ISM@FIRE-2012 Adhoc Retrieval Task and Morpheme Extraction Task

Avinash Yadav, Robins Yadav, Sukomal Pal

Dept. of CSE,

Indian School of Mines, Dhanbad, India.

(raoavinash@yahoo.co.in, robins.yadav@gmail.com, sukomalpal@gmail.com)

Abstract

This paper describes the work that we did at Indian School of Mines, Dhanbad for FIRE 2012. This year we participated in two tasks: Adhoc Retrieval Task and Morpheme Extraction Task (MET). Within the adhoc task, we participated in two monolingual retrieval activities, namely English and Hindi using Lemur and Indri search engine respectively. We submitted a total of 6 runs (3 in English and Hindi respectively) in the adhoc task. In MET, we submitted a morpheme extraction tool which takes a single file of documents in any language as input and produces a two-column file listing words in the language and their corresponding stemmed root words.

1 Introduction

This is second year of participation from Indian School of Mines Dhanbad at FIRE. At FIRE-2012, we have participated in two tasks. The first one is Adhoc Retrieval Task and the second one is Morpheme Extraction Task (MET). Within ADHOC Retrieval Task, we have participated in two monolingual retrieval activities for English and Hindi using Lemur and Indri search engine respectively. We submitted a total of 6 runs (3 in English and Hindi each) in the Adhoc Task. Out of the three runs in English, first run is retrieved without stemming, the second one is retrieved using Krovetz stemmer and the third one is retrieved using our ISMStemmer. Out of three runs in Hindi language the first one has been retrieved without stemming, the second one is retrieved by performing stemming using ISMStemmer on corpus only and the third one has been retrieved by applying ISM Stemmer both on corpus and queries. In MET, we submitted a morpheme extraction tool which takes input a single file of documents in any language and produces a two-column file listing words in the language and their corresponding stemmed root words.

Our main objective for participation is to compare the efficiency and performance of ISMStemmer in comparison to other stemmers in English as well as in several Indian languages.

In adhoc task, our performance is not satisfactory as we were not prepared enough for the participation. However, we did reasonably well in MET task. ISMstemmer, being a language-

independent stemmer, worked on a set of Indian languages and performed better than baseline retrieval results for Marathi, Gujarati, Bengali and comparable to baseline for Odia language.

The paper is organized as follows. Section 2 discusses about related works. Section 3 focuses on stemming. Section 4 discusses about our participation and results and section 5 concludes with directions for future work.

2 Related work

Stemming is a popular technique in IR which has proven to enhance recall in general, but for the morphologically complex languages improves precision as well [2]. Stemming algorithms are broadly classified into two categories, namely rule-based and statistical. Rule-based stemmers work on a set of pre-defined language-specic rules, whereas statistical stemmers employs statistical information from a large corpus of a given language in order to learn the morphology. A statistical stemmer obviates the need of language-specific expertise, and therefore, is often a preferred choice, specifically in information retrieval. There are mainly three kinds of approaches to language-independent stemming. The first kind of methods take a set of words and try to find probable stems and suffixes for each word; other methods look for association between lexicographically-similar words by analyzing their co-occurrence in a corpus. The third group of methods is based on character n-grams [7,8,9].
Paik et al. [7] proposed a graph-based statistical stemmer GRAS, where a set of word classes are formed each having a pivot word or stem. All the words within a class share a common prefix but have different valid suffixes. Valid suffixes are shortlisted pairwise based on their occurrence with other prefix words in the corpus.
However we try to discover suffixes based on the concept of frequent itemset generation technique according to apriori algorithm in market basket data analysis [1].

3 Stemming

The usefulness of stemming is shown to be mixed in languages such as English. But in case of morphologically complex languages, stemming produces a significant performance improvement. A number of linguistic rule-based stemmers are available for most European languages which employ a set of rules to get back the root word from its variants. But for Indian languages which are highly inflectional in nature, devising a linguistic rule-based stemmer needs some additional resources which are not available. We present a purely statistical and corpus based approach which extracts morphemes from inflected form of words.

For morpheme extraction we have used a stemmer developed here in ISM [1] which strips off suffixes from inflected words. This is language-independent as we do not use any language-specific grammar rule. It works on the frequency of words. Stemmer extracts the morphemes from the words depending on their frequency of occurrence (of suffixes). We first find the list of frequent suffixes and then we extract the morphemes by removing the suffixes from the inflected word form.

We first identify in the lexicon a set of all suffixes of length *n* (*n* = 1, 2 . . .) by grouping words that share the same suffix and the number of words in a group becomes the frequency of the corresponding suffix. A suffix is called a potential suffix if its frequency in the lexicon is larger than a certain cut-off threshold. The rationale is that variant word forms of a language are generated by adding suffixes taken from a finite set of suffixes specific to the language and given a large enough corpus they also occur sufficiently frequently. So the frequency is a good indicator of the potentiality of a suffix.

In order to find the valid suffixes reverse the strings appearing in the lexicon (list of unique unstemmed words) collected from the given document collection. Than perform the lexicographical sorting on these words so that all the words sharing the common suffixes will be together. After that derive the suffixes from the entire corpus depending on the predefined threshold value. Only the characters having frequency higher than the threshold qualify as 1-character suffix. The set of 1-character suffixes also serve as the potential candidates while generating 2-character suffixes. A 1-character suffix which did not cross the threshold cannot be part of a 2-character suffix since its frequency is expected to be less or equal to that of 1-character suffix. Hence discard such 1-character suffixes during discovery of 2-character suffixes. Scan again and collected frequency for all possible 2-character suffixes. The set of 2-character suffixes crossing the threshold are considered valid 2-character suffixes and form the candidate-set for 3-character suffixes. Continue the process as long as a frequent n character suffix can be generated i.e. if the frequencies of such suffixes are greater than the predefined threshold value then those suffixes are valid suffixes.

The stemmer works in the following phases:

1. Pre-processing of corpus: our suffix-finding algorithm works on single file having single column. So as a pre-processing step we fetch the contents of all the documents in the corpus and put them into a single file.

2. Data cleaning: An early step of pre-processing is to divide the input text into units called tokens where each token is either a word or a number or a punctuation mark. Words are not always surrounded by white spaces. Often punctuation marks are attached to words, such as commas, semicolons, and period (full stop). So before processing all the punctuation marks, apostrophes, special and numeric characters are removed.

3. Remove stop words: there are many frequently used common words which are of little importance. These words are called stop words. Exclusion of these stop words from the cleaned documents make the collection smaller which save the required space and processing time.

4. Convert the whole refined single file corpus into a single column.

5. Find valid suffixes list as follows:

a. Sort the single file single column corpus.
b. Find the unique words from the sorted word file.
c. Reverse the unique sorted word file.

d. Again sort the reversed word file.
e. Find the frequent suffixes (of length 1-character, 2-characters and so on).
f. Find valid suffixes whose frequency is above a pre-decided threshold value α.

6. Reverse the original single column file.

7. Match each of the reversed word with the valid suffix list. If the word contains any of the suffixes, strip the matched suffix from the word such that the stemmed word has a longest prefix of length at least the threshold β.

8. Reverse the stemmed word to get the original stemmed word.

By the above mentioned algorithm we can find the stemmed word list which is used during indexing of document corpus.

4 Our Participation

We participated in two tasks

1. Adhoc Retrieval Task

2. Morpheme Extraction Task.

4.1 Adhoc Retrieval Task

Adhoc Task is been participated in on two languages. The approach for both of these languages is same at the first step but differs at second level.

4.1.1 Approach

The Adhoc Retrieval task is based on two steps. These are:-

1. Indexing.

2. Retrieval.

4.1.1.1 Indexing

The indexing task approach is same for both English and Hindi Languages.

During **indexing** documents are prepared for use by an IR system. This means preparing the raw document collection into an easily accessible representation of documents. This transformation -from a document text into a *representation of text* is known as **indexing** the documents. Transforming a document into an indexed form involves the use of

- a library or set of regular expressions

- parsers
- a library of stop words (a stop list)
- other miscellaneous filters

This is normally done in five steps: markup & format removal, tokenization, filtration, stemming and weighting. If no markup removal and weighting is required, then this transformation involves only tokenization, filtration and stemming. This type of indexing is frequently found in databases that merely sort text files and raw data. On the Web, however, the above five steps are used especially since documents are created in different formats and relevance scores are needed.

The indexing system builds compressed inverted lists for each term and field in memory. Periodically, as memory gets scarce, this data is flushed to disk. The data that is written to disk is self-contained: it contains all information necessary to perform queries on that data. In a sense, an Indri index can be considered a set of smaller indexes. The retrieval system has been written to be able to query many indexes together. The indexer also stores a copy of the incoming document text in compressed form. This text is commonly used to produce document snippets at retrieval time. The index subsystem is capable of storing any text that can be represented in Unicode.

For Indexing, we used IndriBulidIndex command of Indri search engine.

4.1.1.2 Retrieval

We have used raw-tf based vector space model (VSM) and language modelling (LM) approach for retrieval.

4.1.2 English Retrieval

For English Retrieval, we used raw-tf based VSM of Lemur Project. Firstly, a parameter file needed to be constructed for that. For retrieval we used Vector Space Model of Information Retrieval. A total of 50000 documents are retrieved for 50 queries counting 1000 per query.

4.1.2.1 Data

Documents

Documents on which task has to be carried out were received from FIRE 2012 organisers in form of a corpus in compressed format. Firstly we uncompressed it and then task was carried on the extracted corpus which sized about 1 GB. It consisted of text files from two sources. First was The Telegraph newspaper and the other was BD news (BanglaDesh news) of time span 2006-2010.

Topics

We also got a set of 50 queries (176-225). Each query consists of different sections in English Language.

4.1.2.2 Results

We retrieved 3 runs after performing above mentioned steps. They are:-

RUN#1

ism.english.unstemmed.avinash-EE_TDN.txt.res

This run has been retrieved by performing the task without using any stemmer.

Run # 2

ism.english.krovetzstemmer.result_EE_TDN.txt.res

This run has been retrieved by performing task same as above run except by using Krovetz stemmer.

RUN #3

ism.english.ismstemmer.result_EE_TDN.txt.res

This run has been retrieved by performing the task same as above run except by using ISM stemmer

The summary of the English results is as given below in Table 1.

Table 1. English Adhoc Task

(Number of quries: 50, results retrieved 1000 documents per query)

Run id	EE.ism.unstemmed	EE.ism.krovetzstemmer	EE.ism.ismstemmer
Number of Relevant documents	3539	3539	3539
Number of relevant documents retrieved	2503	2504	2415
Mean Average Precision (MAP) value	0.2264	0.2255	0.2096

4.1.3 Hindi Retrieval

For English Retrieval, command used is IndriRunQuery of Indri Search engine. Firstly, a parameter file needed to be constructed for that. For retrieval we used Vector Space Model of Information Retrieval. A total of 50000 documents are retrieved for 50 queries counting 1000 per query.

4.1.3.1 Data

Documents

Documents on which task has to be carried out were received from FIRE 2012 organisers in form of a corpus in compressed format. Firstly we uncompressed it and then task was carried on the extracted corpus which sized about 1.2 GB. It consisted of text files from two sources. First was The Amar Ujala newspaper and the other was Navbharat Times of time span 2001-2010.

Topics

We also got a set of 50 text queries (from-to). Each query consists of different sections in Hindi Language.

4.1.3.2 Results

We retrieved 3 runs after performing above mentioned steps. They are:-

RUN#1

Hindi unstemmed corpus and unstemmed query evaluation result:-

This run has been retrieved by performing the task without using any stemmer.

Run # 2

Hindi stemmed corpus and unstemmed query evaluation result

This run has been retrieved by stemming the corpus using ISMstemmer but the query is still unstemmed.

RUN #3

Hindi stemmed corpus and stemmed query evaluation result

This run has been retrieved by performing stemming on both corpus and query using ISM stemmer.

The results of our Hindi monolingual run is summarized in Table 2.

Table 2. Hindi Adhoc Task

(Number of quries: 50, results retrieved 1000 documents per query)

Run id	unstemmed	Stemmedcorpus.unstemmedquery	Stemmedcorpus.stemmedquery
Number of Relevant documents	2309	2309	2309
Number of relevant documents retrieved	222	98	209
Mean Average Precision (MAP) value	0.0173	0.0026	0.0137

4.2 Morpheme Extraction Task

We made a morpheme extraction tool using the ISMStemmer which takes input a single file consisting of all the documents in the corpus. The tool does case-folding, removes punctuation marks, tokenizes and then stems the tokens according to apriori algorithm stated above. Since our stemming algorithm is statistical and independent of any language grammar, it can work on any inflectional language. Hence it was applied on a number of Indian languages. The tool produces a 2-column file as output, where the first column gives the list of words in the language and the second column suffix-removed stems corresponding to the words in the first column.

The performance of the stemmer was evaluated through monolingual retrieval on different languages as given below (Table 3-.

TABLE 3. BENGALI

Institute	**Language**	**MAP Obtained**
Baseline	Bengali	0.2740
JU	Bengali	0.3307
DCU	Bengali	0.3300
IIT-KGP	Bengali	0.3225
CVPR-Team1	Bengali	0.3159
ISM	Bengali	0.3103

TABLE 4. GUJARATI

Institute	Language	MAP Obtained
Baseline	Gujarati	0.2677
ISM	Gujarati	0.2824

TABLE 5. HINDI

Institute	Language	MAP Obtained
Baseline	Hindi	0.2821
DCU	Hindi	0.2963
ISM	Hindi	0.2793
IIT-B	Hindi	NA

TABLE 6. MARATHI

Institute	Language	MAP Obtained
Baseline	Marathi	0.2320
ISM	Marathi	0.2797
IIT-B	Marathi	0.2684

TABLE 7. ODIA

Institute	Language	MAP Obtained
Baseline	Odia	0.1537
IIIT-Bh	Odia	0.1537
ISM	Odia	0.1537

TABLE 8. TAMIL[#]

Team	Language	MAP Obtained
Baseline	Tamil	NA
AUCEG	Tamil	NA
ISM	Tamil	NA

NA: Not Available

[+]Bicolumn file was not generated by this tool

[#]Qrels are not available. The results will be published once the qrels are available.

5 Conclusion

This paper describes our participation in FIRE-2012 English & Hindi monolingual task and Morpheme Extraction Task (MET). In adhoc retrieval task, we participated with a bare minimum set up based on open source tools. We have used Lemur platform and Indri Search Engine along with ISMStemmer developed at ISM. Initial results that we received on English and Hindi test collection using ISMStemmer are not up to the mark, but promising. There is obviously scope of further improvement for the stemmer.

Based on our performance in Morpheme Extraction Task (MET), we can say that our Morpheme Extraction tool is working well on all languages on which it has been tested. It is the only tool which has been working on Gujarati language at FIRE 2012, performing quite well on Marathi language and working above reference mark on Bengali and Odia languages. Although the performance on Hindi Language is below reference level but it is working well on Hindi Language based upon its functionality. There are some deficiencies in the tool, especially in rightly choosing the inflected words where suffix-stripping to be done and not to be done. We are working on this which will hopefully boost its performance further in future.

6 References

[1] Raktim Banerjee, Sukomal Pal: ISM@FIRE-2011 Bengali Monolingual Task: A frequency based stemmer, Post-proceedings of FIRE-2011, IIT Bombay. (*accepted*).

[2] www.**isi**cal.ac.in/~**fire**/ (as on 06.12.2012)

[3] *Cristopher D.Manning, Prabhakar Raghawan, Hinrich Schutze- An introduction to information retrieval, Cambridge University press 2008.*

[4] http://en.wikipedia.org/wiki/Information_retrieval (as on 06.12.2012)

[5]] https://sourceforge.net/p/lemur/wiki/Indri%20Query%20Language%20Reference/ ((as on 06.12.2012)

[6] www.lemurproject.org. (as on 06.12.2012)

[7] Paik, J. H., Mitra, M., Parui, S. K., and J¨ arvelin, K. 2011. GRAS: An effective and efficient stemming algorithm for information retrieval. *ACM Trans. Inf. Syst. 29, 4, Article 19 (November 2011),* 24 pages.

[8] Paik, J. H. and Parui, S. K. 2011. A fast corpus-based stemmer. ACM Trans. Asian Lang. *Inform. Process. 10, 2, Article 8 (June 2011),* 16 pages.

[9] Paik J. H.**,** Pal Dipasree**,** Parui S. K. A Novel Corpus-Based Stemming Algorithm using Co-occurrence Statistics. *SIGIR'11,* July 24–28, 2011, Beijing, China.

[10] Xu, J. and Croft, W. B. 1998. Corpus-based stemming using cooccurrence of word variants. *ACM Trans. Inf. Syst. 16,* 1, 61–81.

[11] http://en.wikipedia.org/wiki/Stemming (as on 06.12.2012)

[12]. How Effective Is Suffixing? Donna Harman. *Lister Hill Center for Biomedical Communications, National Library of Medicine, Bethesda, MD 20209.*