is a relational database, whereas unstructured data is a web page. During the 1950s, IR was an activity only a few people engaged in specific domains like library science, judiciary, and medicine would require and use. After personal computers and the World Wide Web (WWW) emerged, hundreds of millions of people started using IR through a web search engine, desktop search or personal device search facilities to meet their information needs.

IR is considered a discipline in computer science that deals with the representation,

storage, organization and access to information items [14]. The block diagram of an IR system is shown in Fig 1.1. The IR model used in the system comprises the following four components.

- D is a set of documents in the collection.
- Q is a set of queries or user's information needs.
- F is a framework for modelling text collections and queries.
- R (q_i, d_j) is a ranking function associated with a real number and ordering the documents for a query q_i.

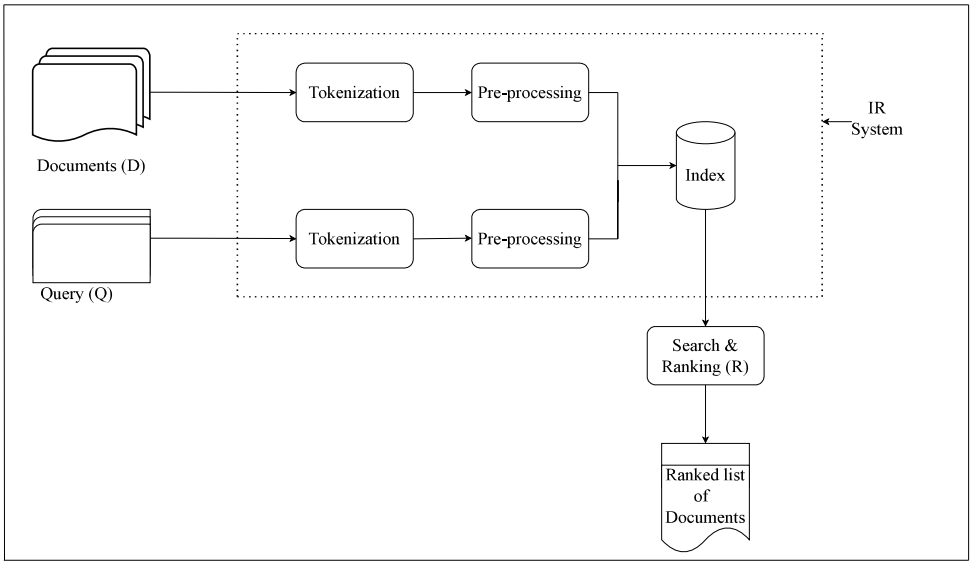

Figure 1.1: Block diagram of an information retrieval system

To build a model, we first think about the representation of documents and information needs. Given these representations, we then conceive the framework in which they can be modelled. This framework should also provide the intuition for constructing a ranking function. For instance, in the classic Boolean model, the framework comprises sets of documents and the standard operation on sets. In the classic vector model, the framework consists a t-dimensional vector space and standard linear algebra operations on vectors. Similarly, in the probabilistic model, the framework consists of sets, standard probability operations and Bayes' theorem [14].

1.2 Input to an IR system

Primarily, three inputs to an IR system are described below.

1.2.1 Document collection

It is the set of documents from which we retrieve a subset of documents meeting some search criteria. It is also called a corpus. The collection can be static or dynamic. In the static collection, the set of documents does not change and contains the same documents during indexing and retrieval, including several of its iterations. However, in the dynamic collection, a document can be added to or deleted from the collection at any time. The corpus may comprise text documents or web pages, including multimedia items (e.g., images, audio and videos). In this thesis, we only use static text collection.

1.2.2 Query

A query is the textual representation of an information need of a user. A query comprises a set of terms. From the system's point of view, a query is expected to represent an apparent, concise, and well-defined information item so that it can be effectively retrieved. Also, a clear and well-defined query can help determine whether retrieval is correct or not. However, it cannot be expected or ensured in a commercial search engine where users pose free queries.

1.2.3 Relevance Judgments

To evaluate the effectiveness of an IR system, ground-truth information. i.e., which documents are relevant to a given query are also required. This information is needed not only for a single query but for a set of queries where the set comprises queries of different difficulty levels covering the entire document collection. This information is fed to the system only during the evaluation of the system and not used during retrieval. Hence, it is not shown in Fig 1.1 and will be discussed further in due course.

1.3 Different steps in an IR system

Both the documents and queries go through several steps in an IR system. We briefly describe the steps below.

1.3.1 Tokenization

This is the first step for both documents and queries. Each document (query) is scanned and split into a sequence of words called tokens. A token can be a single word or a phrase representing a single concept. Tokens are identified based on regular expressions, delimiters, and explicit definitions from a dictionary.

1.3.2 Pre-processing

It is an essential step for an IR system. All the tokens obtained in the previous step are not equally important - while some carry important semantic information, some are used to conform to a syntactic structure. Also, there can be several variations of a query word used in different documents. If we attempt to match a query word in its raw form with that in a document, we may not find it, even though the document discusses the same concept. Hence, we need to clean and normalize the tokens using different pre-processing techniques that are applied to both the queries and the documents. Some of them are described below.

Stopword Removal

In natural languages, a significant number of words are used to complete the syntactic structure of a sentence, but they do not carry much information. For example, in English, articles like 'a', 'an', 'the', prepositions like 'in', 'on', 'at','of', conjunctions like 'and', 'but', 'yet', 'though', etc. These words are called stopwords, noise words, or negative dictionaries in general [16]. An observation in Brown corpus shows that 42% of the words belong to a set of hundred frequently used words in the corpus, and they form only 0.1% in the lexicon. However, 5.7% of the words in the corpus belong to 58% in lexicon [15]. It is also observed that the top-ranked documents are retrieved due to semantically informative words rather than frequently used words or stopwords [78]. Also, the presence of stopwords dilutes the importance of key terms in a query. Hence, considering only the important words

reduces the vocabulary size without much loss of information [121]. These stopwords can be obtained from linguists, who have a general nature and can apply to all kinds of document collection. We call it *non-corpus-based* stopword list. However, we can also find a list of stopwords specific to a given document collection using linguistic knowledge (e.g., statistical techniques). We refer to this stopword list as *corpus-based* one.

Stemming

In a morphologically rich language, many words have a single root but differ in word forms. For example, 'education', 'educating', 'educated', and 'educational' have their root word 'educate'. These words are called morphological inflections of the root word and are obtained after adding suffixes like 'ion', 'ing', 'ed', and 'ional' to the root word. Stemming is a mechanism that transforms morphological variants of a word to its root word by stripping the suffix and/or prefix from the word. If we strip the suffixes using stemming, we get the root word, which may not be semantically meaningful (here 'educat')- we call it a *stem*. Stemming is a mechanical way of getting stems. Another alternative technique is lemmatization, which requires linguistic knowledge and always gives semantically meaningful roots (here 'educate') called *lemma*. This thesis evaluates the effect of stemming techniques in Sanskrit IR.

In IR systems, stemming has two benefits. One, stemming reduces index size significantly by conflating several terms. Two, it improves an IR system's effectiveness by retrieving many potentially relevant documents [55]. In the current state of the art, the stemmers are broadly categorized into rule-based (language-dependent) and statistical (language-independent). In a language with good linguistic resources, such as English, many rule-based stemming algorithms ([98], [79]) have been proposed and evaluated.

Decompounding

In natural languages, compound words are formed by concatenating two or more words. Compound words create challenges for NLP applications like machine translation, text classification, speech recognition, IR, etc. In the IR domain, compound words can cause a vocabulary mismatch between a query and the documents in the collection. For example, if a query comprises the term 'book', and the collection does not contain any document with the term 'book' but a few comprising the term 'notebook', no documents will be

retrieved if the IR system employs term-by-term matching. However, the document containing the term 'notebook' might be relevant. The mismatch problem could be addressed if the compound term 'notebook' was decomposed into its constituents: 'note' and 'book' in both the query and the documents. This method of splitting a compound word is called decompounding. Decompounding is an essential pre-processing step for morphologically rich languages like German, Finnish, Dutch, and many Indian languages. This thesis investigates the effect of decompounding techniques in different Indian languages (Marathi, Hindi and Sanskrit) IR.

1.3.3 Indexing

An index is a data structure listing the essential and unique terms present in a collection where each term points to a list of documents where they occur. The process of creating an index is called indexing, and it is done after preprocessing the document collection. It is represented as a combination of a dictionary (a dictionary can be considered as a list of unique words present in the collection) and a posting list (a list of documents in increasing order of document IDs for each term). It is used for a random search for keywords in the query. An index enables faster search on a vast document collection over a simple linear search. For example, in a web search engine, the system has to provide a search over billions of documents stored on several thousands of computers. Without indexing, the search engine would have to linearly scan every document in the collection, requiring a lot of time. Indexing is thus a technique to construct an index that enables easy and fast search over the documents in a large collection.

1.3.4 Search and Ranking

Search is the technique for preparing a list of documents matching the query terms. Search is performed on the index built from the document collection. For a given query (q), a ranking algorithm ranks the set of documents already searched based on similarity scores computed between the given query and each document (d) in the searched list. The ranking is a fundamental and crucial step in the implementation of an IR system behind search engines. The ranking is used in various applications, including recommender systems. Most search engines and IR systems implement different ranking algorithms.

1.4 Motivation

Since the beginning of the World Wide Web (WWW) (the 1990s), the Web has primarily been comprised of English data. Different researchers proposed linguistic tools in English, such as tokenizers, stemmers, lemmatizers, POS taggers, and morphological analyzers, to improve the effectiveness of NLP and other text-processing tasks. Some of these tools are also used as components of an IR system. However, this is different for non-English languages, especially for low-resource languages with a low presence on the web. With the advancements of technology, there has been a substantial growth of the multilingual data in the WWW [19]. The multilingual data comprises a wide range of low-resource languages other than English. The low-resource languages have different kinds of lexical, morphological, syntactic, and semantic level variations. The morphological variations in low-resource languages complicate the IR process. Hence, researchers proposed different linguistic tools such as POS taggers, stemmers, lemmatizers, and morphological analyzers in low-resource languages.

Diab [35] presented the second-generation tools named AMIRA for Arabic. The AMIRA toolkit includes a clitic tokenizer, a part of speech (POS) tagger and a base phrase chunker-shallow syntactic (check) parser. Shao et al. [124] evaluated a character-based model for joint segmentation and POS tagging for Chinese. The bidirectional RNN-CRF architecture captures rich contextual information and sub-character level features and outperforms the state-of-the-art models in joint segmentation and POS tagging tasks. Straka et al. [134] proposed contextualized word embedding models in Czech text for POS tagging, lemmatization, dependency parsing and named entity recognition. Tolmachev et al. [139] introduced a combination of a feature-rich linear model with a recurrent neural network-based language model (RNNLM) for morphological analysis in the Japanese language. Kovriguina et al. [67] presented a maximum entropy tagging model and neural net dependency parser for the Russian language. The NLP tool allows fast and scalable processing of large text collections and can be easily integrated with other tools in the CoreNLP pipeline. A wide variety of linguistic tools are available for European languages like French, Russian, Czech, etc., and East-Asian languages like Korean and Japanese. However, it is less explored in low-resource Indo-Aryan languages like Marathi, Bengali, and Sanskrit.

In the past few years, different researchers have developed a few linguistic tools for low-resource Indian languages. Raulji et al. [106], Dabre et al. [26], Das and Das [29] proposed morphological analyzer for Sanskrit, Marathi and Bengali languages, respectively. Deshmukh and Kiwelekar [33] and Kabir et al. [61] investigated a deep learning-based POS tagger for Marathi and Bengali languages. Mohnot et al. [88] proposed a hybrid POS tagger for Hindi. Priyadarshi and Saha [101] created the first POS tagger for Maithili. Sahu and Pal [115] explored the effect of stopwords in Bengali, Marathi, Gujarati and Sanskrit languages. They observed that stopword removal improves the effectiveness of the IR system. Dolamic and Savoy [37] evaluated the effect of stopword and stemming method in Bengali, Marathi and Hindi IR. Ganguly et al. [43] investigated the impact of their decompounding model in Bengali IR. It has been observed above that the different morphological analyzers, POS taggers, stopword removals, stemming techniques, and decompounding methods improve effectiveness in different morphologically rich European and East-Asian languages.

From the above studies, we find the impact of different pre-processing strategies (stopword, stemming and decompounding) is less explored in Indo-Aryan languages like Marathi, Gujarati, and Sanskrit IR [57], [126]. This thesis explores and evaluates the effect of different pre-processing strategies, such as stopword removal, stemming methods and decompounding techniques in Indian language IR.

1.5 Dissertation Overview

The overarching goal of this dissertation is to study the impact of different pre-processing strategies, such as stopword removal, stemming techniques, and decompounding methods in different Indian languages IR.

First, we study the effect of non-corpus-based (stopword lists generated from linguistic knowledge and not specific to any document collection) in different Indian languages (Bengali, Marathi, Gujarati and Sanskrit) IR. We notice that stopwords are collection-specific (stopword list extracted from the collection in hand). Only a few stopwords are found in a given collection when stopwords are non-corpus-based, while many are absent. On the other hand, some words in a given collection are repetitive and redundant. These words can also be considered stopwords for the given collection. In corpus-based stopword

removal, we generate different corpus-based stopword lists using statistical approaches. We also investigate the impact of different stopword lists in the IR domain and compare their effectiveness with that of non-corpus-based ones. The proposed corpus-based stopword lists outperform non-corpus-based ones and improve effectiveness in Indian languages IR.

Second, we build a Sanskrit text collection for text-processing tasks. We also propose stemming techniques and evaluate their effectiveness in the NLP and IR domains. Different stemming techniques have been shown to improve effectiveness in Sanskrit IR.

Third, we explore different decompounding models, such as corpus-based, hybrid machine learning-based and deep learning-based, in different Indian languages (Marathi, Hindi and Sanskrit). We observe that the different decompounding models improve effectiveness in Indian languages IR.

We conduct the experiments using the Terrier[1] retrieval system. Terrier supports different retrieval models, such as probabilistic retrieval models (BM25 and TF-IDF), DFR-based models, and the Hiemstra language model. Some of the results obtained are also compared against the standard European languages (German and English) collection.

1.6 Research Goals

The broad objective of the research presented here is to study the impact of pre-processing strategies in different Indian languages and evaluate their effectiveness on text processing, particularly in the framework of IR settings. The following issues broadly guide our work.

1. Why are most of the Indian languages low-resourced? What are the challenges in building text collections in Indian languages?
2. What pre-processing steps should be applied in different Indian languages text processing?
3. How do different pre-processing strategies affect Indian language IR in general?
4. How are the different pre-processing steps interrelated? Whether their combinations add up to the overall retrieval effectiveness? If yes, which combinations and to what extent?

[1] http://terrier.org/

We formulate the following research questions (RQ) to study and explore solutions to the above broad issues.

RQ1: Does stopword removal improve effectiveness in the Indian languages IR? If yes, to what extent? Which stopword lists (non-corpus-based or corpus-based) are the most effective in Indian language IR?
This broad-level RQ is further split into a few micro-level RQs for conducting experiments and observations as follows.
RQ 1.1: *At the gross level, is there any impact of non-corpus-based stopword removal on Indian languages IR?*
RQ 1.2: *Do stopwords have any relationship with average document length from the perspective of retrieval effectiveness? In other words, how does retrieval effectiveness change with the number of stopwords and document length?*
RQ 1.3: *Is there any difference in retrieval effectiveness between non-corpus-based and corpus-based stopword removal?*
RQ 1.4: *Can corpus-based stopword list improve retrieval effectiveness? If yes, to what extent? Among the different corpus-based stopword lists, which one performs best in the IR domain?*
RQ 1.5: *Does the length of a corpus-based stopword list affect retrieval effectiveness of Indian languages? If yes, to what extent?*

RQ 1.1 and **RQ 1.2** on non-corpus-based stopwords are explored in Chapter 4 for a number of Indian languages. Similarly, **RQ 1.3**, **RQ 1.4**, and **RQ 1.5** for different corpus-based stopwords are studied in Chapter 5.

RQ2: 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?
RQ3: Do stemming techniques improve the overall effectiveness of the Sanskrit retrieval system? If yes, to what extent?
RQ4: For Sanskrit IR, what is the most suitable stemmer: whether to use a light stemmer, an aggressive stemmer, or a language-independent one during indexing?
The **RQ2**, **RQ3** and **RQ4** are presented in Chapter 6.
RQ5: Is there any impact of word decompounding in the Indian language IR? If yes, to what extent?

Like before, this broad-level RQ is further divided into the following sub-research-questions.

RQ 5.1: *Can corpus-based decompounding models be used in the Indian language IR? If yes, how?*
RQ 5.2: *Can machine learning and deep learning decompounding models be applied in Indian language IR? If yes, how?*
RQ 5.3: *Among the different decompounding models (corpus-based, hybrid machine learning-based, and deep learning-based), which one provides the best retrieval effectiveness?*
RQ 5.4: *Which IR model provides the best retrieval effectiveness when decompounding is handled?*

all the research questions (**RQs 5.1- 5.4**) are investigated in Chapter 7.

1.7 Contribution and Impact

This thesis studies the impact of different pre-processing strategies in Indian language IR. The main contributions of this thesis are related to the data creation, implementation, and evaluation of the different pre-processing strategies in the IR domain, where we address several existing issues using standard models and methods. Our contributions are instrumental in advancing the state of the art by creating a new dataset: Sanskrit text collection for text processing tasks. The following experiments are conducted on several corpus-based, machine learning-based and deep learning-based techniques in different Indian language text collections that provide insights into the usefulness of different techniques.

1.7.1 A study on stopwords in Indian languages IR

Stopwords are commonly used words that are often removed from text data during natural language processing (NLP) tasks to improve the system's effectiveness and reduce noise in the data. A body of literature exists on stopword removal for improving effectiveness in different computational tasks like information retrieval [78], text classification ([6], [58], [13]), and text segmentation. Although stopword removal is an important pre-processing

step, it is less explored in Indian languages IR. This study examines the effect of different non-corpus-based and corpus-based removal in Indian languages IR.

Our studies reveal that non-corpus-based and corpus-based stopword removal improves retrieval effectiveness in different Indian languages IR. However, using the corpus-based stopword lists outperforms the non-corpus-based ones while IR effectiveness is evaluated. We also find that there can not be a single particular length of the stopword list that can be used for different Indian languages. The recommended length of the stopword list varies from one language to another. It is also found that good retrieval effectiveness can be achieved using a smaller corpus-based stopword list than its non-corpus-based counterpart or its larger version. Furthermore, we study the effect of stopwords on retrieval effectiveness over document length. The impact of stopword removal is generally low in short documents compared to their long counterparts across the Indian languages.

1.7.2 Creation of a text collection and a study on stemming techniques on Sanskrit

Sanskrit is an Indo-Aryan or Indic language belonging to the Indo-European language family. It originated in the second millennium BCE as Vedic Sanskrit. A standardized version of classical Sanskrit is presented in 'Ashtadhyayi'. Many modern languages, like Hindi, Bengali, Gujarati, Punjabi, Bengali, and Assamese, evolved from Sanskrit [135]. Literature in Sanskrit includes a rich body of poetry, drama, philosophical, religious, scientific and technical text. However, an effective and efficient search system is yet to be available for searching Sanskrit documents because of specific linguistic features and difficulties generated. Even though it is not so popular in day-to-day life, Sanskrit is still widely used to worship Hindu deities, for ceremonial purposes, and for some Buddhist practices in chants. It is listed among 22 languages in the eighth schedule of the constitution of India. Sanskrit is the second official language in the Indian province of Uttarakhand. However, despite the importance of Sanskrit in different languages, literature and geographical locations, there is no standard benchmark text collection for IR in Sanskrit.

In this study, we built a text collection for the Sanskrit language and investigated different indexing, stemming, and searching strategies. The text collection is comprised of

daily or periodical newspapers. The corpus was built by extracting the news documents from 'All India Radio Sanskrit News' and 'Samprati Vartah News' between 2015 and 2019. Since the raw text is extracted from multiple sources with different formats, text extraction requires handling images and advertisements along with the text. The original text data is non-standard, so we transcode it into UTF-8. The collection includes national and international information like politics, economics, technology, health and sports. We also propose a *light* stemmer that handles inflectional suffixes like singular and plural endings (e.g., 'girl' and 'girls' as in English) or feminine and masculine variants (e.g., 'hero' and 'heroine' as in English) to the same stem and an *aggressive* stemmer that handles derivational suffixes (e.g., 'assess' and 'assessment' as in English) to the same stem. We observe that the different stemming strategies improve the effectiveness of an IR system. The proposed stemmers are compared with GRAS stemmer [92], language-independent indexing approach (trunc-n) and no stemming approach. Among different stemming methods, the aggressive stemmer is the most effective in the IR domain.

1.7.3 A study on word decompounding methods in Indian languages IR

In natural languages, compound words are formed by concatenating two or more words. The Indian language compound words are categorized into two types. First is a typical compound word: a simple concatenation of two or more words. E.g., 'note' and 'book' are combined to form a 'notebook'. Second is a *sandhied* compound word: according to sandhi rules, the tail constituent of the first component changes at the merging location (end of the first word and start of the second word). For example, विद्या (vidya) and अपायते (apayate) are combined to form विद्यापायते (vidyapayate). These modifications make splitting a sandhied compound word challenging in Indian languages. An additional feature of the Indian language compound word is that one of the compound constituents may not be used independently as a meaningful word. For example, in उपनगर (upanagar), उप (upa means smaller in size or stature) does not have independent use on its own but used as the prefix of another word. In the IR domain, compound words generate a vocabulary mismatch between a query and the documents in the collection. Hence, the decompounding technique improves the effectiveness of an IR system.

This study explores the effect of different corpus-based, hybrid machine learning-based and deep neural network-based decompounding models in Indian language IR. The corpus-based model primarily comprises dictionary-based approaches. We used hybrid machine learning-based models such as SVM, CRF, KNN, Decision Tree, and Random Forest. Additionally, we explore the deep neural network-based models in Indian languages.

We notice that the decompounding models improve the effectiveness of an IR system. Among them, the deep learning-based models outperform the corpus-based and hybrid machine learning-based models in Indian languages IR. The attention-based deep learning model is the most effective in Indian language IR. The corpus-based models exhibit poor retrieval effectiveness in Indian language IR. We also observe that the effect of decompounding models is higher in Marathi than in Hindi and Sanskrit IR.

1.8 Structure of the Thesis

The thesis is organized into eight different chapters. A brief description of the chapters is as follows:

Chapter 2 provides a detailed survey of the different pre-processing strategies in text analysis applications. We formalized the state-of-the-art pre-processing strategies in three subsections. First, we present the literature study of the stopword removal method in text-processing applications. Secondly, we discuss the impact of stemming strategies in a wide range of applications. Finally, we present the effect of decompounding strategies in the text analysis task.

Chapter 3 presents the mathematical formulae used in different retrieval models. It also describes the simulation setup, datasets, evaluation metrics and statistical tests.

Chapter 4 discusses the impact of non-corpus-based stopword list in Indian language IR. An extensive set of experiments has been conducted in different Indian languages IR. We also study the relationship between stopwords and average document length in the IR domain.

Chapter 5 describe the corpus-based stopword generation approaches in different Indian languages. We also investigate the impact of different corpus-based stopword lists in the IR domain. Moreover, we compare different non-corpus-based and corpus-based

stopword lists in the Indian language IR.

Chapter 6 presents the text collection building for Sanskrit text processing. We also investigated different indexing, stemming, and searching strategies in Sanskrit. We propose two stemmers in Sanskrit called 'light' and 'aggressive' stemmers and evaluate their effectiveness in the NLP and IR domains.

Chapter 7 explores different decompounding techniques in Indian language IR. We study three decompounding techniques: corpus-based, hybrid machine learning-based, and deep learning-based in different Indian languages. Moreover, we evaluate the effectiveness of different decompounding models in the IR domain.

Chapter 8 outline the discussions and a few limitations. It also describes the overall impact of preprocessing steps in the Indian language IR.

Chapter 9 brings the concluding remarks and suggestions for future work. A pictorial representation of the structure of the thesis organization is depicted in the figure below:

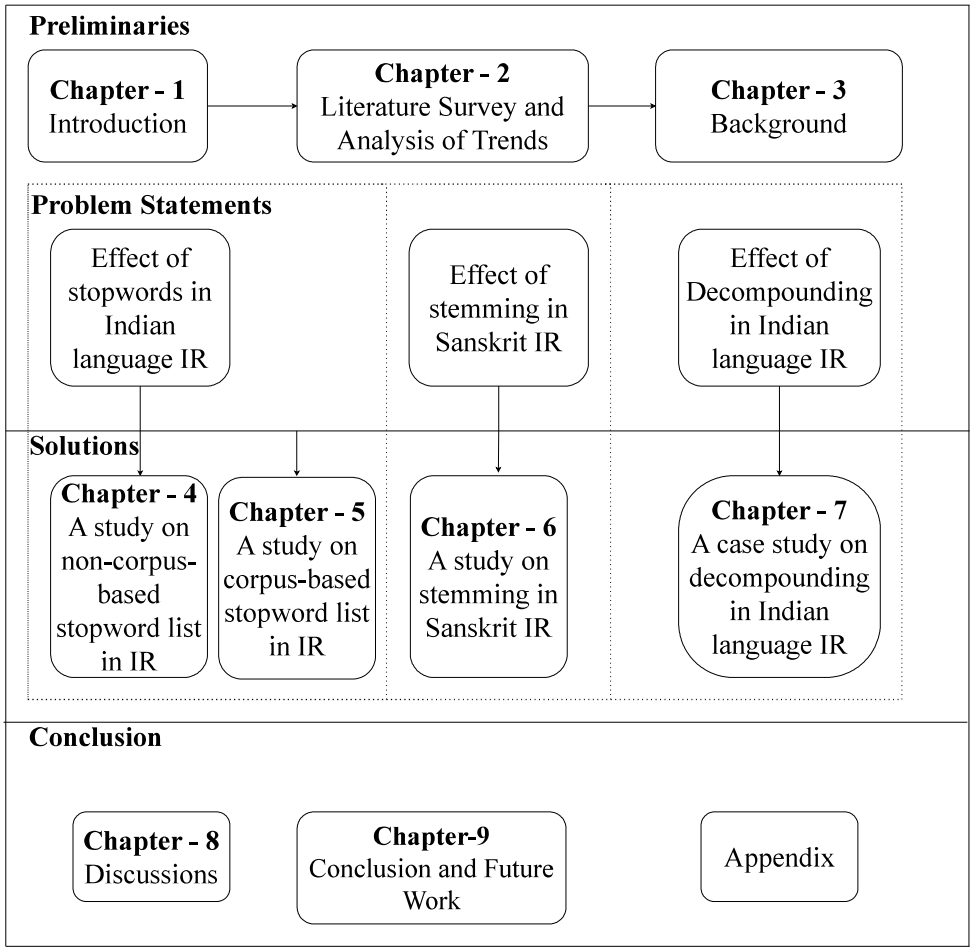

Figure 1.2: Structure and organization of thesis