ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING

DEPARTMENT OF COMPUTING

A MASTER'S THESIS

ON

SILT'E TEXT INFORMATION RETRIEVAL SYSTEM BASED ON VECTOR SPACE MODEL

SUBMITTED IN PARTIAL FULFILLMENT FOR THE REQUIREMENTS OF THE DEGREE

OF

MASTER OF SCIENCE IN INFORMATION SYSTEM

BY

Jemal Zuber

September 2016

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY SCHOOL OF

ELECTRICAL ENGINEERING AND COMPUTING

DEPARTMENT OF COMPUTING

SILT'E TEXT INFORMATION RETRIEVAL SYSTEM BASED ON VECTOR SPACE MODEL

BY: JEMAL ZUBER

Name and Signature of Members of the Examining Board

Name	**Title**	**Signature**	**Date**
_____	Advisor	_____	_____
_____	Internal Examiner	_____	_____
_____	External Examiner	_____	_____

Adama, Ethiopia

September, 2016

APPROVAL SHEET

This Research Thesis entitled "**SILT'E TEXT INFORMATION RETRIEVAL SYSTEM BASED ON VECTOR SPACE MODEL**" has been read and approved as meeting the preliminary research requirements of the Department of Computing in partial fulfillment for the award of the degree of Master Science in Information System, Adama Science and Technology University, Adama, Ethiopia.

_____	_____	_____
Advisor	Signature	Date
_____	_____	_____
External Examiner	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
Institute coordinator for Research, P, GS & C	Signature	Date

Acknowledgments

Alhamdulillah, all praises to Allah for the strengths and His blessing in completing this thesis. Special gratitude goes to my advisor, Dr. Shimelis Assefa, for his constant support from the beginning of my thesis work. His invaluable help of constructive comments and suggestions throughout the thesis work have contributed to the success of this research. I would like also to express my sincere appreciation for his hospitable approach when I contact him.

I would like to express my deepest thankfulness to Dr.Wondwossen Mulugeta for his valuable feedbacks and support.

I would like to express my appreciation to the Silte Zone Education Bureau and the Silte Zone Culture, Tourism and Government Affairs Communication Bureau staff members, especially for Ato Bedru Hussen and Ato Surur for their support and help to get available electronic Silt'e language documents.

I would like to thank my family and my friends, for their support and prayers, and especially my thoughtful mother Mehdiya Yassin for her endless motivation.

Sincere thanks to my classmates for their kindness collaboration and moral support during my study.

Abstract

The main aim of an information retrieval system is to extract appropriate information from enormous collection of data based on user's need. In this research, we were concerned on text information retrieval for Silt'e language. We discussed its basic morphological features that have effect on the implementation and performance of information retrieval systems. In detail, we discussed about the model we used, that is the Vector Space Model, the architecture, prototype development tool and evaluation of the developed prototype. Different text preprocessors are offered by Apache Solr. For our prototype implementation we used the Apache Solr's standard tokenizer, standard stopword removal algorithms and term weighting and similarity computing functions. The prototype is evaluated using the corpus that was prepared from Silt'e school text books. We used ranked retrieval evaluation techniques which are average precision and mean average precision. The experimental analysis shows that the obtained mean average precision 81.4% result is encouraging to develop vertical Silt'e text information retrieval systems and Silt'e search engines.

Table of Contents

Acknowledgments I

Abstract II

List of tables VII

List of figures VIII

Nomenclatures List IX

CHAPTER ONE 1

The problem and its background 1

1.1. Introduction 1

1.2. Background of the study 1

1.3. Statement of the problem 2

1.4. Objectives of the study 3

1.4.1. General objectives 3

1.1.2. Specific objectives 3

1.5. Scope and limitations of the study 4

1.6. Significance of the study 4

1.7. Research methodology 4

1.7.1. The test collection 4

1.7.2. Indexing method 4

1.7.3. Performance evaluation 5

1.7.4. Prototype development tool 5

1.8. Thesis organization 5

CHAPTER TWO 6

Literature Review 6

2.1. Introduction to Silt'e language 6

2.1.1. Silt'e dialects 6

2.1.2. Silt'e writing system 6
2.1.2.1. Silt'e vowels 7
2.1.2.2. Silt'e consonants 8
2.1.3. Silt'e grammar 8
2.2. Overview of information retrieval system 14
2.2.1. Introduction to information retrieval system 14
2.2.2. Development of information retrieval system 14
2.2.3. Information Retrieval Models 16
2.2.3.1. Boolean model 16
2.2.3.2. Vector Space Model 16
2.2.3.3. Probabilistic Model 18
2.3. General process in information retrieval system 19
2.3.1. Indexing 19
2.3.1.1. Preparing documents 19
2.3.1.2. Tokenization 19
2.3.1.3. Stopword removal 19
2.3.1.4. Stemming 20
2.3.1.5. Term weighting 20
2.4. Related Works 22
CHAPTER THREE 25
Detailed Discussion on Research Methodologies 25
3.1. Collection organization 25
3.2. Vector space model techniques 25
3.2.1. Term frequency (tf) 26
3.2.2. Inverse document frequency (idf) 26
3.2.3. Term weighting 26

3.2.4. Measuring document similarity 27
3.3. Document indexing 27
3.3.1. Inverted indexing 28
3.4. Performance evaluation 29
3.4.1. Recall 29
3.4.2. Precision 29
3.4.3. Average precision 30
3.4.4. Mean average precision 30
3.5. Prototype development tool overview 31
3.5.1. Apache Solr 31
3.5.1.1. Attraction of Apache solr 31
3.5.1.2. Characteristics of Apache Solr 31
3.5.1.3. Schema design in solr 32
3.5.1.4. Text analysis in solr 33
3.5.1.5. Indexing and querying in solr 33
3.5.1.6. Collection specific tools in solr 34
3.5.2. Text extracting tools 36
CHAPTER FOUR 38
Processing Silt'e Text and Experimentation 38
4.1. Silt'e information retrieval system architecture 38
4.2. Tokenizing Silt'e text 39
4.3. Removing Silt'e Stopword 41
4.4. Indexing Silt'e document 42
4.5. Searching for Silt'e documents 43
4.6. Performance evaluation 45
4.6.1. Measuring system effectiveness 46

4.6.2. User oriented effectiveness evaluation 50

CHAPTER FIVE 52

Conclusion and Recommendation 52

5.1. Conclusion 52

5.2. Recommendations 53

References 54

Appendix I. Lists of Ethiopic alphabets used for Silt'e 59

Appendix II. Lists of Silt'e stopwords 60

Appendix III. Lists of Silt'e prefixes 61

Appendix IV. Lists of Silt'e suffixes 62

Appendix V. Schema structure for Silt'e text information retrieval system 63

List of tables

Table 2. 1 Phonetic order of Ge'ez alphabets 7
Table 2. 2 Silt'e consonants 8
Table 2. 3 Paucal formation using suffixation 9
Table 2. 4 Paucal formation using consonant reduplication 9
Table 2. 5 Silt'e personal pronouns 10
Table 2. 6 Silt'e gender marker for possessive markers 10
Table 2. 7 Paucal adjective formed by suffixation 11
Table 2. 8 Paucal adjective formed by reduplication 11
Table 2. 9 Adjectives derived from nouns 12
Table 2. 10 Sample Silt'e adverbs 12
Table 2. 11 Noun morphology with prefix and suffix 13
Table 2. 12 sample Silt'e verbs 13
Table 4. 1 Sample Silt'e stopwords list 41
Table 4. 2 Relevancy for the selected queries 46
Table 4. 3 Precision-recall measure based on relevancy 48
Table 4. 4 System effectiveness measures 49
Table 4. 5 System wise evaluations 51

List of figures

Figure 2. 1 Document and query representation on Euclidean space 17
Figure 2. 2 Silt’e text processing steps 22
Figure 3. 1 Inverted index of two Silt’e documents..... 28
Figure 3. 2 The Silt'e_tirs core statistics and instance information 35
Figure 3. 3 Sample analysis tool result 35
Figure 3. 4 Information about the field content_type 36
Figure 4. 1 Silt’e information retrieval system architecture 39
Figure 4. 2 Sample Silt’e text to test tokenizer effectiveness 40
Figure 4. 3 Tokenized Silt’e texts 40
Figure 4. 4 Sample Silt’e text to test performance of stopword removal algorithm..... 42
Figure 4. 5 Removing Silt’e stopword analysis result 42
Figure 4. 6 Number of indexed documents and words 43
Figure 4. 7 Sample search result 44
Figure 4. 8 Prototype interface..... 50

Nomenclatures List

ASTU = Adama Science and Technology University

APS = Apache Lucene Solr

Silte_TIRS = Silt'e Text Information Retrieval System

POI = Poor Obfuscation Implementation

R = Relevant

NR = Non-relevant

Doc Id = Document Id

CHAPTER ONE

The problem and its background

1.1. Introduction

Information retrieval deals with the representation, storage, organization of and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he is interested (Ricardo, 2011). Accessing the desired information depends on efficiency of the retrieval system designed for a particular language. As a result, in this study we designed text retrieval system prototype for the Silt'e language. In this chapter all aspects of the study such as background of the study, the problem statement, objectives, methodology, scope, limitation and purpose of the study were discussed.

1.2. Background of the study

Silt'e is an Ethiopian language that is used as a medium of instruction (MOI) at elementary schools and given as a subject in junior and secondary high schools in the Silt'e Zone since 1995. The Zone administration decided to use the langue as a working language since 2014. It is the South- Ethio-Semetic language family and use Saba Ge'ez (Ethiopic script).

Information retrieval systems are designed to facilitate access to stored information. In addition, they are concerned with the representation, storage and organization of information items. The components of information retrieval system consists of a set of information items, a set of requests and a mechanism to determine which information items are most likely to meet the requirements of the requests. Basically, users of information state their information needs in the form of queries and submit their queries to the information retrieval system. Based on, the user queries the system output information item that is relevant to the user query. In other word, the items that have common entries in both the database and users query are returned to users. This happens when a matching exists between queries and information items. It is difficult to specify information that is needed using exact query

terms. The most critical language issue for retrieval effectiveness is the term mismatch problem. The indexers and the users do not often use the same words. To attain a match between these two components, the items in the documents as well as in the query should be represented in some form (Uma, 2014).

Usually, information retrieval systems do not work on the documents themselves, but on representations or abstraction. Therefore, deciding whether a document is relevant to a query depends on the kind of representation that is being used for the document and the query. Almost all existing information retrieval systems simply represent documents and queries as a 'bag-of-words' with its weight. When users give some query to the system the system tried to match those documents which have terms in the query and the documents will be automatically retrieved but not otherwise (Wordofa, 2013).

There are various methodologies and techniques tried so as to make effective and efficient way of retrieving documents from very large and unstructured corpus. Some of the scientific approaches to resolve the problems are ontology based IR, latent semantic indexing, query expansion and reformulation, statistical method for semantic indexing, and others (Ricardo, 2011).

Different models are used to develop information retrieval system. Among them Vector Space Model has been applied in many information retrieval systems. It is an algebraic model for representing text documents as vectors of identifiers. It is used in information filtering, information retrieval, indexing and relevancy rankings (vector_space_model, 2016). It has a number of advantages over the standard Boolean model. It is simple model based on linear algebra and allows ranking documents according to their possible relevance.

1.3. Statement of the problem

Digital unstructured Silt'e text documents in government organizations and schools increase from time to time in the Silte zone. The growth of digital text information makes the utilization and access of the right information difficult. Therefore, a tool needed to store, organize and helps to access Silt'e text digital text information.

Large numbers of Silte people leave their birth place for work and became urban or semi-urban inhabitants. This number increase from time to time, so accessing Silt'e information by

this portion of the population and others those who go to Silt'e zone for work from different parts of Ethiopia is very difficult, because there is no tool that provide Silte information easily.

The other very important issue is that urban resident of the population uses other languages for communication and information exchange. As a result they fail to recall their native language and culture, especially the young generations. This situation may result in the disappearance of the language and cultural aspects of the nation thorough a long time. To reduce this risk information regarding the zone's social, cultural, economic and political activities need to be preserved in digital format for generations to come and make easily accessible.

In the backdrop of the above discussion, we designed corpus based Silt'e language text information retrieval system prototype to overcome the stated problems or showcase how access to digital resources in Silt'e language can be initiated. Hence, the research find answers for the following research questions.

- What are the linguistic features of the Silt'e language?
- Which text operations and indexing procedures are suitable to organize the Silt'e corpus?
- Is vector space model techniques implemented in Apache Solr appropriate to design effective Silt'e text retrieval system?

1.4. Objectives of the study

1.4.1. General objectives

The main objective of this research is to examine the possibility of developing Silt'e text information retrieval system that can enable users to access the desired document that contains relevant information from the corpus based on vector space model.

1.1.2. Specific objectives

The following lists are specific objectives of this study.

- To investigate the semantic and syntactic structure of the Silt'e language
- To review literatures to get adequate understanding of the topic area

- Develop a prototype to demonstrate search of documents in Silt'e language given specified user queries.
- Evaluate performance of the developed prototype
- Recommend future work directions for further investigations

1.5. Scope and limitations of the study

The main focus of this study is to design a prototype to experiment an information retrieval system that effectively searches from Silt'e text collection. To provide relevance conclusion with enormous documents takes more time. Therefore, in this study we used sample document sets to give relevance conclusion of the prototype.

1.6. Significance of the study

Applying current information retrieval technologies for Silt'e language has several benefits. It enables Silt'e speakers retrieving text documents efficiently, effectively and it used as a foundation for other researchers who want work on this area. The researcher developed skill to solve critical problems found in the society by applying scientific procedures.

1.7. Research methodology

In this section we list research methods used to answer the research questions. The detailed discussions about the methodologies are presented in chapter three. It covers the data collection methods and research procedures.

1.7.1. The test collection

A digital text documents are one of the resources we used to test the prototype. For this purpose mainly school text books prepared in Silt'e language and different reports from Silte culture and tourism bureau were used. The collection consists of 99 documents in different topic.

1.7.2. Indexing method

In this study we preferred inverted index document representation technique, because it is the most popular data structure used in document retrieval systems. The indexing process passes four stages to create the indexed file. These steps are document collection, tokenization, linguistic preprocessing and indexing the documents.

1.7.3. Performance evaluation

Evaluating the degree of relevancy between the retrieved search result and the actual user need is important concept in information retrieval systems. We used relevance based evaluation measures to test the performance of the system prototype. Out of the relevance based evaluation measures for our research we choose the traditional precision-recall and F-measure approaches.

1.7.4. Prototype development tool

The development area is Windows environment and the package used is Apache Lucene Solr (ALS). Apache Solr is highly scalable search platform built using Apache Lucene. It is written in Java and runs as a standalone server (Dikshant, 2015). It is extensively used for indexing a large collection of documents and supporting full text search.

1.8. Thesis organization

We arranged this thesis work in five chapters. The first chapter presents the problem and its background. The main objectives and specific objectives, scope, limitation, significance and research methodology were discussed in this chapter. Chapter two is dedicated to describe literature review. The basics of the Silt'e language, overview of information retrieval system, information retrieval models, general process in information retrieval system and related works were discussed in this chapter. In chapter three applied research methodologies like collection organization, vector space model concepts, collection indexing, performance evaluation, Silt'e information retrieval system architecture, prototype development tool and Silt'e text processing were briefly discussed. Experiment and discussion were presented in chapter four. The last chapter covered conclusion and recommendations.

CHAPTER TWO

Literature Review

2.1. Introduction to Silt’e language

Information retrieval research process comprises many aspects. Among those aspects understanding the language basics are the most important one. As a result, in this section we discussed the Silt’e language basics clearly and precisely. To done this we referred Silt’e language school text books and researches conducted on the Silt’e language.

Silt'e is a Semitic language mainly spoken in Southern Nations, Nationalities and People’s Region in the Silte Zone. Some Silt’e speakers settled in other parts of the country, particularly in Adama, Addis Ababa and Jimma. The Silt’e language is given the International Standard for Language (ISO 639_3) code of STV (Semitic South Transverse). The secondary level (L2) literacy of the language reaches 17%. The literacy has been increasing, because it is used as a medium of instruction in primary schools from grade 1-4 and taken as a subject from grade 5-12.

2.1.1. Silt’e dialects

The Silt’e language consisting of four dialects, these are Azernet-Berbere, Silti, Wuriro and Ulbareg. These four differences are determined on topographical region. They are grouped in to four dialect families but, strong similarity exist among them (Lewis, 2016).

2.1.2. Silt’e writing system

The accurate period that the Ge’ez character set were used for Silt’e writing system is not known. However, since 1980s, Silt'e has been written in the Ge'ez alphabet, or Ethiopic script, writing system, originally developed for the now extinct Ge'ez language and most known today in its use for Amharic and Tigrigna (Silt'e_language, n.d). Portion of the Ge’ez (Ethiopic) alphabets and their phonetic order are shown in the following table.

1st order	2nd order	3rd order	4th order	5th order	6th order	7th order
ሀ ha	ሁ hu	ሂ hi	ሃ haa	ሄ he	ህ hi	ሆ ho
ለ le	ሉ lu	ሊ li	ላ la	ሌ le	ል li	ሎ lo
መ me	ሙ mu	ሚ mi	ማ ma	ሜ me	ም mi	ሞ mo
ረ re	ሩ ru	ሪ ri	ራ ra	ሬ re	ር ri	ሮ ro
ሰ se	ሱ su	ሲ si	ሳ sa	ሴ se	ስ si	ሶ so

Table 2. 1 Phonetic order of Ge'ez alphabets

2.1.2.1. Silt’e vowels

In the Ge’ez scripts only seven vowels are distinguished, but in written Silt’e these seven vowels are mapped onto ten Silt'e vowels, these are the five short vowels አ[a], ኦ[o], እ[i], ኡ[u], ኤ[e] and five long vowels ኣ[aa], ኦኦ[oo], ኢ[ii], ኡኡ[uu], ኤኤ[ee] (Kedir, 2012).The following practical example illustrates Silt’e language vowels clearly (Silt'e_language, n.d).

- ä → a: አለፉ alafa 'he passed'
- u → u, uu: ሙት mut 'death', muut 'thing'
- i →
  - ii: ኢን iin 'eye'
  - word-final i: ሚሪ mari 'friend'
  - i ending a noun stem: ሚሪከ marika 'his friend'
  - impersonal perfect verb i suffix: ባሊ baali 'people said'; በባሊም babaalim 'even if people said'
- a → aa: ጋራሽ gaaraaš 'your (f.) house'
- e → e, ee: ኤፌ eeffe 'he covered'
- ə →
  - i (except as above): እንግር ingir 'foot'
  - consonant not followed by a vowel: አስሮሽት asroošt 'twelve'
- o → o, oo: ቆጬ k'oč'e 'tortoise', k'ooč'e 'he cut'

2.1.2.2. Silt’e consonants

Silt’e language has twenty six consonants which are similar with other Ethiopian Semitic languages (Kedir, 2012). There are ejective consonants in conjunction with plain voiced and voiceless consonants and these can be reduplicated after a short vowel.

ሀ | h |	ለ | l |	መ | m |	ረ | r |	ሰ | s |	ሸ | s |	ቀ | q |	በ | b |	ተ | t |	ቸ | c |	ነ | n |	ኘ | n |	አ | a |
ከ | k |	ወ | w |	ዘ | z |	ዠ | z |	የ | y |	ደ | d |	ጀ | j |	ገ | g |	ጠ | c |	ጨ | c |	ጰ | p |	ፈ | f |	ፐ | p |

Table 2. 2 Silt’e consonants

2.1.3. Silt’e grammar

In linguistic, a grammar deals about the set of fundamental rules governing the arrangement of words in any specified natural language (Grammar, 2016). Like other natural languages, Silt’e has its own grammatical rules and syntax. Mostly used grammatical indicators in Silt’e language are gender, number, definiteness, personal pronouns, adjectives, adverbs, prepositions and negations. The detail descriptions of these indicators are presented as follow.

- **Gender**

In Silt’e language gender is the syntactical category with sex. It is determined in a natural manner; hence peoples and animals assign gender based on their sexual characteristics, but there are a few words available to express gender for inanimateness. For example gender for astronomical elements ወሪ(warii) ‘moon’ define the masculine gender and አይሪቱ(ayiriite) ‘sun’ define the feminine gender.

- **Number**

Silt’e language has formal means to express differences of quantity. Grammatically it uses the three numbering techniques to express number of nouns and pronouns; these are singular, plural and paucal. Among those only paucal is marked the rest are unmarked. As an example

ሲን[siin] refer single or large number of cups, but ሲንቸ[siinca] mention a small number of cups. Noun paucal in Silt'e formed by suffixation of -ቸ[ca]. The following table shows some noun paucal.

Noun	Suffixation	paucal	Meaning
ጣይ[tayii]	-ቸ[ca]	ጣይቸ tayiica]	'sheep'
እንዳች[indaac]	-ቸ[ca]	እንዳችቸ[indaaciica]	'women'
ከራብ[karaab]	-ቸ[ca]	ከራብቸ[karaabcaa]	'oxen'
ባሊቅ[baliiq]	-ቸ[ca]	ባሊቅቸ[baliiqcaa]	'Elderly men'

Table 2. 3 Paucal formation using suffixation

The other way of forming noun paucal is by reduplicating the last consonant of the noun. The following table shows this process.

Noun	Noun paucal with reduplication	meaning
ገና[ganaa]	ገኖኖ[gananoo]	'horses'
ደረሰ[darasaa]	ደረሳሶ[darasasoo]	'religious student'
ገረጃ[garajaa]	ገረጃጆ[garajajoo]	'young females'

Table 2. 4 Paucal formation using consonant reduplication

- **Definiteness**

Definiteness in Silt'e language is a semantic feature of nouns identifying entities that are recognizable in a certain context. There are two definite suffixes -ኢ[ii] for masculine and -ቴ[te] for feminine. Instead of definiteness function, they also play the numbering role. Look at, the grammatical expression ዌሴቴ[wesetee] this expression suggest about a single enset plant and ኡመቲኢ[uumatii] denote the large number of peoples in place of its definiteness.

- **Personal pronoun**

Personal pronouns of the Silt'e language are showed in the following table.

First person		Second person		Third person	
Singular	Plural	Singular	Plural	Singular	Plural
ኢሄ[ihee] 'I'	ኢኘ[inaa] 'We'	አሸ[aash](f) አተ[ataa](m) 'You'	አቱም[atuum] 'You'	ኡሀ[uuha](m) 'He' ኢሸ[iisha](f) 'She'	ኡሁን[uuhun] 'They'

Table 2. 5 Silt'e personal pronouns

Personal pronouns depicted in the table 2.5 also have gender markers. These are defined in the following table.

First person		Second person		Third person	
Singular	Plural	Singular	Plural	Singular	Plural
ኤ[ee] Example ማጤኤ 'My Brother'	**ነ[na]** Example ማጠነ 'Our Brother'	**አ|አሀ[aa|ahaa](m)** Example ማጣአ|ማጣአሀ 'Your Brother' **አሸ[ash](f)** Example ማጣሸ	**አሙ[ammu]** Example ማጣአሙ 'Your Brother'	**ከ[ka](m)** Example ማጠከ 'His Brother' **ሸ[sa](f)** ማጠሸ 'Her Brother'	**ኒሙ[niimu]** Example ማጠኒሙ 'Their Brother'

Table 2. 6 Silt'e gender marker for possessive markers

- **Adjective**

Adjective in Silt'e used to describe characteristics of nouns. By doing this it modifies the meaning of a noun, so it has important function for daily communication.

Sample adjectives: በሬደ [bareeda] ‘beautyfull’ በሬዶ [bareedoo] ‘so beautyfull’

ኢማን [iimaan] ‘faith’ ኢማንቶ [bareedoo] ‘so faithy’

ሸባል [siibal] ‘sung’ ሸባልቶ [bareedoo] ‘singer’

In Silt’e language two techniques are used to form adjectives, these are inflectional and derivational adjectives.

- Inflectional adjectives

Paucal inflectional adjectives are formed by suffixation of -ቸ[ca] and reduplication of the last consonant. The following table shows sample inflectional adjectives.

Nouns used to form adjective	Suffixation of -ቸ	Paucal adjective after suffixation	Meaning
ጣይ[tayii]	-ቸ[ca]	ጣይቸ [tayiica]	‘sheep’
እንዳች[indaac]	-ቸ[ca]	እንዳችቸ[indaaciica]	‘women’
ከራብ[karaab]	-ቸ[ca]	ከራብቸ[karaabcaa]	‘oxen’
ባሊቅ[baliiq]	-ቸ[ca]	ባሊቅቸ[baliiqcaa]	‘Elderly men’

Table 2. 7 Paucal adjective formed by suffixation

Sample paucal adjective formed by reduplication are given by the following table

Nouns used to form adjective	Paucal adjective after reduplication	meaning
ገኛ[ganaa]	ገኛኛ[gananoo]	‘horses’
ደረሰ[darasaa]	ደረሳሶ[darasasoo]	‘student’
ገረጃ[garajaa]	ገረጃጆ[garajajoo]	‘young females’

Table 2. 8 Paucal adjective formed by reduplication

- Derivational adjectives

Silt'e derivational adjectives are derived from nouns using **-አኘ[-anna], -አተኘ[-atanna], -ኣንቺአ[-aancio], -አ[-a], -ም[-m]** and **-ኢታም[-ataam].** The following table shows some denominal adjectives.

-አኘ[-anna]		-አተኘ[-atanna]		- ኣንቺአ[-aancio]		-አ[-a], -ም[-m]		-ኢታም[-ataam]	
Noun	Adjectives	Noun	Adjectives	Noun	Adjectives	Noun	Adjectives	Noun	Adjectives
ጉት	ጉተኘ	ከስብ	ከሰብተኛ	ብተር	ብተራንቾ	መኒጀ	መኒጃም	አይብ	አባታም

Table 2. 9 Adjectives derived from nouns

- **Adverb**

Adverb is part of communication in Silt'e language. It expresses manner, place, time, frequency and level of certainty. Some Silt'e adverbs are listed in the following table.

Adverb of frequency	Adverb of manner	Adverb of place	Adverb of time
ሁለም ነግ 'always'	ፈየኮ 'very'	ሂኔ 'here'	በዞፍ 'later'
ሀደደ ነግ 'sometimes '	ኮሞ 'fast'	ሀኔ 'there'	ሀውጄ 'today'
ሲረም 'never'	ሀዴኘ 'together'	ሁለምኤት 'everywhere'	ጌስ 'tomorrow'
አለፈ አለፋኔ 'occasionally'	ለብቾ 'alone'		ሴስተ 'after tomorrow'

Table 2. 10 Sample Silt'e adverbs

- **Nouns**

Silt'e nouns can end in any consonant or semivowels subject to the general phonological constraints of the language or one of the short vowels **አ**[a], **ኦ**[o], **እ**[i], **ኡ**[u], **ኤ**[e] (Kedir, 2012). It can be derived from verbs. Morphophonemic changes of nouns with prefix and suffix are given in the following table.

Prefix morphophonemic	Suffix morphophonemic
Cአ + ወ ⇨ Cአ Example: -ባ[baa] + ወሪ[waarii] ‘month’ ⇨ ቦሪ[boorii] ‘in a month’ Cአ + እ ⇨ Cኢ Example: -ቢ[bii] + እንግር[ingiir] ‘foot’ ⇨ ቢንግር[biingiir] ‘in a foot’	Final vowel of a word is lengthened:- 1. Before acc.suffix -ነ[-na]. Example: ደቺ[daci] ‘land’⇨ ደቺይነ[daciina] ‘our land’ 2. before pragmatic marker -ሞ[-m] and -ወ[-w] 3. before masculine definite article Example: ጨሎ[cuulo] ⇨ጨሎይ ‘cuulooy’

Table 2. 11 Noun morphology with prefix and suffix

- **Verb**

Silt’e verbs are two types. These are the original verb and derived verb. The derived verb obtained by inflection and derivation. The original verbs have from two to four radicals. Sample original verbs are shown in the following table.

Two radical	Three radical	Four radical
ሄደ[haade] ‘go’	ሀረሰ[haraasa] ‘plough ’	አዋለከ[awaalaka] ‘speak’

Table 2. 12 sample Silt’e verbs

The final letter of a verb can be of four kind consonant only, consonant followed by አ[a], non-palatal consonant followed by አ[a] and palatalized to ኤ[e].

2.2. Overview of information retrieval system

2.2.1. Introduction to information retrieval system

The need to access information from the stored collection begins in the early ages before computers were invented. During that time different researchers tried to develop mechanical and electromechanical systems which were used to accomplish the searching process. These mechanical systems continued to be developed and used until the advent of computers when this approach to information was surpassed (Mark, n.d).

With the growth of digitized unstructured information and, via high speed networks, rapid global access to enormous quantities of that information, the only viable solution to finding relevant items from these large text databases was search, and information retrieval systems became ubiquitous.

Information accessing process begins by formulating queries. The user who wants information about an object such as text document, video, image and etc. specify query and provide to the information retrieval system. The system takes the query and process according to predefined procedures. After processing the system return the relevant result to the user query.

Most information retrieval systems compute a numeric score on how well each object in the database matches the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query (Information_retrieval, 2016).

2.2.2. Development of information retrieval system

The significance of archiving and searching for information has been realized in thousands years ago. The concept of applying computers to pursuit significant pieces of information from the collection was disseminated in the article ‘As We May Think’ by Vannevar Bush in 1945. With the advent of computers, it became possible to store large amounts of information and finding useful information from such collection became a necessity. The field of information retrieval was born in the 1950s out of this necessity (Amit, 2001).

Application of computers to information retrieval started at the conference held by the UK’s Royal Society in 1948. In that conference Holmstrom described the machine called Univac and he stated that, the machine could process at the rate of 120 words per minute (Mark, n.d).

It was the first allusion that computers were involved in content searching. This motivates many researchers to design different projects that improve the searching process using computers. Among those, Mitchell described a project to model the use of a Univac computer to search 1,000,000 records indexed by up to six subject codes; it was estimated that it would take 15 hours to search that many records. The development of document indexing and retrieving the indexed document used to consider information retrieval as a research discipline. Considering computers as ultimate tool for search trigger the growth of a commercial search sector and the consolidation of information retrieval as an increasingly important research area.

Large Information retrieval research group was formed by Gerard Salton in 1960s at Cornell. The group produced numerous technical reports, establishing ideas and concepts that are still major areas of investigation today. The formalization of algorithms to rank documents relative to a query, documents and queries viewed as vectors within an N dimensional space, the introduction of relevance feedback, the clustering of documents with similar content; the statistical association of terms with similar semantic meaning, increasing the number of documents matched with a query by expanding the query with lexical variations were an IR innovations and enhancements contributed by these group (Mark, n.d).

The idea of term frequency – inverse document frequency (tf-idf) was introduced and adopted in 1970s. A number of works done to formalize the retrieval process such as G. Salton synthesized the outputs of his group's work on vectors to produce the vector space model. The manipulation of documents and queries as vectors in a large dimensional space is still common in the information retrieval process.

Achievements of information retrieval between 1980s- mid 1990s were advancement on the basic vector space model in to the most well-known Latent Semantic Indexing (LSI). Here the dimensionality of the vector space of a document was reduced and the Queries were mapped into the reduced space. Computational linguistic approaches like the syntax of words, their semantics; addressing anaphora, ambiguity, and named entities were considered in this period. Stemming algorithms also introduced to match words in to their lexical variant. As a result many information retrieval systems in different linguistic structure were tried to develop.

2.2.3. Information Retrieval Models

Satisfying the users to access information they need by building information retrieval system is one of the problem solving task in information retrieval field. As any science fields, information retrieval also has its own models (Akram, 2015). A model is a framework which helps to understand how some processes are accomplished. Information retrieval models explain the way how efficient retrieval systems are developed. The main classification of the information retrieval models are the Boolean Model, Vector Space Model (VSM), Probabilistic Model (PM), and the Linguistic and Knowledge-based Models (Djoerd, 2009).

2.2.3.1. Boolean model

The Boolean model is the first information retrieval model based on Boolean logic and classical set theory (Standard_Boolean_model, 2016). In this model the basic logical operators used are AND (logical product), OR (logical sum), NOT (logical difference). Query terms are combined by these operators to access the matching sets of documents.

For example, consider a query Science AND Technology. This query contain two index terms Science and Technology. The result will return the set of documents that contain the terms Science and Technology. Here the result set consist only the documents which contain both terms in the index. Forming query using the OR operator i.e. Science OR Technology will respond with larger set of documents, because documents with either of terms or all terms will be retrieved.

The main interesting part of the Boolean model is that it gives users a sense of control over the system (Djoerd, 2009). Even though the Boolean model was the first model and preferred by the experts to control the system, it is the most criticized model in terms of different information retrieval metrics. Among the disadvantages the most one is that it does not provide the ranking of retrieved documents. It also not retrieves anything if no match exists between the query and indexed documents.

2.2.3.2. Vector Space Model

Statistical approach to retrieve information was proposed by Peter Luhn. According to this suggestion to search a document collection, the document should be represented as alike to the document which the user needed it. By doing this similarity between the representation of

the prepared document and the representations of the documents in the collection is used to rank the search results (Vaibhav, 2015).

The vector space model was developed based on the statistical approach. In this model the index representation of a text document and a query are used as a vector to filter information and retrieve it (vector_space_model, 2016). It is representation of index terms and query as vectors embedded in a high dimensional Euclidean space. In the Euclidean space each dimension corresponds to a separate term. The following two dimensional Euclidean spaces visualize the documents (d_1 and d_2) and query (q) vectors.

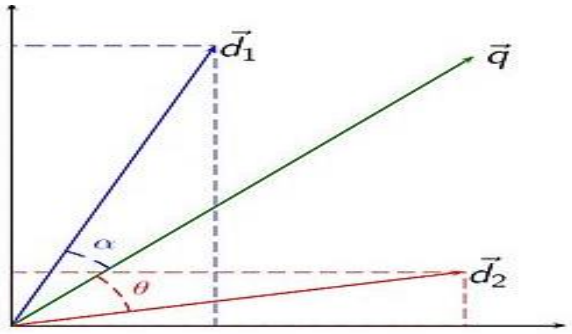

Figure 2. 1 Document and query representation on Euclidean space (sorce wikpedia)

The mathematical expressions of the document and query vectors are given by the following equation.

$$D = \{D_{t1}, D_{t2}, Dt_3, \dots, D_{tn}\}$$
$$Q = \{Q_{t1}, Q_{t2}, Q_{t3}, \dots, Q_{tn}\}$$

……Equation 2.1

Where

D = is a document

Q = is the query

D_{ti} = term i in the document

Q_{ti} = term i in the query

The vector space model steps are categorized into three basic phases. The first phase is indexing a document where content relevant terms are extracted from the document. The second phase involves the weighting of indexed terms to improve retrieval of document appropriate to the user query. In the third phase similarity measure is performed to rank the document with the user query (Introduction_to_vector_space_model, n.d).

2.2.3.3. Probabilistic Model

The other very important information retrieval model is the probabilistic information retrieval model. As we know in the information retrieval process the user translate their information need in to query representation and documents are converted in document representation. Based on these two representations the information retrieval system tries to decide which document fulfills the information need. In the above two information retrieval models, matching is done formally based on defined index terms. But a system encounters the problem of guessing weather a document contain the relevant content which is needed by the user. The probability theory makes available principled grounds for such reasoning under indecision (Christopher, 2008).

In this model documents in a collection are ranked based on their probability of relevance to the query. To accomplish the ranking task there are different techniques are available. Among theses the most familiar is Bayesian inference network (Gezehagn, 2012). It uses the Probabilistic Ranking Principle (PRP). By this principle it tries to estimate the probability that a document will be relevant to a user query. As stated by (Dikshant, 2015), depend on a query each document can be categorized as relevant or non-relevant P(R|D) or P(NR|D).

The ranking of documents in the probabilistic model can be achieved by the following expression

Let

- R be the set of relevant document to a query q.
- R' be the set of non-relevant documents to query q.
- $P(R|d_i)$ be the probability that d_i is relevant to the query q.
- $P(R'|d_i)$ be the probability that d_i is non-relevant to the query q.

The similarity $sim(d_i, q)$ can be defined as

$$\mathrm{Sim}(di, q)\frac{P(R|di)}{P(R'|di)} \text{ …….. Equation 2.2}$$

2.3. General process in information retrieval system

Information retrieval system passes several processing stages to produce the required search result. These processing stages categorized into two interrelated groups. The first one is indexing documents and the second one is retrieving the relevant documents (Vikram, 2014). These basic processing components are presented here briefly.

2.3.1. Indexing

Indexing is the process of extracting terms from the document in a collection and stores it in specified file format. The main aim of an indexing is to speed up the searching operation. The major steps it involves are preparing documents to index, tokenize, remove stopwords, stemming, term weighting and store indexed file (Christopher, 2008).

2.3.1.1. Preparing documents

Well organized documents are an important element to indexing. It is an enormous set of text in a given language. In information retrieval research, documents are properly prepared and stored to do statistical analysis.

2.3.1.2. Tokenization

Splitting up sequence of text characters into individual tokens is called tokenization. Consider the statement: ‘አመተው ለንቅነ ባደነ ላሊቅነ’ the tokens after splitting the sentence are ‘አመተው’, ‘ለንቅነ’, ‘ባደነ’ and ‘ላሊቅነ’. Therefor the purpose of tokenization is to breakdown a document collection in to tokens so that it is easy to match with tokens in the query. As a result it increases the relevancy of a document to a query.

The challenge of tokenization is that it is a language specific process (Christopher, 2008). Handling non-separating whitespace like co-worker, phone numbers and date are also challenged to tokenize.

2.3.1.3. Stopword removal

Stop word are the most common words found in a language. Majority of text document accounts large portion of stop words (Tsz-Wai, 2005). For instance 20-30% of English text document scores of stop word terms (Francis, 1982).

Information expressed by stopwords is insignificant, so document indexing judgment power of these tokens in information retrieval process is very low. As a result neglecting all stopwords tokens when indexing a document and processing the queries make the searching more efficient.

2.3.1.4. Stemming

All natural languages perform morphological change on some words to express an action perfectly. For example: Haile run quickly. The word **quickly** derived from **quick** by adding suffix –**ly**. The inflected words are unproductive for text information retrieval, so the mechanism used to bring words into their root is stemming.

Stemming is the process of dropping inflected words to their root word. It is performed by stripping down word suffixes in order to return the word to its normal form (Aslib, 2000). In general it transforms alteration of words into their most basic pattern. Many researchers prove that stemming enhance processing capacity of information retrieval system.

Various stemming algorithms have been studied since 1960s (Stemming, 2016). Lovins stemmer published in 1968 and Porter stemmer published in 1980 are among those algorithms. Porter's work proposed suffix removing algorithm and widely used algorithm especially in English language (Juhi, 2013) , but the rules are simple so it cannot completely identify English morphology (Wahiba, 2013). Even if different stemming algorithms have been developed they cannot work for all languages, because stemming is natural language dependent process. Since all languages have different inflectional and derivational of morphology of words. Therefore designing stemmers based on each language's characteristic is important.

2.3.1.5. Term weighting

Term weighting is the most important step in text retrieval process. It is numerical statistics that detects how much a word is essential to identify a document from a collection. To accomplish this documents and query are represented by vectors of weighted terms. Text indexing systems based on appropriate term weighting technique returns a good result (Salton, 1988). Information retrieval system tries to return relevant documents to a user query. To achieve this applying appropriate term weighting scheme to increase the performance of information system plays a great role (Cummins, n.d).

There are various term weighting methods are exist such as Binary Weights(BW), Raw Term Frequency(RTF) and term frequency(tf)-inverse document frequency(idf). Term frequency (tf)-inverse document frequency (idf) is a well-known method to evaluate the importance of a word in a document and a means to convert textual document in to a vector space model (Jitendra, 2012). It is simple and most applicable in many information retrieval systems. The main things that should be identified for term weighting calculation are term frequency (tf) and invers document frequency (idf).

Term frequency (tf) is the number of times a term occurs in a document. The term weight that takes place in a document is directly related to its frequency in the document (Luhn, 1957). It is a document level measure.

Inverse document frequency (idf) is used to identify whether a term is common across the corpus or not. A term that occurs in a collection rarely is worthy for discriminating among documents. This achieved by dividing number of the total document to number of documents that contain a term and taking logarithm of the quotient.

Term frequency defines local weight and inverse document frequency defines global weight (khemani, n.d). The combination of these two is used to determine the weight of a term which is used to index a collection. The tf-idf is expressed by the following equation.

$$w_{ij} = tf_{ij} \times idft \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation 2.5}$$

Where, w_{ij} is weight of a term i in document j.

The final stage in information retrieval process is indexing a collection based on the result obtained in the term weighting stage. There are different ways of allocating weight to a term but we prefer the tf-idf approach. Even though information retrieval process has the discussed steps, we used the following architecture to process Silt'e text. In the architecture rounded rectangle represents a process and flowchart magnetic disk represents a collection.

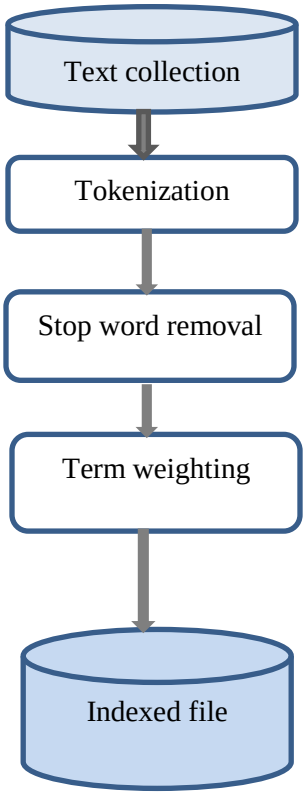

Figure 2. 2 Silt'e text processing steps

2.4.Related Works

The advancement and availability of computing technologies in the 21st century enables to store huge amount of information and data coming from different sources in digital form. As a result, it becomes more and more difficult for target users to select contents they desire. This has an undesirable effect for searching the relevant documents from the entire collection.

In Ethiopia there are more than 86 languages. These languages are grouped in different language families and each family has its own scripting and grammar style. A more documents are available in organizations, schools and on the Internet that are prepared using local languages. Supporting the target users to access and organize the enormous and widespread amount of information is becoming a primary issue. Information retrieval systems ability to retrieve relevant documents became critical due to the existence scripting difference exists in local languages.

It is difficult to specify information that is needed using exact query terms. The most critical language issue for retrieval effectiveness is the term mismatch problem. The indexers and the users do not often use the same words (Uma, 2014).

There are different issues to be considered in designing information retrieval systems such as the test collection, collection indexing approach, retrieval techniques and performance evaluation. Performance evaluation is crucial at many stages in information retrieval system development. At the end of development process it is significant to show that the final retrieval system achieves an acceptable level of performance and that it represents a significant improvement over existing retrieval systems (Keneilwe T. , 2012).

Understanding the role of information retrieval technology in the current information age, different researchers work on many aspects of information retrieval systems for different Ethiopian languages that can solve the information accessing problems on those languages and make the retrieving process more efficient.

Gezehagn works on Afaan Oromo language in the objective of designing information retrieval system that can enable to search for relevant Afaan Oromo text corpus using Vector Space Model. The methods used in his study were literature review, corpus preparation and python 2.7.3 programming language to design the prototype. According to the experimentation he made the system registered 0.575 precision and 0.6264 recall which is promising to design Afaan Oromo information retrieval system that searches within large corpus (Gezehagn, 2012)

Amharic Text Retrieval: An Experiment Using Latent Semantic Indexing (LSI) With Singular Value Decomposition (SVD) research done by (Tewodros, 2012), to examine the potential of Latent Semantic Indexing technique in the retrieval of Amharic text documents and compare its performance with that of exact term matching technique, the standard vector space. Finally he conclude that except at one recall level, the recall-precision graph of the Latent Semantic Indexing (LSI) method was above the standard vector method. The biggest differences between the two approaches were observed at the last two high recall levels (0.90 and 1.00)

The other research work on Amharic text retrieval was done by (Wordofa, 2013). He investigated semantic indexing and document clustering techniques to improve the performance of Amharic information retrieval. His experiment show semantic indexing has improved the performance of Amharic information retrieval system from 60% to 66% F-

measure. Moreover, semantic indexing and document clustering reduced computational cost of the system by 1/k* TermsReductionRatio factor, where K is the number of the cluster.

Probabilistic Information Retrieval System for Amharic Language developed by (Hirpa, 2012), to harness the vagueness problem in query expression observed in previous works using vector space model of Amharic Information Retrieval system. His system has registered F-measure of 0.60.

As we have discussed in the above literatures a number of works tried to design information retrieval system using statistical approach for local languages. Alongside designing information retrieval systems for those languages, cross-lingual information retrieval systems were also developed.

A corpus based approach for Afaan Oromo-English Cross-Lingual Information Retrieval (CLIR) studied by (Bekele, 2011). The result he obtained was a maximum average precision of 0.468 and 0.316 for Afaan Oromo and English respectively. Even though the corpus-based approach is influenced by size and quality of the corpus, the result obtained was encouraging to develop Afaan Oromo-English CLIR by using this approach.

To design text retrieval system for a particular language the study of automatic indexing techniques play important role for that language. Stemming algorithms are among the techniques in automatic indexing. For Silt'e language (Kedir, 2012) tried to design stemming algorithm. To evaluate the stemmer developed in this study, test data of size 1486 words were selected randomly from the sample texts. The experiment showed that the stemmer performs at accuracy of 85.71 % and reduce dictionary size by 34.99% for stems.

Developing information retrieval system has a great impact on the accessing of information in a particular language. We have seen in the literatures there are different information retrieval systems designing approaches and each model has its own strength and limitation. To enhance the correctness and efficiency of the retrieved information from the corpus we have to select appropriate information retrieval system development approaches.

CHAPTER THREE

Detailed Discussion on Research Methodologies

Information retrieval system development process applies a number of interrelated strategies. For our research the strategies we follow are compiling well organized Silt'e collection, indexing the collection by applying vector space model concepts and search sample information to test the effectiveness of the prototype.

3.1. Collection organization

The first thing we have to do in information retrieval system research is collecting documents which are used for indexing. These are all available text documents on different topics. We collected Silt'e text documents for our research. School text books which were collected from Silte Zone Education Bureau takes the highest portion of the corpus because other types of documents like news article, reports were not accessed enough during our data collection time. The collection consists of 99 documents in different topics.

3.2. Vector space model techniques

Vector space model (VSM) is well applied model in many information retrieval system research areas. For our research we choose this model. In this section we discussed the basic concepts of the model.

In vector space model a collection is represented as vectors in a common vector space (Christopher, 2008). Terms and documents are indexed in appropriate way in a vector space .This representation of the collection enables to compute information retrieval operations like term frequency, inverse document frequency, term weighting and document to query scoring.

Fundamental steps in vector space model are content related terms are mined from the document to index, weighting of indexed terms to better retrieval of a document fitting to the user query and computing similarity among documents to rank the document with respect to the user query (Jitendra, 2012). To accomplish these three basic steps some statistical measures are performed. The important statistical measures are explained as follow.

3.2.1. Term frequency (tf)

The Term frequency (tf) deals about the number of occurrence of a term in a document. This specifies the weight of a term in that document. It is one of the requirements needed to compute weight of a term in a document.

$$tf_{ij} = \frac{f_{ij}}{maxftd} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \text{Equation 3.1}$$

Where tf_{ij} is frequency of term i in a document j, maxftd is the frequency of most common term occur in a document.

Term frequency mechanism to determine weight of a term considers that all terms have equal importance. But in the reality all terms do not have equal significance to discriminate among documents in the collection, because some terms occur frequently in all documents. So, alternative technique to reduce the effect of more redundant term is inverse document frequency.

3.2.2. Inverse document frequency (idf)

Inverse document frequency (idf) defined to be the number of documents that contain a term in the corpus. The intention here is that the corpus amount and actual amount of documents which have the term significantly varied. It is determined as

$$idft = \log\left(\frac{N}{dft}\right) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \text{Equation 3.2}$$

where idft is inverse document frequency of term t, N total number of a corpus, dft number of documents that contain a term t in the corpus. We get higher value of **idft** for rare terms and lower value for more common terms.

3.2.3. Term weighting

Next task is combining inverse document frequency and term frequency tf * idft to determine the discriminating power of a term in each document and a query. This computation is called term weighting (tf-idf weighting) and determined as

$$w_{ij} = tf_{ij} \times idft \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \text{Equation 3 .3}$$

Where, w_{ij} is weight of a term i in document j. The value of w_{ij} is high when a term i present many times in a document.

3.2.4. Measuring document similarity

During accessing text information the user provide a query to the system and the system determine similarity between the query and document collection to return the required result. To measure similarity among documents and the query different approaches were applied.

Cosine similarity measure is a common similarity measurement technique. It define angle between the document vector and the query vector (Jitendra, 2012). The angle between two vectors is considered as a measure of divergence between the vectors. The cosine angle is used to calculate the numeric similarity among document with document and the query.

The criteria to quantify similarity between two documents d1 and d2 are to compute the cosine similarity of their vector representations (Christopher, 2008). This is accomplished by applying the following normalized cosine similarity function.

$$\mathrm{sim}(Q, D_i) = \frac{\sum_i w_{Q,j}\ w_{i,j}}{\sqrt{\sum_j w^2 Q,j\ \times \sum_i w^2\ i,j}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation 3.4}$$

Where $\mathrm{sim}(Q, Di)$ is similarity between query and document i, $w_{Q,j}$ is weight of a term in the query, $w_{i,j}$ is weight of a term in a document j.

Based on the similarity result documents are ranked that are more related to the query.

In similarity measure the basic facts we observe are (Gezehagn, 2012).

- If d1 is near to d2, then d2 is near to d1.
- If d1 is near to d2 and d2 is near to d3 and then d1 is not far from d3
- No document is closer to d than d itself.
- Distance between d1 and d2 is the length of the vector |d1 – d2|.

3.3. Document indexing

Information retrieval deals about accessing a document by matching a user query against a set of free-text record. Text retrieval is a branch of information retrieval where the information is stored primarily in the form of text (Majeed, 2014). In our research the free-text documents used are unstructured Silt'e text documents.

The success of text retrieval depends on having suitable data structure in which to store a text. A number of techniques have been proposed such as clustering and signature files to index a document (Salton, 1988), but these were not fit enough to store a large document collection. The method that has proven most successful in IR systems uses a data structure called an inverted index.

3.3.1. Inverted indexing

An inverted index is an index data structure. It also known as posting file or inverted file (Zheng, n.d). It is the most popular data structure used in document retrieval. In this indexing mechanism set of terms (dictionary) are mapped into documents that contain those terms. The set of terms is called a vocabulary. Each term in the inverted index has a pointer that points to the posting list which is a list of all the occurrences of a term (Majeed, 2014).

The basic idea for building an inverted index is to form a dictionary of the unique terms in the collection. For each term in the collection, asseverate a list of documents by identifying a document id in which the term occurs in the specified document. In this research the Silt'e text document passes several text preprocessing operations before it stored as inverted index. The following figure shows sample inverted index structure for two Silt'e documents.

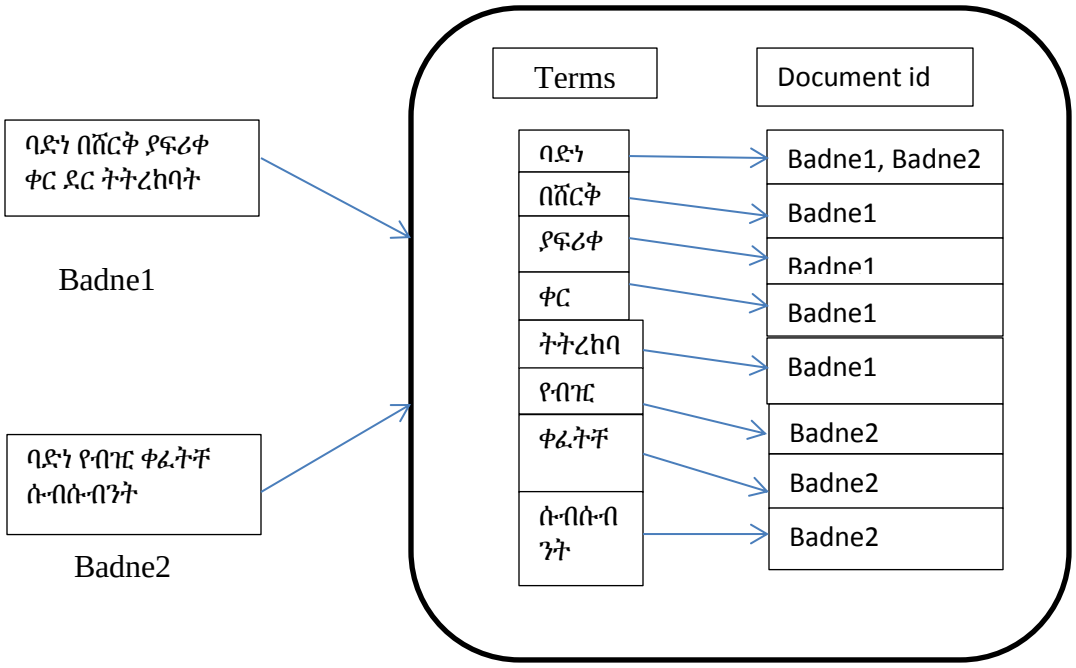

Figure 3. 1 Inverted index of two Silt'e documents

As shown in the figure the term ባድነ appear in both documents Badne1 and Badne2. The terms በሸርቅ, ያፍሪቀ, ቀር and ትትረከባት appear in Badne1 and the term የብዣ , ቀፈትቸ and ሱብሱብንት appear in Badne2. The stopword ደር is removed.

3.4. Performance evaluation

In the case of information retrieval the job is to return a set of relevant documents given for a query. So evaluating performance of an information retrieval system is very important to test the effectiveness of the system for the search result. In this process we focus on measuring system effectiveness, system ability to discriminate between relevant and non-relevant documents with respect to the query.

There are different information retrieval effectiveness measurement mechanisms. Among them precision and recall are well known methods to measure information retrieval effectiveness.

3.4.1. Recall

While searching a user formulates a search query for the needed information and provide to the information retrieval system. Based on a query the system returns relevant documents out of actual documents that are relevant to the query. More specifically recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents.

$$\text{Recall} = \frac{\text{Number of relevant documnets retrieved}}{\text{relevant documents}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation 3.5}$$

3.4.2. Precision

In case of precision we filter more relevant documents from those documents which are returned by considering relevancy to the search item. It can be defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search.

$$\text{Precision} = \frac{\text{Number of relevant documents retrieved}}{\text{Numberof retrieved documents}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation 3.6}$$

Interest to use precision and recall to test information retrieval system effectiveness vary from person to person. Because one person prefers high precision and another person prefers high recall to measure effectiveness based on their domain of interest. These evaluation

measures are inter-dependent measures in that as the number of retrieved items increases the precision usually decreases while recall increases. We see that there is trade-off between these two measures. To compromise this problem a single measure approach was defined. This approach is called F measure which is the weighted harmonic mean of precision and recall (Sasaki, 2007).

$$F = \frac{2PR}{P + R} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation 3.7}$$

Where F is F-measure, P is precision and R is recall.

Precision, recall and F-measure are set oriented measures that means used to measure unordered set of documents (Keneilwe & Tranos, 2012). Most commonly used means to rank the likelihood document relevancy to the information need is statistical or probabilistic approach. The top *k* ranked documents are used to evaluate effectiveness of information retrieval system that rank documents to return ranked set of documents. A number of techniques available for evaluation of ranked retrieval result. Among them we used average precision and mean average precision for our research.

3.4.3. Average precision

Average Precision (AP) is the average of the precision values from the rank position where relevant documents are retrieved, that means the rank position where the recall value increased. It is a single value measure based on the ranking of all the relevant documents and the obtained value highly depends on the ranked relevant documents.

3.4.4. Mean average precision

Mean Average Precision (MAP) is computed as the average of the average precision on each query. It considers all queries are equal. That means taking the mean of the precision at each recall point for all queries together.

In this case interpolated mean is used and precision-recall graph can be plotted based on the obtained mean.

$$\hat{P}(r) = \text{Max}_{r \leq \acute{r}} P(\acute{r}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation 3.8}$$

Where $\hat{P}$ precision at certain recall level, r maximum precision and $\acute{r}$ recall level (Keneilwe & Tranos, 2012)

3.5. Prototype development tool overview

In today's information age, the amount of electronic information becomes huge. It is difficult to find out relevant information from this tremendous storage. The solution for the challenge is developing information retrieval systems. We use Apache Solr version 6.0.0 to develop Silt'e information retrieval system prototype. Here we discussed basic overview of Apache solr.

3.5.1. Apache Solr

Apache Solr is highly scalable search platform built using Apache Lucene. It is written in Java and runs as a stand-alone server (Dikshant, 2015). Simple commands like start and stop are used to start and stop it. It is extensively used for indexing a large collection of documents and supporting full-text search.

3.5.1.1. Attraction of Apache solr

Solr attract peoples to use for their search system because it is the most widely used search solutions in different domains. Some attractiveness features of Solr are:-

- Highly scalable and fault-tolerant: - we can add or remove computing capacity to Solr.
- Enterprise ready:- many organizations in different domain trusted on Solr for their search requirement
- Full-text search: - it provides all the matching capabilities needed and autocomplete.
- Flexible and extensible: - we can modify and customize components by extending the Java classes and adding the configuration for the created class in the appropriate file.
- Easy configuration:- as Solr is configuration based platform it is easy to configure for our work

3.5.1.2. Characteristics of Apache Solr

Solr have a number of features that makes it popular (Dikshant, 2015). These characteristics are strongly aligned with our research strategies. That is why we use Solr for our prototype development and test the system. The basic features are:-

- Solr builds an inverted index of the document that we add to it and at query time it searches the index for matching documents.

- Solr was developed using the vector space model (VSM) to rank and determine the relevance of a document with respect to a user query in order to retrieve the required information.
- In Solr the search query and the documents being indexed go through a chain of analyzers and tokenizers.

3.5.1.3. Schema design in solr

Schema design in solr is a description of document structure that is precondition for indexing and searching, because a search result in solr is a document. It is a basic condition to develop information accessing systems in solr. After structuring the data a series of text analysis process are applied to control the matching actions. The Extensible Markup Language (XML) format is used to define the schema and field type outline. The following XML definition describes a general layout of a given schema design structure in solr.

```
<schema name="SilteTirs" version="1.6">

    Fields are defined here…

</schema>
```

The name and version are attributes, where name specify name of the schema and version define version number for the schema syntax and semantics. Current schema syntax and semantic version for solr-6.0.0 that we use in our research is 1.6. Schema definition for Silt'e text information retrieval system prototype is presented in the appendix.

A free text information document has different information about the document. For example consider a thesis work document as a free text document that provides different information about the thesis such as thesis title, institution where the thesis is done, author and date. All these information are called field in solr and have attributes like name, type, indexed, stored and others. Sample field definition given as follow.

```
<field name=" thesis_title" type="string" indexed="true" stored="true"/>

<field name=" author" type="string" indexed="true" stored="true"/>

<field name=" institution ="string" indexed="true" stored="true"/>
```

<field name=" date " type="Date" indexed="true" stored="true"/>

The other very important field definition in solr is filedType. The fieldType determines the value that the filed can hold and type of text analysis it goes through. Sample filedType definition is defined as follows.

<fieldType name="Silt'e_text" class="solr.TextField" >

Where Silt'e_text is filedType name, solr.TextField is text analysis process applied on Silt'e_text.

3.5.1.4. Text analysis in solr

The function of apache solr is indexing a document and making it searchable when user needs information. To accomplish these functions, it passes a consecutive text analysis phases to convert a sequence of text into terms.

In solr there are a number of analyzer definitions to describe a chain of processes. In every process step, specific text processing operation is performed. The final output of the analyzing process is a term which used for indexing and search. Sample Silt'e text analysis definition is given bellow.

```
<fieldType name="Silt'e_text" class="solr.TextField" positionIncrementGap="100">
<analyzer>
 <tokenizer class="solr.StandardTokenizerFactory"/>
 <filter class="solr.StopFilterFactory" words="lang/stopwords_st.txt" />
</analyzer>
</fieldType>
```

In the definition StandardTokenizerFactory splits the text stream and StopFilterFactory removes if a token is listed in Stopword file list. All these analyzers are java classes defined for text processing operations in solr.

3.5.1.5. Indexing and querying in solr

Indexing and querying are the most important process in information retrieval area. While adding document to solr it passes through index time text processing operations, if we define analyzer in the filedType element with the analyzer type index. The analyzed term indexed

with some information like position and frequency. Sample index time text analysis definition is given as follow.

```
<analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" />
  </analyzer>
```

Like index time text processing, query time text analysis also defined in solr. If we do not define query time text analysis that match with index time text analysis definition, then there is a term mismatch problem. For example we include LowerCaseFilterFactory java class in index time text analysis the same class should include in the query time text analysis. Assume we provide the term "ASTU" for the query if LowerCaseFilterFactory is not included in the query time analysis; we get a result that term is not indexed, because we define LowerCaseFilterFactory in index time text analysis chain. Sample query time analyzer definition given below.

```
<analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
```

3.5.1.6. Collection specific tools in solr

Solr provides collection specific administration tools like overview, analysis, dataimport, plugins, schema, query and others with interactive interface. When we select a particular core that contain the document collection from core sector dropdown menu all available tools are displayed. Among them we described overview, analysis and schema tools because we used them for our testing and analysis purpose.

- Overview tool

The overview tool provide important statistics including the count of documents in the index, total size of the index, last indexed time and core directory-related information. The following screenshot show statistics and instance information of silte_tirs core.

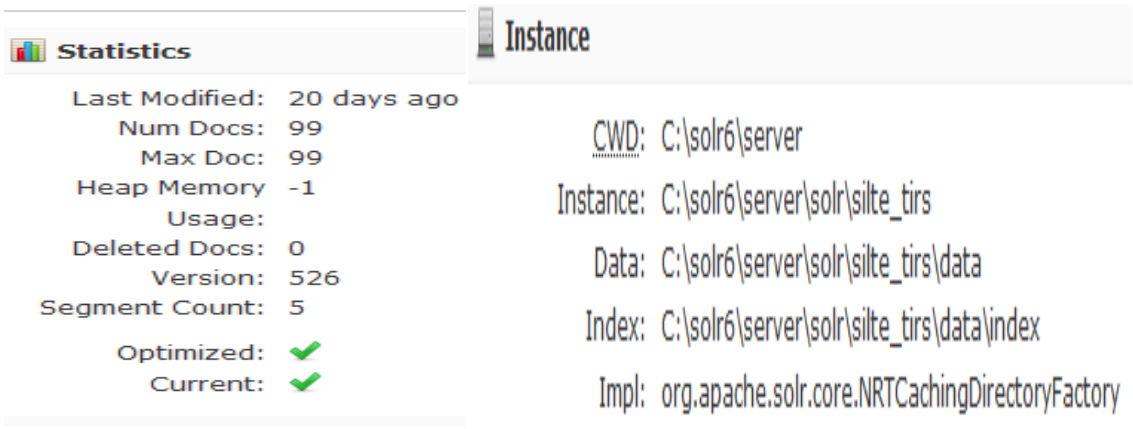

Figure 3. 2 The Silte_tirs core statistics and instance information

- Analysis tool

Apache solr provide collection analysis tool to analyze general characteristics of the document definition in the schema. This tool used to analyze the way how content would be handled during indexing or query processing and view the results separately or at the same time (Cassandra, 2015). Sample screenshot of this tool is given below to analyze "የስልጤ አመሰብ" as a text field value.

Figure 3. 3 Sample analysis tool result

In the above screenshot the analysis tool take “የስልጤ አመሰብ” as a field value that is defined as a text FiledType with the name silt’e_text in the Silt’e text information retrieval schema structure. Text preprocessor StandardTokenizer (ST) applied to tokenize “የስልጤ አመሰብ” into የስልጤ and አመሰብ. The tokenization process starts at position 0 for the token የስልጤ and ends at position 3. Tokenization position for አመሰብ starts at position 5 because position at 4 is white space and ignored. The raw_bytes indicate the equivalent representation of the tokens during indexing.

- Schema browser window

This collection specific tool provides information about data that is defined in the schema definition. It helps us to look at the fields and fieldTypes defined during schema design and we get some information about each field and fieldType. The following sample screenshot shows some information about the field “content_type”.

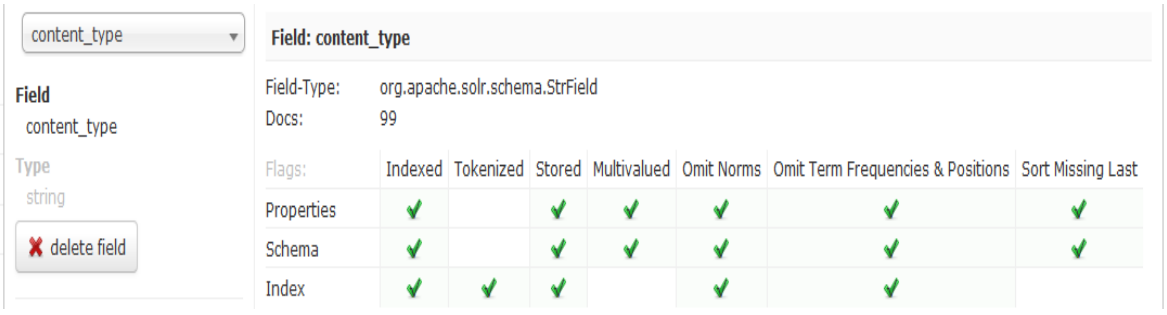

Figure 3. 4 Information about the field content_type

3.5.2. Text extracting tools

Different file formats are used to store documents. File formats like text documents (such as power point, word), PDF, images and multimedia files are sample file formats out of the existing various formats.

We used text documents and PDF file format to store the corpus in our study. Apache solr is used as prototype development tool, therefore some additional tools are needed to extract and index these file formats. The tools integrated with solr are Apache PDFBox and Apache POI file format parsers from Apache Tika framework that is a library used for document type detection and content extraction.

- **PDFBox**: - is open source java library used to perform different operations on PDF files such as verifying and extracting. It contains different components for example FontBox which handle font information.
- **Apache POI** (Poor Obfuscation Implementation): - is a popular library that used to perform operations relating with Microsoft office documents.

CHAPTER FOUR

Processing Silt'e Text and Experimentation

To develop the prototype and process Silt'e text we use window 7 ultimate with Intel(R) core(TM) 2 Duo CPU which have processor speed of 2.26 GHz and 2 GB RAM. Available Java Runtime Environment (JRE) in the system is 1.8.0_73 that is required to run apache lucene solr.

4.1. Silt'e information retrieval system architecture

In most information retrieval systems the system architecture consists of two main parts. These are indexing part and searching part. In this research we design Silt'e information retrieval system architecture that has the two parts.

The indexing part consists of five consecutive process components. These are corpus preparation process, tokenization, stopword removing, term weighting and indexing the corpus.

The other system architecture part is searching. As the indexing part pass a series of steps searching part also pass sequential process phases. Here user provides a query to the system. The system takes the query and performs pre-processing operations like tokenization, stopword removal and calculating weight of each term. Once all these tasks were completed similarity measure between query and documents performed to return a search result. The technique applied here is cosine similarity. Finally the ranked documents are returned based on relevancy to the query.

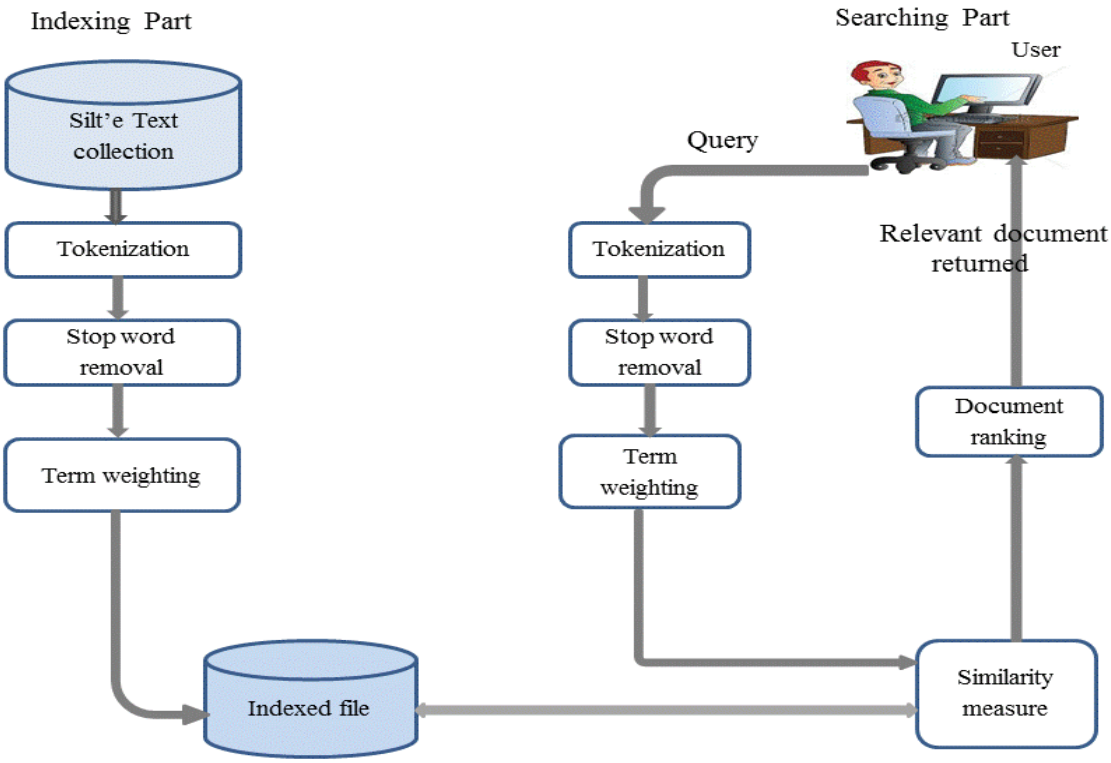

Figure 4. 1 Silt'e information retrieval system architecture

4.2. Tokenizing Silt'e text

To break down Silt'e text stream into tokens we used StandardTokenizerFactory that is implemented in solr. Different tokenizers are provided by solr to use as needed, but we choose StandardTokenizerFactory, because it is advanced and general purpose tokenizer. It is implemented based on Unicode Text Segmentation algorithm. While splitting a text stream it looks for whitespace, punctuations and identifies for sentence boundary. We defined StandardTokenizerFactory in the Silt'e information retrieval system prototype schema as follow.

```
        </analyzer>

            <tokenizer class="solr.StandardTokenizerFactory"/>

         </analyzer>
```

The following sample Silt’e text that consists of Silt’e characters, punctuation marks such - **( ) ; :: ፣ " ?** and white space was prepared to see effect of the selected tokenizer.

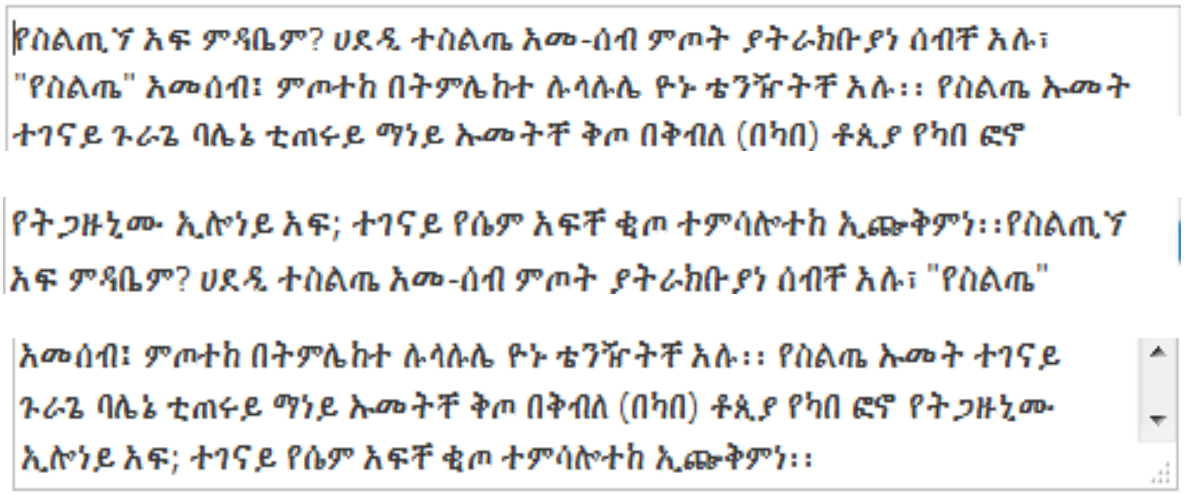

Figure 4. 2 Sample Silt’e text to test tokenizer effectiveness

The following screenshot shows portion of tokenized results.

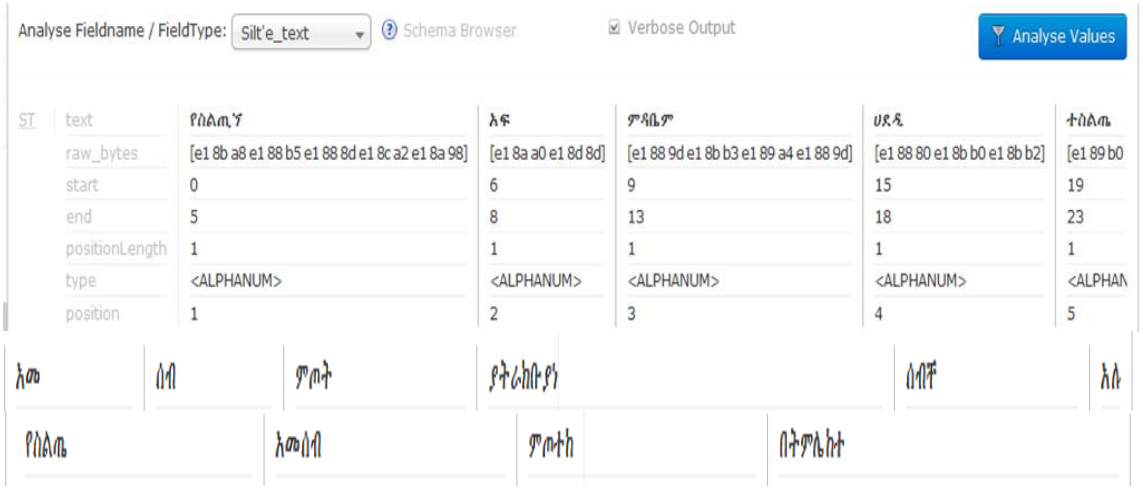

Figure 4. 3 Tokenized Silt’e texts

In the result different information about each Silt’e token is given by the system. For instance information about token “**የስልጢኘ**” it is a text, corresponding raw_bytes e1 8b a8 e1 88 b5 e1 88 8d e1 8c a4, starting position is 0 and ending position is 5, it has a position length of 1 and it is the first token in the text because its position is 1. Note that the token “**የስልጢኘ**” starts at index one and ends at index five, this means that the token has six characters, but the actual numbers of

characters are five. This happens because the system start character index counting with the initial white space.

From tokenized result we observed that the selected tokenizer is properly chop off silt'e text in to individual tokens for further text processing but, has a limitation of handling compound words. For example the compound word አመ-ሰብ is broken in to አመ and ሰብ.

4.3. Removing Silt'e Stopword

In Silt'e language there is no exact predefined stopwords list. To design Silt'e stemmer (Kedir, 2012) conducted a research. In his research he compiled 184 Silt'e stopwords. He used different approaches such as using python program to count frequency of a term in a document; research works done on Silt'e morphology and by referring dictionary to compile stopword list from the collection. We used 140 of the identified stopwords that occur many times in the documents prepared for our research. All stopwords are listed in the appendix. Sample stopwords are given in the following table.

አብተቱ	አቲ
አብታይ	ግን
አነይ	ፈሬ
ዋ	በልዳሌ
አልቀሬ	ለደር

Table 4. 1 Sample Silt'e stopwords list

Avoiding these stopwords from Silte text is helpful in the ranking of documents. To remove stopwords from a text stream solr provides a number of TokenFilters. These TokenFilters are used to process tokens that are the output of the tokenizer. The Solr's TokenFilter we choose in this work is StopFilterFactory and define it next to tokenizer during designing the Silt'e text information retrieval system prototype schema structure. In the schema design we used it as follow.

```
<filter class="solr.StopFilterFactory" words="lang/silte_stopwords.txt" />
```

To see the performance of the token filter we prepared sample Silt’e text that contain some Silt’e stopwords. The following screenshot shows the sample test text.

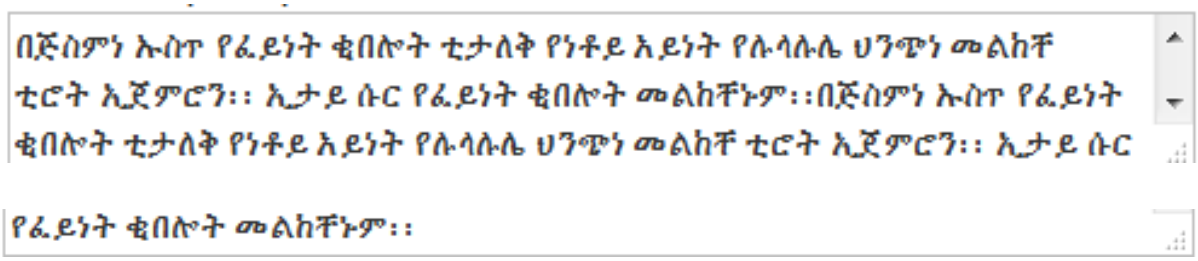

Figure 4. 4 Sample Silt’e text to test performance of stopword removal algorithm

The above test text was given to the system and portion of the analysis result is show by the following figure.

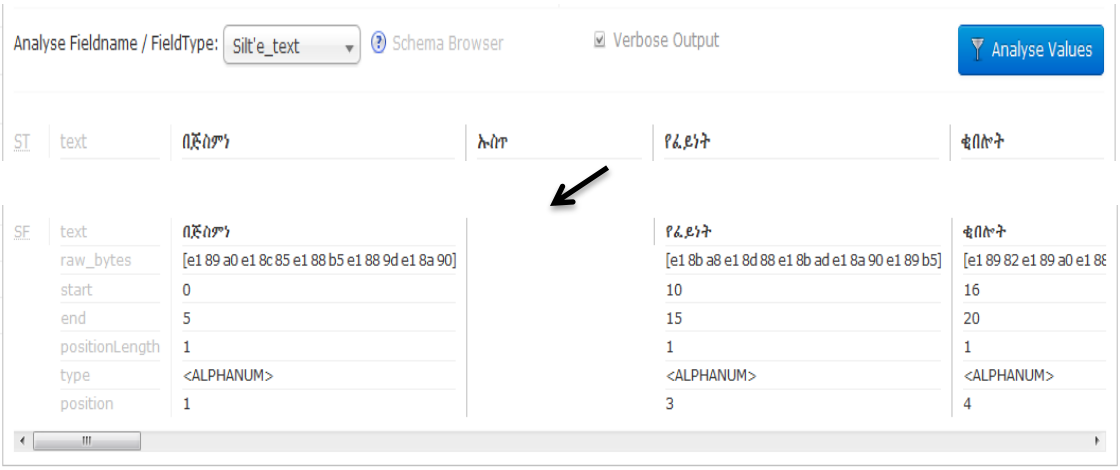

Figure 4. 5 Removing Silt’e stopword analysis result

From the analysis result we observe that in the standard tokenizer (ST) raw the given sample text is tokenized into Silt’e tokens በጅስምነ, ኡስጥ, የፈይነት and ቂበሎት. When we look at standard token filter (SF) raw there are three tokens out of four, because the token ኡስጥ is removed as indicted by the arrow in the screenshot. Generally the selected token filter identifies the stopword and removes it while indexing the Silt’e corpus.

4.4. Indexing Silt’e document

Document indexing structure that we discussed in chapter three is applied to indexing the Silt’e text documents. To do it we used Apache Solr’s default dictionary implementation

factory. After indexing the 99 Silt'e documents that we prepared for this research we get 23736 indexed words. Apache Solr's default term weighting with respect to each documents procedure is used to compute the weight of the indexed Silt'e words with respect to the indexed Silt'e documents. Finally searchable Silt'e text corpus is created in the form of inverted index data stored format. The following screenshot shows number of indexed Silt'e documents and number of indexed terms.

Figure 4. 6 Number of indexed documents and words.

4.5. Searching for Silt'e documents

User interaction with information retrieval system is needed to access the required information. To satisfy user-system interaction, we designed an interface in the prototype. It have only query input search box to access a search result. The following screenshot show sample search result for the query "የስልጤ አፍ".

Figure 4. 7 Sample search result

4.6. Performance evaluation

As we discussed in chapter three, evaluating performance of an information retrieval system is very important to test the effectiveness of the system for the search result. A number of information retrieval researchers agreed that evaluating effectiveness of an information retrieval system helps to design, develop and implement search systems (Clough, 2013). To accomplish effectiveness evaluation test document collections, different topics in the collection, and evaluation measures were involved. Queries are used to test effectiveness of the system. Also queries play an important role to check how well a given information retrieval system differentiate relevant document with non-relevant document in the collection (Fatiha, 2013).

We worked with two experts to identify relevant documents with non-relevant documents for the prepared queries from the corpus that compiled for this research. These experts are Ato Bedru Hussen from Silte Zone Education Bureau the Silt'e language curriculum preparation officer and Surur Bedru from the Silte Zone Culture, Tourism and Government Affairs Communication Bureau the Silt'e language and art organizer head.

For our research we prepared ten free text queries. These queries were identified by reading the test collection thoroughly. We tried to make the queries well representative and inclusive to test the designed prototype. The prepared queries and relevant and non-relevant documents for each query are presented by the following table.

S.N	Query	Relevant document	Non-relevant document
1	የስልጤ አፍ	6	93
2	የስልጤ አደ	7	92
3	ስልጤዋ ያየር ንባረትከ	5	94
4	የአደ ርዝቅቸ	12	87
5	የባድ መሊቅ	10	89
6	የነቶ መንቃቆ	9	90
7	ያፍ አበሮስ	9	90
8	ጎትተኜ ኤትቸ	8	91
9	ቅሬታቶ	6	93
10	ያበሮስ ቅጨ ሃለት	4	95

Table 4. 2 Relevancy for the selected queries

4.6.1. Measuring system effectiveness

There is no standardized test collection for Silt'e language that was used to evaluate information retrieval system before. Due to this fact we carefully prepared Silt'e test collection in different topics. To perform experiment we used this test collection.

In our system oriented evaluation process we focus on measuring system effectiveness. That means the prototype's ability to discriminate between relevant and non-relevant documents for the selected queries.

To accomplish system oriented evaluation process we follow the Cranfield approach (Clough, 2013). These are:-

- We apply vector space model strategy to rank documents for every query in the test collection
- Compute the effectiveness for every query in the test collection as a function of relevant documents retrieved
- Average the scores for all queries to compute overall effectiveness of the system
- Finally we form a critical opinion based on the score

We apply vector space model strategy to rank documents for every query in the test collection. We used rank based information retrieval system effectiveness measures. As we discussed in chapter three there are a number of rank based evaluation techniques out of them we choose average precision and mean average precision, because the prototype that we designed for Silt'e text information retrieval system present ranked set of results for a given user query. These measures are computed as a function of relevant documents retrieved.

We considered the top 10 retrieved results to compute the effectiveness measures. We presented the detailed computational results for precision and recall in the following table as a function of relevant documents retrieved.

Query	Evaluation measures for the selected queries as a function of relevant document retrieved										
የስልጤ አፍ	Rank Position	1	2	3	4	5	6	7	8	9	10
	Doc_Id	silteafitwalekbiyaet	siltafyatfahmotquwa	silteafyetembeadab	silteafmitot	silteafmidabesul	siteweg	yelanzielewdi	silteafsineluge	awalku	aymalemo
	Relevancy	R	R	NR	R	NR	R	R	NR	NR	NR
	Recall	0.167	0.333	0.333	0.5	0.5	0.667	0.833	0.833	0.833	0.833
	Precision	1	1	0.667	0.75	0.6	0.617	0.714	0.625	0.555	0.5
የስልጤ አደ	Doc Id	adeneleqirne	mohamedsirgaga	yelanzielewdi	yesilteyekilfan	feyekoterezqot	yesinqbuqo	kimsire	nud_ahlaqche	silte_weg	afetqirot
	Relevancy	R	NR	NR	R	NR	NR	NR	NR	NR	NR
	Recall	0.143	0.143	0.143	0.333	0.333	0.333	0.333	0.333	0.333	0.333
	Precision	1	0.5	0.333	0.5	0.4	0.333	0.286	0.25	0.222	0.2
ስልጤዋ ያየር ንባረትከ	Doc Id	silte_weather	badene2	yazgagscience	qoto	badne4	Badne6	kerm	yetopiy	zilam	abar
	Relevancy	R	NR	NR	NR	NR	NR	NR	NR	NR	NR
	Recall	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	Precision	1	0.5	0.333	0.25	0.5	0.166	0.143	0.125	0.111	0.1
የአደ ርዝቅቸ	Doc Id	yeaderizqiche	fulewabered	jismeatiregne	tesinqzof	qotwabilo	yesilteyekilfan	bitrezeqbuyanwokt	netoteqrariro	haybwazireche	ujo
	Relevancy	R	R	NR	NR	NR	R	NR	NR	NR	NR
	Recall	0.083	0.166	0.166	0.166	0.166	0.25	0.25	0.25	0.25	0.25
	Precision	1	1	0.666	0.5	0.4	0.5	0.428	0.375	0.333	0.3
የባድ መሊቅ	Doc Id	amaseb1	yakababi	amasebfeyinet	amaseb3	silte_weg	likot	nud_ahlaqche	yetopiyaafaberos	yesilteqiritо	bungechere
	Relevancy	R	NR	NR	R	NR	NR	NR	NR	NR	NR
	Recall	0.125	0.125	0.125	0.25	0.25	0.25	0.25	0.25	0.25	0.25
	Precision	1	0.5	0.333	0.5	0.4	0.333	0.285	0.25	0.222	0.2
የኑዶ መንቃቆ	Doc Id	hichina6	yazgagsci	baliqnet	meytedurero	hinchina8	yetopiyaen	netoteqrarir	yumetliq	feyekotere	baliqnet

			ence		t		dach	o		zeqot	
	Relevancy	R	NR	NR	NR	R	NR	R	NR	NR	NR
	Recall	0.111	0.111	0.111	0.111	0.222	0.222	0.333	0.333	0.333	0.333
	Precision	1	0.5	0.333	0.25	0.4	0.333	0.428	0.375	0.333	0.3
ያፍ አበሮስ	Doc Id	lang1	nilo	lang3	lang2	yetopi yaafab eros	lang4	kush	omo	yetop iyaaf cheru kobo	sinelig a
	Relevancy	R	R	R	R	NR	R	R	R	NR	NR
	Recall	0.111	0.222	0.333	0.444	0.444	0.555	0.666	0.777	0.777	0.777
	Precision	1	1	1	1	0.8	0.833	0.857	0.875	0.777	0.7
ጎትተኜ ኤትቸ	Doc Id	badn e8	badn e4	badne 3	yakab abi	hisab3	hisab 2	badne 6	geba ya	tazga gne	yesilte qiritoz eyari
	Relevancy	R	NR	R	NR	NR	NR	R	NR	NR	NR
	Recall	0.125	0.125	0.25	0.25	0.25	0.25	0.375	0.375	0.375	0.375
	Precision	1	0.5	0.666	0.5	0.4	0.333	0.428	0.375	0.333	0.3
ቅሬታዶ	Doc Id	yesilt eqirit o	yeqir itosij e	yeqirit odews	yakab abi	qirito	kurm obese r	amase b	bade ne	bung echer e	qiritoh inzot
	Relevancy	R	R	R	NR	NR	NR	NR	NR	NR	NR
	Recall	0.166	0.333	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	Precision	1	1	1	0.75	0.6	0.5	0.428	0.375	0.333	0.3
ያበሮስ ቅጨ ሃለት	Doc Id	yaber osqic he	amas ebfey net	yegere dwold hanq	yazga gscien ce	yakab abi	hinch ina4	yegere dwold	adeg nebit er	tazga gne	yezeg netme nqe
	Relevancy	R	R	NR	NR	R	NR	NR	NR	NR	R
	Recall	0.25	0.5	0.5	0.5	0.75	0.75	0.75	0.75	0.75	1
	Precision	1	1	0.666	0.5	0.6	0.5	0.428	0.375	0.333	0.4

Table 4. 3 Precision-recall measures based on relevancy

The computed average precision for each query and mean average precision over all queries of the designed prototype are presented by the following table.

S.N	Query	Average Precision
Q1	የስልጤ አፍ	0.816
Q2	የስልጤ አደ	0.75
Q3	ስልጤዋ ያየር ንባረትከ	1
Q4	የአደ ርዝቅቸ	0.833
Q5	የባድ መሊቅ	0.75
Q6	የነቶ መንቃቆ	0.609
Q7	ያፍ አበሮስ	0.937
Q8	ጎትተኔ ኤትቸ	0.698
Q9	ቅሬታቶ	1
Q10	ያበሮስ ቅጨ ሃለት	0.75
	Mean average precision	0.814
	In percent (%)	81.4%

Table 4. 4 System effectiveness measures

From the analysis table we observe that the system achieve 100% average precision for the queries Q3 and Q9. For the remaining queries the system not scores 100% average precision. The average precision is computed as a function of relevant documents retrieved, but the average precision result for the queries Q1, Q2, Q4, Q5, Q6, Q7, Q8 and Q10 indicates that there are non-relevant documents are retrieved. The reason is that in Silt'e text documents the same set of words are used to express different concepts in different documents. For example consider the words in the query **"የስልጤ አፍ".** Documents retrieved for this query describe about the background of the Silt'e language in the relevant documents, but the same set of words are used in the non-relevant documents to describe other things. When we examine the Silte text information sources that we collected for this study, different scripting style is used for the same term in different documents. For example the words **ጎትተኜ**, **ጎትተኘ** and **ጎትለኜ** have the same meaning with different scripting style. This situation has impact on the retrieved result.

Generally the obtained mean average precision 81.4% result indicates that it is encouraging to develop Silt'e vertical information retrieval systems and Silt'e search engines using vector space model. We can achieve better result by avoiding limitations on the Silt'e scripting system.

4.6.2. User oriented effectiveness evaluation

Different user aspects are considered in the design of prototype for this research. Among the features the most consideration goes to user's interaction with the system. To satisfy user-system interaction, we make the prototype interface very simple to use. It have only query input search box to access a search result. The following screenshot shows the user interface of the prototype.

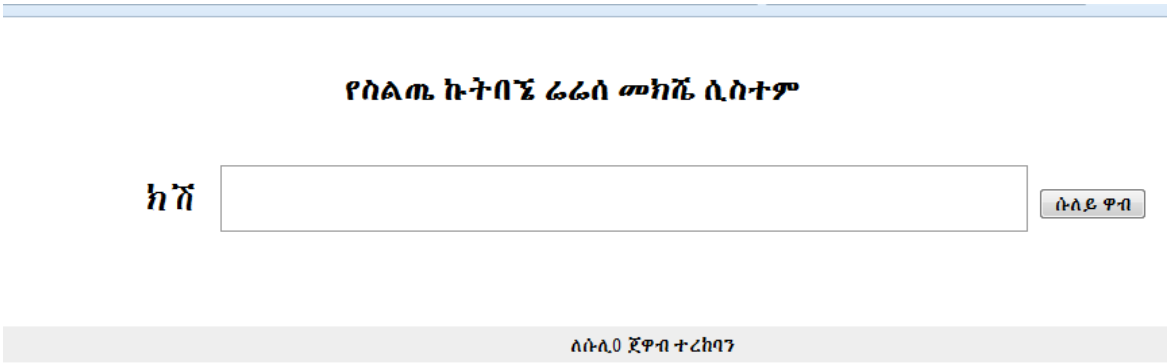

Figure 4. 8 Prototype interface

To perform system wise inspection that means usability and friendliness of the prototype we selected 15 peoples with different background. Due to time factor we limit the number of involved peoples to test the system wise evaluation. We discussed the details of the evaluation in the following table.

Number of peoples involved	Background of the peoples	Inspected elements	Response result
7	Have computing background in the field of Computer Science, Information System and Information Technology	Terminologies, coloring, layout, input and output formats used in the prototype interface	57.14% of the involved peoples provide positive response on the inspected elements. 42.86% of the involved peoples accept the terminology, layout and input/output formats, but they were not satisfied with the coloring of the interface.
8	Understand the language used in the system, but not have computing background	Easiness to use the system and satisfaction with the retrieved result	62.5% satisfied by both easiness and retrieved result. 37.5% satisfied by the retrieved result, but dissatisfied with the easiness because the Silt'e language uses Ethiopic character and they are not enough skilled to type Ethiopic characters from the keyboard

Table 4. 5 System wise evaluations

CHAPTER FIVE

Conclusion and Recommendation

5.1. Conclusion

Organizing scattered and large amount of text information in a manageable format and enabling the target users to access the information easily is the active area of information retrieval research. Consequently, numerous techniques have been proposed to design and develop information retrieval systems. But designing and developing information retrieval system highly depends on the basics of a language. Therefor proper understanding of the target language is needed to develop the required effective information retrieval system that is used by the target users.

Our effort in this thesis was investigating the possibility of designing and developing text information retrieval system to the Silt'e language. The language uses Ethiopic characters in its writing system and has its own grammatical style. Affixation is used for word formation based on the root pattern construction. Most commonly used affixations are prefix, infix and suffix. Reduplicating characters in a given token also used to form a new word. Sometime combination of affixations is involved on the word formation process. As a result, morphologically rich words with different meaning formed from the same root word. Therefore efficient Silt'e text processing procedures are required to get the best Silt'e text information search result from the storage.

We developed Silt'e text information retrieval system prototype based on vector space model using Apache solr as a prototype development tool. The reason we choose Apache solr is that it is java based package developed based on vector space model to index text information and access the indexed information. As a result, we developed Silt'e text information retrieval system prototype using this tool to achieve the best search result. The system architecture is made up of the indexing and the searching components. The searching component comprises the Silt'e text preprocessing elements that are tokenizer and stopword removal. We deal with tokenizing and stopword removing algorithms to preprocess Silt'e text. Sample Silt'e text was tested using the adopted algorithms and the result shows that these algorithms are

effective to process Silt'e text stream. Solr provides a number of stemming algorithms for different languages, but we did not adopt any of them because the Silt'e language has its own morphological rules and it needs its own stemming algorithm

The experimentation performed using the prototype that was developed without invoking stemming algorithm scored 81.4% mean average precision.

To achieve better result Apache Solr allows us to add our own module to process Silt'e text. Despite the fact, the challenge we faced was the incapability to adding Silt'e stemming algorithm in to solr because it needs enough time to do it.

Limitation we observed in our study is that the number of documents returned for each query to evaluate the system oriented effectiveness is less. The cause for this result is the corpus size we used.

5.2. Recommendations

Information retrieval research on Silt'e language is in its infant stage. The results of our work indicate the possibility of designing and developing efficient Silt'e text information retrieval systems. However a number of efforts are remain as a future work.

- Preprocessing morphologically complex languages has a positive effect on the development of information retrieval systems. Stemming algorithms are among the techniques used to return morphologically complex words in to their root form. We recommend implementing and adding Silt'e stemmer in to Apache solr to get enhanced performance.
- In this study we considered only the Silt'e text information documents. To develop fully functional Silt'e information retrieval system, we endorse considering different document types in the future study.
- Silt'e use Unicode data encoding scheme, it is challenged to processes the Silt'e text efficiently. We encourage using translated encoding scheme to design different text preprocessors and to use available Solr's efficient text processing algorithms
- Thesaurus is helpful to handle polysemy and synonymy terms for morphologically complex information retrieval systems. Conducting study by considering thesaurus for Silt'e language is stimulated.

References

Akram, R. (2015). Review: Information Retrieval Techniques and Applications. International Journal of Computer Networks and Communications Security, VOL. 3(NO. 9).

Amit, S. (2001). Modern Information Retrieval: A Brief Overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. Retrieved from http://singhal.info/ieee2001.pdf

Apache Solr News.html. (n.d.). Retrieved March 8, 2016, from http://lucene.apache.org/solr/mirrors-solr-latest-redir.html

Aslib. (2000). Effectiveness of Stemming for Turkish Text Retrieval. The Association for Information Management, 34(2), 195-200.

Bekele, D. (2011, June). Afaan Oromo-English Cross-Lingual Information Retrieval (CLIR): A CORPUS BASED APPROACH. Addis Ababa University. Retrieved from http://hdl.handle.net/123456789/8589

Cassandra, T. (2015, Augest 20). Apache Solr Reference Guide Covering Apache Solr 6.0. Retrieved from http://archive.apache.org/dist/lucene/solr/ref-guide/apache-solr-ref-guid-6.0.pdf

Christopher, D. P. (2008). An Introduction to Information System. Cambridge: Cambridge University Press. Retrieved from www.cambridge.org/9780521865715/2008

Clough, P. &. (2013, July). Evaluting the performance of infromation retrieval systems using test collections. Information Research an International electronic jornal. Retrieved from www.informationr.net/ir/index.html

Cummins, C. O. (n.d). Determining general term weighting schemes for the Vector Space Model of Information Retrieval using Genetic Programming. Galway: National University of Ireland.

Dikshant, S. (2015). Apache Solr: A Practical Approach to Enterprise Search. New York: Springer Science+Business Media.

Djoerd, H. (2009). Information Retrieval Models. Retrieved from http://wwwhome.cs.utwente.nl/~hiemstra/papers/IRModelsTutorial-draft.pdf

Fatiha, B. W. (2013). Concept-Based Indexing in Text Information RetrievaL. International Journal of Computer Science & Information Technology, Vol 5(No 1). Retrieved May 10, 2016, from airccse.org/journal/ijcsit2013_curr.html

Francis, W. (1982). Frequency Analysis of English Usage: Lexicon and Grammar. Houghton Mifflin.

Geza, T. (2012). Language Planning And Policy In The Silt'e Zone. Addis Ababa University. Retrieved January 18, 2016, from http://hdl.handle.net/123456789/8405

Gezehagn, G. (2012). Afaan Oromo Text Retrieval System. Addis Ababa University. Retrieved from http://hdl.handle.net/123456789/2850

Grammar. (2016). Retrieved April 20, 2016, from Wikipedia: htt://en.m.wikipedia.org/wiki/Grammar

Hirpa, A. (2012, June). Probabilistic Information Retrieval System for Amharic Language. Addis Ababa University. Retrieved from http://hdl.handle.net/123456789/8376

Information_retrieval. (2016, July 12). Retrieved July 15, 2016, from Wikipedia: https://en.m.wikipedia.org/wiki/information_retrieval

Introduction_to_vector_space_model. (n.d). Retrieved April 9, 2016, from Introduction to vector space model: http://cogsys.imm.dtu.dk/thor/projects/multimedia/textmining/node5.html

Jitendra, S. (2012). Analysis of Vector Space Model in Information Retrieval. International Journal of Computer Applications, 14-18.

Juhi, A. N. (2013). Improving the Quality of Gujarati-Hindi Machine Translation Through Part of speech Tagging and Stemmer Assisted Trasliteration. International Journal on Natural Language Computing (IJNLC), 2(3), 49-54.

Kedir, M. (2012, June). Designinig a Stemming Algorithm for Silt'e. Addis Ababa University. Retrieved from www.etd.aau.edu.et/handle/123456789/8723

Keneilwe, & Tranos, Z. (2012, June). Evaluation Of Information Retrieval Systems. International Journal of Computer Science & Information Technology, Vol 4(No 3).

Keneilwe, T. (2012, june). Evaluation of Information Retrieval. International Journal of Computer Science & Information Technology, 4(3).

khemani, M. (n.d). Retrieved April 18, 2016, from Mining the Web through Information Retrieval: http://www.miislita.com

Kumaran, A. (2008). Adapting information retrieval systems to user queries. Information Processing and Management, 1838–1862.

Lewis, M. P. (2016). Retrieved April 21, 2016, from Ethnologue: https://www.ethnologue.com/language/stv

Luhn. (1957). A statstical Approach to Mechanized Encoding and Searching of Literary Information. IBM Journal of Research and Development, 309-317.

Majeed, Z. (2014). Search Queries in an Information Retrieval System for Arabic Language Text. University of Kentucky.

Mark, S. B. (n.d). The History of Information Retrieval Research. doi:Retrieved from http://ciir.publications.cs.umass.edu/getpdf.php

Ricardo, B. Y. (2011). Modern Information Retrieval the concepts and technology behind search (Vol. second edition). Pearson Education Limited.

Robert, R. K. (1997). Information Storage and Retrieval . Wily-india: John Wiley & Sons, inc.

Salton, B. (1988). Term Wieghting Approaches in Automatic Text Retrieval.

Sasaki, Y. (2007, October 26). The truth of the F-measure. Schoole of Computer Science, University of Manchester. Retrieved from http://www.cs.odu.edu/f-measure

Silt'e_language. (n.d). Retrieved April 21, 2016, from Wikipedia: http://en.wikipedia.org/wiki/Silte%27e_language

Standard_Boolean_model. (2016, May). Standard Boolean model. Retrieved April 14, 2016, from wikipedia: https://en.m.wikipedia.org/wiki/Standard_Boolean_model

Stemming. (2016, June 1). Retrieved April 29, 2016, from Wikipedia: http://en.m.wikipeia.org/wiki/Stemming

Tewodros, H. (2012, May 2). Amharic Text Retrieval: An Experiment Using Latent Semantic Indexing (LSI) With Singular Value Decomposition (SVD). Addis Abab Univesity. Retrieved from http://hdl.handle.net/123456789/2807

Tsz-Wai, L. B. (2005). Automatically Building a Stopword List for an Information Retrieval System. 5th Dutch-Belgium Information Retrieval Workshop (DIR) ’05 Utrecht. Retrieved from www.cs.uu.nl/dir2005/

Uma, D. G. (2014). Wordnet and Ontology Based Query Expansion for Semantic information retrieval in sports domain. Journal of Computer Science, 11(2), 361-371.

Vaibhav, C. P. (2015). Information Retrieval and Context Based Document summerization Using Vector Space Model. International Journal of Emerging Technology and Advanced Engineering, Volume 5(Issue 9). Retrieved June 22, 2016, from www.ijetae.com/IJETAE_0915_46.pdf

vecctorspacemodel. (2016, February 27). Retrieved March 17, 2016, from Wikipedia: https://en.wikipedia.org/wiki/Vector_space_model

vector_space_model. (2016, April 5). Retrieved April 16, 2016, from Wikiprdia: https://en.m.wikipedia.org/wiki/vector_space_model

Vikram, S. B. (2014). An EffectiveE Tokenization Algorithm for Information Rertieval Systems. Department of Computer Engineering,National Institute of Technology, Kurukshetra, Haryana, India.

Wahiba, B. A. (2013). A New Stemmer to Imorove Information Retrieval. International Journal of Network Security & Its Applications (IJNSA), 05, 143-154. Retrieved from http://www.airccse.org/journal/nsa/5413nsa11.pdf

Wordofa, M. (2013, June). Semantic Indexing and Document Clustering for Amharic Information RetrievaL. Addis Ababa University. Retrieved from http://hdl.handle.net/123456789/8719

Zheng, W. J. (n.d). A Fast Algorithm for Constructing Inverted Files on Heterogeneneous Platforms. Maryland, United States of America. Retrieved from http:/www.umiacs.umd.edu/indexing

Appendix I. Lists of Ethiopic alphabets used for Silt'e

1st order	2nd order	3rd order	4th order	5th order	6th order	7th order
ሀ ha	ሁ hu	ሂ hii	ሃ haa	ሄ he	ህ hi	ሆ ho
ለ le	ሉ lu	ሊ lii	ላ laa	ሌ le	ል li	ሎ lo
መ me	ሙ mu	ሚ mi	ማ maa	ሜ me	ም mi	ሞ mo
ረ re	ሩ ru	ሪ ri	ራ raa	ሬ re	ር ri	ሮ ro
ሰ se	ሱ su	ሲ si	ሳ saa	ሴ se	ስ si	ሶ so
ሸ ša	ሹ šu	ሺ šii	ሻ šaa	ሼ še	ሽ ši	ሾ šo
ቀ qa	ቁ qu	ቂ qii	ቃ qaa	ቄ qe	ቅ qi	ቆ qo
በ ba	ቡ bu	ቢ bii	ባ baa	ቤ be	ብ bi	ቦ bo
ተ te	ቱ tu	ቲ tii	ታ taa	ቴ te	ት ti	ቶ to
ቸ ca	ቹ cu	ቺ cii	ቻ caa	ቼ ce	ች ci	ቾ co
ነ ne	ኑ nu	ኒ nii	ና naa	ኔ ne	ን ni	ኖ no
ኘ ña	ኙ ñu	ኚ ñii	ኛ ñaa	ኜ ñe	ኝ ñi	ኞ ño
አ a	ኡ u	ኢ ii	ኣ aa	ኤ ee	እ i	ኦ o
ከ ka	ኩ ku	ኪ kii	ካ kaa	ኬ ke	ክ ki	ኮ ko
ወ wa	ዉ wu	ዊ wii	ዋ waa	ዌ we	ው wu	ዎ wo
ዘ za	ዙ zu	ዚ zii	ዛ zaa	ዜ ze	ዝ zi	ዞ zo
ዠ ža	ዡ žu	ዢ žii	ዣ žaa	ዤ že	ዥ ži	ዦ žo
የ ya	ዩ yu	ይ yii	ያ yaa	ዬ ye	ዪ yu	ዮ yo
ደ da	ዱ du	ዲ dii	ዳ daa	ዴ de	ድ di	ዶ do
ጀ ja	ጁ ju	ጂ jii	ጃ jaa	ጄ je	ጅ ji	ጆ jo
ገ ga	ጉ gu	ጊ gii	ጋ gaa	ጌ ge	ግ gi	ጎ go

ጠ ta	ጡ tu	ጢ tii	ጣ taa	ጤ te	ጥ ti	ጦ to
ጨ ça	ጩ çu	ጪ çii	ጫ çaa	ጬ çe	ጭ çi	ጮ ço
ጰ pa	ጱ pu	ጲ pii	ጳ paa	ጴ pe	ጵ pi	ጶ po
ፈ fa	ፉ fu	ፊ fii	ፋ faa	ፌ fe	ፍ fi	ፎ fo
ፐ pa	ፑ pu	ፒ pii	ፓ paa	ፔ pe	ፕ pi	ፖ po

Appendix II. Lists of Silt'e stopwords

አብተቴእብታይ	ደር	ናሩ	በለ
እብተቴ	ኤት	ብቾ	ቂጦ
አብታይ	ፈሬ	ለገነ	ለደር
አብተቴ	ፎኖ	ገነ	አበይ
እሊ	ግን	ሁልምከ	ጉት
አሊ	ቀደ	ለሳድባድከ	ኤጋህ
እነይ	ስር	ለሰባድሽ	ኤጋሽ
አነይ	እነኮ	ለሰባድኑም	ኤጋሙ
እቲ	ኡንኮ	ለሰገግሽ	ተዮን
አቲ	ማ	ለሰገግኑም	ገናገናይ
እቢ	ምን	ለሰገግከ	ዮናነይ
አቢ	አይኔ	ለሰገጋሙ	ቢትላይም
ዋ	አይነኮ	ሀዳድኑም	አሎነ
ሱር	ላይሽ	ድባየ	ሂነሚ
አድ	አይታይ	አብሌ	በሁነትንሙ
አድአድ	ወይ	ቱፍትሉፍት	ሁኖተኒመዋ
እነይ	ታሌ	ተሬር	አታይ
በልዳሌ	ናር	ታቼን	ሀነይ
አልቀሬ	ናርት	ቀለ	ሁኖ

ገና	ደር	ኡሀ	የለ
የልስ	አይነኮ	ያሽ	ሱር
ሉላሉሌ	ሉሌ	ዬዬ	ሂንኩምንገ
በሉሌ	አቢሌ	በዬ	ሉሀ
አለቢ	ተደር	ዋ	ለሄ
ኢንኩምንገ	በውኖትምኪ	ለገነገናም	አዩ
ቅጨ	የገግ	ለሄው	እታይ
ብቾ	ገገኑ	በሰቼ	እተቴ
ገገ	ቲታሚ	አተም	አታይ
ኡስጥ	ኦናኔ	ግዝቾ	አተቴ
ግዝ	ገጌ	ሬራቀደን	ትዮኑ
ሆነምታሌ	ሀነግነ	ሬራ	ዩታይ
በሁኖት	እነይ	ቀደን	ዩተቴ
ሉላሉሌ	ሱርዋ	አናግና	ያታይ
በልዳሌ	ገናሚ	ኢንኩምንገ	ያተቴ
ወክት	ሂንኩምንገ	ሉሉሌ	

Appendix III. Lists of Silt’e prefixes

ኢለው	እት	ላ	ይ
እለው	እል	ቃ	ት
በል	እለ	ጫ	ከ
ኢለ	አል	አ	ቲ
በሰ	አት	ሰ	ተ
አይ	ተይ	ለ	ሻ
እተ	በ	የ	ና

ያ እ

Appendix IV. Lists of Silt’e suffixes

ኩሞመ	እሎ	ኩ	ቱ
ቢያኔ	ዬን	ሁ	ካ
አታም	ኔት	ነ	መ
አተኘ	ታም	ን	ሙ
አመ	ተኛ	ሺ	ሚ
ሼታ	ኢ	ኤ	ዋ
ኩመ	ይ	አ	ሶ
ኒመ	ኤ	ተ	ኒ
ሼሸ	ሸ	ት	ኝ
ታት	ኔ	ኡ	ሎ
ኤን	ከ	እ	ክ
አኘ	ሸ	ኦ	ኛ
ተኘ	ቸ	ው	
ንቾ	ኮ	ም	

Appendix V. Schema structure for Silt'e text information retrieval system

```
<?xml version="1.0" encoding="UTF-8" ?>

<schema name="silte_tirs" version="1.6">

    <field name="_version_" type="long" indexed="true" stored="false" />

    <field name="id" type="string" indexed="true" stored="true" required="true"
    multiValued="false" />

    <field name="content_type" type="string" indexed="true" stored="true"
    multiValued="true"/>

    <field name="links" type="string" indexed="true" stored="true" multiValued="true"/>

    <field name="_src_" type="string" indexed="false" stored="true"/>

    <field name="content" type="text_general" indexed="false" stored="true"
    multiValued="true"/>

    <field name="text" type="text_general" indexed="true" stored="false" multiValued="true"/>

    <fieldType name="string" class="solr.StrField" sortMissingLast="true" />

    <fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"/>

    <fieldType name="int" class="solr.TrieIntField" docValues="true" precisionStep="0"
    positionIncrementGap="0"/>

    <fieldType name="float" class="solr.TrieFloatField" docValues="true" precisionStep="0"
    positionIncrementGap="0"/>

    <fieldType name="long" class="solr.TrieLongField" docValues="true" precisionStep="0"
    positionIncrementGap="0"/>

    <fieldType name="double" class="solr.TrieDoubleField" docValues="true" precisionStep="0"
    positionIncrementGap="0"/>
```

<fieldType name="tint" class="solr.TrieIntField" docValues="true" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tfloat" class="solr.TrieFloatField" docValues="true" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tlong" class="solr.TrieLongField" docValues="true" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tdouble" class="solr.TrieDoubleField" docValues="true" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="date" class="solr.TrieDateField" docValues="true" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="tdate" class="solr.TrieDateField" docValues="true" precisionStep="6" positionIncrementGap="0"/>

<fieldType name="binary" class="solr.BinaryField"/>

<fieldType name="random" class="solr.RandomSortField" indexed="true" />

<fieldType name="text_ws" class="solr.TextField" positionIncrementGap="100">

<analyzer>

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

</analyzer>

</fieldType>

<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100">

<analyzer type="index">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />

```
        <filter class="solr.LowerCaseFilterFactory"/>
        </analyzer>
        <analyzer type="query">
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
        <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true"
        expand="true"/>
        <filter class="solr.LowerCaseFilterFactory"/>
        </analyzer>
</fieldType>
<fieldType name="phonetic" stored="false" indexed="true" class="solr.TextField" >
        <analyzer>
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.DoubleMetaphoneFilterFactory" inject="false"/>
        </analyzer>
</fieldType>
<fieldType name="payloads" stored="false" indexed="true" class="solr.TextField" >
        <analyzer>
        <tokenizer class="solr.WhitespaceTokenizerFactory"/>
        <filter class="solr.DelimitedPayloadTokenFilterFactory" encoder="float"/>
        </analyzer>
</fieldType>
```

```
<fieldType name="descendent_path" class="solr.TextField">
        <analyzer type="index">
        <tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/" />
        </analyzer>
        <analyzer type="query">
        <tokenizer class="solr.KeywordTokenizerFactory" />
        </analyzer>
</fieldType>
<fieldType name="ancestor_path" class="solr.TextField">
        <analyzer type="index">
        <tokenizer class="solr.KeywordTokenizerFactory" />
        </analyzer>
        <analyzer type="query">
        <tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/" />
        </analyzer>
</fieldType>
<fieldType name="Silt'e_text" class="solr.TextField" positionIncrementGap="100">
        <analyzer>
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.StopFilterFactory" ignoreCase="true"
        words="lang/stopwords_st.txt" />
        </analyzer>
        </fieldType>
        <fieldType name="preanalyzed" class="solr.PreAnalyzedField">
```

```
            <analyzer type="query">
            <tokenizer class="solr.WhitespaceTokenizerFactory"/>
            </analyzer>
        </fieldType>
</schema>
```