ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

PROBABILISTIC INFORMATION RETRIEVAL SYSTEM FOR AMHARIC LANGUAGE

AMANUEL HIRPA MADESSA

June 2012

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

PROBABILISTIC INFORMATION RETRIEVAL SYSTEM FOR AMHARIC LANGUAGE

BY:

AMANUEL HIRPA MADESSA

A thesis submitted to the School of Information Science of Addis Ababa University in partial fulfillment of the requirement for the Degree of Master of Science in Information Science

June 2012

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

PROBABILISTIC INFORMATION RETRIEVAL SYSTEM FOR AMHARIC LANGUAGE

BY:

AMANUEL HIRPA MADESSA

Name and signature of members of the examining board

Name	Title	Signature	Date
_____	Chairperson	_____	_____
Million Meshesha (PhD)	Advisor	_____	_____
Dereje Teferi (PhD)	Examiner	_____	_____

DEDICATED TO:

MY SISTER MEKDES HIRPA (FIKIR)

DECLARATION

This thesis is my original work. It has not been presented for a degree in any other university and all sources of material used for the thesis have been duly acknowledged.

_____

AMANUEL HIRPA MADESSA

The thesis has been submitted for examination with my approval as university advisors

_____

MILLION MESHESHA (PhD)

June 2012

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHM

LIST OF ACRONYMS AND ABBREVIATION

BIM	Binary Independent Model
BNM	Bayesian Network Model
DAG	Directed Acyclic Graph
DFR	Divergence for Randomness
EBM	Extended Boolean Model
ENA	Ethiopian News Agency
FDRE	Federal Democratic Republic of Ethiopia
GVSM	Generalized Vector Space Model
IDF	Inverse Document Frequency
INM	Inference Network Model
IR	Information Retrieval
LM	Language Model
LSI	Latent Semantic Indexing
MAP	Mean Average Precision
PRP	Probabilistic Ranking Principle
SVD	Singular Value Decomposition
TF	Term Frequency
TREC	Text Retrieval Conference
VSM	Vector Space Model

LIST OF APPENDIXES

ACKNOWLEDGMENT

First and foremost I thank God, who makes everything possible. My greatest gratitude is extended to my advisor Dr. Million Meshesha for his constructive reviews of my work and for his valuable advice giving me to overcome challenges and finish this research at the right time. I offer special thanks to my family particularly my sisters Mekdes (Fikir) and Selam (Baby) and her husband Fayisa, my brothers Tesfaye Hirpa and Tesfaye Asmera, who have been supporting me in so many ways. My thanks also goes to my fellow classmates specially my friend Solomon G/Mariam for his moral support and constructive ideas during the research. Last but not list, I would like to thank Aksum University for giving me the scholarship and taking care of my necessary needs and also Addis Ababa University giving me the chance to enroll in information science masters program.

Abstract

Nowadays, a considerable amount of information has been produced in Ethiopia. This accumulation of information is challenging for archival and searching from the existing huge amount of information particularly written in Amharic language. Thus, developing an information retrieval (IR) system for Amharic language allows searching and retrieving relevant documents that satisfy information need of users. Accordingly, few IR systems have been developed. However, those IR systems have not registered a promising performance because they are developed based on vector space model that do not have the mechanism to define user's information need using relevance feedback and query reformulation techniques unless other modules are integrated. Furthermore, the model does not define uncertainty that exists in IR systems. In order to solve these issues, probabilistic retrieval model that has the capability of reweighting query terms based on relevance feedback can be used.

In this research, a probabilistic based IR system is developed for Amharic language. Both indexing and searching module was constructed. In these modules, different text operations such as: tokenization, normalization, stemming and stop word removal are included. Then, the retrieval system is tested and the experimental results show that probabilistic based IR system returned encouraging result even without controlling the problem of synonyms and polysemous terms that exist in Amharic text. The system registered on the average 73% F-measure. Nevertheless, the performance of the system is greatly affected by synonyms and polysemous terms that exist in the language beside its richness in morphology (variant words).

Keywords: Information Retrieval, Probabilistic Model, Amharic Language.

CHAPTER ONE

INTRODUCTION

1.1. Background

Nowadays, information is everywhere that helps people to make the correct decision at the right time. As the written information becomes large in size and digital documents easily available electronically, it would be difficult to retrieve relevant documents among the accumulated document collections. This exponential growth of information records of all kinds results in the problem (phenomenon) of information explosion [1].

Before 1990s most people preferred getting information from others rather than information retrieval system [1]. However, after optimization of the effectiveness, information retrieval system becomes the most preferred means of accessing information by people. Therefore, the need to store and retrieve written information effectively and efficiently has become increasingly important.

It has been long time since people realized the importance of retrieving relevant information from a given document collection. Before 3000 B.C., the people of Sumerians find out a new way of storing clay tablet with cuneiform inscriptions to make the clay tablet accessible effectively and efficiently [2]. The need for storing and retrieving relevant information becomes important especially after the invention of paper and printing press, which leads to the problem of information explosion. Vanevar Bush first finds out the problem of information explosion in 1945. He wrote an outstanding article titled "As We May Think". This article is a major contribution for the development of automatic access to large amount of accumulated information. In 1952, his idea was applied and named information retrieval [3]. Since then, many scholars have been working on storing and retrieving relevant information from large collection of documents.

Information retrieval (IR) is the process of finding relevant documents from unstructured document collection that satisfies information need of users [1]. According to Ricardo and Ribeiro-Neto [4], "Information retrieval deals with the representation, storage, organization

and access to information items". While the representation and organization of the information items provides the user with easy access to the information he/she is seeking for, the storage of information allows accumulating knowledge to make documents easily accessible for later use. In addition, the access to information item provides the user with effective and efficient retrieval of information based on their need.

One of the mechanisms to create effective communication between reader/user and author/writer is effective IR system [5]. The main goal of information retrieval system is to retrieve relevant information, but there is no a single IR system capable of retrieving only relevant document according to the user's need. Rather, it retrieves irrelevant documents [6].

Predicting relevant documents is one of the core issues in IR system, and IR models are responsible for determining the prediction of what is relevant and what is not [4]. A number of retrieval models have been proposed since the mid 1960s. They have evolved from specific models intended for use with small structured document to recent models that have strong theoretical basis and which are intended to accommodate variety of full text document types such as: Boolean model, vector space model, probabilistic model, etc.

Current models handle documents with complex internal structure and most of them incorporate a learning or relevance feedback component that can improve performance when presented with sample relevant documents [7].

Probabilistic model is a statistical analysis model that estimates the probability of a document relevance given available evidences. It works based on the probability ranking principle, which states that "An information retrieval system is supposed to rank the documents based on their probability of relevance to the query, given all the evidence available" [8].

The probabilistic model incorporates relevance feedback and query expansion mechanisms. It is the oldest but also the hottest research area in IR. There can be variety of sources of evidence that are used by the probabilistic retrieval methods. The most common one is the statistical distribution of the terms in both the relevant and non-relevant

documents. The principle takes into account that there is uncertainty in the representation of the information need and the documents [9]. Table 1.1 depicts the defining characteristics of probabilistic model and vector space model.

	Probabilistic Model	**Vector Space Model**
Motivation	Address uncertainty in query representations	Simplify query formulation
Goal	Rank the output based on probability of relevance	Rank the output based on similarity
Methods	Use of different model	Cosine measure

Table 1.1: Characteristics of vector space model and probabilistic model

IR system is different from other kind of information system by its uncertainty nature [7]. In a database system, a query is formulated precisely to the information need and there is an exact definition of which elements of the database represent the answer. However, in IR system neither a query formulation can be assumed to represent exclusively an information need nor there is a clear procedure that decides whether a document is relevant or not. The most successful approach for handling the uncertainty nature of IR is probabilistic model.

Empirical evidences show that other models like Vector space model (VSM) and its variant models such as, Extended Boolean Model (EBM) and Generalized Vector space model (GVSM) are not attempted to define uncertainty in IR system. They do not have relevance feedback and query expansion mechanism by themselves to do with the external realities of users and information needs unless other modules are integrated to those models so as to register better performance [4][7][9].

There are different information retrieval methods which have a probabilistic basis [10]. The most widely used ones are Binary Independence Model (BIM) and Bayesian Network Model (BNM). BIM works based on representation of queries and documents (i.e. as sets

of terms) and by collecting relevance feedback data from few documents, the model then can be applied in order to estimate the probability of relevance for the remaining documents in the collection. However, in BNM graphical representation of probabilistic model is used to show probabilistic dependency between variables. The model then estimate the probability of relevance based on child and parent causal relationship that is represented by linking each parent node to the child node [4].

Several researches in the area of probabilistic retrieval model has been conducted globally. Robertson et.al [11] developed a probabilistic model based IR system for English language to solve the problem of uncertainty found in IR system. From the experiment, encouraging result is found. However, the consideration of giving the same weights for terms in different document was one of the main challenges. Rijsbergen [12] also developed probabilistic model based IR system to improve the binary independent assumption and the uncertainty nature of IR by considering the semantic relationship of query terms and document.

1.2. Statement of the Problem

These days, a considerable amount of information has been produced in Ethiopia, especially in Amharic language within organizations and outside the organizations rapidly and continuously. This information explosion is challenging for archival and searching from the existing huge amount of information that is written in Amharic language.

There are more than 80 languages in Ethiopia and Amharic is the mother tongue language for more than 25 million people [6]. According to Federal Democratic Republic of Ethiopian (FDRE) population census commission [13], it is the first language for more than 20 million and second language for over 5 million people. Accordingly, there are huge collections of documents available in the form of books, magazines, newspapers, novels, officials and legal documents, etc. Developing an IR system that enables searching and retrieving relevant document written in Amharic language is important.

Several works have been done in the last decade on Amharic information retrieval system, such as: N-Gram based automatic indexing for Amharic text [14], Amharic text retrieval

using latent semantic indexing with singular value decomposition [15], design and implementation of Amharic search engine [16] and semantic based query expansion technique for Amharic IR [6]. However, as depicted in table 1.2, IR systems based on vector space model developed for Amharic language so far has not registered a better performance. According to Justin and Rolf [9], "effective integration of more information about users need should lead to better IR performance". Vector space model and its variants do not have the mechanism to define users need using relevance feedback and query reformulation techniques unless other modules are integrated to the models. In comparison, probabilistic model by itself enables defining user's need using relevant feedback and query reformulation techniques.

As stated by Fuhr [7], the other problem of developing an IR system using vector space model is lack of handling uncertainty nature of IR. How exact is the representation of the document and query? How well is query match to documents? How relevant is the searching result to the query? Are the relevant documents uncertain? The most successful approach for coping with uncertain in IR is a probabilistic model. Probability theory seems to be the most natural way to quantify uncertainty.

Vector space model and its variants do not give emphasis for terms that are not in the user's query even though they describe the user's information need. To the contrary, probabilistic model includes some form of probabilistic relevance feedback where relevant documents reinforce each other [17][18].

There are a number of applications of probabilistic model in information retrieval system designed worldwide for different languages. The research done by Huang and Robertson [19] for Chinese language and Dolamic and Savoy [20] for Russian language are worth mentioning here.

Hence, this research is initiated to experiment the effectiveness of probabilistic model based IR system for Amharic language. This can help to exploit the advantage of probabilistic model in designing a generic IR system that modify its searching based on relevance feedback and term reweighting.

Authors	Year	Title	Corpus Type	Average Precision	Average Recall	F-measure
Betelihem [14]	2002	N-Gram based automatic indexing for Amharic text	News	0.05	0.88	0.09
Tewodros [15]	2003	Amharic text retrieval: an experiment using latent semantic indexing with singular value decomposition	News	0.71	0.55	0.61
Solomon et.al [16]	2009	Design and Implementation of Amharic Search Engine	Web Documents	0.99	0.52	0.68
Abey [6]	2011	Semantic based query expansion technique for Amharic IR	Amharic Bible	0.53	0.73	0.63

Table 1.2: Previous Amharic IR systems performance summary

To this end, this research attempts to find solution for the following research questions:

- What are the properties and word formations in Amharic language?
- What are the suitable components to design probabilistic based retrieval system?
- Does probabilistic model improve effectiveness of the IR system in searching for relevant Amharic documents?

1.3. Objective of the Study

To conduct this research the following general and specific objectives are established.

1.3.1. General Objective

The general objective of this study is to experiment the effectiveness of probabilistic IR model for searching relevant documents from Amharic text corpus so as to design an applicable Amharic information retrieval system.

1.3.2. Specific Objectives

With the aim of achieving the general objective of this study, the following specific objectives are formulated:-

- To understand concepts on probabilistic IR model and review related works from literatures to define the contribution of this study.
- To explore and identify the linguistic features of the Amharic language applicable to probabilistic model.
- To setup the experiment by organizing Amharic documents, queries and relevant judgments.
- To generate content-bearing index terms by applying text-preprocessing techniques such as, stop word removal, stemming and normalization.
- To design a prototype probabilistic Amharic IR system that can search for relevant documents from Amharic corpus.
- To evaluate the performance of the system using IR effectiveness measures such as recall, precision and F-measure.

1.4. Scope and Limitation of the Study

The scope of this research is to develop a prototype IR system by applying a binary independent method of probabilistic model. The IR system is designed to search within large size Amharic document corpus. News items are collected that covers issues such as, politics, sport, economic, social, accident, health, education, tourism and justice.

The system is developed following the procedures of the IR system. Given Amharic document corpus, text operations such as, tokenization, stop words removal, normalization

and stemming are integrated to select content-bearing index terms. Inverted file indexing is used to organize content-bearing index terms. Finally, searching module is enabled to allow users search for relevant documents that satisfy their information need. To improve the performance of the system, term-reweighting based query reformulation is also supported.

Lack of standard large size document corpus and lack of thesaurus to integrate for query expansion mechanism to control Amharic synonyms and polysemy words are the limitations of this research. Due to long processing time to construct relevance matrix based relevance judgment, considering the time constraint, the experiment was conducted only using 300 Amharic news documents.

1.5. Methodology

According to Abiy et al. [21], methodology is the theory and analysis of deciding how research should proceed, and it involves analysis of the principles and procedures followed in a research. Methodology covers the entire approach of research. Therefore, in this study methodology is used to gain fuller understanding of phenomenon, to produce a better quality documentation, to ensure consistency and requirements met completely.

The following methods and procedures are used in order to achieve the stated objectives of this study.

1.5.1. Literature Review

In order to have deeper understanding on philosophies, principles/theories and techniques of information retrieval system, particularly on the probabilistic information retrieval model, extensive literature review is conducted from books and Internet. Previous related research works, printed materials such as, journal articles, conference papers and electronic materials on the web are also assessed. Additional literature review and document analyses are made to investigate and identify the feature of Amharic text, which is important to the research in the course of Amharic text operations.

1.5.2. Dataset Preparation

In this research, Amharic documents collected from Amharic local news articles available on web site of Walta Information Center were used as a test data. Walta Information Center is a private organization, which produces and distributes news on television and radio [22].

The researcher inspired to use news articles as a test data because, news articles are easier to access, available in electronic form, sufficient in size, covers all domain areas and previous researches such as Betelihem [14] and Tewodros [15] used 100 and 200 collections of local news articles respectively.

1.5.3. Implementation Tools

Python programming language is used for developing the system. The researcher initiated to use Python because; Python's syntax is clear and readable. Experts and beginners can easily understand the code and everyone can become productive in Python very quickly [23].

It is simple to get support and fast to code. Python provides fast feedback in several ways. First, the programmer can skip many tasks that other languages require him to take. Therefore, it reduces both the cost of program maintenance and the development time. Python also encourages program reusability by implementing modules and packages. One can easily share functionality between programs by breaking the programs into modules, and reusing the modules as components of other programs. Python is also a portable programming language [24].

1.5.4. Testing Procedure

To test the system Amharic queries are formulated and relevance judgment is prepared to construct document query matrix that shows all relevant documents for each test queries. To assess the effectiveness of the proposed system (i.e., the quality of its search results) the most frequent and basic statistical measures, recall, precision and F-measure are used.

Recall is percentage of relevant documents retrieved from the database in response to users query. Precision is percentage of retrieved documents that are relevant to the query. F-measure is a weighted harmonic mean of precision and recall [1]. To visualize the performance of the system, precision and recall curve, which reflect precision at different standard recall levels were plotted.

1.6. Significance of the Study

This study is expected to produce results in several ways. The main significance is designing an applicable Amharic information retrieval system that can help users search and retrieve relevant documents written in Amharic language. It has also specific contributions. First, this study produces results that can show the advantage of applying a probabilistic information retrieval model for Amharic language. Second, it gives direction for researchers to select the best model for future works on Amharic text retrieval based on the performance achieved. Third, since the model can easily extended it can be applied to any problem where there is a need to retrieve any documents written in local Semitic languages (such as, Guragegna, Tigrigna, Siltegna etc) from large collection. Additionally, it is an academic exercise to fulfill the requirement of masters program the researcher is enrolled in.

1.7. Organization of the Research

This study is organized in to five chapters. The first chapter discusses introduction that provides background information, the problem statement, the research objective, research question, methodology, scope and limitations and significance of the study.

The second chapter reviews the literature covering the various theories/philosophies, techniques and methods of information retrieval system and various information retrieval models especially probabilistic model. In addition, the historical background of Amharic language and its writing system are covered.

The third chapter presents the technique implemented, the architecture adopted and the algorithm used in this study. The experimental settings, test results interpretations and the findings of the experiment are presented in chapter four.

Finally, chapter five presents conclusion drawn from the findings of the study and recommendations that should be considered in future researches for designing an applicable Amharic IR system.

CHAPTER TWO

LITERATURE REVIEW

Information retrieval is a key technology that has been made in the history of humankind. It is the key technology behind search engines and an everyday technology for many web users [25][26].

An IR system has been developed for serving user's purpose of; good reading and magazine articles for an assignments; finding educational material for a learning objective; finding facts for decision making etc. However, the main problem is to retrieve what is useful and leaving what is not or in other word developing a perfect retrieval system. Perfect retrieval system does not exist, because relevant judgment is based on the subjective opinion of the users. What is relevant for one user may not be relevant for other user [6][5][7].

2.1. The Retrieval Process

A simple IR system have at least three basic processes [1][4][5][7][27][28]: indexing, query formulation and the matching process.

2.1.1. The Indexing Process

The indexing process is an offline process of organizing documents using keywords extracted from the collection in which the end user of the system is not directly involved [4]. It is used to speed up access to desired information from document collection as per user's query [27].

Representing documents by its full set of words is an early way of document representation. This way of representing document can be called full text representation [4]. However, using full text indexing reduces retrieval efficiency and it asks higher computational costs [4]. Therefore, representing documents using keyword terms by

applying preprocessing techniques such as, tokenization, stop word elimination, stemming, and normalization is advisable. Figure 2.1 depicts the logical view of document that slowly swings from full text to a set of index terms.

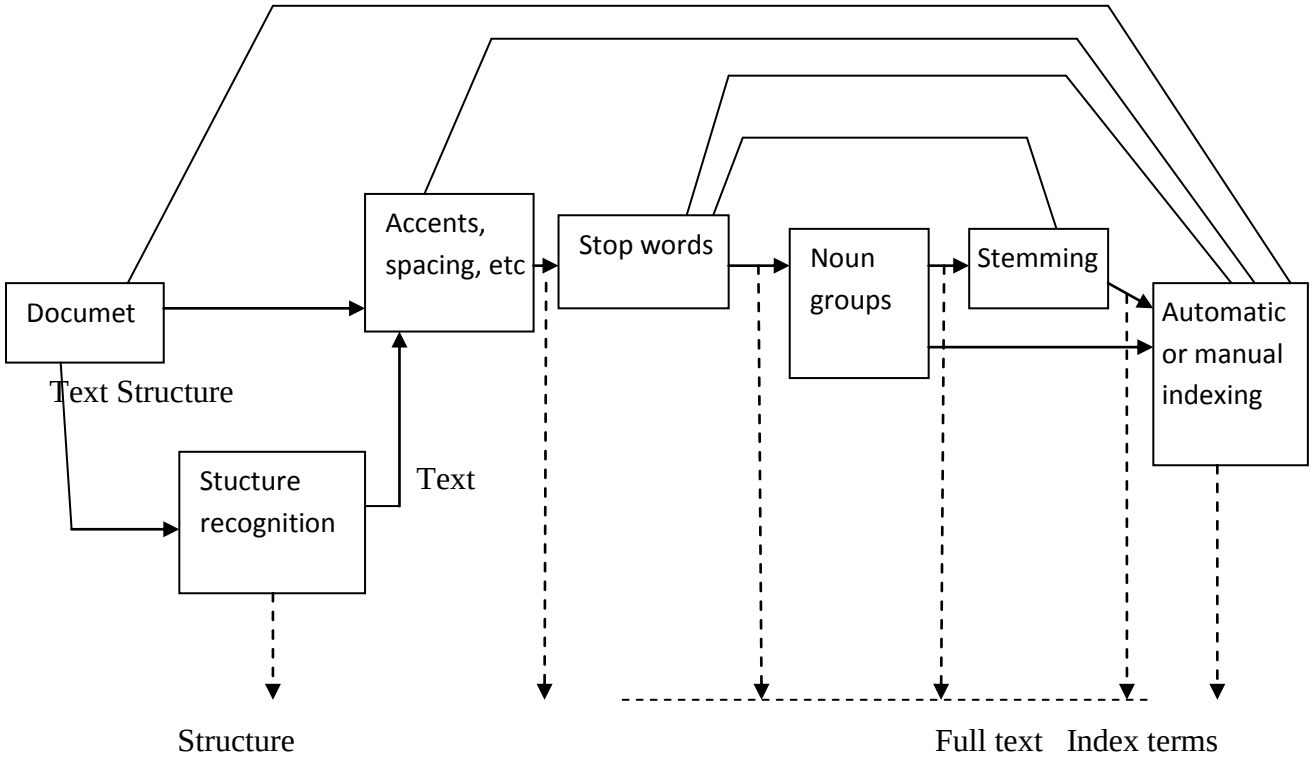

Figure 2.1: Logical view of a document

There are several index structures used for generating index terms [4]; such as, sequential file, inverted file, suffix tree, suffix array and signature file. Sequential file is an indexing structure, which access elements of record in a predetermined ordered sequence. The records are arranged serially one after another in lexicographic order on the value of some key field. Inverted file stores a map from content to its locations in a database file. Suffix tree and suffix array process the suffixes of a given string to allow particular quick implementation string operations. Signature file works based on hash coded. It is a word oriented index structure.

The most popular indexing structure is inverted file, which is also adopted in this research. Inverted file is a mechanism for indexing a text collection so as to make the searching task fast. There are two elements involving in building the inverted file [4]: the vocabulary and

the occurrence. The vocabulary file is the set of index terms in the text collection and it is organized by terms. The vocabulary file stores all of the keywords that appear in any of the documents in lexicographical order and for each word a pointer to posting file. The occurrence contains one record per term, listing all the text locations where the words occurs frequency of each term in a document [4].

2.1.2. Query Processing

Once the document is indexed and ready for retrieval process, the next step is to translate the information need of user into query language provided by the system. This process involves a serious of steps [27]. First, the user specifies his/her information need using the natural language (e.g. English, Amharic, etc.) supported by the IR system. Second, the system transforms the query into logical format by applying text operation, which is also used when the document was indexed. To refine representation of users information need and improve effectiveness of the system query operation will be employed which is discussed in section 2.3. Finally, the query is processed to retrieve relevant documents.

According to Baeza-Yates and Ribeiro-Neto [4], query language specifies regular expressions to search for strings and chapters. One of the query languages used to specify users query expression is Boolean query. It is a query language formulated based on logic concept such as AND, OR, BUT etc. Mostly Boolean query is used by Boolean retrieval model. Other models also integrate Boolean query for instance, probabilistic model. Boolean query describes the information need of the user by relating words with logic operators. For example; the query q_1 OR q_2 selects all documents which satisfy q_1 or q_2. The query q_1 AND q_2 select only documents which satisfy both q_1 and q_2. The query q_1 BUT q_2 selects all documents which satisfy q_1 but not q_2. Sometimes a logic operator NOT is used in place of BUT.

2.1.3. The Matching Process

After documents are logically organized and the user query is processed, the next step is comparing the query against the document representations, which is called, the matching process. The result of the matching is a ranked list of documents according to their

likelihood of relevance [27]. Baeza-Yates and Ribeiro-Neto [4], stated that, one central problem of any information retrieval system is predicting which documents are relevant and which are not. The ranking algorithms are used for such decision. According to Hiemstra [27], the theory behind ranking algorithms is a crucial part of information retrieval system. They attempt to display documents in decreasing order based on their similarity score with the query. Most of the time documents that are considered as relevant to users gets the biggest score and displayed at the top of the retrieved list. Thus, IR models guide the process of matching and ranking relevant documents.

There are different similarity measurement /matching techniques are developed [1] [4][28].such as, Euclidean distance, Cosine similarity measure, dot product, etc. Euclidian distance calculates "the root of square differences between coordinates of a pair of document and query terms". The similarity between vectors for document d_i and query q can be computed as:

$$sim(dj, q) = |dj - q| = \sqrt{\sum_{i=1}^{n} (w_{ij} - w_{iq})^2} \quad \ldots\ldots (2.1)$$

Where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query. Dot product/inner product calculates "the product of the magnitudes of query and document vectors". It is normalized because it favors long documents with a large number of unique terms and it measure only how many terms matched but not how many terms are not matched. Similarity between vectors for the document d_i and query q can be computed as:

$$sim(dj, q) = dj.q = \sum_{i=1}^{n} w_{ij} . w_{iq} \quad \ldots\ldots (2.2)$$

Where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query q. Cosine similarity/normalized inner product calculates "the cosine angle between query and documents vectors by projecting them into a term space". It measures similarity between d_1 and d_2 captured by the cosine of the angle x between them.

$$sim(dj, q) = \frac{\vec{dj}.\vec{q}}{|\vec{dj}|\,|\vec{q}|} = \frac{\sum_{i=1}^{n} w_{ij}.w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2}\sqrt{\sum_{i=1}^{n} w_{iq}^2}} \ldots\ldots (2.3)$$

Where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query q. The denominator involves the lengths of the vectors.

There are also other measures that are widely used in the literature to measure vector similarities. Some of these include the Dice's Coefficient, Jaccard's Coefficient, Euclidian distance and Overlap [4].

2.2. IR Models

Hiemstra [27], stated that, IR model is the mechanism of predicting and explain the need of the user given the query to retrieve relevance documents from the collection. IR models serves as blueprint so as to develop applicable IR system. In addition to that, IR models guide the matching process to retrieve a ranked list of relevant document given a query.

IR models are broadly categorized in to two approaches, which are semantic approach and statistical approach. Semantic approaches models such as, latent semantic indexing and neural network try to work on syntactic and semantic analysis. They attempt to implement some degree of understanding the natural language text that users provide. On the other hand, statistical approaches such as, vector space model and probabilistic model attempts to retrieve documents that are highly ranked in terms of statistical measure [28].

The three most widely used information retrieval model that bases on statistical approaches are [4]: Boolean, vector space, and probabilistic.

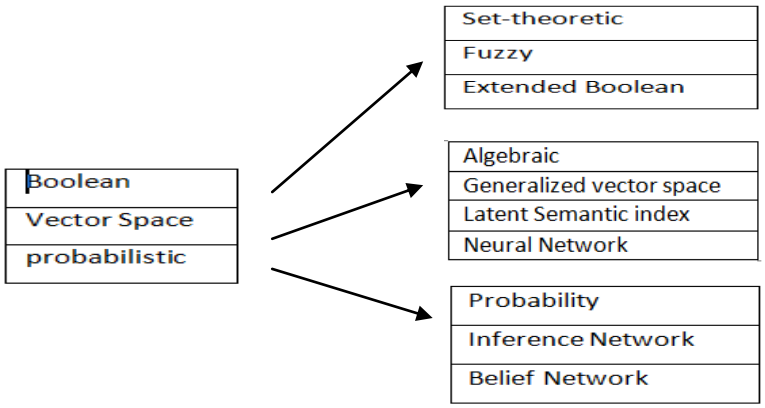

Figure 2.2: Taxonomy of IR models adopted from [4]

Figure 2.2 depict that the three IR models again can be classified based on their mathematical basis. In the boolean model, a set of index terms are used to represent document and query. Therefore, the mathematical basis can be called set theoretic. In the vector space model, a vector in a t-dimensional space is used for representing document and query, therefore, the model is algebraic. In the probabilistic model, probability theory is used for representing document and query. Therefore, the model is probabilistic. Different scholars have been proposed several alternative approaches based on their mathematical basis. For set theoretic models, the alternatives are; the fuzzy and the extended boolean models. For algebraic models, the alternatives are; the generalized vector, the latent semantic indexing, and the neural network. For probabilistic model, the alternatives are; the inference network and the belief network models [4].

2.2.1. Boolean Model

The Boolean model is the first and the simplest model of information retrieval. It works based on set theory and Boolean algebra which allowed users to specify their information need using a combination of classical Boolean operators, ANDs, ORs and NOT [1][2][29]. For instance, query t_1 and query t_2 (t_1 AND t_2) is only satisfied by a given document D_1 if and only if D_1 contains both terms t_1 and t_2. Likewise, the query "t_1 OR t_2" is satisfied by D_1 if and only if it contains t_1 or t_2 or both. On the other hand, the query "t_1 AND NOT t_2" satisfies D_1 if and only if it contains t_1 and does not contain t_2. The model views each document as just a set of words [28].The weight for index terms in Boolean retrieval model are all binary (i.e. W_{ij} E{0, 1}. The similarity of a document dj to the query q is expressed as:

$$sim(dj, q) = \begin{cases} 1 = \ if\ document\ satisfies\ the\ boolean\ query \\ 0 = Otherwise \end{cases}$$

Where, if similarity of document d_j to the query q is equal to 1 (present), the document d_j is relevant. Whereas, if similarity of document d_j to the query q is equal to 0(absent) the document is not relevant [4].

Boolean model have an advantage of giving users a sense of control over the system. It is immediately clear why a document has been retrieved given a query. If the resulting document set is either too small or too big, it directly clear which operators will produce respectively a bigger or smaller set. The other advantage of this model is the clean formalism behind the model and its simplicity [4][28].

However, the model has a number of clear disadvantages. First, most user find it difficult to translate their information need into a Boolean expression. Second, the model has not follow document ranking principle by nature, which means all retrieved documents are considered equally important. Third, the model either retrieves too few or too many documents, or sometimes not retrieves any, which might lead to null result. In addition it's sometimes creates its own problems, such as, misunderstanding and misrepresentation of the users information needs [4][29].

Several attempts has been done so far to make the model effective by developing different alternatives of the model such as fuzzy set model and the extended Boolean model. However, IR communities consider that Boolean system is less effective than ranked retrieval system [8]. Accordingly, a number of models have been proposed for this process; the widely used IR models are the Vector Space Model and the Probabilistic Model [28].

2.2.2. Vector Space Model

Vector space model is one of the commonly used statistical approaches to represent each textual document as a set of terms [27]. In this model, documents and terms are represented as vectors in a k-dimension space where each dimension corresponds to a possible document feature. The word "terms" is not inherent in the model, but terms are typically words and phrases. The values assigned to elements in the vector space describe the degree of importance of term in representing the semantics of the document. Sometimes a given term may receive a different weight in each document in which it occurs. If the term is not appearing in a given document the weight of that term will be zero in that document [4]. For a given document (d_j) the weight of the terms in it can be expressed as the coordinates of d_j in the document space. A document collection

containing a total of documents (‘d’) described by terms (‘t’) is represented as term by document matrix (T x D). Each row of this matrix is a term vector and each column of this matrix is a document vector. The element at row i, column j, is the weight of term j in document I [1].

Term list	Doc1	Doc2	Doc3	Doc4	……	Doc n
Term 1	w11	w12	w13	w14	……	w1n
Term 2	w21	w22	w23	w24	……	w2n
Term 3	w31	w32	w33	w34	……	w3n
Term 4	w41	w42	w43	w44	……	w4n
Term 5	w51	w52	w53	w54	……	w5n
……	…….	……	……	……	……	…….
Term m	wm1	wm2	wm3	wm4		wnm

Table 2.1: Example of Term Document Matrix

In this model, query is interpreted as another document in document space. If the term that is not in the collection but appear in the query, this will add additional dimension in the document space [4].

The weight of terms in the documents or in the queries assigned by using term frequency (TF) and inverse document frequency (TF*IDF) scheme which are the most successful and widely used automatic generation of weights [4]. The term frequency is the frequency of occurrence of the given term within the given document. This scheme attempts to measure the degree of importance of the term within the given document. Inverse document frequency (idf) is a frequency of a given term within an entire collection of documents. It attempts to measure how widely the term is distributed over the given collection.

The similarity of document and query in vector space model is determined by correlation between the vectors dj and vector q. The correlation is quantified by the associative coefficients based on the inner product(dot product) of the document and query vector,

where documents whose vectors are close to the query vector are more relevant to the query than documents whose vectors are far from the query vector [4].

As discussed in section 2.1.3 there are different similarity measurements. However, the most widely used by vector space model and popular similarity measure is the cosine coefficient, which measures the angle between the document vector and the query vector [4].

According to Baeza-Yates and Ribeiro-Neto[4], Vector space model (VSM) have several advantages. First, it improves retrieval performance using its term-weighting scheme. Second, the partial matching strategy of VSM allows retrieval of document approximate the query condition. Third, it sort and rank the documents according to their degree of similarity to the query using cosine similarity measurement. Finally, it is simple to implement and fast.

However, the model has also several disadvantages. It considers terms as unrelated objects in the semantic space. This means, if no common words are shared between the query and documents in text collection, the similarity value will be zero and no document will be retrieved. This is usually happened when users and authors prefer to use different words which have the same meaning [4]. Vector space model is not attempt to define uncertainty in IR system and its ranked answers sets is difficult to improve upon without the integration of query expansion and reformulation modules to the model [7].

Literature suggested that one of the alternative modeling paradigm, which is capable of solving the above mentioned problem of vector space model is probabilistic IR model [7][30].

2.2.3. Probabilistic Model

In information retrieval process, user first start with information needs which is then translated into query representation. On the other side, documents are also translated into document representation. Finally, the system attempts to determine the relevance of the document to the information need of the users. In IR models such as, Boolean and Vector

space model, given a query and document representation matching is done without considering the semantic relationship between query and documents. IR systems build upon those models has an uncertain guess of whether a document has content relevant to the information need. However probabilistic theory provides a principled foundation for such reasoning under uncertainty [1][7].

Maron and Kuhns was the first to introduce ranking by the probability of relevance, soon after Stephen Robertson brought new idea called, probabilistic ranking principle in 1977 [2][4].

Probabilistic information retrieval is the estimation of the probability of relevance that a document d_i will be judged relevant by the user with respect to query q. which is expressed as, $P(R|q,d_i)$, where, R is the set of relevant document. Typically, in probabilistic model, based on the query of user the documents are divided in to two parts. The first contain relevant documents and the second contain non-relevant (irrelevant) documents. However, the probability of any document is relevant or irrelevant with respect to user query is initially unknown. Therefore, the probabilistic model needs to guess at the beginning of searching process. The user then observe the first retrieved documents and gives feedback for the system by selecting relevant documents as relevant and irrelevant documents as irrelevant. By collecting relevance feedback data from a few documents, the model then can be applied in order to estimate the probability of relevance for the remaining documents in the collection. This process iteratively applied to improve the performance of the system so as to retrieve relevant documents which satisfies users need [4].

In probabilistic model, the order in which documents are presented to the user is to rank documents by their estimated probability of relevance with respect to the user information need. The principle behind this assumption is called, probability ranking principle (PRP) [1][4]. Probability ranking principle asserts that [12] :

> If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately

as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data

Several retrieval models have been developed, which has a probabilistic basis. The mostly used and recent developed methods of probabilistic model are, Binary Independent Model, Inference Network Model and Belief Network Model [4].

- Binary Independent model

The classical Binary independence model (BIM) was introduced in 1976 by Roberston and Sparck Jones[4]. According to Greengrass [28], the model has been called 'binary independence' because it has been assumed that to arrive at the model one must make the simplifying assumption that the document properties that serve as clues to relevance are independent of each other in both the set of relevant documents and the set of non-relevant documents.

In BIM, binary is equivalent to Boolean; queries and documents are represented as binary incidence vectors of terms. $D=\{d_1,d_2,\ldots,d_n\}$ where, $d_i=1$ if term i is present in document d and $d_i=0$ if term i is not present in document d. Moreover, query q represented by the incidence vector$\vec{q}$. As expressed above, in BIM model, the probability of P(R|d,q) that a document is relevant through the probability in terms of term incidence vectors $P(R|\vec{x},\vec{q})$ in both document and query.

- Bayesian Networks Model

Bayesian networks model was introduced by Turtle and Croft [1] as information retrieval model on the basis of probabilistic theory. Bayesian networks use a form of probabilistic graphical model. They use directed acyclic graphs (DAGs) to show probabilistic dependencies between variables, in which the nodes represent random variables, the arcs depict causal relationships between these variables. They have child and parent causal relationship, which is represented by linking each parent node to the child node. Only the root nodes are without parents [4].

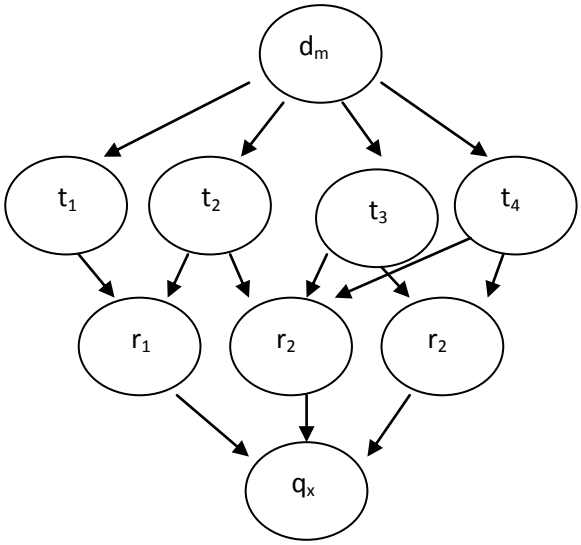

Figure 2.3: Example of Bayesian Network model adopted from [7]

Figure 2.3 depicts the graphical representation of Bayesian Network. According to Ruggeri [31], the network has two sets. The first set is the nodes and the second set is the directed edges. Arrows draw the edges between nodes that represent direct reliance among the variables. For instance, an edge from dm to t_1 represents a arithmetical reliance between the corresponding variables. Therefore, the value taken by variable t_1 depends on the value taken by variable d_m. In other word, the node dm is parent node and t_1 is the child node. Even if the edge represents direct causal relationship among the nodes, the reasoning process functions by propagating information in any path.

There are two models for information retrieval based on Bayesian networks. The first model is called inference network and the second model is called belief network.

- Bayesian Inference Network Model

According to Baeza-Yates amd Ribeiro-Neto [4], there are two traditional schools of thought in probability which are based on the frequentist view and the epistemologist view. The frequentist views probability as a statistical notion related to the laws of chance. The epistemologist views probability as a degree of belief whose specification might be devoid of statistical experimentation.

Bayesian Inference Network model is built up on epistemologist view of the information retrieval problem. It associates random variables with the index terms, the documents and the user queries. The model computes P(I|D), the probability that a user's information need (I) is satisfied given a particular document (D). This probability can be computed separately for each document in a collection. The documents can then be ranked by probability, from highest probability of satisfying the user's need to lowest [4].

- Bayesian Belief Network Model

Bayesian belief network is the use of Bayesian calculus to determine the probabilities of each node from the predetermined conditional and prior probabilities [32]. As Baeza-Yates and Ribeiro-Neto [4], stated in Bayesian belief network the users query q is modeled as a network node to its associated random variable. Whenever q completely covers the concept space C the variable is set to 1. Therefore belief network computes the probability of q, (i.e. P(q)) by the degree of coverage of the space C by q.

Document d_j is modeled as network node to its associated binary random variable. If d_j completely covers the concept space C the variable set to 1. To compute the probability of d_j ($P(d_j)$), compute the degree of coverage of the space C by d_j. In belief network documents and the user query modeled as subsets of index terms. Each subset is interpreted as concept in the concept space C [4].

The ranking principle in belief network expressed as, the degree of coverage provided to the concept d_j by the concept q. $P\{d_j|q)$ is adopted as the rank of the document d_j with respect to the query q[4].

In general, probabilistic models attempt to capture the information retrieval problem within a probabilistic framework. Unfortunately, the probabilistic model has got its own drawbacks. First, the probability theory analysis takes much more time and efforts, and it offer unnecessary theoretical burden on the researcher. Second, probabilistic model need to guess the initial separation of documents into relevant and non-relevant sets. Third, the model does not take into account the frequency with which an index term occurs inside a document. In other word, all weights are binary [4][28].

However, probabilistic model have several potential advantages [4][28]. The first, advantage is the expectation of retrieval effectiveness that is near to optimal relative to the evidence used is high. Second, it has less reliance on traditional trial and error retrieval experiments. Third, each document's probability of relevance estimate can be reported to the user in ranked output. It would presumably be easier for most users to understand and base their stopping behavior (i.e., when they stop looking at lower ranking documents).

2.3. Query Operation

Query operation is the process of representing user information need [27]. Most of users without deep knowledge of documents representation and searching environment spent their much valuable time on reformulating their queries so as to get relevant document that satisfies their need. The first query formulation is considered as initial try and the second reformulation is considered as attempt to get better and additional relevant documents. There are two ways in query reformulation [4]; query expansion and term reweighting. Query expansion is the process of adding terms which share similar or related meaning with the query terms. Term reweighting on the other hand, involves attaching various weights to query terms, so that documents carrying g a query term with the highest weight can be superior [4][27].

For the query operations to happen relevance feedback and query reformulation is necessary. Relevance feedback enables to identify relevant document retrieved for the users query and query reformulation enables to expand the original query with the new terms and reweight the terms in the expanded query so as to retrieve relevant documents which satisfy user's information need. There are various relevance feedback and query reformulation mechanisms as discussed below.

2.3.1. Relevance Feedback

Relevance feedback is a mechanism of engaging users or system in retrieval process so as to improve the final result of the IR system. There are two relevance feedback mechanisms [1][4][28][33], user relevance feedback and pseudo relevance feedback. User relevance

feedback is used to improve the final result of the IR system by involving the users in relevance feedback during the retrieval process. The procedures followed in user relevance feedback are the following. First, the user provides a query based on which the IR system returns initial relevant documents. Second, the user marks some returned documents as relevant or non-relevant. Third, the system computes a better representation of the information need based on the user feedback. Finally, the system displays a revised set of retrieval results.

Pseudo-relevance feedback is a method that eliminates the need of user's involvement. In other word, it automates the manual feedback of the user so as to get the improved result. This method works by assuming the top k ranked documents are relevant and automatically process the relevance feedback search component [33][5].

2.3.2. Query Reformulation

Query reformulation is a mechanism used to enhance the performance of the retrieval system by using two different methods called query expansion and term reweighting [4].

Query expansion technique is a process of adding a new term from relevant documents. There are two types of query expansion strategies [4]: global analysis and local analysis. Global analysis strategy examines all documents in the collection so as to expand query. Local analysis examine only documents retrieved automatically for a given query q to determine query expansion.

Term reweighting technique is a process of adjusting the weight of the term based on the users or system relevance judgment. There are different techniques of term reweighting [1]. Rocchio algorithm, probabilistic term reweighting etc.

Rocchio algorithm in one the most widely used algorithm designed for vector space model [34]. It finds a query vector which increases similarity with relevant document while decreases similarity with non-relevant documents [1].

Probabilistic reweighting technique is designed for probabilistic model. It attempts to predict the probability that a given document will be relevant to a given query. The similarity of document dj to a query q can be expressed as [4]:

$$sim(d_{j,}q) \propto \sum_{i=1}^{t} w_i\,, q.w_i, j\ (log\frac{P(k_i|R)}{1-P(k_i|R)} + log\frac{1-P(k_i|\bar{R}}{P(k_i|\overline{\bar{R}})} \ldots\ldots\ldots (2.4)$$

Where, $P(k_i|R)$ express the probability of getting term k_i in the relevant documents of R and $P(k_i|\bar{R})$ represent the probability of getting term k_i in the non-relevant documents of $\bar{R}$. However, initially equation 2.4 is not used because of the probabilities of $P(k_i|R)$ and $P(k_i|\bar{R})$ are unknown. Therefore, two assumptions are made for the initial search in which there are no any retrieved documents. The first assumption is $P(k_i|R)$ is constant for all terms k_i (usually 0.5) and the second assumption is, the term probability distribution $P(k_i|\bar{R})$ can be approximated by the distribution in the whole collection [4]. The two assumptions expressed as:

$$P(k_i|R) = 0.5 \quad and \ \ P(ki|\bar{R} = \frac{n_i}{N} \ldots\ldots\ldots (2.5)$$

Where, n_i stands for the number of documents in the collection which contain the term k_i. Substituting into equation 2.4, it becomes [4];

$$sim_{initial}\left(d_j\,, q\right) = \sum_{i}^{t} w_{iq}\ .w_{ij}\ log\frac{N-n_i}{n_i} \ldots\ldots (2.6)$$

After initially documents are retrieved the user marks the retrieved documents as relevant and non-relevant. Then the information is used so as to evaluate the probability of $P(k_i|R)$ and $P(k_i|\bar{R})$. The probabilities $P(k_i|R)$ and $P(k_i|\bar{R})$ can be approximated by [4]:

$$P(ki|R) = \frac{|Dr,i|}{|Dr|}\ and\ P(ki|\bar{R}) = \frac{ni-|Dr,i|}{N=|Dr|} \ldots\ldots (2.7)$$

Where, D_r is the set of relevant retrieved documents as the user judgment and $D_{r,j}$ is the subset of D_r composed of the documents which contain the term k_i. However, equation 2.7,

have a problem for some small values of $|D_r|$ and $|D_{r,i}|$. Thus, a 0.5 adjustment factor is added to the estimation of $P(k_i|R)$ and $P(k_i|\bar{R})$. This will give [4];

$$P(k_i|R) = \frac{|Dr,i| + 0.5}{|Dr| + 0.5} \quad and \; P(ki|\bar{R}) = \frac{n_i - |Dr,i| + 0.5}{N - |Dr| + 0.5} \ldots\ldots\ldots (2.8)$$

The adjustment fact made at equation 2.9 may provide inadequate estimation in some cases. In this case, alternative adjustment factors have been formulated. For instance, replacing adjustment factor 0.5 by $\frac{n_i}{N}$[4]:

$$P(k_i|R) = \frac{|Dr,i| + \frac{n_i}{N}}{|Dr| + 0.5} \quad and \; P(ki|\bar{R}) = \frac{n_i - |Dr,i| + \frac{n_i}{N}}{N - |Dr| + 0.5} \ldots\ldots\ldots (2.9)$$

2.4. IR System Evaluation

Evaluation of information retrieval system is done before the system is implemented [4]. Several reasons are stated why evaluation is needed in IR. For instance, evaluation provides the ability to [1];

- Validate and verify the system to check whether the system is right or not.
- Measure which one is the better system than the other one.
- Measure how good the IR system works.
- Identify techniques/ algorithms that work well and do not work.
- Identify specific components of techniques or algorithms that work better.
- Provide future direction for further studies

In IR system evaluation, the two common measure of system performance are efficiency and effectiveness. Efficiency is the time and space used by the system in retrieval process. To be called efficient system, the retrieval and indexing time of the system should be shorter and the space used in indexing file should be smaller. On the other hand, effectiveness refers how much the system meets its designed objective. To be called

effective system the system should be capable of retrieving relevant documents from the collection and the system should retrieved documents that satisfy users need [4].

2.5. The Amharic Writing system and its Features

Ethiopia has several languages, which are spoken by different nations and nationalities. Among the languages, Amharic or "አማርኛ" is the national language of Ethiopia until 1995. Following the declaration of constitution of Ethiopian federal democratic government, it becomes the working/official language. Because of its wide application, Amharic documents are become increasing in both hard copy and electronic forms.

Since this research is conducted considering Amharic documents, it is important to investigate the potential features of the language that have the capability of representing the contents of the documents that demands one to understand the characteristics of the language.

2.5.1. History of Amharic Writing System

The origin of Amharic writing system traced back when Semitic scripts were flourished in the Middle East before three thousand years ago [35], as depicted in figure 2.4.

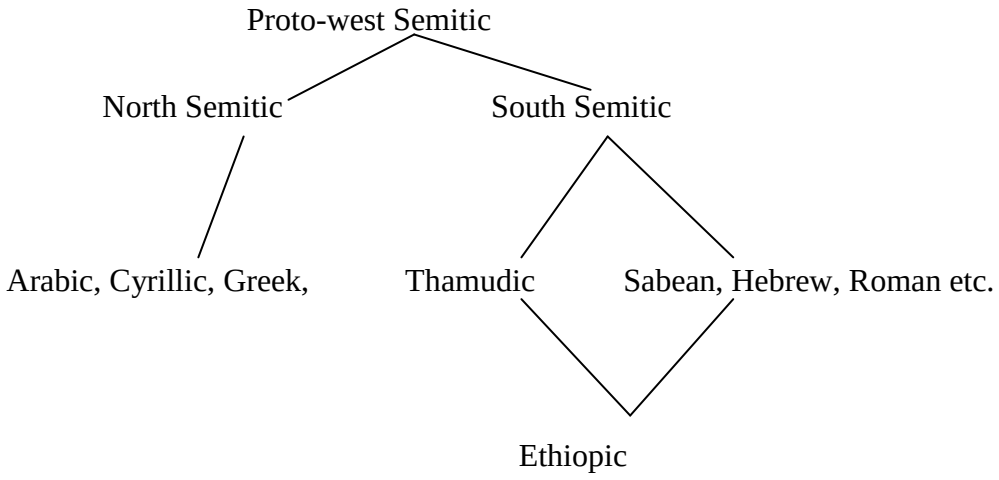

Figure 2.4: Origins of Ethiopic a family tree model adopted from [35]

The model shows two main branches downward from Proto-West Semitic; the North Semitic and South Semitic. The children in North Semitic branch are Hebrew, Arabic, Greek, Roman and Cyrillic. The South Semitic side produces Ethiopic via the Sabean system, which is speculatively dated as emerging in the 11th and 10th centuries BC. The three Semitic languages which are only found in Ethiopia and Eritrea are Geez or Ge'ez, Amharic and Tigrinya which are used in a representation for Ethiopic system [35].

Geez play a significant role in the development and expansion of Amharic language and writing system. Several religious texts, such as Bible, translations of Arabic Christian texts from Egypt and literatures such as the 'qine' (ቅኔ) and poems are all written in Geez. The emergence and expansion of Geez inscriptions in the Ethiopic script traced back to the 4th century AD, when Geez was the language of the empire of Aksum North Ethiopia. Even if the use of geez is limited to Orthodox Church, it is still a source for the coining of Ethiopian literary and technical terms [35].

The Sabean script is attributed as the source of the Geez script, likewise, the Geez writing system attributed as the source of Amharic writing system. In the early age, Amharic has been the language of the politically dominant ethnic group in Ethiopia and it is now used in every part of the country.

2.5.2. Amharic Alphabet

The Amharic language alphabet consists 33 basic characters having 7 different forms for each consonant vowel combination [35]. The basic character occurs in one form and the remaining six different forms are build based on the basic characters having a little modification form. For example, the seven orders of the consonants of ሀ (hä) and ዘ (ze) expressed as follows;

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
hä	hu	hi	hä	he	h	ho
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
zä	zu	zi	zä	ze	z	zo

Generally, there are 231 characters. According to Bloor [35], Amharic is a syllabic language in which a character is used to represent a phoneme, which is a combination of a vowel and a consonant. In Amharic language, there are additional nearly forty characters which contains special feature representing labialization. For instance, ሏ (lua) from ለ (lä) ሟ(mua) from መ (mä) and ሯ (rua) fromረ (rä).

As every language has its own way of representing numbers, despite the absence of representing number zero (‘0’), Amharic has also its own way of representing numbers. The Amharic numbering system consists of twenty (20) single characters derived from Greek letters by modifying the characters into Amharic character form. The symbols are modified by adding a horizontal stroke above and below. The numbers consists of a single character for one to ten, for multiples of ten (twenty to ninety), hundred and thousand [36].

The Amharic writing system also consists around 17 (seventy) punctuation marks. The most commonly used punctuation marks are [36];

- Two dots (‘:’) called “ሁለት ነጥብ”, which is similar to a colon used to separate words, however, nowadays the influence of foreign languages lead to the use of blank space instead.
- Four dots (‘::’) called “አራት ነጥብ”, which is used as full stop.
- Two dots with horizontal stroke line over them (‘፣’) called “ነጠላ ሰረዝ” which is used as comma.
- Two dots with horizontal stroke above and below them (‘፤’) called “ድርብ ሰረዝ”, which is used as semi-colon. The other punctuations are mostly borrowed symbols such as, ?,!,/.\, “,’ etc.

2.5.3. Features and Challenges of the Amharic Writing System in Designing Amharic Text retrieval

Several characteristics of Amharic writing system have been investigated in many literatures. Some of the characteristics are [6] [15] [35][36][37];

- It has different forms of writing compound nouns and abbreviations.

- It has different ways of writing the same word.
- It has consonants with the same sound.
- It is reach in word variants

These different features of the language cause several challenges in designing Amharic information retrieval system.

- Different forms of wring compound nouns:

Amharic writing system has two ways of writing compound nouns without affecting the meaning of the word. In other word, two words are sometimes written as one word or one word might be written as two words. There is no similar convention that specifies how these words are written. For instance, the word "bête kiristian" which means church might be written as one word "ቤተክርስትያን"or two words "ቤተ ክርስትያን" which does not make any difference in the meaning of the word. And also the word "bête mengist" which means palace might be written as one word "ቤተመንግስት" or two words "ቤተ መንግስት". Other compound nouns such as, "መአድ ቤት","ብርድልብስ","ብረትድስት", "ቤተ መዘክር", "ወጥ ቤት", etc. which means dining room, blanket, cooking pot, museum, kitchen respectively, can be written as one word or two words. This situation makes an IR system difficult to differentiate those words.

- Different forms of wring abbreviation

The other characteristic of Amharic writing system is it has different forms of writing abbreviation. Here also, there is no the same convention for using the same form of abbreviations. For instance, the phrase "አዲስአበባ" which means the city Addis Ababa might be abbreviated as; "አአ","አ/አ","አ.አ". Therefore, there should be a mechanism to handle these problems in retrieving Amharic documents.

- Different ways of writing the same word and consonants with the same sound

In Amharic writing system there are characters or symbols which have the same sound during reading. For example, the characters "ሀ, ሐ, ኀ, አ and ዐ" produces the same sound

with characters "ሃ, ሓ, ኃ, ኣ and ዓ" respectively during reading. On the other hand, there are also alphabets that share the same sound, such as, ሀ, ሐ, and ኀ, ሰ and ሠ, አ and ዓ, and ጸ and ፀ. As a result, words which have the same meaning may have different spelling structure. For example, the word "አለም" which means world can be written differently as; "አለም", "ዓለም", "ዐለም", "ኣለም".

In IR system since it only match the character in query words to check whether a word found in a document has the same meaning as in the query, it consider the word 'አለም' and 'ዓለም' are dissimilar.

Other problems of Amharic writing system are the existence of several synonym and polysemous words. Synonym words are the words with the same meaning but written differently. For instance, the word "ኮፍያ" which means 'hat' can be called differently in different region, such as, "ባርኔጣ", "ቆብ" etc. Polysemous words are words that have different meaning by stressing some characters in the word while reading. For instance, the word "ገና" which means 'Christmas day' can have another meaning "not yet" when 'n' is stressed [15].

- Reach in word variants

Amharic language is reach in word variants. One term may have different variant terms. For instance, a word "ሰጠ" meaning 'to give' may have several variants such as; 'ሰጡ', 'ሰጥ', 'ሰጠች', 'ሰጠን', 'ሰጣቸዉ', 'ሰጠከው' 'ሰጠካት', 'ሰጠላቸዉ', 'ሰጠላት', 'ሰጠለት', 'ሰጠሀቸው', etc. Such variant of words subject a system to consider those words as morphologically different, but the same in meaning.

In general, such problems makes designing information retrieval system for Amharic language difficult and challenging, and it needs a lots of works expected from experts so as to develop a better IR system considering the language problems discussed above.

2.6. Review of Related Works

Probabilistic information retrieval has been studied for decades, which results several body of literature on the topic.

2.6.1. Global Probabilistic IR Research

Probabilistic retrieval model computes the probability of relevance of a document for a query [30].Given a query, a document is assumed either relevant or non-relevant. However, the system not knows whether a document is relevant or not. Therefore, it as to estimate the relevancy of the document based on probabilistic methods [4].For instance, let D and Q represents document and query, respectively. Let R represent a binary random variable that indicates whether D is relevant to Q or not. R takes two values which are relevant (r) and non-relevant ($\bar{r}$). Thus, to estimate the probability of relevance (i.e. p(R=r |D,Q) or p(R= $\bar{r}$|D,Q)), several ways of probabilistic model have been developed.

Robertson and Sparck Jones [11], developed the well-known classical probabilistic model called The Binary Independence Retrieval (BIR) model so as to estimate the probability of relevance (i.e. p(R=r |D,Q) or p(R= $\bar{r}$|D,Q)).

Robertson and Sparck Jones work was based on two basic assumptions: the independent assumption and ordering principle. The independent assumption assumed that, terms are distributed independently and randomly. Specifically, independence assumption one (I1) stated that, "the distribution of terms in relevant documents is independent and their distribution in all documents is independent". The second independent assumption (I2) stated that, "the distribution of terms in relevant documents is independent and their distribution in non-relevant documents is in-dependent". The ordering principle states, documents should be ordered by their probable relevance to the query. Specifically, the ordering principle one (O1) stated that, "probable relevance is based only on the presence of search terms in documents" and the ordering principle (O2) stated that, "probable relevance is based on both the presence of search terms in documents and their absence from documents". Five relevance weighting functions (i.e. F0-F4), which are derived from

a formal probability theory of relevance weighting was investigated. The experiment conducted using manually indexed 1400 document collection written in English language. Table 2.2 shows the retrieval performance registered.

Weighting function	Precision	Recall
F0	29%	90%
F1	50%	90%
F2	60%	90%
F3	66%	80%
F4	70%	80%

Table 2.2: Experimental result of Robertson and Sparck Jones work

As the result indicates, the ordering principle O2 is correct and O1 is incorrect. The performance also shows that F3 and F4 performed consistently better than F1 and F2. On the other hand, the experiment depict that, the performance of the system improves when information about the occurrences of terms in relevant documents is added to information about their simple document incidence. Specifically, relevance weights give a better performance than simple term matching [11].

Several attempts have been made to improve the binary independent representation. For instance, Rijsbergen [12] try to improve the binary independence model by capturing some term dependence as defined by a minimum spanning tree weighted by average mutual information. The result of the research shows that, the dependency model achieved significant performance over independence model. However, the experiment was done only on very small document collections written in English language and the estimation of many more parameters is a problem in practice.

The application of probabilistic retrieval model is not only limited to English language. Several works have been done in different languages such as, Chinese [19] and Russian [20] by using different probabilistic models.

Xiangji and Stephen [19] tried to investigate the effect of probabilistic approach on Chinese text retrieval, the difference between word approach and character approach and the effect of different phrase weighting functions and of varying their parameters. The

weighting function used is based on the basic weighting function of Rovertson-Sparck Jones weight [11], which approximates to inverse collection frequency when there is no relevance information. For evaluation of the system in Test Retrieval Conference (TREC 6), three automatic runs (i.e. run1, run2, run3) were submitted. Run1 and run3 are for word indexing approach and run2 is for character indexing approach. Table 2.3 depicts the result of the experiment.

	Average precision	R precision	Precision for 100 documents
Run1	48%	51%	48%
Run2	50%	52%	49%
Run3	49%	51%	48%

Table 2.3: Experimental result of Xiangji and Stephen work

The overall result suggested that a new way of phrase weighting schemes for Chinese should be developed [18].

Ljiljana and Jacques [20] attempted to use probabilistic information retrieval model so as to evaluate several stemming and indexing approaches for Russian language. The probabilistic methods used in this research including Okapi, Divergence form Randomness (DFR) and statistical language model (SLM). To evaluate the performance of the IR models used in this research, mean average precision (MAP) were used. The MAP values were computed by TREC_EVAL software using a 1000 retrieved documents. To determine the performance of the model two sided t-test with a significance level of α= 5% were used. The experiment result shows that, the mean average precision for Okapi model without stemming is 0.0881. The mean average precision for DFR model without stemming is 0.0879. The mean average precision for SLM model without stemming is 0.0964. This work was mainly conducted for suggesting appropriate stemming strategy. The experiment shows that stemming procedure improves retrieval effectiveness especially in the case of the collection containing short document [20].

2.6.2. Local IR System Works

Information explosion in electronic text written in Amharic language caused an increasing need of designing effective and efficient IR system for Amharic texts. A number of IR systems developed so far for retrieving Amharic texts.

Betelihem [14], developed n-gram-based automatic indexing for Amharic text retrieval. This research mainly conducted to solve the problem of not having standard stemming procedure and stop word list for Amharic language. Thus, to solve such problem a method called n-gram automatic indexing has been developed. The experiment conducted using 100 Amharic news articles, which are collected form Walta Information Center. Vector space model was used for the purpose of document representation. The results showed that tri-grams indexing methods performed 5% and 88% precision and recall respectively. One of the recommendations forwarded in this research is using combinations of different n-grams for indexing.

Tewodros [15] developed Amharic text retrieval using latent semantic indexing (LSI) strategy with singular value decomposition. His work mainly focuses on indexing process, to solve the problem found in exact term matching retrieval system. In exact term matching system, the term in the user's query may not appear in a relevant document since there are many ways to express the same concept. Moreover, many words can have also more than one meaning. The result leads to the retrieval system that misses relevant documents and retrieve non-relevant ones. Thus, LSI indexing technique were used which partially handle the problems of exact term matching by organizing terms and document into a semantic structure. 25 queries and 206 Amharic document collections were used for experimentation. The result of the experiment shows that, the recall-precision graph of the LSI method was above standard vector method. The average precision registered by the system was 71%. One of the recommendations given by this research is to design a model that consider relevance feedback that gives information for the system about which documents are relevant and which are not relevant so as to enhance the performance of the system.

Tessema and Solomon [16], designed and implement Amharic search engine, which retrieve web documents written in Amharic language. Even if general search engine such as Google, Yahoo have the way to accept Amharic query and retrieve relevant document, they simply match patterns without considering the feature of Amharic language that affect the retrieval performance. In this research, a complete language specific process has been done, such as, crawler, indexer and query engine component. The experiment result shows, the average precision and recall of 99% and 52% respectively. In future work, the need for considering additional features of Amharic text was recommended.

Alemayehu [38] and Abey [6], developed Amharic IR system by integrating query expansion techniques to enhance the performance of the system. The research is conducted to solve the problem of polysemous and synonymous exists in Amharic text, which decreases the performance of the IR system. Several query expansion techniques were tested. Such as, global analysis, local analysis, bi-gram analysis, bi-gram based thesaurus and statistical co-occurrence method. The result of the experiment show statistical co-occurrence method registered better performance than other methods by scoring 73% recall and 53 precision. One of the recommendations forwarded is to design ontology based query expansion in order to control expanding terms that are polysemous by themselves.

In conclusion, even if several IR systems have been developed in the last decade, most of works are attempt to design an Amharic IR system using vector space model. However, the use of vector space model may not control uncertainty nature of IR system. Thus in this study an attempt is made to develop probabilistic based Amharic IR system to enhance the performance of the Amharic IR system.

CHAPTER THREE

DESIGNING PROBABILISTIC IR SYSTEM

In designing probabilistic Amharic information retrieval system, the two major components which are indexing and searching are used [4]. Indexing is an offline process used to facilitate access to preferred information from document corpus accurately and efficiently as per users query. Searching is an offline process used to scan document collection so as to find relevant documents that satisfy users need .

Figure 3.1 depicts the probabilistic based IR system architecture designed and implemented in this research. It indicates that, the IR system has online (Searching) and offline (Indexing) processes. During the offline process, Amharic documents are organized using inverted indexing structure. To ease the indexing task, text operations are applied on the text of the documents in order to transform them in to their logical representation. The documents are indexed and the index is used to execute the search. The searching task is an online process that accepts users query. The query is processed to identify terms using which searching is done to identify and retrieve relevant documents. After ranked documents are retrieved, the users provide feedback that can be used to refine the query and restart the search for improved results [4].

3.1. Amharic Document Indexing

In information retrieval, searching is possible or efficient when the text is small. However, in large databases searching will take much more time and space unless indexing structure is used to organize documents. Therefore, constructing and maintaining indexing on large database is not optional.

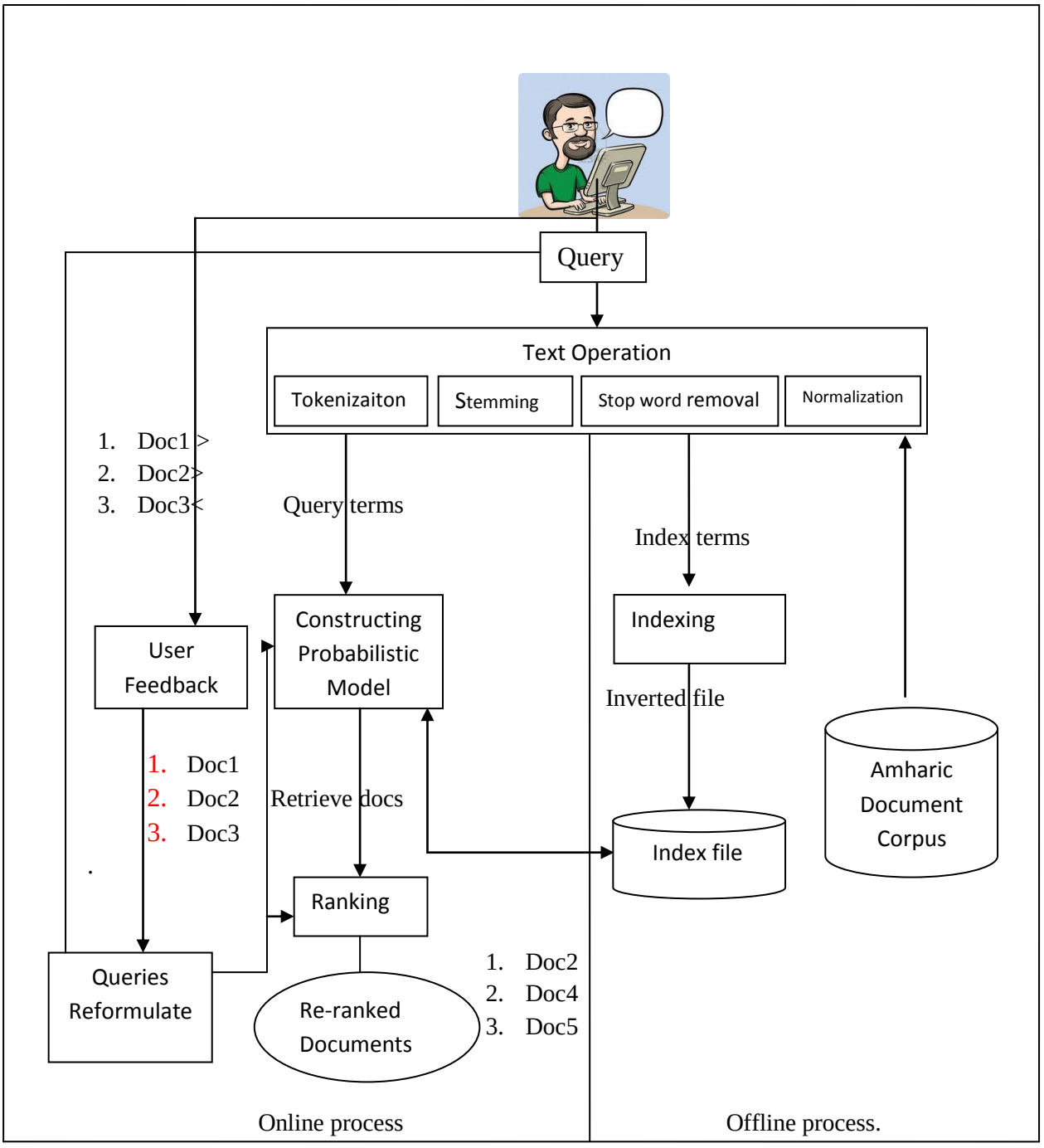

Figure 3.1: Probabilistic based IR system architecture

Several works has been done so as to index the document collection/corpus used in this research. These tasks include; stemming, stop word removal and normalization.

3.1.1. Word Stemming

In Amharic writing system, words are morphologically variant. Morphological variant words have similar semantic interpretations. In IR system, those words considered as equivalent words [1]. Therefore, words have to be reduced to their root using stemming technique. On the other hand, stemming also used to reduces the dictionary size (i.e. the number of distinct terms used in representing a set of documents). The smaller the dictionary size results the smaller storage space and processing time required.

Stemming techniques are language dependent. Therefore, every language needs to have language specific stemming technique. In Amharic text, there are many word variants/affixes [36]. To conflate them into stem word, stemming technique/ algorithm developed by Nega [39] were used. He developed the stemmer that involves the removal of both prefixes and suffixes and also consider letter inconsistency and reiterative verb forms.

According to Nega [39], a stemmer that stem words without consideration of remaining stem, which removes words that are similar to prefix and suffix list but that are not actually affixes is called context-free stemmer. For instance, English word ‘regular’ and ‘metal’ will be ‘gular’ and ‘met’ respectively if context-free stemmer removes ‘re’ and ‘al’. Similarly, weak result will appear if such technique is applied for Amharic language. For example, for Amharic word ‘ከለከለ’ which means forbid, ‘ከለከላቸዉ’ which means forbid them and ‘ከለከላት’ which means forbid her, if we remove string ከ ‘ke’ which is found in prefix list, the result will be ለከለ which does not have any relation with the correct stem. Therefore, context-sensitive approach should be used with affix removal being controlled by two action codes and five conditions.

The action codes are:

A1: do not perform affix removal

A2: remove a shorter form affix

The five conditions are:

C1: characters following the assumed affix (i.e. if the assumed affix is followed by:). This is used to cater for words where vowel elision occurs at morpheme boundaries in the case of prefixing, and to reduce the removal of terminal consonants in the case of suffixing.

C2: characters preceding the assumed affix (i.e. if the assumed affix is preceded by :). This is used to avoid the removal of terminal consonants that are not genuine suffixes.

C3: if the assumed affix is part of an identified consonants structure this is used to avoid the removal of non-genuine affixes from specific words and their variants.

C4: A minimum stem-length condition for the specified affix. The stem length includes both vowels and consonants and the condition is used to reduce over stemming.

C5: A combination of the above.

Table 3.1 shows the prefix and suffix lists used in this research, which is adopted from Nega [38]

Prefix lists					Suffix lists				
የ	ስለ	የሚ	እየ	ስለሚ	ች	ኝ	ችን	ቸው	ዊት
እያ	እንደ	አል	አለ	በ	ና	ዎች	ኛ	ዎቻቸው ም	ውም
ለ	ከ	ይ	እንዳ ይ	ሲ	ው	ዎች ም	ውያን	ዎቹ	ናቸው
እንዲ	እስከ	ከነ	እን	እነ	ባቸው	ዊያን	ነት	ያዊ	ን
					ት	ሉ	ችው	ዊ	ዊቷ
					ቹ ን	ዬ	ዎ	ህ	ሽ
					ዋ	ሁ	ለት	ላት	ላቸው
					ላችሁ	በት	ባት	ባቸው	ባችሁ
					ቱ	ሽ	ይቱ	የው	ኞች

Table 3.1: Prefix and Suffix list

Based on the prefix and suffix list and the context sensitive rule discussed in section 3.1.1, words are conflated to their stem using the algorithm 3.1 and algorithm 3.2. In this research prefix is removed prior to suffix.

Step 1: Get WORD and count the number of radicals (or consonants)

Step 2: IF number of radicals < 3 TJEN stop and return WORD

Step 3: IF length (Word) < -(MAXPL + 2) THEN assign the first length (WORD) -2

Characters to a temporary prefix, TEMPP

ELSE assign the first MAXPL characters to TEMPP (where MAXPL is the

Length of the longest prefix in the list)

Step 4: Find TEMPP in the prefix list

Step 5: IF found THEN

IF context sensitive THEN check the context-sensitive rules cl-c5

Apply al or a2 and GOTO step 7

ELSE GOTO step 6

ELSE

IF length (TEMPP) >1 THEN assign length (TEMPP) -1 Characters to

TEMPP and GOTO step 4

ELSE return WORD

Step 6: Remove prefix

Step 7: IF number of radicals of WORD > 3 or length (WORD) > 4 THEN GOTO step 1

ELSE GOTO step 2

Algorithm 3.1: Prefix remover algorithm

Step 1: Get WORD and count the number of radicals (or consonants)

Step 2: IF number of radicals < 3 TJEN stop and return WORD

Step 3: IF length (Word) < -(MAXPL + 2) THEN assign the last length (WORD) -2

Characters to a temporary prefix, TEMPP

ELSE assign the first MAXPL characters to TEMPP (where MAXPL is the

Length of the longest prefix in the list)

Step 4: Find TEMPP in the prefix list

Step 5: IF found THEN

IF the ending required recoding THEN recode and GOTO step 1

ELSE

IF context sensitive THEN check the context-sensitive rules

C1-c5 apply a1 or a2 and GOTO step 7

ELSE GOTO step 6

ELSE

IF length (TEMPP) > 1 THEN assign length (TEMPP) – 1 characters to

TEMPP and GOTO step 4

ELSE return WORD

Step 6: Remove prefix

Step 7: IF number of radicals of WORD >2 THEN GOTO step 1

Algorithm 3.1: Suffix remover algorithm

3.1.2. Stop Word Removal

Not all terms found in the document are equally important to represent documents they exist in. Some terms are common in most documents. Therefore, removing those terms, which are not used to identify some portions of the document collection, is important. Such terms are removed based on two methods. The first method is to remove high frequent terms by counting the number of occurrences (frequency). The second method is using stop word list for the language [40][41].

In this research the second method were used to apply stop word removal. The stop word list is adopted form Nega [39]. Removing stop words were applied after stemming is implemented. This is because, Amharic language is reach in word variant, thus applying stemming prior to stop word removal reduces the problem of considering morphologically variant words as different in meaning by IR system.
Table 3.2 shows stop word list used in this research.

Stop Word Lost									
ነው	እኔ	እኛ	እነሱ	እሱ	እሷ	አንተ	አናንተ	አና	ወደ
ነይ	ወይ	ከ	ናቸው	ትናት	ጥቂት	በርካታ	ብቻ	ሁሉም	ሌላ
ሌሎች	ሁሉ	እያንዳንዱ	እያንዳንዳችው	ስለ	እንዲሁም	እንጂ	ደግሞ	መካከልከ	ሰሞኑን
ከሰሞኑ	በሰሞኑ	የሰሞኑ	ትናንት	ትናንትና	ጋራ	የጋራ	ከጋራ	ተለያዩ	ተለያየ
ድረስ	እስከ	በጣም	ግን	ሲሆን	ሲል	ዉስጥ	ላይ	ናት	ነበሩ
ነበረች	ያ	ወይዘሮ	ወይዘሪት	ነገሮች	ከፊት	ከላይ	ታች	ከታች	በታች
የታች	በውስጥ	ከውስጥ	ጋር	ናቸው	ይህ	በላይ	ወደ	ወዘተ	እና
ወይም	እንደ	አቶ	ፊት	ወደፊት	ነገር	በፊት	በሆላ	በኩል	

Table 3.2: Stop Word List

3.1.3. Normalizing Characters and Words

As discussed in section 2.4.3, Amharic writing system contains characters or symbols, which have the same sound during reading. For example, the characters "ሀ, ሐ, ኀ, አ and ዐ" produces the same sound during reading with characters "ሃ, ሓ, ኃ, ኣ and ዓ" respectively. On the other hand, there are also alphabets that share the same sound, such as, "ሀ, ሐ, and ኀ", "ሰ and ሠ", "አ and ዓ", and "ጸ and ፀ". As a result, words which have the same meaning, may have different spelling structure. For example, the word "አለም" can be written in different way such as, "አለም", "ዓለም" "ዐለም" "ኣለም". Likewise, there are also synonymous words that have spelled differently but have similar meaning. For instance, the word 'ኢትዮጵያ' (Ethiopia) can be written using different words 'አቢሲኒያ' (Abyssinia) and 'የአስራራትወርጻጋ' (thirty month of sunshine). In this research such kind of problems are solved by changing the characters and the word to their common standard form.

3.2. Searching Using the Probabilistic Model

As briefly described in section 2.2.3, in designing probabilistic IR system, one can choose from several methods. Binary independent model is used to design probabilistic Amharic information retrieval system. This is because, according to Greengrass [28], the first step in most of probabilistic methods is to make some simplifying assumption. Thus, BIM is the model that has been used with the probabilistic ranking principle by introducing some simple assumptions which makes estimating the probability function P(R|d, q) practical.

The feedback process is also directly related to the derivation of new weights for query terms and the term reweighting is optimal under the assumptions of term independence [4]. In addition, it is the first model that has been used in several researches and it has clear and simple assumptions and mathematical and theoretical basis.

According to Robertson and Sparck [11], binary independent model based on two assumptions and principles; the independence assumption and ordering principle.

The independent assumption stated that (A1) the distribution of terms in relevant documents as well as in non-relevant documents and (A2) their distribution in all document is independent.

The ordering principle stated also that probable relevance is based only on the presence of query terms in the document (OP1) and/or on both the presence and absence of query terms in the documents (OP2).

	Independent Assumption		
Ordering Principle		A1	A2
	OP1	F1	F2
	OP2	F3	F4

Table 3.3: Assumption- Principle Summary Table

The records in table 3.3 F1, F2, F3 and F4 are weighting functions that related the two independent assumptions and ordering principle. Robertson and Sparck [11], stated that, the assumption A2 is more truthful than A1 while the ordering principle OP2 is correct and 0P1 is incorrect. This yield F4 is most likely to give the best result and it is the best function. Table 3.4 summarizes the weighting functions.

In binary independent model there are three steps to compute term probability. The first step compute terms when there is no retrieved document at initial stage. The second step compute terms after documents are retrieved and feedback is provided by the user. The third step compute terms when partial feedback is given [4].

At first, since the properties used to retrieve relevant information are unknown and only index terms are known properties, attempt has to make the initial guessing. The assumptions made in this step are [4];

- $P(ki|R)$ is constant for all index terms k (usually, its equal to 0.5)

- The distribution of index terms among the non-relevant documents can be approximated by the distribution of index terms among all the documents in the collection.

Weight number	Formula	Function number	Description
W1	$log \frac{(\frac{r}{R})}{(\frac{n}{N})}$	F1	Evaluates the ratio of the proportion of relevant documents in which the term occurs to the proportion of the entire collection in which it occurs.
W2	$log \frac{(\frac{r}{R})}{(\frac{n-r}{N-R})}$	F2	Evaluates the ratio of the proportion of relevant documents to that of non-relevant documents.
W3	$log \frac{(\frac{r}{R-r})}{(\frac{n}{N-n})}$	F3	Evaluates the ratio between the number of relevant documents in which it does occur and the number in which it does not occur and the collection likelihoods for the term.
W4	$log \frac{(\frac{r}{R-r})}{(\frac{n-r}{N-n-R})}$	F4	Evaluates the ratio between the term relevance likelihoods and its non-relevance likelihoods.

Table 3.4: Summary of weighting functions

These two assumptions will give as;

$$P(k_i|R) = 0.5 \; and \; P(k_i|\bar{R}) = \; log \frac{N-ni+0.5}{ni+0.5} \ldots\ldots .(3.1)$$

Where, N is the total number of documents in the collection and n_i is the number of documents which contain the index term k_i.

Using this initial guess, documents are retrieved which contain query terms and provide an initial probabilistic ranking. After documents are retrieved, the user looks at the retrieved documents and marks them as relevant and non-relevant. The system then uses this feedback to refine the description of the answer set. At this stage, initial ranking is shown and more discriminating information about terms is available (i.e. relevance feedback), this will allow more accurate estimation [4][11]. Therefore, relevant documents retrieved

should be improved using probabilistic relevance weighting technique. This technique uses the concept in term incidence contingency table 3.5 shown below [11].

	Relevant	Non-relevant	Total
Containing the term	r	n - r	n
Not containing the term	R - r	N – n – R + r	N - n
Total	R	N - R	N

Table 3.5: Term incidence contingency table

Where,

- r is the number of relevant documents that contain the term,
- n – r is the number of non-relevant documents that contain the term,
- n is the number of documents that contain the term,
- R- r is the number of relevant documents that do not contain the term,
- N – n – R + r is the number of non-relevant documents that do not contain the term,
- N – n is the number of documents that do not contain the term, R is the number of relevant documents,
- N – R is the number of non-relevant documents and N is the total number of documents in the collection.

After the knowledge of relevant documents and non-relevant documents for a given query is completed, the next step is estimating the probability of finding term (ti) in relevance document using equation 3.2 and the probability of finding term (ti) in non-relevant document using equation 3.3 [11];

$$P(ti|R) = \frac{r}{R} \dots\dots\dots\dots\dots\dots (3.2)$$

$$P(ti|\bar{R}) = \frac{n-r}{N-R} \ldots\ldots\ldots\ldots (3.3)$$

According to Sparck et al. [31], the above equations can be rewritten to compute term presence weighting function as;

$$\mathrm{w} = \log \frac{\mathrm{r}\,(\mathrm{N}-\mathrm{n}|-\mathrm{R}+\mathrm{r})}{(\mathrm{R}-\mathrm{r})(\mathrm{n}-\mathrm{r})} \ldots\ldots\ldots\ldots\ldots\ldots (3.4)$$

However, Robertson and Jones [11], noted different assumptions lead to a different formula for computing term weighting. They argue "in practice users may find themselves in the situation where, even if they know some relevant documents are retrieved, they wish to continue searching". They assume that "users may not found all the relevant documents that would satisfy their need". Therefore, the record in the center of the contingency table (i.e. N – n – R + r) may not be taken as absolute. The estimation of document relevance when considering new items has to allow for uncertainty. This estimation adds 0.5 to all the central record and it derives a specific term relevance weighting formula;

$$\text{Relevance Weighting (Rw)} = \log \frac{(\mathrm{r}+0.5)(\mathrm{N}-\mathrm{n}-\mathrm{R}+\mathrm{r}+0.5)}{(\mathrm{R}-\mathrm{r}+0.5)(\mathrm{n}-\mathrm{r}+0.5)} \ldots\ldots (3.5)$$

3.3. IR System Evaluation

Retrieving relevant document from the collection that satisfies users need is the heart of IR system evaluation in determining its effectiveness. Two strategies are identified in measuring the effectiveness performance of IR systems. The first is the user-centered strategy, which uses relevant judgment so as to evaluate the performance of the system and the second is system centered strategies which work based on reference judgment available prior to testing process [4]. Based on the concept of relevance (i.e. to a given query or information need), there are several techniques of measures of IR performance available, such as, precision and recall, F-measure, E-measure, MAP (Mean average precision), R-measure, etc [28]. In this study, the three widely used techniques precision, recall, and F-measure are used to measure the effectiveness of the IR system designed.

Precision and recall are the two most frequent and basic statistical measures. Recall is the percentage of relevant documents retrieved from the database in response to users query, whereas precision is percentage of retrieved documents that are relevant to the query [1].

To show these metrics, assume the document collection be D. Let R_t is all retrieve documents from the collection D and R_l a number of relevant documents in D. The joint of R_t and R_l is a set of documents retrieved and relevant, R_A. Therefore, the recall and precision can be calculated using equation 3.5 and 3.6 respectively.

$$\text{Recall} = \frac{|RA||R_t|}{|R_l|} \ldots\ldots\ldots\ldots (3.5)$$

$$\text{Precision} = \frac{|RA||R_t|}{|R_t|} \ldots\ldots\ldots (3.6)$$

The formula show above for precision and recall assume that, all documents retrieved (R_t) is examined by user. Thus, the retrieved document cannot be presented for the user at once. Rather, the retrieved documents are presented according to their degree of relevance as per the user query. Then, the user examines the ranked documents starting from the top. This kind of examination of documents by the user leads the recall and precision measures to vary. Therefore, for appropriate evaluation of recall and precision, plotting a precision versus recall curve is necessary [4].

To draw precision-recall curve, for example let documents retrieved by the system is Dr Where Dr = {d_3, d_{33}, d_9, d_1, d_{10}, d_{11}, d_{14}, d_7, d_{44}, d_{49}, d_{50}, d_{53}, d_{55}, d_{77}, d_{133}} in ranked order. Assume Rq contain a set of relevant documents for the query. Where, Rq = {d_1, d_3, d_7, d_{10}, d_{14}, d_{33}, d_{44}, d_{49}, d_{55}, d_{133}}. Based on the given information recall- precession curve is constructed. However, when constricting the curve based on the original recall and precession may result in saw tooth curve, there is a need to smooth the curve using interpolation technique.

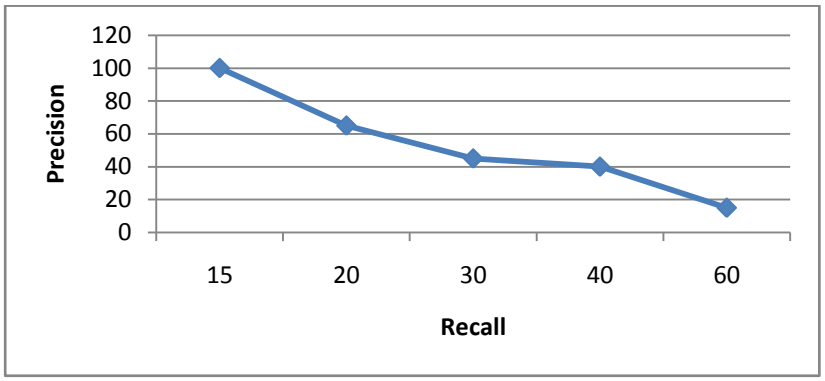

Figure 3.2: Example of precision and recall graph

The example shown above for plotting precision and recall is carried out for a single query. When measuring the performance of IR system multiple queries are used. To evaluate the performance of the system using k queries, average precision at each recall level is used. This can be expressed as follows [4];

$$\vec{P}(r) = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.7)$$

Where,$\vec{P}(r)$ is the average precision at the recall level r, Nq is the number of queries used, and Pi(r) is the precision at recall level r for the i^{th} query. It is an empirical fact that on average as recall increases, precision decreases.

The recall levels for each query might be deferent from the standard recall levels, which is difficult to compare performance across queries. Therefore, interpolation procedure is necessary. As a result, a single precision value for each query can be used that takes a precision at some recall level for each single query. The interpolated precision versus recall curve is shown as follows; Let rj, j E {0, 1,2, ... , 10}, be a reference to the j^{th} standard recall level. Then, P(rj) = max rj ≤ r≤rj+1 P(r), "which states that the interpolated precision at the j^{th} standard recall level is the maximum known precision at any recall level between the j^{th} recall level and the $(j + l)^{th}$ recall level"[4].

In general, precision and recall have been used widely so as to evaluate information retrieval system performance. However, they measure two different aspects of the system

and thus they are inversely relative. If recall of a system is improved then the precision is reduced. The reason behind this is that, when attempt is made to include many of the relevant documents, irrelevant documents more and more exist in the answer set. On the other hand, if precision of a system is improved then the recall is reduced. This is because; there are retrieved relevant documents among the whole relevant documents found in the corpus. Achieving both precision and recall 100% is ideal [4].

Several problems have been distinguished. First, to make appropriate estimation of maximum recall for a query, it needs deep knowledge of all the documents in the collection. Second, even if many situations consider the use of a single measure, which combines both, recall and precision capture different aspects of the set of retrieved documents. One of the methods developed to alleviate the above recall and precision problems is the F-measure [4].

In this research probabilistic model is used to enhance the performance of Amharic IR system by increasing the precision without affecting the recall ability of the system. Thus, the F-measure is used to measure the performance of the system since it balances the precision and recall values.

F-measure is a single measure that trades off precision versus recall. It is the weighted harmonic mean of precision and recall expressed as follows [1];

$$f(j) = \frac{2.P.R}{P+R} \ldots\ldots\ldots\ldots\ldots (3.8)$$

Where $R=(r_1,r_2,\ldots r_{11},\ldots r_n)$ and $P=(p_1,p_2,\ldots p_{11},\ldots p_n)$ are recall and precision at the j^{th} document respectively. F-measure will be 0 when no relevant document is retrieved and 1 when all the first ranked documents are relevant. Moreover, the harmonic mean F assumes a high value only when both recall and precision are high.

In summary, there are different methods in designing probabilistic based IR system. However, the binary independent method is used to develop probabilistic based IR system for Amharic language so as to enhance the performance of the IR and to alleviate the

problem of uncertainty exists in IR. To evaluate the performance of the method, the model is implemented and tested using Amharic news articles.

CHAPTER FOUR

EXPERIMENTATION

In this research an attempts has been made to design a probabilistic information retrieval system for Amharic language. The system has both the indexing and searching parts. Inverted file indexing structure is used to organize documents so as to speed up searching. The probabilistic model that attempts to simulate the uncertainty nature of an IR system guides the searching.

To test the prototype system developed, 300 Amharic News articles were used as a document corpus. All news articles are obtained from the web site of Walta Information Center [22]. As shown in table 4.1, the news articles contain seven clusters of news, which are accident, health, education, sport, tourism, justice and politics.

No.	Types of News	Number of Documents
1	Accidental news	40
2	Tourism news	40
3	Justice related news	40
4	Political related news	40
5	Educational news	40
6	Sport news	30
7	Health care news	70
	Total	300

Table 4.1: Types of news article used

Each news articles are saved under common folder using .txt format, which is supported by most of programming language. Additionally, 10 test queries were selected by the researcher to test the performance of the system. Relevance judgment is also done for identifying which document is relevant for a given test query.

4.1. Description of the Prototype System

Before queries are projected to the term document matrix, preprocessing performed on the test document collection is also performed on the queries. However, unlike vector space model, which obtain frequency of each query terms in the documents, probabilistic model obtain the absence and presence of the query term in documents. In addition to that, queries are weighted two times (i.e. before and after relevance feedback is given) using the probabilistic weighting assumption discussed in section 3.2.1. Table 4.2 and 4.3 shows example of terms document matrix constructed for weighting query before and after relevance feedback is given.

	Query terms		
Documents	Term 1	Term 2	Term 3
Doc 1	0 or 1	0 or 1	0 or 1
Doc 2	0 or 1	0 or 1	0 or 1
Doc 3	0 or 1	0 or 1	0 or 1
Doc 4	0 or 1	0 or 1	0 or 1
Doc 5	0 or 1	0 or 1	0 or 1
weight	Weight t1	Weight t2	Weight t3

Table 4.2: Example of term document matrix before relevance feedback is given

		Query terms		
Documents	Term 1	Term 2	Term 3	Relevance Feedback
Doc 1	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
Doc 2	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
Doc 3	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
Doc 4	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
Doc 5	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
weight	Weight t1	Weight t2	Weight t3	

Table 4.3: Example of term document matrix after relevance feedback is given

In developing the prototype system of probabilistic based IR for Amharic language, several components are integrated together as depicted in Figure 3.1 in chapter three. Those components describe the major activities done in the probabilistic retrieval model development. The system has been built using python version IDLE 2.6.2. Figure 4.1, presents a screen shot which shows the first list of retrieved document using a given query.

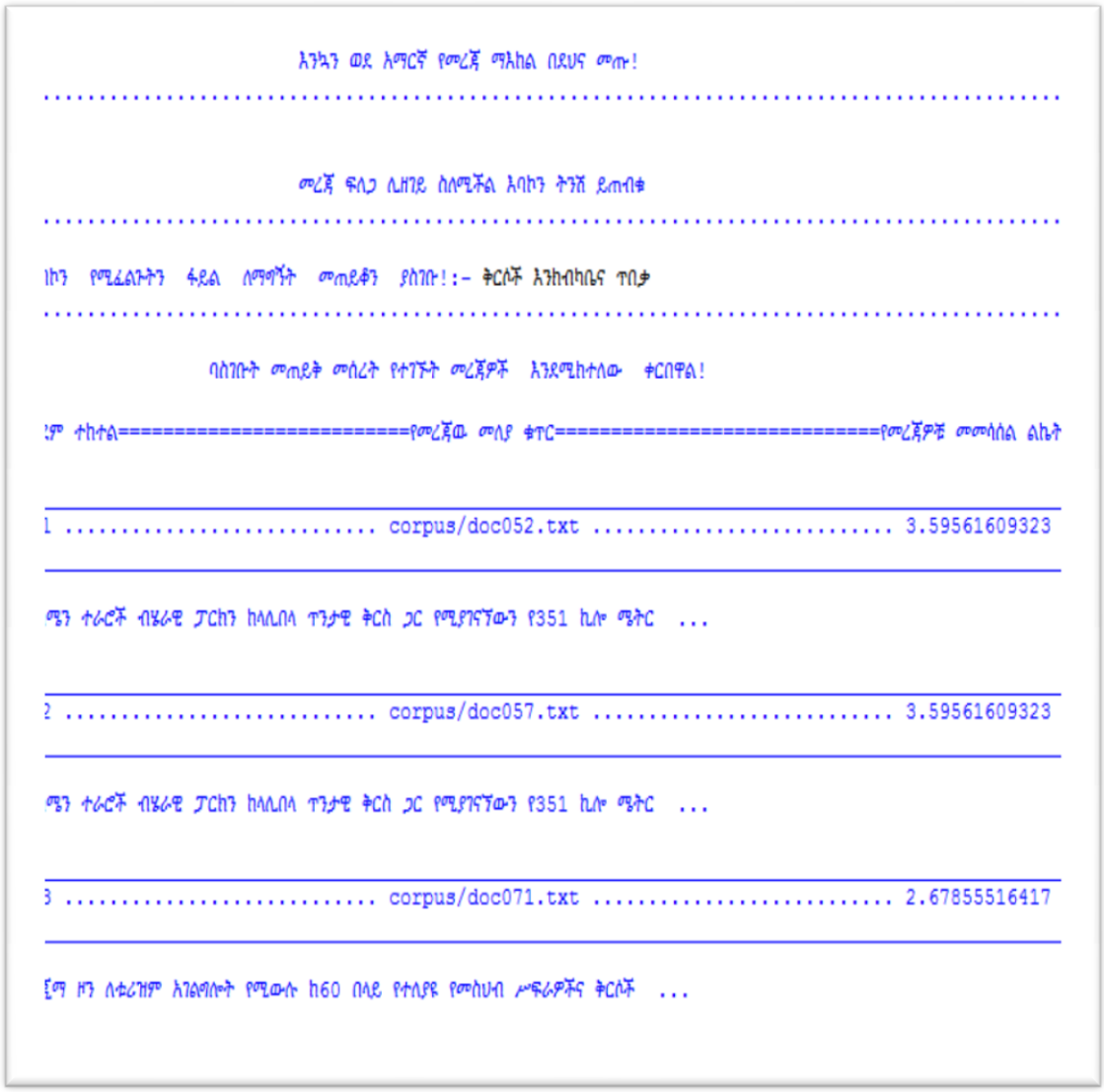

Figure 4.1: A Screen shot of retrieved document for a given query

The main objective of the prototype system is to project query terms and documents in the matrix to make comparison between documents and query, then calculating the weight of each query terms based on the notion implemented by probabilistic model and finally

calculating the score of each document and ranks in decreasing order. To fulfill this objective the following procedures are followed; first documents are pre-process for extracting index terms, removing stop word, stemming and normalization. Matrix construction such as, mapping terms and documents into space, weighting and reweighting terms using relevance feedback given by users. Testing and evaluating the prototype.

The indexing process is implemented to construct inverted index. As discussed in section 2.1.1, when implementing inverted index there are several tasks need to be done. Such as, stemming, normalization, tokenization and stop word removal.

The first step in constructing inverted index is generating index terms from document collection. In this step, the first task is tokenizing terms. Figure 4.2, shows python code implemented to tokenize terms found in document collection.

```
fileterms=fileReader(docid)
 for term in fileterms:
    indexterms=[] store each terms in the documents that are un
    for index in term.split():
       if index.isalpha():
             indexterms.append ( punctuationremove (index))
       for k in indexterms:
          List1.append(k)
       List2 = sorted(List2 + List1)
       List3 = list(set(List2))
       List5 = sorted(List3 + List1)
```

Figure 4.2: Python code for tokenization

The code first read files from all documents and extract each words, punctuations and numbers. Then, by splitting each of them, it removes punctuations and numbers. After the index terms are sorted, it store the unique index terms in ‘List5’ list variable for further processing.

As discussed in section 2.5.3, in Amharic writing system there are characters, which have the same sound during reading. In addition, there are alphabets that share the same word sound. To convert characters and words to their common form, normalization module was implemented.

Once of the problem faced when implementing normalization was finding standard list of Amharic words and characters with their common form. In this research, a limited list is prepared by collecting some of the common forms of Amharic words from different sources. Figure 4.3, shows how characters and words are converted to their common form.

```
def normal(str):
        sim=codecs.open("similarWord.txt",'r', encoding = 'utf-8' )
        similarWord=sim.read()
        li1=[]
        for m in similarWord.split():
            li1.append(m)
        for i in range(1,len(li1),2):
            if str==li1[i] or str==li1[i+1]:
                str=li1[i+1]
        temp=codecs.open("similarChar.txt",'r', encoding = 'utf-8' )
        similarChar=temp.read()
        li=[]
        for j in similarChar.split():
            li.append(j)
        for i in range(1,len(li),2):
            str=str.replace(li[i],li[i+1])
        return str
```

Figure 4.3: Python code for normalization

The code first take index term and read if similar term is found form text file. If the word is similar, it converts the term to its common form. The same process is followed to convert the character to its common form.

In Amharic writing system words are morphologically rich. Since morphological variant words have similar semantic interpretations, there is a need to stem words to their root. In this research, the stemming implementation has two modules, the prefix removal module and the suffix removal module [39]. Figure 4.4, depicts python code implementation of prefix removal of Amharic words.

```
def preffix(word):
        if len(word)<3:
            return word
        else:
            while len(word)>=3:
                tempp=word[ :3]
                p=codecs.open("prefix.txt",'r', encoding = 'utf-8' )
                pr = p.read()
                y=pr.split()
                p.close()
                if y.__contains__(tempp):
                    if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                    else:
                        word=word[3: ]
                        preffix(word)
                else:
                    tempp=word[ :2]
                    if  y.__contains__(tempp):
                        if (condition1(tempp,word)):
                            word=condition2(tempp,word)
                            return word
                        else:
                            word=word[2: ]
                            preffix(word)
                    else:
                        tempp=word[ :1]
                        if  y.__contains__(tempp):
                            if (condition1(tempp,word)):
                                word=condition2(tempp,word)
                                return word
                            else:
                                word=word[1: ]
                                preffix(word)

                        else:
                            return word
```

Figure 4.4: Python code for prefix removal

The code takes index terms and check if the length is greater that two or not. If the length of the word is less than two, the word is returned without further processing. If the length of the word is greater than two, the code iterate on word and check characters if they matched with one of the prefix found in prefix list. If the character is matched, processing continues for checking context sensitivity of the word and finally the stemmed word is returned.

After the prefix is removed from index terms, the next step is removing suffix. When removing suffix, similar procedure is followed like prefix. Figure 4.5, shows python code implemented for removing suffix.

```
def suffix(word):
        if len(word)<3:
           return word
        else:
           while len(word)>=3:
              tempp=word[len(word)-3: ]
              s=codecs.open("sufix.txt",'r', encoding = 'utf-8' )
              su = s.read()
              x=su.split()
              s.close()
              if x.__contains__(tempp):
                 if (condition1(tempp,word)):
                    word=condition2(tempp,word)
                    return word
                 else:
                    word=word[ :len(word)-3]
                    suffix(word)
              else:
                 tempp=word[len(word)-2: ]
                 if  x.__contains__(tempp):
                    if (condition1(tempp,word)):
                       word=condition2(tempp,word)
                       return word
                    else:
                       word=word[ :len(word)-2]
                       suffix(word)
                 else:
                    tempp=word[len(word)-1: ]
                    if  x.__contains__(tempp):
                       if (condition1(tempp,word)):
                          word=condition2(tempp,word)
                          return word
                       else:
                          word=word[ :len(word)-1]
                          suffix(word)
                    else:
                       return word
        return word
```

Figure 4.5: Python code for suffix removal

One of the challenges faced when implementing stemming on Amharic word is the algorithm adopted from Nega [39], sometimes overstem words that are inflected using prefix. This results the stem become too short word. For instance the word "ማግኘት" meaning 'to find' and "ማግባት" meaning 'to get marry' are over stemmed to "ማግ"; as a result of which both words are indexed as the same word located in different document. This situation in turn misrepresents two different words as similar and hence retrieval of irrelevant documents. This results the semantic relationship with other terms and statistical information for that term become poor.

After stemming is implemented, the next step is removing stop words from stop word list. Figure 4.6, shows how Amharic stop word is removed.

```
decs.open("stopw.txt",'r',encoding = 'utf-8')
  n=stop_words.read()
  s=n.split()
  stop=0
  for i in range(0,len(s)):
     if  (index) in s[i]:
        stop=1
  if stop==0:
     indexterms2.append (index)
```

Figure 4.6: Python code for removing stop words

The code read stop word list from text file and compared it with tokenized and stemmed index term. Then, if word is similar, it removes from index terms.

4.2. Performance evaluation

As discussed in section 3.3, Precision, Recall and F-measure are the most frequent and basic statistical measures which are widely used measures to assess the effectiveness of IR system (i.e., the quality of the search results). These three parameters are used in this research so as to measure the effectiveness of the designed probabilistic based IR system.

Before testing is done, ten Amharic test queries were formulated. The relevance judgment is prepared to construct document query matrix that shows all relevant documents for each test queries. After each test queries are measured by precision, recall and F-measure, then the average of each test queries represents the performance registered by the system. However, to facilitate computing average performance over a set of queries each with a different numbers of relevant documents, precision and recall curves which reflect precision at different standard recall levels from 0 to 1 inclusive in step of 0.1 is plotted to draw the curve.

The performance of the system is evaluated before and after relevance feedback using ten queries. Based on the performance registered, an attempt has been made to compare the result of probabilistic based IR system for Amharic language with the previously done Amharic IR system using vector space model.

To get the maximum optimal performance, a recursive testing has been made to fix threshold. For retrieving a set of relevant documents initially, the weight greater than one (>1) has been found that the optimal threshold. The weight greater than 1.5 is the optimal threshold value fixed for retrieving a set of relevant documents after relevance feedback is given. For the maximum number of iteration in giving a relevance feedback to the system, two times iterations has been found that the optimal thresholds.

Using the information given in table 4.4, evaluation is done by measuring the recall, precision and F-measure of each test queries and the average of each queries result are obtained as the initial performance of the system. Table 4.4 presents relevant documents retrieved from the Amharic corpus for each test query.

Query number	List of queries	List of relevant documents for each queries	Relevant document retrieved
1	የአደጋ ጊዜ እርዳታዎች	3, 4, 7, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21, 22, 23, 25, 30, 31, 33, 34, 39	3, 4, 7, 10, 11, 14, 18, 20, 21, 22, 23,
2	የኤች አይ ቪ ምርመራ	243, 249, 250, 262, 267, 268, 276, 293, 291, 298	243, 249, 250, 262, 267, 276
3	የመማሪያ ክፍል ግንባታ	191, 194, 197, 198, 199, 201, 203, 205, 216, 208, 221, 222, 225, 228, 229	203, 197, 194, 222, 205, 229, 199
4	ቅርሶች እንክብካቤና ጥበቃ	41, 42, 45, 49, 50, 51, 52, 53, 55, 57, 62, 64, 71, 75, 76, 78, 79	52, 57, 71, 51, 53, 55, 78
5	ጤና ጣቢያ ማስፋፊያ ስራዎች	233, 236, 239, 247, 248, 255, 256, 263, 264, 271, 279, 284, 297	236.
6	የእግርኳስ ስልጠና	162, 164, 167, 168, 173, 174, 177	164, 162, 174
7	ቴክኒክና ሙያ ማሰልጠኛ ተቋም	191, 201, 210, 208, 213,	208, 201, 191, 213
8	የሞትና የአካል ጉዳት አደጋ	1, 8, 19, 26, 27, 28, 35, 36, 38	1, 8, 19, 27, 35
9	የወባ በሽታ መከላከልና ቁጥጥር	4, 11, 238, 239, 240, 241, 250, 251, 258, 260, 267, 266, 275, 286, 287, 288, 296	4, 11, 258, 239, 250, 260, 287, 296, 288, 286, 278, 275
10	ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	123, 129, 130, 137, 143, 149, 148	143, 148, 123

Table 4.4: Test query, relevant document list and ranked list of output documents

Table 4.5 shows the effectiveness of the probabilistic Amharic IR system based on 10 queries selected for the experiment.

Query	Relevant	Retrived	Relevant retrived	P	R	F
የአደጋ ጊዜ እርዳታዎች	15	26	14	0.54	0.9	0.68
የኤች አይ ቪ ምርመራ	6	35	6	0.17	1.0	0.29
የመማሪያ ክፍል ግንባታ	10	29	9	0.31	0.9	0.46
ቅርሶች እንክብካቤና ጥበቃ	14	23	13	0.57	0.9	0.70
ጤና ጣቢያ ማስፋፊያ ስራዎች	9	25	9	0.36	1.0	0.53
የእግርኳስ ስልጠና	6	17	5	0.29	0.8	0.43
ቴክኒክናሙያ ማሰልጠኛ ተቋም	5	22	5	0.23	1.0	0.37
የሞትና የአካልጉዳት አደጋ	6	29	4	0.14	0.7	0.23
የወባ በሽታ መከላከልና ቁጥጥር	12	23	12	0.52	1.0	0.69
ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	4	45	4	0.09	1.0	0.16
			Average	0.32	0.80	0.48

Tabel 4.5: The initial performance of the system considering normalization and word variant

As it is observed from table 4.5, the average result of precision and recall of the system using the initial guess made by the model about the relevance of documents are 32% and 80% respectively. This shows that the percentage of recall dominates the percentage of precision by 48%. Finally, the F-measure, score is 48%, which indicates the performance of the system is not satisfactory.

The result depicts the system retrieved most of the relevant documents in the collection out of the total relevant documents in the corpus. However, the result of the precision indicates that, the non-relevant documents retrieved are higher than the relevant documents retrived. This is because; documents containing one of query terms but not-relevant are retrieved.

These documents are irrelevant because, the query term found in those documents not express the meaning of the query with respect to other terms found in the query. For example, for query "የአደጋ ጊዜ እርዳታዎች" which express the aid given at accidental time, the system retrieved irrelevant documents such as 'doc216', 'doc245', 'doc254' and 'doc253' because they contains query term 'አደጋ'. However, in these documents the term ''አደጋ' is used to expresses different accidental cases which are not related with aid. The same problem is also revealed for other queries.

On the other hand, in probabilistic model the initial guess of relevant document is based on Boolean expression. Thus, all terms that match one of user queries will be retrieved which increases the number of denominator used for calculating precision, thereby decreasing the percentage of precision..

Therefore, in order to increase the performance of the system, the probabilistic model uses relevance feedback from the users so as to apply query terms reweighting in order to increase the weight of terms found in relevant documents and decrease the weight of terms found in non-relevant documents.

As can be seen from Table 4.6, after users provide feedback on initially retrieved documents as relevant and non-relevant, the average percentage of precision is increased by 45%, recall is decreased by 19%. Thus, the performance of the system increased with an improvement of 25% F-measure. In this stage, the system retrieved documents that are judged as relevant by the user and other documents nearer to the judged relevant documents. However, the documents judged as non-relevant by the user are not retrieved because the weight of the query terms found in those documents is decreased, which results in excluding such documents from retrieval as relevant. The stemming algorithm implemented for the system also lacks controlling some of query word variant. For instance, relevant documents containing variant of query term "ግንባታ" such as "መገት", "ተገነባ", "መገንባት", "ሊገነባ", "እየተገነባ" are not retrieved. Because of the problem of stemmer, there are documents which are relevant but not retrieved by the system there by decreases the recall ability of the system.

Query	Relevant	Retrived	Rel-retri	P	R	F
የአደጋ ጊዜ እርዳታዎች	15	11	11	1.00	0.73	0.85
የኤች አይ ቪ ምርመራ	6	11	6	0.55	1.00	0.71
የመማሪያ ክፍል ግንባታ	10	7	7	1.00	0.70	0.82
ቅርሶች እንክብካቤና ጥበቃ	14	8	7	0.88	0.50	0.64
ጤና ጣቢያ ማስፋፊያ ስራዎች	9	1	1	1.00	0.11	0.20
የእግርኳስ ስልጠና	6	4	3	0.75	0.50	0.60
ቴክኒክናሙያ ማሰልጠኛ ተቋም	5	15	4	0.27	0.80	0.40
የሞትና የአካልጉዳት አደጋ	6	4	5	1.25	0.83	1.00
የወባ በሽታ መከላከልና ቁጥጥር	12	16	12	0.75	1.00	0.86
ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	4	12	3	0.25	0.75	0.38
			Average	0.77	0.69	0.73

Table 4.6: The performance of the system after relevance feedback is given without considering synonym words

Obtaining the value of precision at standard recall levels for each available 10 queries is important so as to show the performance of the system. Thus, interpolation of precision/recall curve is done. Figure 4.7 shows the interpolated recall-precision curve to depict the performance of the designed prototype system before and after relevance feedback is given.

As it is observed from Figue 4.7 the performance of the system register better performance when the user provides relevant feedback. The curve at the upper side of the graph in which recall and precision reaches maximum point indicates the highest performance registered by the system. At recall level, 0.6 the maximum precision registered is 0.77 this represents the average performance registered by the system.

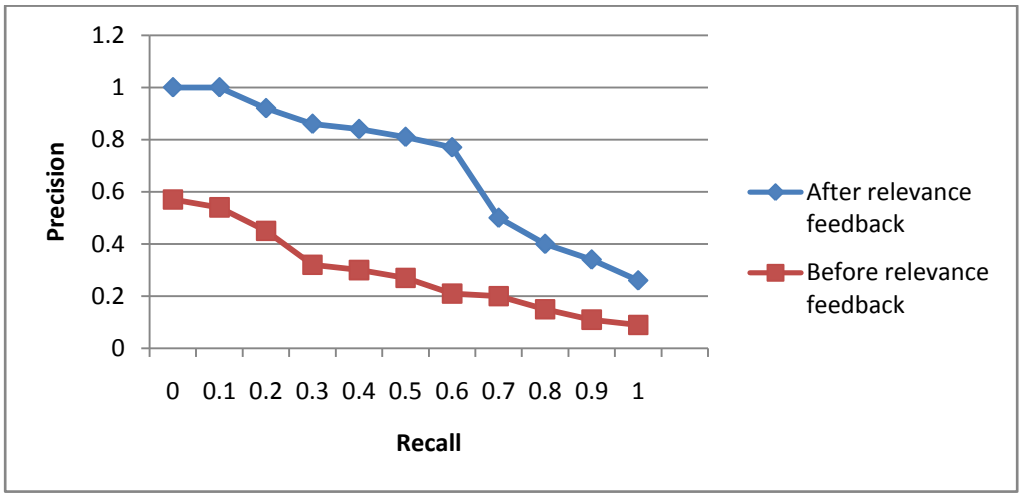

Figure 4.7: Precision/Recall curve before and after relevance feedback

4.3. Test Result on Synonymous and Polysemous

The system is also greatly affected by synonymous and polysemous nature of Amharic words. Considering documents having synonym terms of query word, system performance is evaluated. Table 4.7, shows the result of the system after users provide their relevance feedback.

As the test result in Table 4.7 indicates, as compared to the result obtained without considering synonymous word, the recall ability of the system decreases by 13% when documents having synonym words of query terms are considered. This is because, since synonym terms are not handled in this system, the result of recall highly decreases. There are several relevant documents containing synonym word of query terms. For instance, for the query ‘የመማሪያ ክፍል ግንባታ’’ relevant document ‘doc194’ not retrieved because it expressed by synonym words of query terms called ‘የትምህርት ቤት ማስፋፊያ’.

Query	Relevant	Retrived	Rel-retrie	P	R	F
የአደጋ ጊዜ እርዳታዎች	21	11	11	1.00	0.52	0.69
የኤች አይ ቪ ምርመራ	10	11	6	0.55	0.60	0.57
የመማሪያ ክፍል ግንባታ	15	7	7	1.00	0.47	0.64
ቅርሶች እንክብካቤና ጥበቃ	17	8	7	0.88	0.41	0.56
ጤና ጣቢያ ማስፋፊያ ስራዎች	13	1	1	1.00	0.08	0.14
የእግርኳስ ስልጠና	7	4	3	0.75	0.43	0.55
ቴክኒክናሙያ ማሰልጠኛ ተቋም	5	15	4	0.27	0.80	0.40
የሞትና የአካልጉዳት አደጋ	9	4	5	1.25	0.56	0.77
የወባበሽታመከላከልናቁጥጥር	18	16	12	0.75	0.67	0.71
ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	7	12	3	0.25	0.43	0.32
			Average	0.77	0.50	0.60

Table 4.7: The performance of the system after relevance feedback is given considering synonym words

4.4. Findings and Challenges

The obtained result indicates probabilistic based IR system for Amharic language register encouraging performance that increases in 10% F-measure as compared to the recent IR system developed for Amharic language by Abey [6].

However, there are several challenges faced which limits to register the optimum performance expected from the model in order to outperform the entire IR system developed for Amharic language.

The first challenge comes from the probabilistic model itself. In probabilistic model the initial guess is made based on Boolean expression. This results every documents that matches one of the term in query are retrieved. It can be possible limiting the retrieved documents using threshold; however, from the experiment it is found that if the user enter less than or equal to two queries there is a high probability of having similar weight. This will result retrieving no documents or retrieving all documents containing those terms. In this case, the precision decreases highly and the recall becomes 1.0.

On the other hand, since probabilistic model used Boolean expression for initial guess of relevant document, it does not consider the importance of the document based on the frequency of the terms in the document. For this reason, sometimes those documents having query terms with highest frequency than others could be ranked lately. In this case, users faced with the problem of having to choose the appropriate words that are also used in the relevant documents. Hence, poor result could be displayed when the system retrieve documents after feedback is given.

Additionally, one of the experimentation result shows that, in a very rare case a relevant document may not be retrieved. This is because when the weight of the terms found in documents which are similar with query terms are a sequence of negative (-) and positive (+) result, adding the score may generate a negative result. Since in probabilistic a negative result is considered as non-relevant document, the document will not be retrieved.

The other challenge comes from the problem of synonym and polysemy terms. Like any other information retrieval model, the probabilistic model has not incorporated a mechanism to control synonym or polysemous terms. As discussed in section 2.3.4, Amharic language encompasses full of synonym and polysemous words. In IR system unless there is a mechanism to control those kinds of words, the performance of the system highly decreases because relevant documents containing synonym word with of query term are not retrieved, while irrelevant documents that contains polysemy words are retrieved. For instance, for a query "እርዳታ" meaning they aid, a document contain word "ሰጡ" meaning they give and "ለገሱ" meaning they donate could not be retrieved unless it contain

the query word itself "እርዳታ". The combination of the above results leads to a decrease in the performance of the system in both precision and recall.

Table 4.6 shows, some examples of test queries with their word variants uncontrolled by the stemming algorithm implemented for the system and synonym terms of queries found in relevant documents but not retrieved.

Queries	Synonyms words	Variants words
የአደጋ ጊዜ እርዳታዎች	'ልገሳ', 'ሰጡ', 'አበረከቱ'	'ረዱ', 'በመርዳት', 'ሲረዱ'
የኤች አይ ቪ ምርመራ	'ኤድስ'	'ተመረመሩ', 'እንዲመረመሩ'
የመማሪያ ክፍል ግንባታ	'ታንጸዉ', 'ተሰሩ', 'ትምህርት ቤት' 'ህንጻ'	'ተገነቡ', 'ይገነባሉ', 'በመገንባትላይ', 'አስገንብቶ'
ጤና ጣቢያ ማስፋፊያ ስራዎች	'ሆስፒታል', 'ጤና ኬላ', 'ተግባራት'	
ቅርሶች እንክብካቤና ጥበቃ	'ክትትል' 'እድሳት'	
የሞትና የአካልጉዳት አደጋ	'ህይወቱ አለፈ' 'አረፉ'	
የወባ በሽታ መከላከልና ቁጥጥር	'ህመምተኞች'	'ወባኝክ'
ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	'ተቋማት' 'ተግማራት' 'ስራ'	

Table 4.8: Test queries with their synonym word and word variant

Finding a large size and standard corpus for Amharic language is also considered as one of the challenge faced in this research. The state of the art in the area of text processing indicates that, there is no any developed standard corpus for Amharic language. Thus, in this research the researcher uses small size corpus extracted from Walta Information Center website [21]. This result not only weakens the performance of the system but also makes it difficult to compare the result obtained with several researches since there is different in test queries, document content and size used for testing.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1. Conclusion

Information retrieval is the mechanism for finding relevant documents from unstructured document collection that satisfies information need of the users [4]. It is obvious that the main goal of information retrieval system is to retrieve relevant information. Several models have been developed considering that goal. However, most of them have the problem of defining uncertainty exists in IR system. One of the model which have the ability to control the uncertainty in IR system is probabilistic model. It incorporates a learning or relevance feedback component that can improve performance when presented with sample relevant documents.

Therefore, in this research an attempt has been made to design and develop a probabilistic based information retrieval system for Amharic language in order to enhance the retrieval performance of Amharic IR system considering the advantage of the probabilistic model.

In designing the IR system, Binary Independent Model (BIM) is selected and implemented. At first step when the search component initiated the system generates the first ranked list of relevant documents according to the initial guess made based on the notion of the model, then the system asked for evidences from the user about the relevancy or non-relevancy of the document. Finally based on the evidence and users relevance feedback the system improves its performance.

System evaluation has been done to discover the extent to which the designed system enhances the performance of Amharic IR system based on the F-measure. As the experimental result show, probabilistic based Amharic IR system register a better performance and score on the average 73% F-measure. This is a promising result to design an applicable IR system if polysemous and synonymous nature of Amharic words is controlled with the help of thesaurus and co-occurrence analysis.

5.2. Recommendation

The designed probabilistic based Amharic IR system is just the first attempt to see the advantage of the model in order to enhance the performance of IR system for Amharic text. Therefore, to obtain the optimum performance expected form the model, this work can be further pursued in several future directions.

- Probabilistic model make the initial guess based on Boolean expression, which inhibit to know important words to represent a document and, accordingly may not retrieve relevant documents that contain large number of terms found in a given query. Hence, there is a need to build hybrid system that uses vector space model to guess relevant documents for user query using non-binary weighting technique and then use probabilistic relevance feedback to improve the performance of the system.

- One of the problems in enhancing the performance of Amharic IR system is the existence of synonyms and polysemy terms in Amharic text. We recommended integrating mechanisms of controlling synonym and polysemy terms in the probabilistic model to enhance precision and recall of the system.

- The stemming algorithm used in this research is the best algorithm developed so far for Amharic language. However, it frequently overstems word variants greatly affecting the performance of the system. Therefore, future word need to consider designing ontology based stemming algorithm that conflates based on meaning understanding.

- Finding a standard corpus and test queries with relevance judgement for testing the designed system is one of the challenges faced in this research. Therefore, future research need to consider the development of standard Amharic corpus that can be used by researcher to evaluate progress made in designing Amharic IR system.

- Sometimes users may need a document which contains combination of query terms by using conjugations words such as 'ና' or'እና' meaning AND,'ወይም' meaning OR and 'ሳይሆን' meaning NOT. This demands an intelligent IR system that reconstructs query terms as per user's requirement. Hence, there is a need to integrate query reconstruction mechanism to the Amharic IR system.

REFERENCES

[1] H.S. Christopher D.Manning, Prabhakar Raghavan, "An Introduction to Information Retrieval", 1st Edition, Cambridge University Press, Cambridge England, 2009.

[2] A. Singhal, "Modern Information Retrieval: A Brief Overview", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 24(4): pp. 35-42, 2001.

[3] T. Saracevic, "Information science: Encyclopedia of Library and Information Science", 3rd Edition, Taylor and Francis publisher, New Jersey, USA, 2009.

[4] B. R., Ribeiro Neto, "Modern Information Retrieval", 2nd Edition, Addison-Wesley-Longman Publishers, New York, USA 1999.

[5] D. Hiemstra, T. Pothoven, M.V. Vliet, and D. Harman, "Behind the Scenes of the Digital Museum of Information Retrieval Research", In Proceedings of the 9th Dutch-Belgian Information Retrieval Workshop, 9(1):99-100, 2009.

[6] Abey Bruck, "Semantic Based Query Expansion Technique for Amharic IR", MSc Thesis, School of Information Science, Addis Ababa University, Ethiopia, 2011.

[7] N. Fuhr, "Probabilistic Models in Information Retrieval", The Computer Journal, 35(3):243-255, 1992.

[8] "Informaiton Retrieval Model", Available at; http://comminfo.rutgers.edu /~aspoerri/InfoCrystal/ Ch_2.htm, Accessed Date; 12/5/2012, 2012.

[9] J. Picard and R. Haenni, "Modeling Information Retrieval with Probabilistic Argumentation Systems", 20th BCS-IRSG Annual Colloquium on IR, 14(7):10-25,1998.

[10] Nlp.stanford.edu, "Probabilistic Information Retrieval", Available at; http://nlp.stanford.edu/IR-book/html/htmledition/probabilistic-information-retrieval-1.html, Accessed Date; 27/1/2012, 2008.

[11] K.S. S.E.Robertson, "Relevance Weighting of Search Terms", Journal of the American Society for Information Science, 27(4):129-146, 1976.

[12] C.J. Rijsbergen, "A Theoretical Basis for the Use of Co-occurrence Data in Information Retrieval", Journal of Documentation, 23(2):106-–119, 1977.

[13] F.D. Commission, "Summary and Statistical Report of the 2007 Population and Housing Census", pp. 1-113, 2008.

[14] Betelihem M., "N-Gram-Based Automatic Indexing for Amharic Text", MSc Thesis, Addis Ababa University School of Information science, Ethiopia, 2002.

[15] Tewodros H., "Amharic text retrieval: An Experiment Using Latent Semantic Indexing (LSI) with Singular Value Decomposition (SVD)" , MSc Thesis, Addis Ababa University, School of Information science, Ethiopia, 2003.

[16] Tesema M. and Solomon A. , "Design and Implementation of Amharic Search Engine", fifth international conference on signal image technology and internet based systems, 3(1):318-325, 2009.

[17] J.B. Teevan, "Bayesian Model for Information Retrieval", Available at; http://www.ai.mit.edu/research/abstracts/abstracts2000/pdf/z-teevan1.pdf, Accessed Date; 29/1/2012, 2000.

[18] V.H. Surajit Chaudhuri, Gautam Das, "Probabilistic Information Retrieval Approach for Ranking of Database Query Results", ACM Transactions on Database systems (TODS), 31(3): 1-43, 2004.

[19] S.E. Xiangji Huang, "Okapi Chinese Text Retrieval Experiments at TREC-6", in Proceedings of TREC'1997, 16(4) 552-569, 2000.

[20] L. Dolamic and J. Savoy, "Indexing and Searching Strategies for the Russian Language", Journal of the American Society for Information Science, 60(24):2540-2547, 2009.

[21] Abiy Z. , Alemayehu W. , Daniel T. , Melese G. and Yilma S. , "Introduction to Research Methods", 1st Edition, Graduate studies and Research Office of Addis Ababa University, 2009.

[22] Walta-Information-Centre, "Local News Papers", Available at; http://www.newspapersites.net/ newspaper/ walta- information-centre. Accessed Date; 11/1/2012, 2012.

[23] M. Lutuz, "Learning Python" , O'Reilly publish, USA ,4th Edition, 2009.

[24] K. D.Lee, "Python Programming Fundamentals" , Springer-verlog press, USA, 1st Edition, 2011.

[25] T. Mandl, "Recent Developments in the Evaluation of Information Retrieval Systems : Moving Towards Diversity and Practical Relevance", the European journal for the information professional, . 32(5) 27-38, 2008.

[26] www.dsoergel.com, "The Scope of IR Utility , Relevance , and IR System Performance", Available at; http:// www.dsoergel.com%2FNewPublications%2FHCIEncyclopediaIRShortEF orDS .pd, Accessed Date; 7/2/2012, 1997.

[27] D. Hiemstra, "Information Retrieval Models", John Wiley and sons publisher, Enschede-Noord, Netherlands , 1st Edition, 2009, 2009.

[28] Ed Greengrass, "Information Retrieval : A Survey by Ed Greengrass", Available at; www.csee.umbc.edu/cadip/readings/IR.report.120600.book.pdf, Accessed Date; 3/12/2011, 2011.

[29] H.R. Turtle and W.B. Croft, "A Comparison of Text Retrieval Models", The Computer Journal, 35(2): 279-290,1992.

[30] S.W. k. Sparck Jones, "A probabilistic Model of Information Retrieval: Development And Comparative Experiments", Information Processing and Management, 36(4): 779-808, 2000.

[31] Ruggeri F, Faltin F and Kenett R, Bayesian Networks, Available at; http://www.eng.tau.ac.il /~bengal/BN.pdf, Accessed Date; 17/3/2012, 2012

[32] M.L. Krieg, "A Tutorial on Bayesian Belief Networks", Available at; http://www.dsto.defence. gov . au, Accessed Date; 10/1/2012, 2012.

[33] Shipeng y, Deng C, Ji-Rong W and Wei-Ying M, "Improving Pseudo-Relevance Feedback in Web page Segmentation", Proceedings of the 12th International Conference on World Wide Web, 1(1) pp. 11 - 18, 2003.

[34] Arnaud F and Renata D, "Rocciho's Relevance feedback Algorithm in Basic Vector Space Comparison and LSI Models", Available at; http://www.mpi-inf.mpg.de/~bast/ir-seminar-ws04/project-proposals/renata_dividino_arnaud_fietzke.pdf, Accessed date: 15/4/2012, 2012

[35] T. Bloor, "The Ethiopic Writing System: a Profile", Journal of the Simplified Spelling Society, 19(1): 30-36, 1995.

[36] M. Bender, "The Ethiopic Writing System", Oxford University Press, London, 1976.

[37] Saba A, "The Application of Information Retrieval Techniques to Amharic Documents on the Web", MSc Thesis, Addis Ababa University School of Information Studies for Africa, Ethiopia, 2001.

[38] Alemayehu ,"Application of Query Expansion for Amharic information retrieval System", MSc Thesis, Addis Ababa University, School of Information Science, Ethiopia, 2002.

[39] Nega A. and P. Willett, "Stemming of Amharic Words for Information Retrieval", in Literary and Linguistic Computing. Oxford University press, Oxford London, 17(1): pp. 1-18, 2002.

[40] C.J. Rijsbergen, "Information Retrieval" , Butterworth-Heinemann publisher, USA, 2nd edition, 1979.

[41] M.J. Gerard Salton, "Introduction to Modern Information Retrieval" , Available at; http://www.sifaka.cs.uiuc.edu/course/410s12/mir.pdf , Accessed Date; 20/2/2012, 2012.

APPENDIXES

Appendix 1: Amharic character set

	ā/ä [a]	u [u]	ī/i [i]	a [a]	ē/e [e/ɛ]	(i)/(ə) [ə]	0 [o/ɔ]
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
[h]	ha	hu	hi	ha	he	h(ə)	ho
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
[l]	le	lu	li	la	le	l(ə)	lo
h/ḥ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
[h]	ha	hu	hi	ha	he	h(ə)	ho
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
[m]	me	mu	mi	ma	me	m(ə)	mo
s/ś	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
[s]	se	su	si	sa	se	s(ə)	so
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
[r]	re	ru	ri	ra	re	r(ə)	ro
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
[s]	se	su	si	sa	se	s(ə)	so
sh/š	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
[ʃ]	ʃe	ʃu	ʃi	ʃa	ʃe	ʃ(ə)	ʃo
k’/q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
[k’]	k’e	k’u	k’i	k’a	k’e	k’(ə)	k’o
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
[b]	be	bu	bi	ba	be	b(ə)	bo
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
[t]	te	tu	ti	ta	te	t(ə)	to
ch/č	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ

	ā/ä [a]	u [u]	ī/i [i]	a [a]	ē/e [e/ɛ]	(i)/(ə) [ə]	o [o/ɔ]
h/ḫ	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
[h]	he	hu	hi	ha	he	h(ə)	ho
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
[w]	we	wu	wi	wa	we	w(ə)	wo
‘/ʿ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
[ʔ]	ʔa	ʔu	ʔi	ʔa	ʔe	ʔ(ə)	ʔo
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
[z]	ze	zu	zi	za	ze	z(ə)	zo
zh/ž	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
[ʒ]	ʒe	ʒu	ʒi	ʒa	ʒe	ʒ(ə)	ʒo
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
[j]	je	ju	ji	ja	je	j(ə)	jo
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
[d]	de	du	di	da	de	d(ə)	do
j/ǧ	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
[ʤ]	ʤe	ʤu	ʤi	ʤa	ʤe	ʤ(ə)	ʤo
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
[g]	ge	gu	gi	ga	ge	g(ə)	go
t’/ṭ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
[t’]	t’e	t’u	t’i	t’a	t’e	t’(ə)	t’o
ch’/č̣	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
[ʧ’]	ʧ’e	ʧ’u	ʧ’i	ʧ’a	ʧ’e	ʧ’(ə)	ʧ’o
p’/ṗ	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ

[ɸ]	ɸe	ɸu	ɸi	ɸa	ɸe	ɸ(ə)	ɸo
h/ḫ	ሃ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
[h]	ha	hu	hi	ha	he	h(ə)	ho
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
[n]	ne	nu	ni	na	ne	n(ə)	no
ny/ñ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
[ɲ]	ɲe	ɲu	ɲi	ɲa	ɲe	ɲ(ə)	ɲo
’/’	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
[ʔ]	(ʔ)a	(ʔ)u	(ʔ)i	(ʔ)a	(ʔ)e	(ʔ)(ə)	(ʔ)o
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
[k]	ke	ku	ki	ka	ke	k(ə)	ko

[p’]	p’e	p’u	p’i	p’a	p’e	p’(ə)	p’o
ts’/ṣ	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
[ts]	ts’e	ts’u	ts’i	ts’a	ts’e	ts’(ə)	ts’o
ts’/ṣ́	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
[ts]	ts’e	ts’u	ts’i	ts’a	ts’e	ts’(ə)	ts’o
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
[f]	fe	fu	fi	fa	fe	f(ə)	fo
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
[p]	pe	pu	pi	pa	pe	p(ə)	po
v	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ
[v]	ve	vu	vi	va	ve	v(ə)	vo

Appendix 2: Amharic punctuation mark list set

፣	።	፤	፥	፦	፧
comma	full stop / period	colon	semi-colon	preface colon	question mark (no longer used)

Appendix 3: Amharic numbering set

፩	፪	፫	፬	፭	፮	፯	፰	፱	፲
1	2	3	4	5	6	7	8	9	10
፳	፴	፵	፶	፷	፸	፹	፺	፻	፼
20	30	40	50	60	70	80	90	100	10000

Appendix 4: Relevance judgment used

Documents	የኢደጋ ጊዜ እርዳታዎች	የኤች አይ ቪ ምርመራ	የመማሪያ ክፍል ማንባታ	ቅርሶች እንክብካቤ ና ጥበቃ	ጤናጣቢያ ማስፋፊያ ስራዎች	የእግርኳስ ስልጠና	ቴክኒክና ሙያ ማሰልጠኛ ተቋም	የሞትናየአካል ጉዳትአደጋ	የወባ በሽታ መከላከልና ቁጥጥር	ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ
doc001	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc002	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc003	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc004	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc005	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc006	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc007	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc008	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc009	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc010	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc011	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc012	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc013	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc014	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc015	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc016	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc017	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc018	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc019	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc020	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc021	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc022	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc023	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc024	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc025	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc026	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc027	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc028	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc029	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc030	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc031	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc032	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc033	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc034	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc035	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc036	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc037	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc038	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc039	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc040	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc041	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc042	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc043	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc044	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc045	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc046	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc047	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc048	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc049	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc050	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc051	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc052	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc053	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc054	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc055	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc056	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc057	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc058	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc059	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc060	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc061	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc062	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc063	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc064	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc065	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc066	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc067	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc068	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc069	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc070	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc071	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc072	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc073	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc074	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc075	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc076	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc077	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc078	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc079	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc080	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc081	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc082	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc083	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc084	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc085	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc086	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc087	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc088	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc089	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc090	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc091	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc092	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc093	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc094	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc095	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc096	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc097	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc098	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc099	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc100	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc101	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc102	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc103	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc104	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc105	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc106	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc107	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc108	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc109	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc110	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc111	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc112	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc113	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc114	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc115	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc116	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc117	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc118	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc119	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc120	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc121	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc122	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc123	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc124	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc125	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc126	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc127	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc128	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc129	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc130	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc131	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc132	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc133	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc134	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc135	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc136	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc137	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant

doc138	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc139	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc140	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc141	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc142	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc143	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc144	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc145	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc146	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc147	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc148	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc149	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc150	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc151	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc152	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc153	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc154	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc155	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc156	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc157	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc158	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc159	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc160	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc161	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc162	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc163	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc164	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc165	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc166	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc167	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc168	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc169	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc170	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc171	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc172	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc173	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc174	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc175	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc176	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc177	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc178	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc179	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc180	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc181	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc182	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc183	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc184	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc185	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc186	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc187	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc188	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc189	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc190	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc191	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc192	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc193	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc194	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc195	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc196	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc197	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc198	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc199	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc200	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc201	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc202	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc203	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc204	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc205	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc206	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc207	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc208	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc209	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc210	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc211	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc212	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc213	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc214	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc215	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc216	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc217	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc218	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc219	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc220	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc221	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc222	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc223	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc224	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc225	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc226	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc227	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc228	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc229	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc230	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc231	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc232	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc233	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc234	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc235	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc236	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc237	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc238	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc239	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc240	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc241	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc242	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc243	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc244	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc245	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc246	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc247	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc248	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc249	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc250	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc251	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc252	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc253	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc254	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc255	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc256	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc257	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc258	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc259	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc260	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc261	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc262	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc263	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc264	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc265	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc266	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc267	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc268	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc269	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc270	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc271	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc272	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc273	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc274	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc275	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc276	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc277	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc278	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc279	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc280	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc281	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc282	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc283	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc284	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc285	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc286	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc287	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc288	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc289	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc290	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc291	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc292	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc293	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc294	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc295	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc296	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc297	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc298	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc299	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc300	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

Appendix 5: Probabilistic Amharic IR system python code

- Indexing module

```
import os
import math
import codecs
import string
import re
import glob
import sys
class amhariIndexing:
   def index(self, path):
      def condition1(affix, tempp):
         if len(tempp)> 2:
            if affix.__eq__(u'\u12a8'):
                 return 1
            elif affix.__eq__(u'\u1260'):
                 return 1
            elif affix.__eq__(u'\u129E\u127D'):
                 return 1
            elif affix.__eq__(u'\u1295'):
                 return 1
            elif affix.__eq__(u'\u1275'):
                 return 1
            elif affix.__eq__(u'\u1290\u1275'):
                 return 1
            elif affix.__eq__(u'\u12CE\u127D'):
                 return 1
            elif affix.__eq__(u'\u127D'):
                 return 1
            elif affix.__eq__(u'\u127D\u1295'):
                 return 1
            elif affix.__eq__(u'\u127D\u1201'):
                 return 1
            elif affix.__eq__(u'\u1278\u12CD'):
                 return 1
            elif affix.__eq__(u'\u1271'):
                 return 1
            elif affix.__eq__(u'\u1279'):
                 return 1
            elif affix.__eq__(u'\u12CA'):
                 return 1
            elif affix.__eq__(u'\u12CA\u1275'):
                 return 1
            elif affix.__eq__(u'\u12CD\u12EB\u1295'):
                 return 1
            elif affix.__eq__(u'\u1277\u120D'):
                 return 1
            elif affix.__eq__(u'\u121D'):
```

```
            return 1
        elif affix.__eq__(u'\u1275') or affix.__eq__(u'\u12A0'):
            return 1
        elif affix.__eq__(u'\u129B'):
            return 1
        elif affix.__eq__(u'\u1290\u1275') or affix.__eq__(u'\u12A5'):
            return 1
        elif affix.__eq__(u'\u121D') or affix.__eq__(u'\u12A5'):
            return 1
        elif affix.__eq__(u'\u1208\u1275'):
            return 1
    else:
        return 0

def condition2(affix,tempp):
    if len(tempp) > 2:
        if affix.__eq__(u'\u12A8'):
            if tempp.__eq__(u'\u12A8\u1265\u1275') or tempp.__eq__(u'\u12A8\u1208\u12A8\u1208') or
       tempp.__eq__(u'\u12A8\u1290\u12A8\u1290') or tempp.__eq__(u'\u12A8\u1228\u12A8\u1228') or
       tempp.__eq__(u'\u12A8\u1128\u12A8\u1218') or tempp.__eq__(u'\u12A8\u1230\u12A8\u1230') or
       tempp.__eq__(u'\u12A8\u1270\u12A8\u1270') or tempp.__eq__(u'\u12A8\u1270\u121B') or
       tempp.__eq__(u'\u12A8\u1228\u1295') or tempp.__eq__(u'\u12A8\u1230\u120D') or
       tempp.__eq__(u'\u12A8\u134D\u1270\u129B') or tempp.__eq__(u'\u12A8\u1265\u1275'):
                return tempp
            else:
                tempp = tempp[1: ]
                tempp = preffix(tempp);
                return tempp
        elif affix.__eq__(u'\u1260'):
            if tempp.__eq__(u'\u1260\u123D\u1273') or tempp.__eq__(u'\u1260\u1300\u1275') or
       tempp.__eq__(u'\u1260\u1228\u1260\u1228') or tempp.__eq__(u'\u1260\u1230\u1260\u1230'):
                return tempp
            else:
                tempp = tempp[1: ]
                tempp = preffix(tempp);
                return tempp
        elif affix.__eq__(u'\u129B'):
            if tempp.__eq__(u'\u1260\u123D\u1273'):
                return tempp
            else:
                tempp = tempp[ :-1]
                tempp = preffix(tempp);
                return tempp
        elif affix.__eq__(u'\u1295'):
            if tempp.__eq__(u'\u12D8\u1218\u1295') or tempp.__eq__(u'\u1261\u12F5\u1295') or
       tempp.__eq__(u'\u123D\u134B\u1295') or tempp.__eq__(u'\u12C8\u1308\u1295') or
       tempp.__eq__(u'\u1261\u12F5\u1295') or tempp.__eq__(u'\u123D\u134B\u1295') or
       tempp.__eq__(u'\u12C8\u1308\u1295') or tempp.__eq__(u'\u132D\u1241\u1295') or
       tempp.__eq__(u'\u1205\u133B\u1295') or tempp.__eq__(u'\u12A5\u12CD\u1295') or
       tempp.__eq__(u'\u12A5\u121D\u1295') or tempp.__eq__(u'\u12D8\u1348\u1295'):
                return tempp
            else:
```

```
        tempp = tempp[ :-1]
        tempp=sadisConverter(tempp)
        tempp = suffix(tempp)
        return tempp
 elif affix.__eq__(u'\u1208\u1275'):
    if tempp.__eq__(u'\u12A0\u1208\u1275') or tempp.__eq__(u'\u12A5\u1208\u1275') or
tempp.__eq__(u'\u1201\u1208\u1275'):
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp);
        return tempp
 elif affix.__eq__(u'\u1275'):
    if tempp.__eq__(u'\u12A0\u1263\u1275') or tempp.__eq__(u'\u12A0\u12EB\u1275') or
tempp.__eq__(u'\u1205\u12ED\u12C8\u1275') or tempp.__eq__(u'\u12A0\u122B\u12CA\u1275') or
tempp.__eq__(u'\u1230\u12A3\u1275') or tempp.__eq__(u'\u1218\u1265\u1275') or
tempp.__eq__(u'\u1218\u1230\u1228\u1275') or tempp.__eq__(u'\u12C8\u1245\u1275') or
tempp.__eq__(u'\u1325\u1228\u1275') or tempp.__eq__(u'\u1275\u122D\u134D') or
tempp.__eq__(u'\u1201\u1208\u1275') or tempp.__eq__(u'\u1236\u1235\u1275') or
tempp.__eq__(u'\u12A0\u122B\u1275') or tempp.__eq__(u'\u12A0\u121D\u1235\u1275') or
tempp.__eq__(u'\u1235\u12F5\u1235\u1275') or tempp.__eq__(u'\u1230\u1263\u1275') or
tempp.__eq__(u'\u1235\u121D\u1295\u1275') or tempp.__eq__(u'\u1275\u121D\u1205\u122D\u1275')
or tempp.__eq__(u'\u1325\u1245\u121D\u1275') or tempp.__eq__(u'\u1233\u121D\u1295\u1275') or
tempp.__eq__(u'\u12AD\u122B\u12CA\u1275'):
       return tempp
    else:
        tempp = tempp[ :-1]
        #change to sades
        tempp = suffix(tempp);
        return tempp
 elif affix.__eq__(u'\u1290\u1275'):
    if tempp.__eq__(u'\u12A5\u12CD\u1290\u1275') or tempp.__eq__(u'\u12A5\u121D\u1290\u1275'):
        return tempp
    else:
        tempp = tempp[ :-1]
        #change to sades
        tempp = suffix(tempp);
        return tempp
 elif affix.__eq__(u'\u12CE\u127D'):
    if tempp.__eq__(u'\u1230\u12CE\u127D'):
        str1 = tempp[-1: ]
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp)
        return tempp
 elif affix.__eq__(u'\u129E\u127D'):
    if tempp.__eq__(u'\u12F3\u129E\u127D'):
        tempp = tempp[ :-1]
        ##change to sades
```

```
        return tempp
    else:
        tempp = tempp[ :-1]
        #change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D\u1295'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D\u1201'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1278\u12CD'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1271'):
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1279'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u12CA'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u12CA\u1275'):
    if tempp.__eq__(u'\u12A0\u122B\u12CA\u1275'):
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp);
        return tempp
elif affix.__eq__(u'\u12CD\u12EB\u1295'):
        tempp = tempp[ :-3]
        ##change to sades
        tempp = suffix(tempp)
```

```
            return tempp
        elif affix.__eq__(u'\u1277\u120D'):
            if tempp.__eq__(u'\u121E\u1277\u120D'):
                tempp = tempp[ :-1]
                ##change to sades
                return tempp
            else:
                tempp = tempp[ :-2]
                #change to sades
                tempp = suffix(tempp)
                return tempp
        elif affix.__eq__(u'\u121D'):
            if tempp.__eq__(u'\u1308\u12F3\u121D') or tempp.__eq__(u'\u1230\u120B\u121D') or
       tempp.__eq__(u'\u12A0\u1208\u121D'):
                return tempp
            else:
                tempp = tempp[ :-1]
                #change to sades
                tempp = suffix(tempp);
                return tempp
    else:
        return tempp
def punctuationremove(str):
    if str.endswith('\n'):
        str=str.rstrip('\n')
    if str.startswith(' '):
        str=str.lstrip(' ')
    p=codecs.open("Punctuation.txt",'r', encoding = 'utf-8')
    Punctuation = p.read()
    for pc in Punctuation:
        if str.endswith(pc):
            str=str.rstrip(pc)
        if str.startswith(pc):
            str=str.lstrip(pc)
    for j in Punctuation:
        for i in range(1,len(str)):
            if j in Punctuation and j in str:
                str=str.strip(j)
    return str
def normal(str):
    sim=codecs.open("similarWord.txt",'r', encoding = 'utf-8' )
    similarWord=sim.read()
    li1=[]
    for m in similarWord.split():
        li1.append(m)
    for i in range(1,len(li1),2):
        if str==li1[i] or str==li1[i+1]:
            str=li1[i+1]
    temp=codecs.open("similarChar.txt",'r', encoding = 'utf-8' )
    similarChar=temp.read()
    li=[]
    for j in similarChar.split():
```

```
        li.append(j)
    for i in range(1,len(li),2):
        str=str.replace(li[i],li[i+1])
    return str
def preffix(word):
    if len(word)<3:
        return word
    else:
        while len(word)>=3:
            tempp=word[ :3]
            p=codecs.open("prefix.txt",'r', encoding = 'utf-8' )
            pr = p.read()
            y=pr.split()
            p.close()
            if y.__contains__(tempp):
                if (condition1(tempp,word)):
                    word=condition2(tempp,word)
                    return word
                else:
                    word=word[3: ]
                    preffix(word)
            else:
                tempp=word[ :2]
                if  y.__contains__(tempp):
                    if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                    else:
                        word=word[2: ]
                        preffix(word)
                else:
                    tempp=word[ :1]
                    if  y.__contains__(tempp):
                        if (condition1(tempp,word)):
                            word=condition2(tempp,word)
                            return word
                        else:
                            word=word[1: ]
                            preffix(word)

                    else:
                        return word

    return word
def suffix(word):
    if len(word)<3:
        return word
    else:
        while len(word)>=3:
            tempp=word[len(word)-3: ]
            s=codecs.open("sufix.txt",'r', encoding = 'utf-8' )
            su = s.read()
```

```
            x=su.split()
            s.close()
            if x.__contains__(tempp):
               if (condition1(tempp,word)):
                  word=condition2(tempp,word)
                  return word
               else:
                  word=word[ :len(word)-3]
                  suffix(word)
            else:
               tempp=word[len(word)-2: ]
               if  x.__contains__(tempp):
                  if (condition1(tempp,word)):
                     word=condition2(tempp,word)
                     return word
                  else:
                     word=word[ :len(word)-2]
                     suffix(word)
               else:
                  tempp=word[len(word)-1: ]
                  if  x.__contains__(tempp):
                     if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                     else:
                        word=word[ :len(word)-1]
                        suffix(word)

                  else:
                     return word


      return word
   def fileReader(filename):
      indexterms=[]
      infile=codecs.open(filename, encoding='utf-8')
      lterms=infile.readlines()
      infile.close()
      for str in lterms:
         if str.endswith('\n'):
            str=str.rstrip('\n')
         indexterms.append(str)
      return indexterms
   def docsReader(pst):
      ldocid=[]
      infile=open(pst)
      lterms=infile.readlines()
      infile.close()
      for str in lterms:
         if str.endswith('\n'):
            str=str.rstrip('\n')
         ldocid.append(str)
```

```
    return ldocid
def indexTerm(ldocid):
    fileterms=[]
    List2 = []
    List3 = []
    L7 = []
    List4 = []#store terms in each document in sorted order??
    List22 = []#store unique terms from the document
    indexterm=[]
    cff = {}#store terms and its collection frequency in a dictinay
    tff = {}#store terms and its term frequency in a dictinay
    dff = {}#store terms and its document frequency in a dictinay
    posFile=[]#store docid and tf of terms
    docID = 0#store docid
    L9 = []#L9 store term, docid and tf for all document
    L10=[]#L10 stor the docid and tf
    stop=codecs.open("stopw.txt",'r', encoding = 'utf-8')
    stopw=stop.read()
    # the loop that reads the id of each document togather
    for docid in ldocid:
        docID = docID + 1
        List5 = []
        aList1 = []#store the key of dictionay which is unique index terms as a list
        aList2 = []#store the value(tf)
        aList3 = []#store the tf for further purpose
        res = []
        L8 = []#store term,docid and tf
        #fileterms store terms in each documents
        fileterms=fileReader(docid)
        for term in fileterms:
            List1 = []#store every index in every docs terms as a list
            L6 = []#have terms
            indexterms=[]#indexterms[] store each terms in the documents that are stemmed and punc removed
            for index in term.split():
                if index.isalpha():
                    stop_words=codecs.open("stopw.txt",'r',encoding = 'utf-8')
                    n=stop_words.read()
                    s=n.split()
                    stop=0
                    for i in range(0,len(s)):
                        if suffix(preffix(punctuationremove(index)))in s[i]:
                            stop=1
                    if stop==0:
                        indexterms.append(suffix(preffix(punctuationremove(index))))
                for k in indexterms:
                    List1.append(k)
                List4 = sorted(List4 + List1)
                List22 = list(set(List4))
                List5 = sorted(List5 + List1)
    #term frequency count loop
        for tf in List5:
            try:
```

```
            tff[tf] = tff[tf] + 1
        except:
            tff[tf] = 1
        tfff = sorted(tff.items())
    for keyValue in range(len(tfff)):
        aList2.append(tfff[keyValue][0])#index terms
        aList1.append(tfff[keyValue][1])#tf of terms
    for kk in aList1:
        aList3.append(kk)#aList3 store tf for futher process purpose
        posFile.append((docID, kk))
    ListDict = dict(zip(aList2,aList3))
    for key in ListDict:
        L6.append(key)#have terms
        L8.append((key, docID, ListDict[key]))#store term,docid and tf for a single doc
    L7 = sorted(L7 + L6)
    tff = {}
    List3 = sorted(List3 + List2)
    L9 = sorted(list(L9 + L8))#L9 store term, docid and tf for all document
#the loop count the collection frequency of terms
for cf in List4:
    try:
        cff[cf] = cff[cf] + 1
    except:
        cff[cf] = 1
    cfreq = sorted(cff.items())
#the loop count the document frequency of terms
for df in L7:
    try:
        dff[df] = dff[df] + 1
    except:
        dff[df] = 1
    dfreq2 = sorted(dff.items())
y=[]#store terms by extracting from cf
for kk2 in range(len(cfreq)):
    y.append(cfreq[kk2][0])

TDFCF = []#store index term df and cf as a single list
for k1 in range(len(dfreq2) and len(cfreq)):
    TDFCF.append((dfreq2[k1][0], dfreq2[k1][1], cfreq[k1][1]))
L33 = []
ID = 0
L44=[]
for j in ldocid:
    ID = ID + 1
    f=codecs.open(j, 'r', encoding = 'utf-8')
    f2=f.read()
    dic={}
    for term in y:
        if term not in dic:
            found=f2.find(term)
            while found >-1:
                if term not in dic:
```

```
                    dic[term]=found
                    found=f2.find(term, found+1)
                else:
                    dic[term]=((str(dic[term]))+ "    "+(str(found)))
                    found=f2.find(term, found+1)
            for k in range(len(L9)):
                if term == L9[k][0] and ID == L9[k][1]:
                    L10.append((term, ID, L9[k][2], dic[term]))
                    L33.append((term,ID))
                    L44=sorted(L10)
                else: continue
        f.close()
        L11=[]#store term, df and cf
        for i in range(len(TDFCF)):
            L22 = []
            L11.append((normal(TDFCF[i][0]), TDFCF[i][1], TDFCF[i][2]))
            for j in range(len(L33)):
                if TDFCF[i][0] == L33[j][0]:
                    L22.append(L33[j][0])
                else: continue
            L11.append(tuple(L22))
        fiLe=codecs.open("collectionfrequency.txt", 'w', encoding = 'utf-8')
        fiLe.write(str("Term")+ "\t" + str("CF") + "\t" + "\n")
        for i in range(0, len(L11), 2):
            fiLe.write(L11[i][0]  + "\t" + str(L11[i][2]) + "\t")
            fiLe.write("\n")
        fiLe.close()
        fiLe=codecs.open("vocabularyFile.txt", 'w', encoding = 'utf-8')
        fiLe.write(str("Term") + "\t" + str("DF") + "\t" + str("CF") + "\t" + "\n")
        for i in range(0, len(L11), 2):
            fiLe.write(L11[i][0] + "\t" + str(L11[i][1]) + "\t" + str(L11[i][2]) + "\t")
            fiLe.write("\n")
        fiLe.close()
        fiLe=codecs.open("postingFile.txt", 'w', encoding = 'utf-8')
        #fiLe.write(str("docID") + "\t" + str("TF") + "\n")
        for key in range(len(L44)):
            print >> fiLe,(L44[key][1]),"\t" ,
            print >> fiLe,(L44[key][2]),"\t" ,
            print >> fiLe,(L44[key][3])
        fiLe.close()

    def test(indexlist, ldocid):
        totDocs = int(len([fiLe for fiLe in os.listdir("./corpus") if os.path.isfile(os.path.join("./corpus", fiLe))]))
    return test(indexTerm(docsReader(path)), docsReader(path))
ob=amhariIndexing()
f1=open("documentsID.txt",'w')
for i in os.listdir('C:\Users\Amanuel\Desktop\probability prototype2\corpus'):
    docs="corpus"
    docs=docs + '/' + i
    f1.write(docs)
    f1.write("\n")
f1.close()
```

ob.index("documentsID.txt")

- Searching Module

```
import os
import sys
import math
import codecs
import string
print "\n\t\t\tእንኳን ወደ አማርኛ የመረጃ ማእከል በደህና መጡ!"
print "....................................................................................."
print
print "\n\t\t\tመረጃ ፍለጋ ሊዘገይ ስለሚችል እባኮን ትንሽ ይጠበቁ"
print "....................................................................................."
print
def relevancefeedback(UserQuery,vector,flag):
   while flag<3:
      L1,L2,L01,L02=[],[],[],[]
      N = int(len([fiLe for fiLe in os.listdir("./corpus") if os.path.isfile(os.path.join("./corpus", fiLe))]))
      querylist=[]
      userInput=UserQuery.split()
      for i in userInput:
         querylist.append(punctuationremove(suffix(preffix(i))))
      Relevancedoc=[]
      termination=int(raw_input("\n\nየቀረቡት መረጃዎች አጥጋቢ ናቸዉ? ከሆኑ (1) ይጫኑ ካልሆኑ (0) ይጫኑ:"))
      if termination ==1:
         print "እናመሰግናለን ደህናይሁኑ"
         sys.exit()
      else:
         cont=1
         dh=int(raw_input("\n\nምንያህሎቹ መረጃዎች አስፈላጊ ናቸዉ? :"))
         print "የዶኩመንትቶቹን መለያ ቁጥር እዚ ያስገቡ!"
         while cont <= dh:
            R=int(raw_input("===> " +str(cont)+ unicode(" ኛ :", 'utf-8')))
            Relevancedoc.append(R)
            Relevancedocument=sorted(Relevancedoc)
            cont=cont+1
         ch=int(raw_input("\n\nምንያህሎቹ መረጃዎች የማያስፈልጉ ናቸዉ? :"))
         contt=1
         Ireldocs=[]
         while contt <= ch:
            IR=int(raw_input("===> " +str(contt)+ unicode(" ኛ :", 'utf-8')))
            Ireldocs.append(IR)
            Irelevantdocument=sorted(Ireldocs)
            contt=contt+1
         NN=len(Irelevantdocument)
         check=sorted(Relevancedocument+Irelevantdocument)
         provector1=[]
         for n in range(len(querylist)):
            for j in range(1, N+1):
```

```
        for t in range(len(vector)):
            if j==vector[t][1]:
                if querylist[n]==vector[t][0]:
                    provector1.append((vector[t][0],j,1))
                    temp1=sorted(provector1)
                else:
                    pass
rel=[]

for g in range(len(Relevancedocument)):
    for k in range(len(temp1)):
        if Relevancedocument[g]==temp1[k][1]:
            rel.append((temp1[k][0],temp1[k][1],1,1))

cont=0
rel2=[]
for i in range(len(querylist)):
    cont=0
    for j in range(len(rel)):
        for k in range(len(Relevancedocument)):
            if querylist[i]==rel[j][0] and rel[j][1]==Relevancedocument[k]:
                if rel[j][3]==1 and rel[j][1]==k and rel[j][2]==1:
                cont=cont+1
    rel2.append((querylist[i],cont))
count=0
forweightt=[]
for i in range(len(querylist)):
    count=0
    for j in range(len(temp1)):
        if querylist[i]==temp1[j][0]:
            for k in range(len(check)):
                if check[k]==temp1[j][1]:
                    count=count+1
    forweightt.append((querylist[i],count))
weightt=[]
R=len(Relevancedocument)
B=R+NN
for i in range(len(forweightt)):
    mat1=float((rel2[i][1]+ 0.5))
    mat01=float((B)-(R))
    mat02=float(forweightt[i][1])
    mat03=(mat01)-(mat02)
    mat04=(mat1)+(mat03)
    mat2=float((rel2[i][1]) + 0.5)
    mat3=float((R-(rel2[i][1]))+0.5)
    mat4=float(((forweightt[i][1])- (rel2[i][1])) + 0.5)
    sim1=(mat04*mat2)
    sim2=(mat3*mat4)
    sim=float(sim1/sim2)
    simm=math.log((sim),10)
    weightt.append(simm)
countt=1
```

```
rankk=[]
countt=1
scoree=[]
scoree2=[]
for i in range(len(querylist)):
   rankk.append((querylist[i],countt))
   countt=countt+1
for i in range(len(querylist)):
   for j in range(len(provector1)):
      if querylist[i]==provector1[j][0]:
         for m in range(1,(N+1)):
            if provector1[j][1]==m:
               scoree.append((m,rankk[i][1]))
               scoree2=sorted(scoree)

            else:
               pass
addd=0
scoreee=[]

for k in range(len(weightt)):
   for j in range(1,(N+1)):
      for i in range(len(scoree2)):
         if scoree2[i][0]==j:
            addd=(addd)+(weightt[(scoree2[i][1])-1])
            scoreee.append((scoree2[i][0],addd))
      addd=0
   break
Ran=[]
if len(userInput)>0:
   if len(userInput)>=1:
      finalRess={}
      finalRess=dict(scoreee)
      Rankin=[]
      Rankingg=[]
      Rankingg=finalRess.keys()
      f11=open("documentsID.txt", 'r')
      f22=f11.readlines()
      DocLL=[]
      for i in f22:
         recordd=i.split()
         DocLL.append(recordd)
      DocNN=1
      DocL22=[]
      pathListt=[]
      for i in DocLL:
         DocL22.append((DocNN,i[0]))
         DocNN=DocNN+1
      docDictt=dict(DocL22)
      chekpointt=0
      for i in range(0,len(Rankingg)):
         if finalRess[Rankingg[i]] !=0:
```

```
            chekpointt=1
        for i in range(len(Rankingg)):
            for j in range(i+1,len(Rankingg)):
                if finalRess[Rankingg[i]]<finalRess[Rankingg[j]]:
                    rankk=Rankingg[i]
                    Rankingg[i]=Rankingg[j]
                    Rankingg[j]=rankk

        if chekpointt==1:
            print "\n\t\tባስገቡት መጠይቅ መሰረት የተገኙት መረጃዎች  እንደሚከተለው  ቀርበዋል! "
            print"                                    "
            print "ቅደም ተከተል==========================የመረጃዉ መለያ
ቁጥር============================የመረጃዎቹ መመሳሰል ልኬት"
            print
            for i in range(len(Rankingg)):
                if finalRess[Rankingg[i]]>=1.5:
                    pathListt.append(docDictt[Rankingg[i]])
                    print
"_____________________________________________________________________________________"
                    print "",i+1, ".....", docDictt[Rankingg[i]], ".....", finalRess[Rankingg[i]]
                    print
"_____________________________________________________________________________________"
                    f11=codecs.open(docDictt[Rankingg[i]],'r', encoding="utf-8")
                    print
                    print f11.read(50), "..."
                    print
                    f11.close()
            docs = './'
            ch=int(raw_input("\n\nስንት መረጃ ማየት ይፈልጋሉ? : "))
            count=1
            while count <= ch:
                docs = './'
                pathF = raw_input("\nየትኛውን ዶኪውመንት መመልከት ይፈልጋሉ? : ")
                if pathF in pathListt:
                    docs = docs + '/' + pathF
                    showfile = codecs.open(docs,'r', encoding="utf-8")
                    for line in showfile:
                        print line,
                    showfile.close()
                    pathF = ""
                count=count+1
            flag=flag+1
            relevancefeedback(UserQuery,vector,flag)
        else:
            print "ይቅርታ! ከጠየቁት መጠይቅ ጋር የሚመሳሰል መረጃ አልተገኘም።"
        print "ከዚ ዙር ቦሃላ ስይስተሙ መረጃን የመስጠት አቅሙ እየቀነሰ ስለሚሄድ እንደገና መጠይቆን አስተካክለዉ ቢመለሱ ይመከራል!!"
        loop=raw_input("\n\nመጠይቆን መቀጠል ይፈልጋሉ? ከፈለጉ 1 ቁጥርን ይጫኑ፤  መጠይቆን ከጨረሱ ሌላ የፊደል ቁልፉን ይጫኑ!
")
        if loop == '1':
            flag=0
            relevancefeedback(UserQuery,vector,flag)
        else:
```

```
                print "\n\t\tእናመሰግናለን!  ደህና ይሁኑ!!"
                sys.exit()
            else:
                print "ያስገቡት መጠይቅ ተፈላጊ የሆኑ መረጃዎችን ለማግኘት አያስችልም።"
                readIndexData()
        else:
            print "ምንም ዓይነት መጠይቅ አላስገቡም፤ እባክዎ መጠይቆን እንደገና ያስገቡት! "
            readIndexData()
def condition1(affix, tempp):
    if len(tempp)> 2:
        if affix.__eq__(u'\u12a8'):
            return 1
        elif affix.__eq__(u'\u1260'):
            return 1
        elif affix.__eq__(u'\u129E\u127D'):
            return 1
        elif affix.__eq__(u'\u1295'):
            return 1
        elif affix.__eq__(u'\u1275'):
            return 1
        elif affix.__eq__(u'\u1290\u1275'):
            return 1
        elif affix.__eq__(u'\u12CE\u127D'):
            return 1
        elif affix.__eq__(u'\u127D'):
            return 1
        elif affix.__eq__(u'\u127D\u1295'):
            return 1
        elif affix.__eq__(u'\u127D\u1201'):
            return 1
        elif affix.__eq__(u'\u1278\u12CD'):
            return 1
        elif affix.__eq__(u'\u1271'):
            return 1
        elif affix.__eq__(u'\u1279'):
            return 1
        elif affix.__eq__(u'\u12CA'):
            return 1
        elif affix.__eq__(u'\u12CA\u1275'):
            return 1
        elif affix.__eq__(u'\u12CD\u12EB\u1295'):
            return 1
        elif affix.__eq__(u'\u1277\u120D'):
            return 1
        elif affix.__eq__(u'\u121D'):
            return 1
        elif affix.__eq__(u'\u1275') or affix.__eq__(u'\u12A0'):
            return 1
        elif affix.__eq__(u'\u129B'):
            return 1
        elif affix.__eq__(u'\u1290\u1275') or affix.__eq__(u'\u12A5'):
            return 1
```

```
    elif affix.__eq__(u'\u121D') or affix.__eq__(u'\u12A5'):
        return 1
    elif affix.__eq__(u'\u1208\u1275'):
        return 1
  else:
    return 0

def condition2(affix,tempp):
  if len(tempp) > 2:
    if affix.__eq__(u'\u12A8'):
      if tempp.__eq__(u'\u12A8\u1265\u1275') or tempp.__eq__(u'\u12A8\u1208\u12A8\u1208') or
        tempp.__eq__(u'\u12A8\u1290\u12A8\u1290') or tempp.__eq__(u'\u12A8\u1228\u12A8\u1228') or
        tempp.__eq__(u'\u12A8\u1128\u12A8\u1218') or tempp.__eq__(u'\u12A8\u1230\u12A8\u1230') or
        tempp.__eq__(u'\u12A8\u1270\u12A8\u1270') or tempp.__eq__(u'\u12A8\u1270\u121B') or
        tempp.__eq__(u'\u12A8\u1228\u1295') or tempp.__eq__(u'\u12A8\u1230\u120D') or
        tempp.__eq__(u'\u12A8\u134D\u1270\u129B') or tempp.__eq__(u'\u12A8\u1265\u1275'):
         return tempp
      else:
         tempp = tempp[1: ]
         tempp = preffix(tempp);
         return tempp
    elif affix.__eq__(u'\u1260'):
      if tempp.__eq__(u'\u1260\u123D\u1273') or tempp.__eq__(u'\u1260\u1300\u1275') or
        tempp.__eq__(u'\u1260\u1228\u1260\u1228') or tempp.__eq__(u'\u1260\u1230\u1260\u1230'):
         return tempp
      else:
         tempp = tempp[1: ]
         tempp = preffix(tempp);
         return tempp
    elif affix.__eq__(u'\u129B'):
      if tempp.__eq__(u'\u1260\u123D\u1273'):
         return tempp
      else:
         tempp = tempp[ :-1]
         tempp = preffix(tempp);
         return tempp
    elif affix.__eq__(u'\u1295'):
      if tempp.__eq__(u'\u12D8\u1218\u1295') or tempp.__eq__(u'\u1261\u12F5\u1295') or
        tempp.__eq__(u'\u123D\u134B\u1295') or tempp.__eq__(u'\u12C8\u1308\u1295') or
        tempp.__eq__(u'\u1261\u12F5\u1295') or tempp.__eq__(u'\u123D\u134B\u1295') or
        tempp.__eq__(u'\u12C8\u1308\u1295') or tempp.__eq__(u'\u132D\u1241\u1295') or
        tempp.__eq__(u'\u1205\u133B\u1295') or tempp.__eq__(u'\u12A5\u12CD\u1295') or
        tempp.__eq__(u'\u12A5\u121D\u1295') or tempp.__eq__(u'\u12D8\u1348\u1295'):
         return tempp
      else:
         tempp = tempp[ :-1]
         tempp=sadisConverter(tempp)
         tempp = suffix(tempp)
         return tempp
    elif affix.__eq__(u'\u1208\u1275'):
      if tempp.__eq__(u'\u12A0\u1208\u1275') or tempp.__eq__(u'\u12A5\u1208\u1275') or
        tempp.__eq__(u'\u1201\u1208\u1275'):
```

```
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp);
        return tempp
elif affix.__eq__(u'\u1275'):
    if tempp.__eq__(u'\u12A0\u1263\u1275') or tempp.__eq__(u'\u12A0\u12EB\u1275') or
      tempp.__eq__(u'\u1205\u12ED\u12C8\u1275') or tempp.__eq__(u'\u12A0\u122B\u12CA\u1275') or
      tempp.__eq__(u'\u1230\u12A3\u1275') or tempp.__eq__(u'\u1218\u1265\u1275') or
      tempp.__eq__(u'\u1218\u1230\u1228\u1275') or tempp.__eq__(u'\u12C8\u1245\u1275') or
      tempp.__eq__(u'\u1325\u1228\u1275') or tempp.__eq__(u'\u1275\u122D\u134D') or
      tempp.__eq__(u'\u1201\u1208\u1275') or tempp.__eq__(u'\u1236\u1235\u1275') or
      tempp.__eq__(u'\u12A0\u122B\u1275') or tempp.__eq__(u'\u12A0\u121D\u1235\u1275') or
      tempp.__eq__(u'\u1235\u12F5\u1235\u1275') or tempp.__eq__(u'\u1230\u1263\u1275') or
      tempp.__eq__(u'\u1235\u121D\u1295\u1275') or tempp.__eq__(u'\u1275\u121D\u1205\u122D\u1275')
      or tempp.__eq__(u'\u1325\u1245\u121D\u1275') or tempp.__eq__(u'\u1233\u121D\u1295\u1275') or
      tempp.__eq__(u'\u12AD\u122B\u12CA\u1275'):
      return tempp
    else:
      tempp = tempp[ :-1]
      #change to sades
      tempp = suffix(tempp);
      return tempp
elif affix.__eq__(u'\u1290\u1275'):
    if tempp.__eq__(u'\u12A5\u12CD\u1290\u1275') or tempp.__eq__(u'\u12A5\u121D\u1290\u1275'):
      return tempp
    else:
      tempp = tempp[ :-1]
      #change to sades
      tempp = suffix(tempp);
      return tempp
elif affix.__eq__(u'\u12CE\u127D'):
    if tempp.__eq__(u'\u1230\u12CE\u127D'):
      str1 = tempp[-1: ]
      return tempp
    else:
      tempp = tempp[ :-2]
      #change to sades
      tempp = suffix(tempp)
      return tempp
elif affix.__eq__(u'\u129E\u127D'):
    if tempp.__eq__(u'\u12F3\u129E\u127D'):
      tempp = tempp[ :-1]
      ##change to sades
      return tempp
    else:
      tempp = tempp[ :-1]
      #change to sades
      tempp = suffix(tempp)
      return tempp
elif affix.__eq__(u'\u127D'):
```

```
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D\u1295'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D\u1201'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1278\u12CD'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1271'):
        ##change to sades
       tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1279'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u12CA'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u12CA\u1275'):
    if tempp.__eq__(u'\u12A0\u122B\u12CA\u1275'):
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp);
        return tempp
elif affix.__eq__(u'\u12CD\u12EB\u1295'):
        tempp = tempp[ :-3]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1277\u120D'):
    if tempp.__eq__(u'\u121E\u1277\u120D'):
        tempp = tempp[ :-1]
        ##change to sades
        return tempp
    else:
```

```
            tempp = tempp[ :-2]
            #change to sades
            tempp = suffix(tempp)
            return tempp
        elif affix.__eq__(u'\u121D'):
          if tempp.__eq__(u'\u1308\u12F3\u121D') or tempp.__eq__(u'\u1230\u120B\u121D') or
            tempp.__eq__(u'\u12A0\u1208\u121D'):
            return tempp
          else:
            tempp = tempp[ :-1]
            #change to sades
            tempp = suffix(tempp);
            return tempp
    else:
      return tempp
def punctuationremove(str):
    if str.endswith('\n'):
      str=str.rstrip('\n')
    if str.startswith(' '):
      str=str.lstrip(' ')
    p=codecs.open("Punctuation.txt",'r', encoding = 'utf-8')
    Punctuation = p.read()
    for pc in Punctuation:
      if str.endswith(pc):
        str=str.rstrip(pc)
      if str.startswith(pc):
        str=str.lstrip(pc)
    for j in Punctuation:
      for i in range(1,len(str)):
        if j in Punctuation and j in str:
          str=str.strip(j)
    return str
def normal(str):
    sim=codecs.open("similarWord.txt",'r', encoding = 'utf-8' )
    similarWord=sim.read()
    li1=[]
    for m in similarWord.split():
      li1.append(m)
    for i in range(1,len(li1),2):
      if str==li1[i] or str==li1[i+1]:
        str=li1[i+1]
    temp=codecs.open("similarChar.txt",'r', encoding = 'utf-8' )
    similarChar=temp.read()
    li=[]
    for j in similarChar.split():
      li.append(j)
    for i in range(1,len(li),2):
      str=str.replace(li[i],li[i+1])
    return str
def preffix(word):
    if len(word)<3:
      return word
```

```
    else:
        while len(word)>=3:
            tempp=word[ :3]
            p=codecs.open("prefix.txt",'r', encoding = 'utf-8' )
            pr = p.read()
            y=pr.split()
            p.close()
            if y.__contains__(tempp):
                if (condition1(tempp,word)):
                    word=condition2(tempp,word)
                    return word
                else:
                    word=word[3: ]
                    preffix(word)
            else:
                tempp=word[ :2]
                if  y.__contains__(tempp):
                    if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                    else:
                        word=word[2: ]
                        preffix(word)
                else:
                    tempp=word[ :1]
                    if  y.__contains__(tempp):
                        if (condition1(tempp,word)):
                            word=condition2(tempp,word)
                            return word
                        else:
                            word=word[1: ]
                            preffix(word)

                    else:
                        return word

    return word
def suffix(word):
    if len(word)<3:
        return word
    else:
        while len(word)>=3:
            tempp=word[len(word)-3: ]
            s=codecs.open("sufix.txt",'r', encoding = 'utf-8' )
            su = s.read()
            x=su.split()
            s.close()
            if x.__contains__(tempp):
                if (condition1(tempp,word)):
                    word=condition2(tempp,word)
                    return word
                else:
```

```
                word=word[ :len(word)-3]
                suffix(word)
            else:
                tempp=word[len(word)-2: ]
                if  x.__contains__(tempp):
                    if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                    else:
                        word=word[ :len(word)-2]
                        suffix(word)
                else:
                    tempp=word[len(word)-1: ]
                    if  x.__contains__(tempp):
                        if (condition1(tempp,word)):
                            word=condition2(tempp,word)
                            return word
                        else:
                            word=word[ :len(word)-1]
                            suffix(word)

                    else:
                        return word


    return word
def readIndexData():
    flag=0
    L1,L2,L01,L02=[],[],[],[]
    N = int(len([fiLe for fiLe in os.listdir("./corpus") if os.path.isfile(os.path.join("./corpus", fiLe))]))
    a=codecs.open("vocabularyFile.txt", 'r', encoding = 'utf-8')
    b=a.readlines()
    for j in b:
        c=j.split()
        f=c[0],int(c[1])
        L01.append(f)
    for i in L01:
        L1.append((i[0],i[1]))
    a.close()
    n=codecs.open("postingFile.txt", 'r', encoding = 'utf-8')
    m=n.readlines()
    for i in m:
        s=i.split()
        f2=int(s[0]),int(s[1])
        L02.append(f2)
    for i in L02:
        L2.append((i[0],i[1]))
    n.close()
    L3,L6,vector=[],[],[]
    d,h,g,a,m,c=0,0,0,0,0,0
    while c < len(L1):
        L=[]
```

```
        L30=[]
        for h in range(L1[d][1]):
            L.append((L1[m][0], L2[a][0]))
            a = a + 1
        m = m + 1
        d = d + 1
        c=c+1
        vector = vector + L
querylist=[]
provector=[]
UserQuery=raw_input("እባኮን የሚፈልጉትን ፋይል ለማግኘት መጠይቆን ያስገቡ!:- ")
print "....."
userInput=UserQuery.split()
for i in userInput:
    querylist.append(punctuationremove(suffix(preffix(i))))
for n in range(len(querylist)):
    for j in range(1, N+1):
        for t in range(len(vector)):
            if j==vector[t][1]:
                if querylist[n]==vector[t][0]:
                    provector.append((vector[t][0],j,1))
                else:
                    pass
count=0
forweight=[]
for i in range(len(querylist)):
    count=0
    for j in range(len(provector)):
        if querylist[i]==provector[j][0]:
            count=count+1
    forweight.append((querylist[i],count))
weight=[]
for i in range(len(forweight)):
    sim1 = float(((N)-(forweight[i][1])) + 0.5)
    sim2=float((forweight[i][1]) + 0.5)
    sim=float(sim1/sim2)
    simm=math.log((sim),10)
    weight.append(simm)
count=1
rank=[]
count=1
score=[]
score2=[]
for i in range(len(querylist)):
    rank.append((querylist[i],count))
    count=count+1
for i in range(len(querylist)):
    for j in range(len(provector)):
        if querylist[i]==provector[j][0]:
            for m in range(1,(N+1)):
                if provector[j][1]==m:
                    score.append((m,rank[i][1]))
```

```
            score2=sorted(score)

        else:
            pass
add=0
scoree=[]

for k in range(len(weight)):
    for j in range(1,(N+1)):
        for i in range(len(score2)):
            if score2[i][0]==j:
                add=(add)+(weight[(score2[i][1])-1])
                scoree.append((score2[i][0],add))
        add=0
    break
if len(userInput)>0 and len(userInput)<10:
    if len(userInput)>=1 and len(userInput)<10:
        finalRes={}
        finalRes=dict(scoree)
        Ranking=[]
        Ranking=finalRes.keys()
        f1=open("documentsID.txt", 'r')
        f2=f1.readlines()
        DocL=[]
        for i in f2:
            record=i.split()
            DocL.append(record)
        DocN=1
        DocL2=[]
        pathList=[]
        for i in DocL:
            DocL2.append((DocN,i[0]))
            DocN=DocN+1
        docDict=dict(DocL2)
        chekpoint=0
        for i in range(0,len(Ranking)):
            if finalRes[Ranking[i]] !=0:
                chekpoint=1
        for i in range(len(Ranking)):
            for j in range(i+1,len(Ranking)):
                if finalRes[Ranking[i]]<finalRes[Ranking[j]]:
                    rank=Ranking[i]
                    Ranking[i]=Ranking[j]
                    Ranking[j]=rank
        if chekpoint==1:
            print "\n\t\tባስገቡት መጠይቅ መሰረት የተገኙት መረጃዎች እንደሚከተለው ቀርበዋል! "
            print"                                    "
            print "ቅደም ተከተል==========================የመረጃዉ መለያ
          ቁጥር============================የመረጃዎቹ መመሳሰል ልኬት"
            print
            for i in range(len(Ranking)):
                if finalRes[Ranking[i]]>1:
```

```
                pathList.append(docDict[Ranking[i]])
                print "_____"
                print "",i+1, ".....", docDict[Ranking[i]], ".....", finalRes[Ranking[i]]
                print "_____"
                f=codecs.open(docDict[Ranking[i]],'r', encoding="utf-8")
                print
                print f.read(50), "..."
                print
                f.close()
            docs = './'
            ch=int(raw_input("\n\nስንት መረጃ ማየት ይፈልጋሉ? : "))
            count=1
            while count <= ch:
               docs = './'
               pathF = raw_input("\nየትኛውን ዶኪውመንት መመልከት ይፈልጋሉ? : ")
               if pathF in pathList:
                  docs = docs + '/' + pathF
                  showfile = codecs.open(docs,'r', encoding="utf-8")
                  for line in showfile:
                     print line,
                  showfile.close()
                  pathF = ""
               count=count+1
            relevancefeedback(UserQuery,vector,flag)
         else:
            print "ይቅርታ! ከጠየቁት መጠይቅ ጋር የሚመሳሰል መረጃ አልተገኘም።"
         loop=raw_input("\n\nማሰተካከል ይፈልጋሉ? ከፈለጉ 1 ቁጥርን ይጫኑ፤  መጠይቆን ከጨረሱ ሌላ የፊደል ቁልፉን ይጫኑ! ")
         if loop == '1':
            readIndexData()
         else:
            print "\n\t\tእናመሰግናለን!  ደህና ይሁኑ!!"
            sys.exit()

      else:
         print "ያስገቡት መጠይቅ ተፈላጊ የሆኑ መረጃዎችን ለማግኘት አያስችልም።"
         readIndexData()
   else:
      print "ምንም ዓይነት መጠይቅ አላስገቡም፤ ወይም መጠይቆ ከሚፈቀደዉ መጠን በላይ ነዉ! እባክዎ መጠይቆን እንደገና ያስገቡት! "
      readIndexData()
readIndexData()
```

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

PROBABILISTIC INFORMATION RETRIEVAL SYSTEM FOR AMHARIC LANGUAGE

AMANUEL HIRPA MADESSA

June 2012

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

PROBABILISTIC INFORMATION RETRIEVAL SYSTEM FOR AMHARIC LANGUAGE

BY:

AMANUEL HIRPA MADESSA

A thesis submitted to the School of Information Science of Addis Ababa University in partial fulfillment of the requirement for the Degree of Master of Science in Information Science

June 2012

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

PROBABILISTIC INFORMATION RETRIEVAL SYSTEM FOR AMHARIC LANGUAGE

BY:

AMANUEL HIRPA MADESSA

Name and signature of members of the examining board

Name	Title	Signature	Date
_____	Chairperson	_____	_____
Million Meshesha (PhD)	Advisor	_____	_____
Dereje Teferi (PhD)	Examiner	_____	_____

DEDICATED TO:

MY SISTER MEKDES HIRPA (FIKIR)

DECLARATION

This thesis is my original work. It has not been presented for a degree in any other university and all sources of material used for the thesis have been duly acknowledged.

_____

AMANUEL HIRPA MADESSA

The thesis has been submitted for examination with my approval as university advisors

_____

MILLION MESHESHA (PhD)

June 2012

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHM

LIST OF ACRONYMS AND ABBREVIATION

BIM	Binary Independent Model
BNM	Bayesian Network Model
DAG	Directed Acyclic Graph
DFR	Divergence for Randomness
EBM	Extended Boolean Model
ENA	Ethiopian News Agency
FDRE	Federal Democratic Republic of Ethiopia
GVSM	Generalized Vector Space Model
IDF	Inverse Document Frequency
INM	Inference Network Model
IR	Information Retrieval
LM	Language Model
LSI	Latent Semantic Indexing
MAP	Mean Average Precision
PRP	Probabilistic Ranking Principle
SVD	Singular Value Decomposition
TF	Term Frequency
TREC	Text Retrieval Conference
VSM	Vector Space Model

LIST OF APPENDIXES

ACKNOWLEDGMENT

First and foremost I thank God, who makes everything possible. My greatest gratitude is extended to my advisor Dr. Million Meshesha for his constructive reviews of my work and for his valuable advice giving me to overcome challenges and finish this research at the right time. I offer special thanks to my family particularly my sisters Mekdes (Fikir) and Selam (Baby) and her husband Fayisa, my brothers Tesfaye Hirpa and Tesfaye Asmera, who have been supporting me in so many ways. My thanks also goes to my fellow classmates specially my friend Solomon G/Mariam for his moral support and constructive ideas during the research. Last but not list, I would like to thank Aksum University for giving me the scholarship and taking care of my necessary needs and also Addis Ababa University giving me the chance to enroll in information science masters program.

Abstract

Nowadays, a considerable amount of information has been produced in Ethiopia. This accumulation of information is challenging for archival and searching from the existing huge amount of information particularly written in Amharic language. Thus, developing an information retrieval (IR) system for Amharic language allows searching and retrieving relevant documents that satisfy information need of users. Accordingly, few IR systems have been developed. However, those IR systems have not registered a promising performance because they are developed based on vector space model that do not have the mechanism to define user's information need using relevance feedback and query reformulation techniques unless other modules are integrated. Furthermore, the model does not define uncertainty that exists in IR systems. In order to solve these issues, probabilistic retrieval model that has the capability of reweighting query terms based on relevance feedback can be used.

In this research, a probabilistic based IR system is developed for Amharic language. Both indexing and searching module was constructed. In these modules, different text operations such as: tokenization, normalization, stemming and stop word removal are included. Then, the retrieval system is tested and the experimental results show that probabilistic based IR system returned encouraging result even without controlling the problem of synonyms and polysemous terms that exist in Amharic text. The system registered on the average 73% F-measure. Nevertheless, the performance of the system is greatly affected by synonyms and polysemous terms that exist in the language beside its richness in morphology (variant words).

Keywords: Information Retrieval, Probabilistic Model, Amharic Language.

CHAPTER ONE

INTRODUCTION

1.1. Background

Nowadays, information is everywhere that helps people to make the correct decision at the right time. As the written information becomes large in size and digital documents easily available electronically, it would be difficult to retrieve relevant documents among the accumulated document collections. This exponential growth of information records of all kinds results in the problem (phenomenon) of information explosion [1].

Before 1990s most people preferred getting information from others rather than information retrieval system [1]. However, after optimization of the effectiveness, information retrieval system becomes the most preferred means of accessing information by people. Therefore, the need to store and retrieve written information effectively and efficiently has become increasingly important.

It has been long time since people realized the importance of retrieving relevant information from a given document collection. Before 3000 B.C., the people of Sumerians find out a new way of storing clay tablet with cuneiform inscriptions to make the clay tablet accessible effectively and efficiently [2]. The need for storing and retrieving relevant information becomes important especially after the invention of paper and printing press, which leads to the problem of information explosion. Vanevar Bush first finds out the problem of information explosion in 1945. He wrote an outstanding article titled "As We May Think". This article is a major contribution for the development of automatic access to large amount of accumulated information. In 1952, his idea was applied and named information retrieval [3]. Since then, many scholars have been working on storing and retrieving relevant information from large collection of documents.

Information retrieval (IR) is the process of finding relevant documents from unstructured document collection that satisfies information need of users [1]. According to Ricardo and Ribeiro-Neto [4], "Information retrieval deals with the representation, storage, organization

and access to information items". While the representation and organization of the information items provides the user with easy access to the information he/she is seeking for, the storage of information allows accumulating knowledge to make documents easily accessible for later use. In addition, the access to information item provides the user with effective and efficient retrieval of information based on their need.

One of the mechanisms to create effective communication between reader/user and author/writer is effective IR system [5]. The main goal of information retrieval system is to retrieve relevant information, but there is no a single IR system capable of retrieving only relevant document according to the user's need. Rather, it retrieves irrelevant documents [6].

Predicting relevant documents is one of the core issues in IR system, and IR models are responsible for determining the prediction of what is relevant and what is not [4]. A number of retrieval models have been proposed since the mid 1960s. They have evolved from specific models intended for use with small structured document to recent models that have strong theoretical basis and which are intended to accommodate variety of full text document types such as: Boolean model, vector space model, probabilistic model, etc.

Current models handle documents with complex internal structure and most of them incorporate a learning or relevance feedback component that can improve performance when presented with sample relevant documents [7].

Probabilistic model is a statistical analysis model that estimates the probability of a document relevance given available evidences. It works based on the probability ranking principle, which states that "An information retrieval system is supposed to rank the documents based on their probability of relevance to the query, given all the evidence available" [8].

The probabilistic model incorporates relevance feedback and query expansion mechanisms. It is the oldest but also the hottest research area in IR. There can be variety of sources of evidence that are used by the probabilistic retrieval methods. The most common one is the statistical distribution of the terms in both the relevant and non-relevant

documents. The principle takes into account that there is uncertainty in the representation of the information need and the documents [9]. Table 1.1 depicts the defining characteristics of probabilistic model and vector space model.

	Probabilistic Model	**Vector Space Model**
Motivation	Address uncertainty in query representations	Simplify query formulation
Goal	Rank the output based on probability of relevance	Rank the output based on similarity
Methods	Use of different model	Cosine measure

Table 1.1: Characteristics of vector space model and probabilistic model

IR system is different from other kind of information system by its uncertainty nature [7]. In a database system, a query is formulated precisely to the information need and there is an exact definition of which elements of the database represent the answer. However, in IR system neither a query formulation can be assumed to represent exclusively an information need nor there is a clear procedure that decides whether a document is relevant or not. The most successful approach for handling the uncertainty nature of IR is probabilistic model.

Empirical evidences show that other models like Vector space model (VSM) and its variant models such as, Extended Boolean Model (EBM) and Generalized Vector space model (GVSM) are not attempted to define uncertainty in IR system. They do not have relevance feedback and query expansion mechanism by themselves to do with the external realities of users and information needs unless other modules are integrated to those models so as to register better performance [4][7][9].

There are different information retrieval methods which have a probabilistic basis [10]. The most widely used ones are Binary Independence Model (BIM) and Bayesian Network Model (BNM). BIM works based on representation of queries and documents (i.e. as sets

of terms) and by collecting relevance feedback data from few documents, the model then can be applied in order to estimate the probability of relevance for the remaining documents in the collection. However, in BNM graphical representation of probabilistic model is used to show probabilistic dependency between variables. The model then estimate the probability of relevance based on child and parent causal relationship that is represented by linking each parent node to the child node [4].

Several researches in the area of probabilistic retrieval model has been conducted globally. Robertson et.al [11] developed a probabilistic model based IR system for English language to solve the problem of uncertainty found in IR system. From the experiment, encouraging result is found. However, the consideration of giving the same weights for terms in different document was one of the main challenges. Rijsbergen [12] also developed probabilistic model based IR system to improve the binary independent assumption and the uncertainty nature of IR by considering the semantic relationship of query terms and document.

1.2. Statement of the Problem

These days, a considerable amount of information has been produced in Ethiopia, especially in Amharic language within organizations and outside the organizations rapidly and continuously. This information explosion is challenging for archival and searching from the existing huge amount of information that is written in Amharic language.

There are more than 80 languages in Ethiopia and Amharic is the mother tongue language for more than 25 million people [6]. According to Federal Democratic Republic of Ethiopian (FDRE) population census commission [13], it is the first language for more than 20 million and second language for over 5 million people. Accordingly, there are huge collections of documents available in the form of books, magazines, newspapers, novels, officials and legal documents, etc. Developing an IR system that enables searching and retrieving relevant document written in Amharic language is important.

Several works have been done in the last decade on Amharic information retrieval system, such as: N-Gram based automatic indexing for Amharic text [14], Amharic text retrieval

using latent semantic indexing with singular value decomposition [15], design and implementation of Amharic search engine [16] and semantic based query expansion technique for Amharic IR [6]. However, as depicted in table 1.2, IR systems based on vector space model developed for Amharic language so far has not registered a better performance. According to Justin and Rolf [9], "effective integration of more information about users need should lead to better IR performance". Vector space model and its variants do not have the mechanism to define users need using relevance feedback and query reformulation techniques unless other modules are integrated to the models. In comparison, probabilistic model by itself enables defining user's need using relevant feedback and query reformulation techniques.

As stated by Fuhr [7], the other problem of developing an IR system using vector space model is lack of handling uncertainty nature of IR. How exact is the representation of the document and query? How well is query match to documents? How relevant is the searching result to the query? Are the relevant documents uncertain? The most successful approach for coping with uncertain in IR is a probabilistic model. Probability theory seems to be the most natural way to quantify uncertainty.

Vector space model and its variants do not give emphasis for terms that are not in the user's query even though they describe the user's information need. To the contrary, probabilistic model includes some form of probabilistic relevance feedback where relevant documents reinforce each other [17][18].

There are a number of applications of probabilistic model in information retrieval system designed worldwide for different languages. The research done by Huang and Robertson [19] for Chinese language and Dolamic and Savoy [20] for Russian language are worth mentioning here.

Hence, this research is initiated to experiment the effectiveness of probabilistic model based IR system for Amharic language. This can help to exploit the advantage of probabilistic model in designing a generic IR system that modify its searching based on relevance feedback and term reweighting.

Authors	Year	Title	Corpus Type	Average Precision	Average Recall	F-measure
Betelihem [14]	2002	N-Gram based automatic indexing for Amharic text	News	0.05	0.88	0.09
Tewodros [15]	2003	Amharic text retrieval: an experiment using latent semantic indexing with singular value decomposition	News	0.71	0.55	0.61
Solomon et.al [16]	2009	Design and Implementation of Amharic Search Engine	Web Documents	0.99	0.52	0.68
Abey [6]	2011	Semantic based query expansion technique for Amharic IR	Amharic Bible	0.53	0.73	0.63

Table 1.2: Previous Amharic IR systems performance summary

To this end, this research attempts to find solution for the following research questions:

- What are the properties and word formations in Amharic language?
- What are the suitable components to design probabilistic based retrieval system?
- Does probabilistic model improve effectiveness of the IR system in searching for relevant Amharic documents?

1.3. Objective of the Study

To conduct this research the following general and specific objectives are established.

1.3.1. General Objective

The general objective of this study is to experiment the effectiveness of probabilistic IR model for searching relevant documents from Amharic text corpus so as to design an applicable Amharic information retrieval system.

1.3.2. Specific Objectives

With the aim of achieving the general objective of this study, the following specific objectives are formulated:-

- To understand concepts on probabilistic IR model and review related works from literatures to define the contribution of this study.
- To explore and identify the linguistic features of the Amharic language applicable to probabilistic model.
- To setup the experiment by organizing Amharic documents, queries and relevant judgments.
- To generate content-bearing index terms by applying text-preprocessing techniques such as, stop word removal, stemming and normalization.
- To design a prototype probabilistic Amharic IR system that can search for relevant documents from Amharic corpus.
- To evaluate the performance of the system using IR effectiveness measures such as recall, precision and F-measure.

1.4. Scope and Limitation of the Study

The scope of this research is to develop a prototype IR system by applying a binary independent method of probabilistic model. The IR system is designed to search within large size Amharic document corpus. News items are collected that covers issues such as, politics, sport, economic, social, accident, health, education, tourism and justice.

The system is developed following the procedures of the IR system. Given Amharic document corpus, text operations such as, tokenization, stop words removal, normalization

and stemming are integrated to select content-bearing index terms. Inverted file indexing is used to organize content-bearing index terms. Finally, searching module is enabled to allow users search for relevant documents that satisfy their information need. To improve the performance of the system, term-reweighting based query reformulation is also supported.

Lack of standard large size document corpus and lack of thesaurus to integrate for query expansion mechanism to control Amharic synonyms and polysemy words are the limitations of this research. Due to long processing time to construct relevance matrix based relevance judgment, considering the time constraint, the experiment was conducted only using 300 Amharic news documents.

1.5. Methodology

According to Abiy et al. [21], methodology is the theory and analysis of deciding how research should proceed, and it involves analysis of the principles and procedures followed in a research. Methodology covers the entire approach of research. Therefore, in this study methodology is used to gain fuller understanding of phenomenon, to produce a better quality documentation, to ensure consistency and requirements met completely.

The following methods and procedures are used in order to achieve the stated objectives of this study.

1.5.1. Literature Review

In order to have deeper understanding on philosophies, principles/theories and techniques of information retrieval system, particularly on the probabilistic information retrieval model, extensive literature review is conducted from books and Internet. Previous related research works, printed materials such as, journal articles, conference papers and electronic materials on the web are also assessed. Additional literature review and document analyses are made to investigate and identify the feature of Amharic text, which is important to the research in the course of Amharic text operations.

1.5.2. Dataset Preparation

In this research, Amharic documents collected from Amharic local news articles available on web site of Walta Information Center were used as a test data. Walta Information Center is a private organization, which produces and distributes news on television and radio [22].

The researcher inspired to use news articles as a test data because, news articles are easier to access, available in electronic form, sufficient in size, covers all domain areas and previous researches such as Betelihem [14] and Tewodros [15] used 100 and 200 collections of local news articles respectively.

1.5.3. Implementation Tools

Python programming language is used for developing the system. The researcher initiated to use Python because; Python's syntax is clear and readable. Experts and beginners can easily understand the code and everyone can become productive in Python very quickly [23].

It is simple to get support and fast to code. Python provides fast feedback in several ways. First, the programmer can skip many tasks that other languages require him to take. Therefore, it reduces both the cost of program maintenance and the development time. Python also encourages program reusability by implementing modules and packages. One can easily share functionality between programs by breaking the programs into modules, and reusing the modules as components of other programs. Python is also a portable programming language [24].

1.5.4. Testing Procedure

To test the system Amharic queries are formulated and relevance judgment is prepared to construct document query matrix that shows all relevant documents for each test queries. To assess the effectiveness of the proposed system (i.e., the quality of its search results) the most frequent and basic statistical measures, recall, precision and F-measure are used.

Recall is percentage of relevant documents retrieved from the database in response to users query. Precision is percentage of retrieved documents that are relevant to the query. F-measure is a weighted harmonic mean of precision and recall [1]. To visualize the performance of the system, precision and recall curve, which reflect precision at different standard recall levels were plotted.

1.6. Significance of the Study

This study is expected to produce results in several ways. The main significance is designing an applicable Amharic information retrieval system that can help users search and retrieve relevant documents written in Amharic language. It has also specific contributions. First, this study produces results that can show the advantage of applying a probabilistic information retrieval model for Amharic language. Second, it gives direction for researchers to select the best model for future works on Amharic text retrieval based on the performance achieved. Third, since the model can easily extended it can be applied to any problem where there is a need to retrieve any documents written in local Semitic languages (such as, Guragegna, Tigrigna, Siltegna etc) from large collection. Additionally, it is an academic exercise to fulfill the requirement of masters program the researcher is enrolled in.

1.7. Organization of the Research

This study is organized in to five chapters. The first chapter discusses introduction that provides background information, the problem statement, the research objective, research question, methodology, scope and limitations and significance of the study.

The second chapter reviews the literature covering the various theories/philosophies, techniques and methods of information retrieval system and various information retrieval models especially probabilistic model. In addition, the historical background of Amharic language and its writing system are covered.

The third chapter presents the technique implemented, the architecture adopted and the algorithm used in this study. The experimental settings, test results interpretations and the findings of the experiment are presented in chapter four.

Finally, chapter five presents conclusion drawn from the findings of the study and recommendations that should be considered in future researches for designing an applicable Amharic IR system.

CHAPTER TWO

LITERATURE REVIEW

Information retrieval is a key technology that has been made in the history of humankind. It is the key technology behind search engines and an everyday technology for many web users [25][26].

An IR system has been developed for serving user's purpose of; good reading and magazine articles for an assignments; finding educational material for a learning objective; finding facts for decision making etc. However, the main problem is to retrieve what is useful and leaving what is not or in other word developing a perfect retrieval system. Perfect retrieval system does not exist, because relevant judgment is based on the subjective opinion of the users. What is relevant for one user may not be relevant for other user [6][5][7].

2.1. The Retrieval Process

A simple IR system have at least three basic processes [1][4][5][7][27][28]: indexing, query formulation and the matching process.

2.1.1. The Indexing Process

The indexing process is an offline process of organizing documents using keywords extracted from the collection in which the end user of the system is not directly involved [4]. It is used to speed up access to desired information from document collection as per user's query [27].

Representing documents by its full set of words is an early way of document representation. This way of representing document can be called full text representation [4]. However, using full text indexing reduces retrieval efficiency and it asks higher computational costs [4]. Therefore, representing documents using keyword terms by

applying preprocessing techniques such as, tokenization, stop word elimination, stemming, and normalization is advisable. Figure 2.1 depicts the logical view of document that slowly swings from full text to a set of index terms.

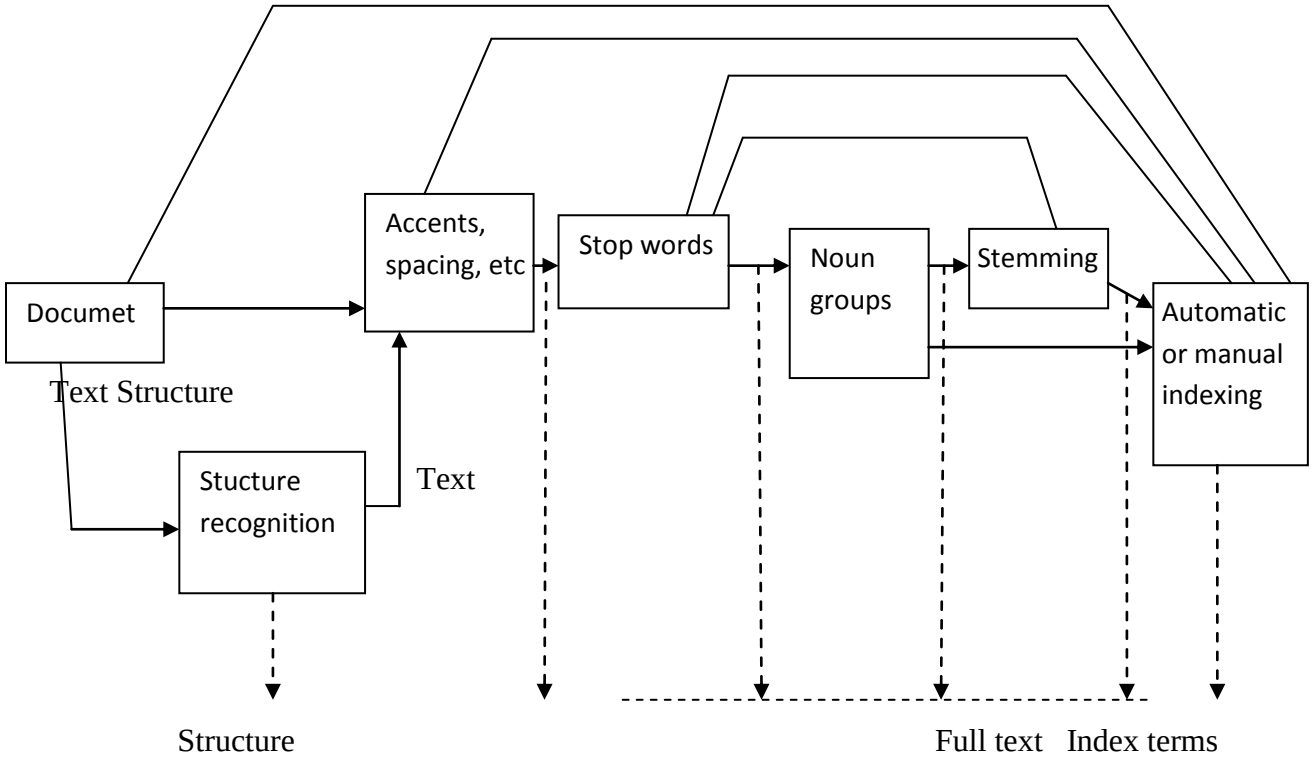

Figure 2.1: Logical view of a document

There are several index structures used for generating index terms [4]; such as, sequential file, inverted file, suffix tree, suffix array and signature file. Sequential file is an indexing structure, which access elements of record in a predetermined ordered sequence. The records are arranged serially one after another in lexicographic order on the value of some key field. Inverted file stores a map from content to its locations in a database file. Suffix tree and suffix array process the suffixes of a given string to allow particular quick implementation string operations. Signature file works based on hash coded. It is a word oriented index structure.

The most popular indexing structure is inverted file, which is also adopted in this research. Inverted file is a mechanism for indexing a text collection so as to make the searching task fast. There are two elements involving in building the inverted file [4]: the vocabulary and

the occurrence. The vocabulary file is the set of index terms in the text collection and it is organized by terms. The vocabulary file stores all of the keywords that appear in any of the documents in lexicographical order and for each word a pointer to posting file. The occurrence contains one record per term, listing all the text locations where the words occurs frequency of each term in a document [4].

2.1.2. Query Processing

Once the document is indexed and ready for retrieval process, the next step is to translate the information need of user into query language provided by the system. This process involves a serious of steps [27]. First, the user specifies his/her information need using the natural language (e.g. English, Amharic, etc.) supported by the IR system. Second, the system transforms the query into logical format by applying text operation, which is also used when the document was indexed. To refine representation of users information need and improve effectiveness of the system query operation will be employed which is discussed in section 2.3. Finally, the query is processed to retrieve relevant documents.

According to Baeza-Yates and Ribeiro-Neto [4], query language specifies regular expressions to search for strings and chapters. One of the query languages used to specify users query expression is Boolean query. It is a query language formulated based on logic concept such as AND, OR, BUT etc. Mostly Boolean query is used by Boolean retrieval model. Other models also integrate Boolean query for instance, probabilistic model. Boolean query describes the information need of the user by relating words with logic operators. For example; the query q_1 OR q_2 selects all documents which satisfy q_1 or q_2. The query q_1 AND q_2 select only documents which satisfy both q_1 and q_2. The query q_1 BUT q_2 selects all documents which satisfy q_1 but not q_2. Sometimes a logic operator NOT is used in place of BUT.

2.1.3. The Matching Process

After documents are logically organized and the user query is processed, the next step is comparing the query against the document representations, which is called, the matching process. The result of the matching is a ranked list of documents according to their

likelihood of relevance [27]. Baeza-Yates and Ribeiro-Neto [4], stated that, one central problem of any information retrieval system is predicting which documents are relevant and which are not. The ranking algorithms are used for such decision. According to Hiemstra [27], the theory behind ranking algorithms is a crucial part of information retrieval system. They attempt to display documents in decreasing order based on their similarity score with the query. Most of the time documents that are considered as relevant to users gets the biggest score and displayed at the top of the retrieved list. Thus, IR models guide the process of matching and ranking relevant documents.

There are different similarity measurement /matching techniques are developed [1] [4][28].such as, Euclidean distance, Cosine similarity measure, dot product, etc. Euclidian distance calculates "the root of square differences between coordinates of a pair of document and query terms". The similarity between vectors for document d_i and query q can be computed as:

$$sim(dj, q) = |dj - q| = \sqrt{\sum_{i=1}^{n} (w_{ij} - w_{iq})^2} \quad \ldots\ldots (2.1)$$

Where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query. Dot product/inner product calculates "the product of the magnitudes of query and document vectors". It is normalized because it favors long documents with a large number of unique terms and it measure only how many terms matched but not how many terms are not matched. Similarity between vectors for the document d_i and query q can be computed as:

$$sim(dj, q) = dj.q = \sum_{i=1}^{n} w_{ij} . w_{iq} \quad \ldots\ldots (2.2)$$

Where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query q. Cosine similarity/normalized inner product calculates "the cosine angle between query and documents vectors by projecting them into a term space". It measures similarity between d_1 and d_2 captured by the cosine of the angle x between them.

$$sim(dj, q) = \frac{\vec{dj}.\vec{q}}{|\vec{dj}|\,|\vec{q}|} = \frac{\sum_{i=1}^{n} w_{ij}.w_{iq}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2}\sqrt{\sum_{i=1}^{n} w_{iq}^2}} \ldots\ldots (2.3)$$

Where w_{ij} is the weight of term i in document j and w_{iq} is the weight of term i in the query q. The denominator involves the lengths of the vectors.

There are also other measures that are widely used in the literature to measure vector similarities. Some of these include the Dice's Coefficient, Jaccard's Coefficient, Euclidian distance and Overlap [4].

2.2. IR Models

Hiemstra [27], stated that, IR model is the mechanism of predicting and explain the need of the user given the query to retrieve relevance documents from the collection. IR models serves as blueprint so as to develop applicable IR system. In addition to that, IR models guide the matching process to retrieve a ranked list of relevant document given a query.

IR models are broadly categorized in to two approaches, which are semantic approach and statistical approach. Semantic approaches models such as, latent semantic indexing and neural network try to work on syntactic and semantic analysis. They attempt to implement some degree of understanding the natural language text that users provide. On the other hand, statistical approaches such as, vector space model and probabilistic model attempts to retrieve documents that are highly ranked in terms of statistical measure [28].

The three most widely used information retrieval model that bases on statistical approaches are [4]: Boolean, vector space, and probabilistic.

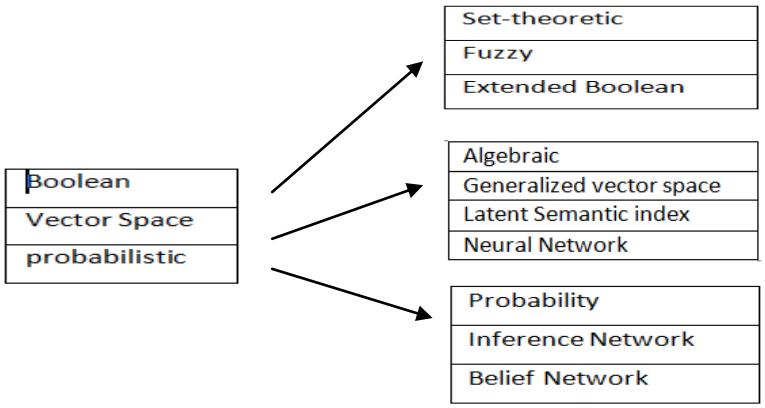

Figure 2.2: Taxonomy of IR models adopted from [4]

Figure 2.2 depict that the three IR models again can be classified based on their mathematical basis. In the boolean model, a set of index terms are used to represent document and query. Therefore, the mathematical basis can be called set theoretic. In the vector space model, a vector in a t-dimensional space is used for representing document and query, therefore, the model is algebraic. In the probabilistic model, probability theory is used for representing document and query. Therefore, the model is probabilistic. Different scholars have been proposed several alternative approaches based on their mathematical basis. For set theoretic models, the alternatives are; the fuzzy and the extended boolean models. For algebraic models, the alternatives are; the generalized vector, the latent semantic indexing, and the neural network. For probabilistic model, the alternatives are; the inference network and the belief network models [4].

2.2.1. Boolean Model

The Boolean model is the first and the simplest model of information retrieval. It works based on set theory and Boolean algebra which allowed users to specify their information need using a combination of classical Boolean operators, ANDs, ORs and NOT [1][2][29]. For instance, query t_1 and query t_2 (t_1 AND t_2) is only satisfied by a given document D_1 if and only if D_1 contains both terms t_1 and t_2. Likewise, the query "t_1 OR t_2" is satisfied by D_1 if and only if it contains t_1 or t_2 or both. On the other hand, the query "t_1 AND NOT t_2" satisfies D_1 if and only if it contains t_1 and does not contain t_2. The model views each document as just a set of words [28].The weight for index terms in Boolean retrieval model are all binary (i.e. W_{ij} E{0, 1}. The similarity of a document dj to the query q is expressed as:

$$sim(dj, q) = \begin{cases} 1 = \ if\ document\ satisfies\ the\ boolean\ query \\ 0 = Otherwise \end{cases}$$

Where, if similarity of document d_j to the query q is equal to 1 (present), the document d_j is relevant. Whereas, if similarity of document d_j to the query q is equal to 0(absent) the document is not relevant [4].

Boolean model have an advantage of giving users a sense of control over the system. It is immediately clear why a document has been retrieved given a query. If the resulting document set is either too small or too big, it directly clear which operators will produce respectively a bigger or smaller set. The other advantage of this model is the clean formalism behind the model and its simplicity [4][28].

However, the model has a number of clear disadvantages. First, most user find it difficult to translate their information need into a Boolean expression. Second, the model has not follow document ranking principle by nature, which means all retrieved documents are considered equally important. Third, the model either retrieves too few or too many documents, or sometimes not retrieves any, which might lead to null result. In addition it's sometimes creates its own problems, such as, misunderstanding and misrepresentation of the users information needs [4][29].

Several attempts has been done so far to make the model effective by developing different alternatives of the model such as fuzzy set model and the extended Boolean model. However, IR communities consider that Boolean system is less effective than ranked retrieval system [8]. Accordingly, a number of models have been proposed for this process; the widely used IR models are the Vector Space Model and the Probabilistic Model [28].

2.2.2. Vector Space Model

Vector space model is one of the commonly used statistical approaches to represent each textual document as a set of terms [27]. In this model, documents and terms are represented as vectors in a k-dimension space where each dimension corresponds to a possible document feature. The word "terms" is not inherent in the model, but terms are typically words and phrases. The values assigned to elements in the vector space describe the degree of importance of term in representing the semantics of the document. Sometimes a given term may receive a different weight in each document in which it occurs. If the term is not appearing in a given document the weight of that term will be zero in that document [4]. For a given document (d_j) the weight of the terms in it can be expressed as the coordinates of d_j in the document space. A document collection

containing a total of documents (‘d’) described by terms (‘t’) is represented as term by document matrix (T x D). Each row of this matrix is a term vector and each column of this matrix is a document vector. The element at row i, column j, is the weight of term j in document I [1].

Term list	Doc1	Doc2	Doc3	Doc4	……	Doc n
Term 1	w11	w12	w13	w14	……	w1n
Term 2	w21	w22	w23	w24	……	w2n
Term 3	w31	w32	w33	w34	……	w3n
Term 4	w41	w42	w43	w44	……	w4n
Term 5	w51	w52	w53	w54	……	w5n
……	…….	……	……	……	……	…….
Term m	wm1	wm2	wm3	wm4		wnm

Table 2.1: Example of Term Document Matrix

In this model, query is interpreted as another document in document space. If the term that is not in the collection but appear in the query, this will add additional dimension in the document space [4].

The weight of terms in the documents or in the queries assigned by using term frequency (TF) and inverse document frequency (TF*IDF) scheme which are the most successful and widely used automatic generation of weights [4]. The term frequency is the frequency of occurrence of the given term within the given document. This scheme attempts to measure the degree of importance of the term within the given document. Inverse document frequency (idf) is a frequency of a given term within an entire collection of documents. It attempts to measure how widely the term is distributed over the given collection.

The similarity of document and query in vector space model is determined by correlation between the vectors dj and vector q. The correlation is quantified by the associative coefficients based on the inner product(dot product) of the document and query vector,

where documents whose vectors are close to the query vector are more relevant to the query than documents whose vectors are far from the query vector [4].

As discussed in section 2.1.3 there are different similarity measurements. However, the most widely used by vector space model and popular similarity measure is the cosine coefficient, which measures the angle between the document vector and the query vector [4].

According to Baeza-Yates and Ribeiro-Neto[4], Vector space model (VSM) have several advantages. First, it improves retrieval performance using its term-weighting scheme. Second, the partial matching strategy of VSM allows retrieval of document approximate the query condition. Third, it sort and rank the documents according to their degree of similarity to the query using cosine similarity measurement. Finally, it is simple to implement and fast.

However, the model has also several disadvantages. It considers terms as unrelated objects in the semantic space. This means, if no common words are shared between the query and documents in text collection, the similarity value will be zero and no document will be retrieved. This is usually happened when users and authors prefer to use different words which have the same meaning [4]. Vector space model is not attempt to define uncertainty in IR system and its ranked answers sets is difficult to improve upon without the integration of query expansion and reformulation modules to the model [7].

Literature suggested that one of the alternative modeling paradigm, which is capable of solving the above mentioned problem of vector space model is probabilistic IR model [7][30].

2.2.3. Probabilistic Model

In information retrieval process, user first start with information needs which is then translated into query representation. On the other side, documents are also translated into document representation. Finally, the system attempts to determine the relevance of the document to the information need of the users. In IR models such as, Boolean and Vector

space model, given a query and document representation matching is done without considering the semantic relationship between query and documents. IR systems build upon those models has an uncertain guess of whether a document has content relevant to the information need. However probabilistic theory provides a principled foundation for such reasoning under uncertainty [1][7].

Maron and Kuhns was the first to introduce ranking by the probability of relevance, soon after Stephen Robertson brought new idea called, probabilistic ranking principle in 1977 [2][4].

Probabilistic information retrieval is the estimation of the probability of relevance that a document d_i will be judged relevant by the user with respect to query q. which is expressed as, $P(R|q,d_i)$, where, R is the set of relevant document. Typically, in probabilistic model, based on the query of user the documents are divided in to two parts. The first contain relevant documents and the second contain non-relevant (irrelevant) documents. However, the probability of any document is relevant or irrelevant with respect to user query is initially unknown. Therefore, the probabilistic model needs to guess at the beginning of searching process. The user then observe the first retrieved documents and gives feedback for the system by selecting relevant documents as relevant and irrelevant documents as irrelevant. By collecting relevance feedback data from a few documents, the model then can be applied in order to estimate the probability of relevance for the remaining documents in the collection. This process iteratively applied to improve the performance of the system so as to retrieve relevant documents which satisfies users need [4].

In probabilistic model, the order in which documents are presented to the user is to rank documents by their estimated probability of relevance with respect to the user information need. The principle behind this assumption is called, probability ranking principle (PRP) [1][4]. Probability ranking principle asserts that [12] :

> If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately

as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data

Several retrieval models have been developed, which has a probabilistic basis. The mostly used and recent developed methods of probabilistic model are, Binary Independent Model, Inference Network Model and Belief Network Model [4].

- Binary Independent model

The classical Binary independence model (BIM) was introduced in 1976 by Roberston and Sparck Jones[4]. According to Greengrass [28], the model has been called 'binary independence' because it has been assumed that to arrive at the model one must make the simplifying assumption that the document properties that serve as clues to relevance are independent of each other in both the set of relevant documents and the set of non-relevant documents.

In BIM, binary is equivalent to Boolean; queries and documents are represented as binary incidence vectors of terms. $D=\{d_1,d_2,\ldots,d_n\}$ where, $d_i=1$ if term i is present in document d and $d_i=0$ if term i is not present in document d. Moreover, query q represented by the incidence vector$\vec{q}$. As expressed above, in BIM model, the probability of P(R|d,q) that a document is relevant through the probability in terms of term incidence vectors $P(R|\vec{x},\vec{q})$ in both document and query.

- Bayesian Networks Model

Bayesian networks model was introduced by Turtle and Croft [1] as information retrieval model on the basis of probabilistic theory. Bayesian networks use a form of probabilistic graphical model. They use directed acyclic graphs (DAGs) to show probabilistic dependencies between variables, in which the nodes represent random variables, the arcs depict causal relationships between these variables. They have child and parent causal relationship, which is represented by linking each parent node to the child node. Only the root nodes are without parents [4].

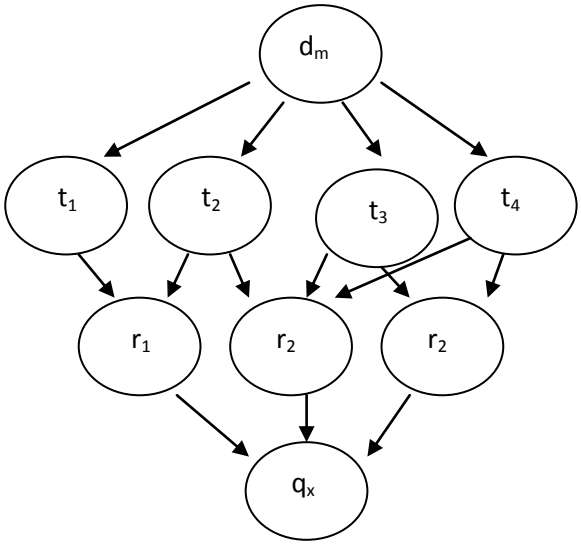

Figure 2.3: Example of Bayesian Network model adopted from [7]

Figure 2.3 depicts the graphical representation of Bayesian Network. According to Ruggeri [31], the network has two sets. The first set is the nodes and the second set is the directed edges. Arrows draw the edges between nodes that represent direct reliance among the variables. For instance, an edge from dm to t_1 represents a arithmetical reliance between the corresponding variables. Therefore, the value taken by variable t_1 depends on the value taken by variable d_m. In other word, the node dm is parent node and t_1 is the child node. Even if the edge represents direct causal relationship among the nodes, the reasoning process functions by propagating information in any path.

There are two models for information retrieval based on Bayesian networks. The first model is called inference network and the second model is called belief network.

- Bayesian Inference Network Model

According to Baeza-Yates amd Ribeiro-Neto [4], there are two traditional schools of thought in probability which are based on the frequentist view and the epistemologist view. The frequentist views probability as a statistical notion related to the laws of chance. The epistemologist views probability as a degree of belief whose specification might be devoid of statistical experimentation.

Bayesian Inference Network model is built up on epistemologist view of the information retrieval problem. It associates random variables with the index terms, the documents and the user queries. The model computes P(I|D), the probability that a user's information need (I) is satisfied given a particular document (D). This probability can be computed separately for each document in a collection. The documents can then be ranked by probability, from highest probability of satisfying the user's need to lowest [4].

- Bayesian Belief Network Model

Bayesian belief network is the use of Bayesian calculus to determine the probabilities of each node from the predetermined conditional and prior probabilities [32]. As Baeza-Yates and Ribeiro-Neto [4], stated in Bayesian belief network the users query q is modeled as a network node to its associated random variable. Whenever q completely covers the concept space C the variable is set to 1. Therefore belief network computes the probability of q, (i.e. P(q)) by the degree of coverage of the space C by q.

Document d_j is modeled as network node to its associated binary random variable. If d_j completely covers the concept space C the variable set to 1. To compute the probability of d_j ($P(d_j)$), compute the degree of coverage of the space C by d_j. In belief network documents and the user query modeled as subsets of index terms. Each subset is interpreted as concept in the concept space C [4].

The ranking principle in belief network expressed as, the degree of coverage provided to the concept d_j by the concept q. $P\{d_j|q)$ is adopted as the rank of the document d_j with respect to the query q[4].

In general, probabilistic models attempt to capture the information retrieval problem within a probabilistic framework. Unfortunately, the probabilistic model has got its own drawbacks. First, the probability theory analysis takes much more time and efforts, and it offer unnecessary theoretical burden on the researcher. Second, probabilistic model need to guess the initial separation of documents into relevant and non-relevant sets. Third, the model does not take into account the frequency with which an index term occurs inside a document. In other word, all weights are binary [4][28].

However, probabilistic model have several potential advantages [4][28]. The first, advantage is the expectation of retrieval effectiveness that is near to optimal relative to the evidence used is high. Second, it has less reliance on traditional trial and error retrieval experiments. Third, each document's probability of relevance estimate can be reported to the user in ranked output. It would presumably be easier for most users to understand and base their stopping behavior (i.e., when they stop looking at lower ranking documents).

2.3. Query Operation

Query operation is the process of representing user information need [27]. Most of users without deep knowledge of documents representation and searching environment spent their much valuable time on reformulating their queries so as to get relevant document that satisfies their need. The first query formulation is considered as initial try and the second reformulation is considered as attempt to get better and additional relevant documents. There are two ways in query reformulation [4]; query expansion and term reweighting. Query expansion is the process of adding terms which share similar or related meaning with the query terms. Term reweighting on the other hand, involves attaching various weights to query terms, so that documents carrying g a query term with the highest weight can be superior [4][27].

For the query operations to happen relevance feedback and query reformulation is necessary. Relevance feedback enables to identify relevant document retrieved for the users query and query reformulation enables to expand the original query with the new terms and reweight the terms in the expanded query so as to retrieve relevant documents which satisfy user's information need. There are various relevance feedback and query reformulation mechanisms as discussed below.

2.3.1. Relevance Feedback

Relevance feedback is a mechanism of engaging users or system in retrieval process so as to improve the final result of the IR system. There are two relevance feedback mechanisms [1][4][28][33], user relevance feedback and pseudo relevance feedback. User relevance

feedback is used to improve the final result of the IR system by involving the users in relevance feedback during the retrieval process. The procedures followed in user relevance feedback are the following. First, the user provides a query based on which the IR system returns initial relevant documents. Second, the user marks some returned documents as relevant or non-relevant. Third, the system computes a better representation of the information need based on the user feedback. Finally, the system displays a revised set of retrieval results.

Pseudo-relevance feedback is a method that eliminates the need of user's involvement. In other word, it automates the manual feedback of the user so as to get the improved result. This method works by assuming the top k ranked documents are relevant and automatically process the relevance feedback search component [33][5].

2.3.2. Query Reformulation

Query reformulation is a mechanism used to enhance the performance of the retrieval system by using two different methods called query expansion and term reweighting [4].

Query expansion technique is a process of adding a new term from relevant documents. There are two types of query expansion strategies [4]: global analysis and local analysis. Global analysis strategy examines all documents in the collection so as to expand query. Local analysis examine only documents retrieved automatically for a given query q to determine query expansion.

Term reweighting technique is a process of adjusting the weight of the term based on the users or system relevance judgment. There are different techniques of term reweighting [1]. Rocchio algorithm, probabilistic term reweighting etc.

Rocchio algorithm in one the most widely used algorithm designed for vector space model [34]. It finds a query vector which increases similarity with relevant document while decreases similarity with non-relevant documents [1].

Probabilistic reweighting technique is designed for probabilistic model. It attempts to predict the probability that a given document will be relevant to a given query. The similarity of document dj to a query q can be expressed as [4]:

$$sim(d_{j,}q) \propto \sum_{i=1}^{t} w_i\,, q.w_i, j\ (log\frac{P(k_i|R)}{1-P(k_i|R)} + log\frac{1-P(k_i|\bar{R}}{P(k_i|\overline{\bar{R}})} \ldots\ldots\ldots (2.4)$$

Where, $P(k_i|R)$ express the probability of getting term k_i in the relevant documents of R and $P(k_i|\bar{R})$ represent the probability of getting term k_i in the non-relevant documents of $\bar{R}$. However, initially equation 2.4 is not used because of the probabilities of $P(k_i|R)$ and $P(k_i|\bar{R})$ are unknown. Therefore, two assumptions are made for the initial search in which there are no any retrieved documents. The first assumption is $P(k_i|R)$ is constant for all terms k_i (usually 0.5) and the second assumption is, the term probability distribution $P(k_i|\bar{R})$ can be approximated by the distribution in the whole collection [4]. The two assumptions expressed as:

$$P(k_i|R) = 0.5 \quad and \ \ P(ki|\bar{R} = \frac{n_i}{N} \ldots\ldots\ldots (2.5)$$

Where, n_i stands for the number of documents in the collection which contain the term k_i. Substituting into equation 2.4, it becomes [4];

$$sim_{initial}\left(d_j\,, q\right) = \sum_{i}^{t} w_{iq}\ .w_{ij}\ log\frac{N-n_i}{n_i} \ldots\ldots (2.6)$$

After initially documents are retrieved the user marks the retrieved documents as relevant and non-relevant. Then the information is used so as to evaluate the probability of $P(k_i|R)$ and $P(k_i|\bar{R})$. The probabilities $P(k_i|R)$ and $P(k_i|\bar{R})$ can be approximated by [4]:

$$P(ki|R) = \frac{|Dr,i|}{|Dr|}\ and\ P(ki|\bar{R}) = \frac{ni-|Dr,i|}{N=|Dr|} \ldots\ldots (2.7)$$

Where, D_r is the set of relevant retrieved documents as the user judgment and $D_{r,j}$ is the subset of D_r composed of the documents which contain the term k_i. However, equation 2.7,

have a problem for some small values of $|D_r|$ and $|D_{r,i}|$. Thus, a 0.5 adjustment factor is added to the estimation of $P(k_i|R)$ and $P(k_i|\bar{R})$. This will give [4];

$$P(k_i|R) = \frac{|Dr,i| + 0.5}{|Dr| + 0.5} \quad and \; P(ki|\bar{R}) = \frac{n_i - |Dr,i| + 0.5}{N - |Dr| + 0.5} \ldots\ldots\ldots (2.8)$$

The adjustment fact made at equation 2.9 may provide inadequate estimation in some cases. In this case, alternative adjustment factors have been formulated. For instance, replacing adjustment factor 0.5 by $\frac{n_i}{N}$[4]:

$$P(k_i|R) = \frac{|Dr,i| + \frac{n_i}{N}}{|Dr| + 0.5} \quad and \; P(ki|\bar{R}) = \frac{n_i - |Dr,i| + \frac{n_i}{N}}{N - |Dr| + 0.5} \ldots\ldots\ldots (2.9)$$

2.4. IR System Evaluation

Evaluation of information retrieval system is done before the system is implemented [4]. Several reasons are stated why evaluation is needed in IR. For instance, evaluation provides the ability to [1];

- Validate and verify the system to check whether the system is right or not.
- Measure which one is the better system than the other one.
- Measure how good the IR system works.
- Identify techniques/ algorithms that work well and do not work.
- Identify specific components of techniques or algorithms that work better.
- Provide future direction for further studies

In IR system evaluation, the two common measure of system performance are efficiency and effectiveness. Efficiency is the time and space used by the system in retrieval process. To be called efficient system, the retrieval and indexing time of the system should be shorter and the space used in indexing file should be smaller. On the other hand, effectiveness refers how much the system meets its designed objective. To be called

effective system the system should be capable of retrieving relevant documents from the collection and the system should retrieved documents that satisfy users need [4].

2.5. The Amharic Writing system and its Features

Ethiopia has several languages, which are spoken by different nations and nationalities. Among the languages, Amharic or "አማርኛ" is the national language of Ethiopia until 1995. Following the declaration of constitution of Ethiopian federal democratic government, it becomes the working/official language. Because of its wide application, Amharic documents are become increasing in both hard copy and electronic forms.

Since this research is conducted considering Amharic documents, it is important to investigate the potential features of the language that have the capability of representing the contents of the documents that demands one to understand the characteristics of the language.

2.5.1. History of Amharic Writing System

The origin of Amharic writing system traced back when Semitic scripts were flourished in the Middle East before three thousand years ago [35], as depicted in figure 2.4.

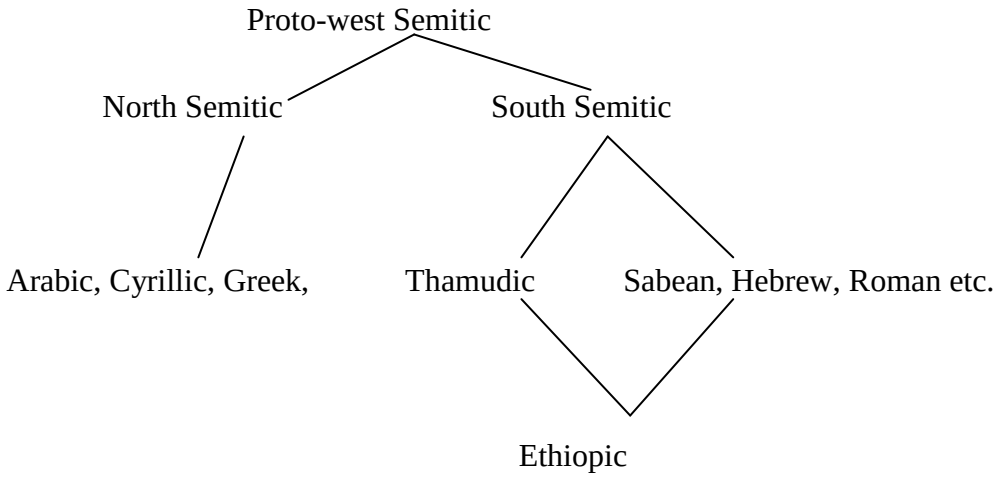

Figure 2.4: Origins of Ethiopic a family tree model adopted from [35]

The model shows two main branches downward from Proto-West Semitic; the North Semitic and South Semitic. The children in North Semitic branch are Hebrew, Arabic, Greek, Roman and Cyrillic. The South Semitic side produces Ethiopic via the Sabean system, which is speculatively dated as emerging in the 11th and 10th centuries BC. The three Semitic languages which are only found in Ethiopia and Eritrea are Geez or Ge'ez, Amharic and Tigrinya which are used in a representation for Ethiopic system [35].

Geez play a significant role in the development and expansion of Amharic language and writing system. Several religious texts, such as Bible, translations of Arabic Christian texts from Egypt and literatures such as the 'qine' (ቅኔ) and poems are all written in Geez. The emergence and expansion of Geez inscriptions in the Ethiopic script traced back to the 4th century AD, when Geez was the language of the empire of Aksum North Ethiopia. Even if the use of geez is limited to Orthodox Church, it is still a source for the coining of Ethiopian literary and technical terms [35].

The Sabean script is attributed as the source of the Geez script, likewise, the Geez writing system attributed as the source of Amharic writing system. In the early age, Amharic has been the language of the politically dominant ethnic group in Ethiopia and it is now used in every part of the country.

2.5.2. Amharic Alphabet

The Amharic language alphabet consists 33 basic characters having 7 different forms for each consonant vowel combination [35]. The basic character occurs in one form and the remaining six different forms are build based on the basic characters having a little modification form. For example, the seven orders of the consonants of ሀ (hä) and ዘ (ze) expressed as follows;

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
hä	hu	hi	hä	he	h	ho
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
zä	zu	zi	zä	ze	z	zo

Generally, there are 231 characters. According to Bloor [35], Amharic is a syllabic language in which a character is used to represent a phoneme, which is a combination of a vowel and a consonant. In Amharic language, there are additional nearly forty characters which contains special feature representing labialization. For instance, ሏ (lua) from ለ (lä) ሟ(mua) from መ (mä) and ሯ (rua) fromረ (rä).

As every language has its own way of representing numbers, despite the absence of representing number zero (‘0’), Amharic has also its own way of representing numbers. The Amharic numbering system consists of twenty (20) single characters derived from Greek letters by modifying the characters into Amharic character form. The symbols are modified by adding a horizontal stroke above and below. The numbers consists of a single character for one to ten, for multiples of ten (twenty to ninety), hundred and thousand [36].

The Amharic writing system also consists around 17 (seventy) punctuation marks. The most commonly used punctuation marks are [36];

- Two dots (‘:’) called “ሁለት ነጥብ”, which is similar to a colon used to separate words, however, nowadays the influence of foreign languages lead to the use of blank space instead.
- Four dots (‘::’) called “አራት ነጥብ”, which is used as full stop.
- Two dots with horizontal stroke line over them (‘፣’) called “ነጠላ ሰረዝ” which is used as comma.
- Two dots with horizontal stroke above and below them (‘፤’) called “ድርብ ሰረዝ”, which is used as semi-colon. The other punctuations are mostly borrowed symbols such as, ?,!,/.\, “,’ etc.

2.5.3. Features and Challenges of the Amharic Writing System in Designing Amharic Text retrieval

Several characteristics of Amharic writing system have been investigated in many literatures. Some of the characteristics are [6] [15] [35][36][37];

- It has different forms of writing compound nouns and abbreviations.

- It has different ways of writing the same word.
- It has consonants with the same sound.
- It is reach in word variants

These different features of the language cause several challenges in designing Amharic information retrieval system.

- Different forms of wring compound nouns:

Amharic writing system has two ways of writing compound nouns without affecting the meaning of the word. In other word, two words are sometimes written as one word or one word might be written as two words. There is no similar convention that specifies how these words are written. For instance, the word "bête kiristian" which means church might be written as one word "ቤተክርስትያን"or two words "ቤተ ክርስትያን" which does not make any difference in the meaning of the word. And also the word "bête mengist" which means palace might be written as one word "ቤተመንግስት" or two words "ቤተ መንግስት". Other compound nouns such as, "መአድ ቤት","ብርድልብስ","ብረትድስት", "ቤተ መዘክር", "ወጥ ቤት", etc. which means dining room, blanket, cooking pot, museum, kitchen respectively, can be written as one word or two words. This situation makes an IR system difficult to differentiate those words.

- Different forms of wring abbreviation

The other characteristic of Amharic writing system is it has different forms of writing abbreviation. Here also, there is no the same convention for using the same form of abbreviations. For instance, the phrase "አዲስአበባ" which means the city Addis Ababa might be abbreviated as; "አአ","አ/አ","አ.አ". Therefore, there should be a mechanism to handle these problems in retrieving Amharic documents.

- Different ways of writing the same word and consonants with the same sound

In Amharic writing system there are characters or symbols which have the same sound during reading. For example, the characters "ሀ, ሐ, ኀ, አ and ዐ" produces the same sound

with characters "ሃ, ሓ, ኃ, ኣ and ዓ" respectively during reading. On the other hand, there are also alphabets that share the same sound, such as, ሀ, ሐ, and ኀ, ሰ and ሠ, አ and ዓ, and ጸ and ፀ. As a result, words which have the same meaning may have different spelling structure. For example, the word "አለም" which means world can be written differently as; "አለም", "ዓለም", "ዐለም", "ኣለም".

In IR system since it only match the character in query words to check whether a word found in a document has the same meaning as in the query, it consider the word 'አለም' and 'ዓለም' are dissimilar.

Other problems of Amharic writing system are the existence of several synonym and polysemous words. Synonym words are the words with the same meaning but written differently. For instance, the word "ኮፍያ" which means 'hat' can be called differently in different region, such as, "ባርኔጣ", "ቆብ" etc. Polysemous words are words that have different meaning by stressing some characters in the word while reading. For instance, the word "ገና" which means 'Christmas day' can have another meaning "not yet" when 'n' is stressed [15].

- Reach in word variants

Amharic language is reach in word variants. One term may have different variant terms. For instance, a word "ሰጠ" meaning 'to give' may have several variants such as; 'ሰጡ', 'ሰጥ', 'ሰጠች', 'ሰጠን', 'ሰጣቸዉ', 'ሰጠከው' 'ሰጠካት', 'ሰጠላቸዉ', 'ሰጠላት', 'ሰጠለት', 'ሰጠሀቸው', etc. Such variant of words subject a system to consider those words as morphologically different, but the same in meaning.

In general, such problems makes designing information retrieval system for Amharic language difficult and challenging, and it needs a lots of works expected from experts so as to develop a better IR system considering the language problems discussed above.

2.6. Review of Related Works

Probabilistic information retrieval has been studied for decades, which results several body of literature on the topic.

2.6.1. Global Probabilistic IR Research

Probabilistic retrieval model computes the probability of relevance of a document for a query [30].Given a query, a document is assumed either relevant or non-relevant. However, the system not knows whether a document is relevant or not. Therefore, it as to estimate the relevancy of the document based on probabilistic methods [4].For instance, let D and Q represents document and query, respectively. Let R represent a binary random variable that indicates whether D is relevant to Q or not. R takes two values which are relevant (r) and non-relevant ($\bar{r}$). Thus, to estimate the probability of relevance (i.e. p(R=r |D,Q) or p(R= $\bar{r}$|D,Q)), several ways of probabilistic model have been developed.

Robertson and Sparck Jones [11], developed the well-known classical probabilistic model called The Binary Independence Retrieval (BIR) model so as to estimate the probability of relevance (i.e. p(R=r |D,Q) or p(R= $\bar{r}$|D,Q)).

Robertson and Sparck Jones work was based on two basic assumptions: the independent assumption and ordering principle. The independent assumption assumed that, terms are distributed independently and randomly. Specifically, independence assumption one (I1) stated that, "the distribution of terms in relevant documents is independent and their distribution in all documents is independent". The second independent assumption (I2) stated that, "the distribution of terms in relevant documents is independent and their distribution in non-relevant documents is in-dependent". The ordering principle states, documents should be ordered by their probable relevance to the query. Specifically, the ordering principle one (O1) stated that, "probable relevance is based only on the presence of search terms in documents" and the ordering principle (O2) stated that, "probable relevance is based on both the presence of search terms in documents and their absence from documents". Five relevance weighting functions (i.e. F0-F4), which are derived from

a formal probability theory of relevance weighting was investigated. The experiment conducted using manually indexed 1400 document collection written in English language. Table 2.2 shows the retrieval performance registered.

Weighting function	Precision	Recall
F0	29%	90%
F1	50%	90%
F2	60%	90%
F3	66%	80%
F4	70%	80%

Table 2.2: Experimental result of Robertson and Sparck Jones work

As the result indicates, the ordering principle O2 is correct and O1 is incorrect. The performance also shows that F3 and F4 performed consistently better than F1 and F2. On the other hand, the experiment depict that, the performance of the system improves when information about the occurrences of terms in relevant documents is added to information about their simple document incidence. Specifically, relevance weights give a better performance than simple term matching [11].

Several attempts have been made to improve the binary independent representation. For instance, Rijsbergen [12] try to improve the binary independence model by capturing some term dependence as defined by a minimum spanning tree weighted by average mutual information. The result of the research shows that, the dependency model achieved significant performance over independence model. However, the experiment was done only on very small document collections written in English language and the estimation of many more parameters is a problem in practice.

The application of probabilistic retrieval model is not only limited to English language. Several works have been done in different languages such as, Chinese [19] and Russian [20] by using different probabilistic models.

Xiangji and Stephen [19] tried to investigate the effect of probabilistic approach on Chinese text retrieval, the difference between word approach and character approach and the effect of different phrase weighting functions and of varying their parameters. The

weighting function used is based on the basic weighting function of Rovertson-Sparck Jones weight [11], which approximates to inverse collection frequency when there is no relevance information. For evaluation of the system in Test Retrieval Conference (TREC 6), three automatic runs (i.e. run1, run2, run3) were submitted. Run1 and run3 are for word indexing approach and run2 is for character indexing approach. Table 2.3 depicts the result of the experiment.

	Average precision	R precision	Precision for 100 documents
Run1	48%	51%	48%
Run2	50%	52%	49%
Run3	49%	51%	48%

Table 2.3: Experimental result of Xiangji and Stephen work

The overall result suggested that a new way of phrase weighting schemes for Chinese should be developed [18].

Ljiljana and Jacques [20] attempted to use probabilistic information retrieval model so as to evaluate several stemming and indexing approaches for Russian language. The probabilistic methods used in this research including Okapi, Divergence form Randomness (DFR) and statistical language model (SLM). To evaluate the performance of the IR models used in this research, mean average precision (MAP) were used. The MAP values were computed by TREC_EVAL software using a 1000 retrieved documents. To determine the performance of the model two sided t-test with a significance level of α= 5% were used. The experiment result shows that, the mean average precision for Okapi model without stemming is 0.0881. The mean average precision for DFR model without stemming is 0.0879. The mean average precision for SLM model without stemming is 0.0964. This work was mainly conducted for suggesting appropriate stemming strategy. The experiment shows that stemming procedure improves retrieval effectiveness especially in the case of the collection containing short document [20].

2.6.2. Local IR System Works

Information explosion in electronic text written in Amharic language caused an increasing need of designing effective and efficient IR system for Amharic texts. A number of IR systems developed so far for retrieving Amharic texts.

Betelihem [14], developed n-gram-based automatic indexing for Amharic text retrieval. This research mainly conducted to solve the problem of not having standard stemming procedure and stop word list for Amharic language. Thus, to solve such problem a method called n-gram automatic indexing has been developed. The experiment conducted using 100 Amharic news articles, which are collected form Walta Information Center. Vector space model was used for the purpose of document representation. The results showed that tri-grams indexing methods performed 5% and 88% precision and recall respectively. One of the recommendations forwarded in this research is using combinations of different n-grams for indexing.

Tewodros [15] developed Amharic text retrieval using latent semantic indexing (LSI) strategy with singular value decomposition. His work mainly focuses on indexing process, to solve the problem found in exact term matching retrieval system. In exact term matching system, the term in the user's query may not appear in a relevant document since there are many ways to express the same concept. Moreover, many words can have also more than one meaning. The result leads to the retrieval system that misses relevant documents and retrieve non-relevant ones. Thus, LSI indexing technique were used which partially handle the problems of exact term matching by organizing terms and document into a semantic structure. 25 queries and 206 Amharic document collections were used for experimentation. The result of the experiment shows that, the recall-precision graph of the LSI method was above standard vector method. The average precision registered by the system was 71%. One of the recommendations given by this research is to design a model that consider relevance feedback that gives information for the system about which documents are relevant and which are not relevant so as to enhance the performance of the system.

Tessema and Solomon [16], designed and implement Amharic search engine, which retrieve web documents written in Amharic language. Even if general search engine such as Google, Yahoo have the way to accept Amharic query and retrieve relevant document, they simply match patterns without considering the feature of Amharic language that affect the retrieval performance. In this research, a complete language specific process has been done, such as, crawler, indexer and query engine component. The experiment result shows, the average precision and recall of 99% and 52% respectively. In future work, the need for considering additional features of Amharic text was recommended.

Alemayehu [38] and Abey [6], developed Amharic IR system by integrating query expansion techniques to enhance the performance of the system. The research is conducted to solve the problem of polysemous and synonymous exists in Amharic text, which decreases the performance of the IR system. Several query expansion techniques were tested. Such as, global analysis, local analysis, bi-gram analysis, bi-gram based thesaurus and statistical co-occurrence method. The result of the experiment show statistical co-occurrence method registered better performance than other methods by scoring 73% recall and 53 precision. One of the recommendations forwarded is to design ontology based query expansion in order to control expanding terms that are polysemous by themselves.

In conclusion, even if several IR systems have been developed in the last decade, most of works are attempt to design an Amharic IR system using vector space model. However, the use of vector space model may not control uncertainty nature of IR system. Thus in this study an attempt is made to develop probabilistic based Amharic IR system to enhance the performance of the Amharic IR system.

CHAPTER THREE

DESIGNING PROBABILISTIC IR SYSTEM

In designing probabilistic Amharic information retrieval system, the two major components which are indexing and searching are used [4]. Indexing is an offline process used to facilitate access to preferred information from document corpus accurately and efficiently as per users query. Searching is an offline process used to scan document collection so as to find relevant documents that satisfy users need .

Figure 3.1 depicts the probabilistic based IR system architecture designed and implemented in this research. It indicates that, the IR system has online (Searching) and offline (Indexing) processes. During the offline process, Amharic documents are organized using inverted indexing structure. To ease the indexing task, text operations are applied on the text of the documents in order to transform them in to their logical representation. The documents are indexed and the index is used to execute the search. The searching task is an online process that accepts users query. The query is processed to identify terms using which searching is done to identify and retrieve relevant documents. After ranked documents are retrieved, the users provide feedback that can be used to refine the query and restart the search for improved results [4].

3.1. Amharic Document Indexing

In information retrieval, searching is possible or efficient when the text is small. However, in large databases searching will take much more time and space unless indexing structure is used to organize documents. Therefore, constructing and maintaining indexing on large database is not optional.

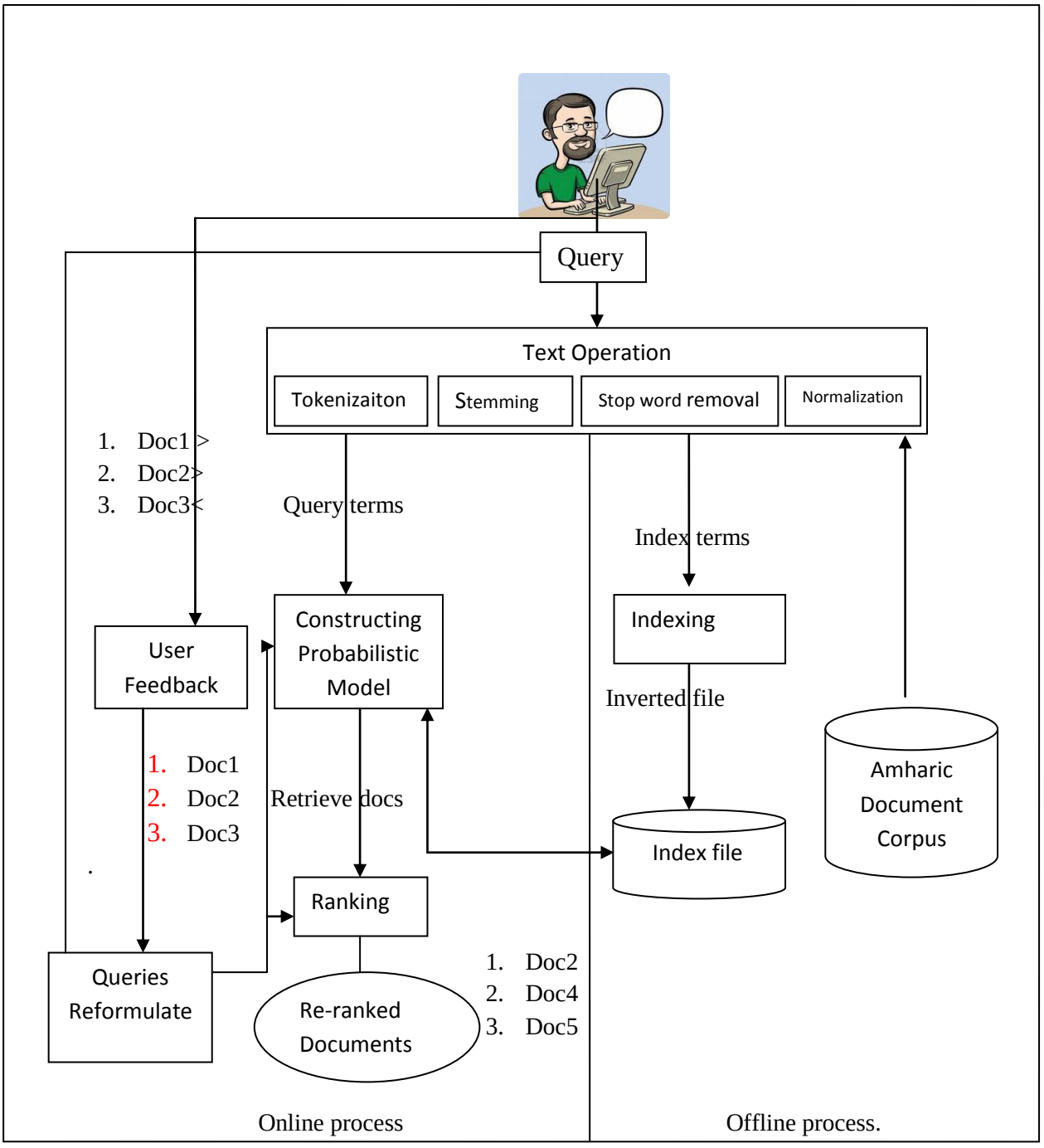

Figure 3.1: Probabilistic based IR system architecture

Several works has been done so as to index the document collection/corpus used in this research. These tasks include; stemming, stop word removal and normalization.

3.1.1. Word Stemming

In Amharic writing system, words are morphologically variant. Morphological variant words have similar semantic interpretations. In IR system, those words considered as equivalent words [1]. Therefore, words have to be reduced to their root using stemming technique. On the other hand, stemming also used to reduces the dictionary size (i.e. the number of distinct terms used in representing a set of documents). The smaller the dictionary size results the smaller storage space and processing time required.

Stemming techniques are language dependent. Therefore, every language needs to have language specific stemming technique. In Amharic text, there are many word variants/affixes [36]. To conflate them into stem word, stemming technique/ algorithm developed by Nega [39] were used. He developed the stemmer that involves the removal of both prefixes and suffixes and also consider letter inconsistency and reiterative verb forms.

According to Nega [39], a stemmer that stem words without consideration of remaining stem, which removes words that are similar to prefix and suffix list but that are not actually affixes is called context-free stemmer. For instance, English word ‘regular’ and ‘metal’ will be ‘gular’ and ‘met’ respectively if context-free stemmer removes ‘re’ and ‘al’. Similarly, weak result will appear if such technique is applied for Amharic language. For example, for Amharic word ‘ከለከለ’ which means forbid, ‘ከለከላቸዉ’ which means forbid them and ‘ከለከላት’ which means forbid her, if we remove string ከ ‘ke’ which is found in prefix list, the result will be ለከለ which does not have any relation with the correct stem. Therefore, context-sensitive approach should be used with affix removal being controlled by two action codes and five conditions.

The action codes are:

A1: do not perform affix removal

A2: remove a shorter form affix

The five conditions are:

C1: characters following the assumed affix (i.e. if the assumed affix is followed by:). This is used to cater for words where vowel elision occurs at morpheme boundaries in the case of prefixing, and to reduce the removal of terminal consonants in the case of suffixing.

C2: characters preceding the assumed affix (i.e. if the assumed affix is preceded by :). This is used to avoid the removal of terminal consonants that are not genuine suffixes.

C3: if the assumed affix is part of an identified consonants structure this is used to avoid the removal of non-genuine affixes from specific words and their variants.

C4: A minimum stem-length condition for the specified affix. The stem length includes both vowels and consonants and the condition is used to reduce over stemming.

C5: A combination of the above.

Table 3.1 shows the prefix and suffix lists used in this research, which is adopted from Nega [38]

Prefix lists					Suffix lists				
የ	ስለ	የሚ	እየ	ስለሚ	ች	ኝ	ችን	ቸው	ዊት
እያ	እንደ	አል	አለ	በ	ና	ዎች	ኛ	ዎቻቸው ም	ውም
ለ	ከ	ይ	እንዳ ይ	ሲ	ው	ዎች ም	ውያን	ዎቹ	ናቸው
እንዲ	እስከ	ከነ	እን	እነ	ባቸው	ዊያን	ነት	ያዊ	ን
					ት	ሉ	ችው	ዊ	ዊቷ
					ቹ ን	ዬ	ዎ	ህ	ሽ
					ዋ	ሁ	ለት	ላት	ላቸው
					ላችሁ	በት	ባት	ባቸው	ባችሁ
					ቱ	ሽ	ይቱ	የው	ኞች

Table 3.1: Prefix and Suffix list

Based on the prefix and suffix list and the context sensitive rule discussed in section 3.1.1, words are conflated to their stem using the algorithm 3.1 and algorithm 3.2. In this research prefix is removed prior to suffix.

Step 1: Get WORD and count the number of radicals (or consonants)

Step 2: IF number of radicals < 3 TJEN stop and return WORD

Step 3: IF length (Word) < -(MAXPL + 2) THEN assign the first length (WORD) -2

Characters to a temporary prefix, TEMPP

ELSE assign the first MAXPL characters to TEMPP (where MAXPL is the

Length of the longest prefix in the list)

Step 4: Find TEMPP in the prefix list

Step 5: IF found THEN

IF context sensitive THEN check the context-sensitive rules cl-c5

Apply al or a2 and GOTO step 7

ELSE GOTO step 6

ELSE

IF length (TEMPP) >1 THEN assign length (TEMPP) -1 Characters to

TEMPP and GOTO step 4

ELSE return WORD

Step 6: Remove prefix

Step 7: IF number of radicals of WORD > 3 or length (WORD) > 4 THEN GOTO step 1

ELSE GOTO step 2

Algorithm 3.1: Prefix remover algorithm

Step 1: Get WORD and count the number of radicals (or consonants)

Step 2: IF number of radicals < 3 TJEN stop and return WORD

Step 3: IF length (Word) < -(MAXPL + 2) THEN assign the last length (WORD) -2

Characters to a temporary prefix, TEMPP

ELSE assign the first MAXPL characters to TEMPP (where MAXPL is the

Length of the longest prefix in the list)

Step 4: Find TEMPP in the prefix list

Step 5: IF found THEN

IF the ending required recoding THEN recode and GOTO step 1

ELSE

IF context sensitive THEN check the context-sensitive rules

C1-c5 apply a1 or a2 and GOTO step 7

ELSE GOTO step 6

ELSE

IF length (TEMPP) > 1 THEN assign length (TEMPP) – 1 characters to

TEMPP and GOTO step 4

ELSE return WORD

Step 6: Remove prefix

Step 7: IF number of radicals of WORD >2 THEN GOTO step 1

Algorithm 3.1: Suffix remover algorithm

3.1.2. Stop Word Removal

Not all terms found in the document are equally important to represent documents they exist in. Some terms are common in most documents. Therefore, removing those terms, which are not used to identify some portions of the document collection, is important. Such terms are removed based on two methods. The first method is to remove high frequent terms by counting the number of occurrences (frequency). The second method is using stop word list for the language [40][41].

In this research the second method were used to apply stop word removal. The stop word list is adopted form Nega [39]. Removing stop words were applied after stemming is implemented. This is because, Amharic language is reach in word variant, thus applying stemming prior to stop word removal reduces the problem of considering morphologically variant words as different in meaning by IR system.
Table 3.2 shows stop word list used in this research.

Stop Word Lost									
ነው	እኔ	እኛ	እነሱ	እሱ	እሷ	አንተ	አናንተ	አና	ወደ
ነይ	ወይ	ከ	ናቸው	ትናት	ጥቂት	በርካታ	ብቻ	ሁሉም	ሌላ
ሌሎች	ሁሉ	እያንዳንዱ	እያንዳንዳችው	ስለ	እንዲሁም	እንጂ	ደግሞ	መካከልከ	ሰሞኑን
ከሰሞኑ	በሰሞኑ	የሰሞኑ	ትናንት	ትናንትና	ጋራ	የጋራ	ከጋራ	ተለያዩ	ተለያየ
ድረስ	እስከ	በጣም	ግን	ሲሆን	ሲል	ዉስጥ	ላይ	ናት	ነበሩ
ነበረች	ያ	ወይዘሮ	ወይዘሪት	ነገሮች	ከፊት	ከላይ	ታች	ከታች	በታች
የታች	በውስጥ	ከውስጥ	ጋር	ናቸው	ይህ	በላይ	ወደ	ወዘተ	እና
ወይም	እንደ	አቶ	ፊት	ወደፊት	ነገር	በፊት	በሆላ	በኩል	

Table 3.2: Stop Word List

3.1.3. Normalizing Characters and Words

As discussed in section 2.4.3, Amharic writing system contains characters or symbols, which have the same sound during reading. For example, the characters "ሀ, ሐ, ኀ, አ and ዐ" produces the same sound during reading with characters "ሃ, ሓ, ኃ, ኣ and ዓ" respectively. On the other hand, there are also alphabets that share the same sound, such as, "ሀ, ሐ, and ኀ", "ሰ and ሠ", "አ and ዓ", and "ጸ and ፀ". As a result, words which have the same meaning, may have different spelling structure. For example, the word "አለም" can be written in different way such as, "አለም", "ዓለም" "ዐለም" "ኣለም". Likewise, there are also synonymous words that have spelled differently but have similar meaning. For instance, the word 'ኢትዮጵያ' (Ethiopia) can be written using different words 'አቢሲኒያ' (Abyssinia) and 'የአስራራትወርጻጋ' (thirty month of sunshine). In this research such kind of problems are solved by changing the characters and the word to their common standard form.

3.2. Searching Using the Probabilistic Model

As briefly described in section 2.2.3, in designing probabilistic IR system, one can choose from several methods. Binary independent model is used to design probabilistic Amharic information retrieval system. This is because, according to Greengrass [28], the first step in most of probabilistic methods is to make some simplifying assumption. Thus, BIM is the model that has been used with the probabilistic ranking principle by introducing some simple assumptions which makes estimating the probability function P(R|d, q) practical.

The feedback process is also directly related to the derivation of new weights for query terms and the term reweighting is optimal under the assumptions of term independence [4]. In addition, it is the first model that has been used in several researches and it has clear and simple assumptions and mathematical and theoretical basis.

According to Robertson and Sparck [11], binary independent model based on two assumptions and principles; the independence assumption and ordering principle.

The independent assumption stated that (A1) the distribution of terms in relevant documents as well as in non-relevant documents and (A2) their distribution in all document is independent.

The ordering principle stated also that probable relevance is based only on the presence of query terms in the document (OP1) and/or on both the presence and absence of query terms in the documents (OP2).

	Independent Assumption		
Ordering Principle		A1	A2
	OP1	F1	F2
	OP2	F3	F4

Table 3.3: Assumption- Principle Summary Table

The records in table 3.3 F1, F2, F3 and F4 are weighting functions that related the two independent assumptions and ordering principle. Robertson and Sparck [11], stated that, the assumption A2 is more truthful than A1 while the ordering principle OP2 is correct and 0P1 is incorrect. This yield F4 is most likely to give the best result and it is the best function. Table 3.4 summarizes the weighting functions.

In binary independent model there are three steps to compute term probability. The first step compute terms when there is no retrieved document at initial stage. The second step compute terms after documents are retrieved and feedback is provided by the user. The third step compute terms when partial feedback is given [4].

At first, since the properties used to retrieve relevant information are unknown and only index terms are known properties, attempt has to make the initial guessing. The assumptions made in this step are [4];

- $P(ki|R)$ is constant for all index terms k (usually, its equal to 0.5)

- The distribution of index terms among the non-relevant documents can be approximated by the distribution of index terms among all the documents in the collection.

Weight number	Formula	Function number	Description
W1	$log \frac{(\frac{r}{R})}{(\frac{n}{N})}$	F1	Evaluates the ratio of the proportion of relevant documents in which the term occurs to the proportion of the entire collection in which it occurs.
W2	$log \frac{(\frac{r}{R})}{(\frac{n-r}{N-R})}$	F2	Evaluates the ratio of the proportion of relevant documents to that of non-relevant documents.
W3	$log \frac{(\frac{r}{R-r})}{(\frac{n}{N-n})}$	F3	Evaluates the ratio between the number of relevant documents in which it does occur and the number in which it does not occur and the collection likelihoods for the term.
W4	$log \frac{(\frac{r}{R-r})}{(\frac{n-r}{N-n-R})}$	F4	Evaluates the ratio between the term relevance likelihoods and its non-relevance likelihoods.

Table 3.4: Summary of weighting functions

These two assumptions will give as;

$$P(k_i|R) = 0.5 \; and \; P(k_i|\bar{R}) = \; log \frac{N-ni+0.5}{ni+0.5} \ldots\ldots .(3.1)$$

Where, N is the total number of documents in the collection and n_i is the number of documents which contain the index term k_i.

Using this initial guess, documents are retrieved which contain query terms and provide an initial probabilistic ranking. After documents are retrieved, the user looks at the retrieved documents and marks them as relevant and non-relevant. The system then uses this feedback to refine the description of the answer set. At this stage, initial ranking is shown and more discriminating information about terms is available (i.e. relevance feedback), this will allow more accurate estimation [4][11]. Therefore, relevant documents retrieved

should be improved using probabilistic relevance weighting technique. This technique uses the concept in term incidence contingency table 3.5 shown below [11].

	Relevant	Non-relevant	Total
Containing the term	r	n - r	n
Not containing the term	R - r	N – n – R + r	N - n
Total	R	N - R	N

Table 3.5: Term incidence contingency table

Where,

- r is the number of relevant documents that contain the term,
- n – r is the number of non-relevant documents that contain the term,
- n is the number of documents that contain the term,
- R- r is the number of relevant documents that do not contain the term,
- N – n – R + r is the number of non-relevant documents that do not contain the term,
- N – n is the number of documents that do not contain the term, R is the number of relevant documents,
- N – R is the number of non-relevant documents and N is the total number of documents in the collection.

After the knowledge of relevant documents and non-relevant documents for a given query is completed, the next step is estimating the probability of finding term (ti) in relevance document using equation 3.2 and the probability of finding term (ti) in non-relevant document using equation 3.3 [11];

$$P(ti|R) = \frac{r}{R} \dots\dots\dots\dots\dots\dots (3.2)$$

$$P(ti|\bar{R}) = \frac{n-r}{N-R} \ldots\ldots\ldots\ldots (3.3)$$

According to Sparck et al. [31], the above equations can be rewritten to compute term presence weighting function as;

$$\mathrm{w} = \log \frac{\mathrm{r}\,(\mathrm{N}-\mathrm{n}|-\mathrm{R}+\mathrm{r})}{(\mathrm{R}-\mathrm{r})(\mathrm{n}-\mathrm{r})} \ldots\ldots\ldots\ldots\ldots\ldots (3.4)$$

However, Robertson and Jones [11], noted different assumptions lead to a different formula for computing term weighting. They argue "in practice users may find themselves in the situation where, even if they know some relevant documents are retrieved, they wish to continue searching". They assume that "users may not found all the relevant documents that would satisfy their need". Therefore, the record in the center of the contingency table (i.e. N – n – R + r) may not be taken as absolute. The estimation of document relevance when considering new items has to allow for uncertainty. This estimation adds 0.5 to all the central record and it derives a specific term relevance weighting formula;

$$\text{Relevance Weighting (Rw)} = \log \frac{(\mathrm{r}+0.5)(\mathrm{N}-\mathrm{n}-\mathrm{R}+\mathrm{r}+0.5)}{(\mathrm{R}-\mathrm{r}+0.5)(\mathrm{n}-\mathrm{r}+0.5)} \ldots\ldots (3.5)$$

3.3. IR System Evaluation

Retrieving relevant document from the collection that satisfies users need is the heart of IR system evaluation in determining its effectiveness. Two strategies are identified in measuring the effectiveness performance of IR systems. The first is the user-centered strategy, which uses relevant judgment so as to evaluate the performance of the system and the second is system centered strategies which work based on reference judgment available prior to testing process [4]. Based on the concept of relevance (i.e. to a given query or information need), there are several techniques of measures of IR performance available, such as, precision and recall, F-measure, E-measure, MAP (Mean average precision), R-measure, etc [28]. In this study, the three widely used techniques precision, recall, and F-measure are used to measure the effectiveness of the IR system designed.

Precision and recall are the two most frequent and basic statistical measures. Recall is the percentage of relevant documents retrieved from the database in response to users query, whereas precision is percentage of retrieved documents that are relevant to the query [1].

To show these metrics, assume the document collection be D. Let R_t is all retrieve documents from the collection D and R_l a number of relevant documents in D. The joint of R_t and R_l is a set of documents retrieved and relevant, R_A. Therefore, the recall and precision can be calculated using equation 3.5 and 3.6 respectively.

$$\text{Recall} = \frac{|RA||R_t|}{|R_l|} \ldots\ldots\ldots\ldots (3.5)$$

$$\text{Precision} = \frac{|RA||R_t|}{|R_t|} \ldots\ldots\ldots (3.6)$$

The formula show above for precision and recall assume that, all documents retrieved (R_t) is examined by user. Thus, the retrieved document cannot be presented for the user at once. Rather, the retrieved documents are presented according to their degree of relevance as per the user query. Then, the user examines the ranked documents starting from the top. This kind of examination of documents by the user leads the recall and precision measures to vary. Therefore, for appropriate evaluation of recall and precision, plotting a precision versus recall curve is necessary [4].

To draw precision-recall curve, for example let documents retrieved by the system is Dr Where Dr = {d_3, d_{33}, d_9, d_1, d_{10}, d_{11}, d_{14}, d_7, d_{44}, d_{49}, d_{50}, d_{53}, d_{55}, d_{77}, d_{133}} in ranked order. Assume Rq contain a set of relevant documents for the query. Where, Rq = {d_1, d_3, d_7, d_{10}, d_{14}, d_{33}, d_{44}, d_{49}, d_{55}, d_{133}}. Based on the given information recall- precession curve is constructed. However, when constricting the curve based on the original recall and precession may result in saw tooth curve, there is a need to smooth the curve using interpolation technique.

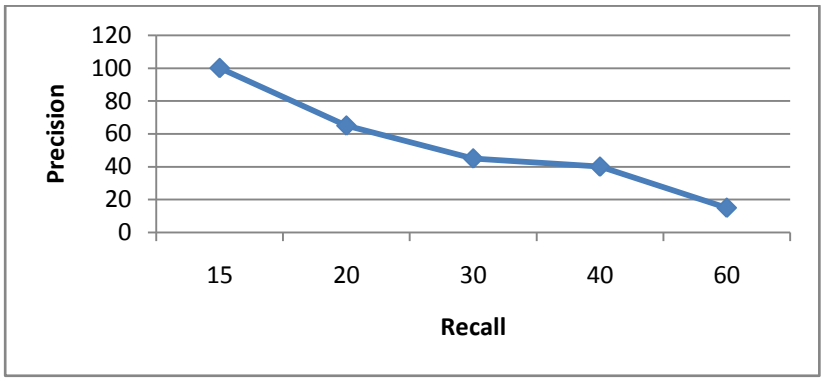

Figure 3.2: Example of precision and recall graph

The example shown above for plotting precision and recall is carried out for a single query. When measuring the performance of IR system multiple queries are used. To evaluate the performance of the system using k queries, average precision at each recall level is used. This can be expressed as follows [4];

$$\vec{P}(r) = \sum_{i=1}^{N_q} \frac{P_i(r)}{N_q} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.7)$$

Where,$\vec{P}(r)$ is the average precision at the recall level r, Nq is the number of queries used, and Pi(r) is the precision at recall level r for the i^{th} query. It is an empirical fact that on average as recall increases, precision decreases.

The recall levels for each query might be deferent from the standard recall levels, which is difficult to compare performance across queries. Therefore, interpolation procedure is necessary. As a result, a single precision value for each query can be used that takes a precision at some recall level for each single query. The interpolated precision versus recall curve is shown as follows; Let rj, j E {0, 1,2, ... , 10}, be a reference to the j^{th} standard recall level. Then, P(rj) = max rj ≤ r≤rj+1 P(r), "which states that the interpolated precision at the j^{th} standard recall level is the maximum known precision at any recall level between the j^{th} recall level and the $(j + l)^{th}$ recall level"[4].

In general, precision and recall have been used widely so as to evaluate information retrieval system performance. However, they measure two different aspects of the system

and thus they are inversely relative. If recall of a system is improved then the precision is reduced. The reason behind this is that, when attempt is made to include many of the relevant documents, irrelevant documents more and more exist in the answer set. On the other hand, if precision of a system is improved then the recall is reduced. This is because; there are retrieved relevant documents among the whole relevant documents found in the corpus. Achieving both precision and recall 100% is ideal [4].

Several problems have been distinguished. First, to make appropriate estimation of maximum recall for a query, it needs deep knowledge of all the documents in the collection. Second, even if many situations consider the use of a single measure, which combines both, recall and precision capture different aspects of the set of retrieved documents. One of the methods developed to alleviate the above recall and precision problems is the F-measure [4].

In this research probabilistic model is used to enhance the performance of Amharic IR system by increasing the precision without affecting the recall ability of the system. Thus, the F-measure is used to measure the performance of the system since it balances the precision and recall values.

F-measure is a single measure that trades off precision versus recall. It is the weighted harmonic mean of precision and recall expressed as follows [1];

$$f(j) = \frac{2.P.R}{P+R} \ldots\ldots\ldots\ldots\ldots (3.8)$$

Where $R=(r_1,r_2,\ldots r_{11},\ldots r_n)$ and $P=(p_1,p_2,\ldots p_{11},\ldots p_n)$ are recall and precision at the j^{th} document respectively. F-measure will be 0 when no relevant document is retrieved and 1 when all the first ranked documents are relevant. Moreover, the harmonic mean F assumes a high value only when both recall and precision are high.

In summary, there are different methods in designing probabilistic based IR system. However, the binary independent method is used to develop probabilistic based IR system for Amharic language so as to enhance the performance of the IR and to alleviate the

problem of uncertainty exists in IR. To evaluate the performance of the method, the model is implemented and tested using Amharic news articles.

CHAPTER FOUR

EXPERIMENTATION

In this research an attempts has been made to design a probabilistic information retrieval system for Amharic language. The system has both the indexing and searching parts. Inverted file indexing structure is used to organize documents so as to speed up searching. The probabilistic model that attempts to simulate the uncertainty nature of an IR system guides the searching.

To test the prototype system developed, 300 Amharic News articles were used as a document corpus. All news articles are obtained from the web site of Walta Information Center [22]. As shown in table 4.1, the news articles contain seven clusters of news, which are accident, health, education, sport, tourism, justice and politics.

No.	Types of News	Number of Documents
1	Accidental news	40
2	Tourism news	40
3	Justice related news	40
4	Political related news	40
5	Educational news	40
6	Sport news	30
7	Health care news	70
	Total	300

Table 4.1: Types of news article used

Each news articles are saved under common folder using .txt format, which is supported by most of programming language. Additionally, 10 test queries were selected by the researcher to test the performance of the system. Relevance judgment is also done for identifying which document is relevant for a given test query.

4.1. Description of the Prototype System

Before queries are projected to the term document matrix, preprocessing performed on the test document collection is also performed on the queries. However, unlike vector space model, which obtain frequency of each query terms in the documents, probabilistic model obtain the absence and presence of the query term in documents. In addition to that, queries are weighted two times (i.e. before and after relevance feedback is given) using the probabilistic weighting assumption discussed in section 3.2.1. Table 4.2 and 4.3 shows example of terms document matrix constructed for weighting query before and after relevance feedback is given.

	Query terms		
Documents	Term 1	Term 2	Term 3
Doc 1	0 or 1	0 or 1	0 or 1
Doc 2	0 or 1	0 or 1	0 or 1
Doc 3	0 or 1	0 or 1	0 or 1
Doc 4	0 or 1	0 or 1	0 or 1
Doc 5	0 or 1	0 or 1	0 or 1
weight	Weight t1	Weight t2	Weight t3

Table 4.2: Example of term document matrix before relevance feedback is given

		Query terms		
Documents	Term 1	Term 2	Term 3	Relevance Feedback
Doc 1	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
Doc 2	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
Doc 3	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
Doc 4	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
Doc 5	0 or 1	0 or 1	0 or 1	Relevant/Non-Relevant
weight	Weight t1	Weight t2	Weight t3	

Table 4.3: Example of term document matrix after relevance feedback is given

In developing the prototype system of probabilistic based IR for Amharic language, several components are integrated together as depicted in Figure 3.1 in chapter three. Those components describe the major activities done in the probabilistic retrieval model development. The system has been built using python version IDLE 2.6.2. Figure 4.1, presents a screen shot which shows the first list of retrieved document using a given query.

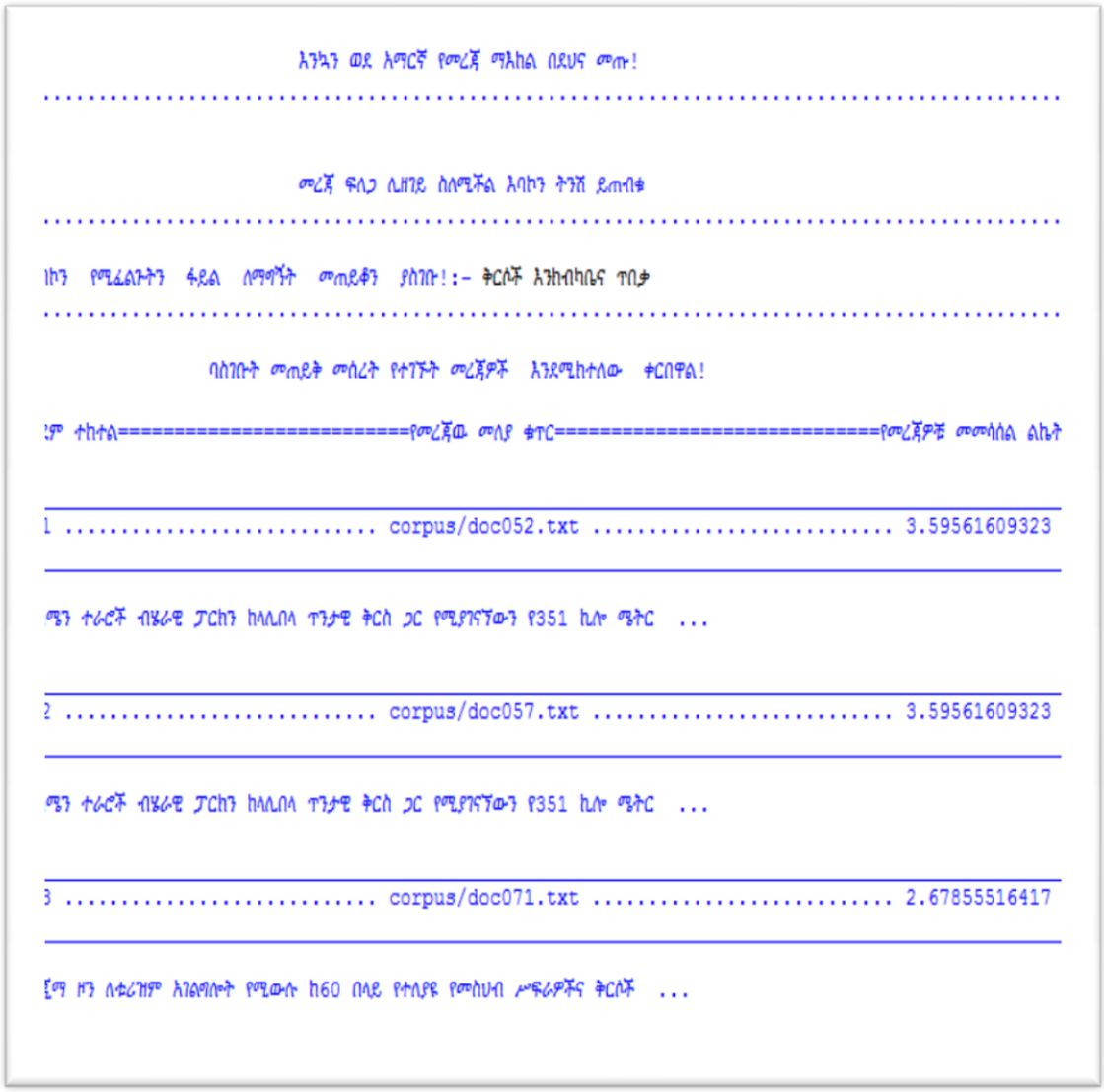

Figure 4.1: A Screen shot of retrieved document for a given query

The main objective of the prototype system is to project query terms and documents in the matrix to make comparison between documents and query, then calculating the weight of each query terms based on the notion implemented by probabilistic model and finally

calculating the score of each document and ranks in decreasing order. To fulfill this objective the following procedures are followed; first documents are pre-process for extracting index terms, removing stop word, stemming and normalization. Matrix construction such as, mapping terms and documents into space, weighting and reweighting terms using relevance feedback given by users. Testing and evaluating the prototype.

The indexing process is implemented to construct inverted index. As discussed in section 2.1.1, when implementing inverted index there are several tasks need to be done. Such as, stemming, normalization, tokenization and stop word removal.

The first step in constructing inverted index is generating index terms from document collection. In this step, the first task is tokenizing terms. Figure 4.2, shows python code implemented to tokenize terms found in document collection.

```
fileterms=fileReader(docid)
 for term in fileterms:
    indexterms=[] store each terms in the documents that are un
    for index in term.split():
       if index.isalpha():
             indexterms.append ( punctuationremove (index))
       for k in indexterms:
          List1.append(k)
       List2 = sorted(List2 + List1)
       List3 = list(set(List2))
       List5 = sorted(List3 + List1)
```

Figure 4.2: Python code for tokenization

The code first read files from all documents and extract each words, punctuations and numbers. Then, by splitting each of them, it removes punctuations and numbers. After the index terms are sorted, it store the unique index terms in ‘List5’ list variable for further processing.

As discussed in section 2.5.3, in Amharic writing system there are characters, which have the same sound during reading. In addition, there are alphabets that share the same word sound. To convert characters and words to their common form, normalization module was implemented.

Once of the problem faced when implementing normalization was finding standard list of Amharic words and characters with their common form. In this research, a limited list is prepared by collecting some of the common forms of Amharic words from different sources. Figure 4.3, shows how characters and words are converted to their common form.

```
def normal(str):
        sim=codecs.open("similarWord.txt",'r', encoding = 'utf-8' )
        similarWord=sim.read()
        li1=[]
        for m in similarWord.split():
            li1.append(m)
        for i in range(1,len(li1),2):
            if str==li1[i] or str==li1[i+1]:
                str=li1[i+1]
        temp=codecs.open("similarChar.txt",'r', encoding = 'utf-8' )
        similarChar=temp.read()
        li=[]
        for j in similarChar.split():
            li.append(j)
        for i in range(1,len(li),2):
            str=str.replace(li[i],li[i+1])
        return str
```

Figure 4.3: Python code for normalization

The code first take index term and read if similar term is found form text file. If the word is similar, it converts the term to its common form. The same process is followed to convert the character to its common form.

In Amharic writing system words are morphologically rich. Since morphological variant words have similar semantic interpretations, there is a need to stem words to their root. In this research, the stemming implementation has two modules, the prefix removal module and the suffix removal module [39]. Figure 4.4, depicts python code implementation of prefix removal of Amharic words.

```
def preffix(word):
        if len(word)<3:
            return word
        else:
            while len(word)>=3:
                tempp=word[ :3]
                p=codecs.open("prefix.txt",'r', encoding = 'utf-8' )
                pr = p.read()
                y=pr.split()
                p.close()
                if y.__contains__(tempp):
                    if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                    else:
                        word=word[3: ]
                        preffix(word)
                else:
                    tempp=word[ :2]
                    if  y.__contains__(tempp):
                        if (condition1(tempp,word)):
                            word=condition2(tempp,word)
                            return word
                        else:
                            word=word[2: ]
                            preffix(word)
                    else:
                        tempp=word[ :1]
                        if  y.__contains__(tempp):
                            if (condition1(tempp,word)):
                                word=condition2(tempp,word)
                                return word
                            else:
                                word=word[1: ]
                                preffix(word)

                        else:
                            return word
```

Figure 4.4: Python code for prefix removal

The code takes index terms and check if the length is greater that two or not. If the length of the word is less than two, the word is returned without further processing. If the length of the word is greater than two, the code iterate on word and check characters if they matched with one of the prefix found in prefix list. If the character is matched, processing continues for checking context sensitivity of the word and finally the stemmed word is returned.

After the prefix is removed from index terms, the next step is removing suffix. When removing suffix, similar procedure is followed like prefix. Figure 4.5, shows python code implemented for removing suffix.

```
def suffix(word):
        if len(word)<3:
           return word
        else:
           while len(word)>=3:
              tempp=word[len(word)-3: ]
              s=codecs.open("sufix.txt",'r', encoding = 'utf-8' )
              su = s.read()
              x=su.split()
              s.close()
              if x.__contains__(tempp):
                 if (condition1(tempp,word)):
                    word=condition2(tempp,word)
                    return word
                 else:
                    word=word[ :len(word)-3]
                    suffix(word)
              else:
                 tempp=word[len(word)-2: ]
                 if  x.__contains__(tempp):
                    if (condition1(tempp,word)):
                       word=condition2(tempp,word)
                       return word
                    else:
                       word=word[ :len(word)-2]
                       suffix(word)
                 else:
                    tempp=word[len(word)-1: ]
                    if  x.__contains__(tempp):
                       if (condition1(tempp,word)):
                          word=condition2(tempp,word)
                          return word
                       else:
                          word=word[ :len(word)-1]
                          suffix(word)
                    else:
                       return word
        return word
```

Figure 4.5: Python code for suffix removal

One of the challenges faced when implementing stemming on Amharic word is the algorithm adopted from Nega [39], sometimes overstem words that are inflected using prefix. This results the stem become too short word. For instance the word "ማግኘት" meaning 'to find' and "ማግባት" meaning 'to get marry' are over stemmed to "ማግ"; as a result of which both words are indexed as the same word located in different document. This situation in turn misrepresents two different words as similar and hence retrieval of irrelevant documents. This results the semantic relationship with other terms and statistical information for that term become poor.

After stemming is implemented, the next step is removing stop words from stop word list. Figure 4.6, shows how Amharic stop word is removed.

```
decs.open("stopw.txt",'r',encoding = 'utf-8')
  n=stop_words.read()
  s=n.split()
  stop=0
  for i in range(0,len(s)):
     if  (index) in s[i]:
        stop=1
  if stop==0:
     indexterms2.append (index)
```

Figure 4.6: Python code for removing stop words

The code read stop word list from text file and compared it with tokenized and stemmed index term. Then, if word is similar, it removes from index terms.

4.2. Performance evaluation

As discussed in section 3.3, Precision, Recall and F-measure are the most frequent and basic statistical measures which are widely used measures to assess the effectiveness of IR system (i.e., the quality of the search results). These three parameters are used in this research so as to measure the effectiveness of the designed probabilistic based IR system.

Before testing is done, ten Amharic test queries were formulated. The relevance judgment is prepared to construct document query matrix that shows all relevant documents for each test queries. After each test queries are measured by precision, recall and F-measure, then the average of each test queries represents the performance registered by the system. However, to facilitate computing average performance over a set of queries each with a different numbers of relevant documents, precision and recall curves which reflect precision at different standard recall levels from 0 to 1 inclusive in step of 0.1 is plotted to draw the curve.

The performance of the system is evaluated before and after relevance feedback using ten queries. Based on the performance registered, an attempt has been made to compare the result of probabilistic based IR system for Amharic language with the previously done Amharic IR system using vector space model.

To get the maximum optimal performance, a recursive testing has been made to fix threshold. For retrieving a set of relevant documents initially, the weight greater than one (>1) has been found that the optimal threshold. The weight greater than 1.5 is the optimal threshold value fixed for retrieving a set of relevant documents after relevance feedback is given. For the maximum number of iteration in giving a relevance feedback to the system, two times iterations has been found that the optimal thresholds.

Using the information given in table 4.4, evaluation is done by measuring the recall, precision and F-measure of each test queries and the average of each queries result are obtained as the initial performance of the system. Table 4.4 presents relevant documents retrieved from the Amharic corpus for each test query.

Query number	List of queries	List of relevant documents for each queries	Relevant document retrieved
1	የአደጋ ጊዜ እርዳታዎች	3, 4, 7, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21, 22, 23, 25, 30, 31, 33, 34, 39	3, 4, 7, 10, 11, 14, 18, 20, 21, 22, 23,
2	የኤች አይ ቪ ምርመራ	243, 249, 250, 262, 267, 268, 276, 293, 291, 298	243, 249, 250, 262, 267, 276
3	የመማሪያ ክፍል ግንባታ	191, 194, 197, 198, 199, 201, 203, 205, 216, 208, 221, 222, 225, 228, 229	203, 197, 194, 222, 205, 229, 199
4	ቅርሶች እንክብካቤና ጥበቃ	41, 42, 45, 49, 50, 51, 52, 53, 55, 57, 62, 64, 71, 75, 76, 78, 79	52, 57, 71, 51, 53, 55, 78
5	ጤና ጣቢያ ማስፋፊያ ስራዎች	233, 236, 239, 247, 248, 255, 256, 263, 264, 271, 279, 284, 297	236.
6	የእግርኳስ ስልጠና	162, 164, 167, 168, 173, 174, 177	164, 162, 174
7	ቴክኒክና ሙያ ማሰልጠኛ ተቋም	191, 201, 210, 208, 213,	208, 201, 191, 213
8	የሞትና የአካል ጉዳት አደጋ	1, 8, 19, 26, 27, 28, 35, 36, 38	1, 8, 19, 27, 35
9	የወባ በሽታ መከላከልና ቁጥጥር	4, 11, 238, 239, 240, 241, 250, 251, 258, 260, 267, 266, 275, 286, 287, 288, 296	4, 11, 258, 239, 250, 260, 287, 296, 288, 286, 278, 275
10	ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	123, 129, 130, 137, 143, 149, 148	143, 148, 123

Table 4.4: Test query, relevant document list and ranked list of output documents

Table 4.5 shows the effectiveness of the probabilistic Amharic IR system based on 10 queries selected for the experiment.

Query	Relevant	Retrived	Relevant retrived	P	R	F
የአደጋ ጊዜ እርዳታዎች	15	26	14	0.54	0.9	0.68
የኤች አይ ቪ ምርመራ	6	35	6	0.17	1.0	0.29
የመማሪያ ክፍል ግንባታ	10	29	9	0.31	0.9	0.46
ቅርሶች እንክብካቤና ጥበቃ	14	23	13	0.57	0.9	0.70
ጤና ጣቢያ ማስፋፊያ ስራዎች	9	25	9	0.36	1.0	0.53
የእግርኳስ ስልጠና	6	17	5	0.29	0.8	0.43
ቴክኒክናሙያ ማሰልጠኛ ተቋም	5	22	5	0.23	1.0	0.37
የሞትና የአካልጉዳት አደጋ	6	29	4	0.14	0.7	0.23
የወባ በሽታ መከላከልና ቁጥጥር	12	23	12	0.52	1.0	0.69
ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	4	45	4	0.09	1.0	0.16
			Average	0.32	0.80	0.48

Tabel 4.5: The initial performance of the system considering normalization and word variant

As it is observed from table 4.5, the average result of precision and recall of the system using the initial guess made by the model about the relevance of documents are 32% and 80% respectively. This shows that the percentage of recall dominates the percentage of precision by 48%. Finally, the F-measure, score is 48%, which indicates the performance of the system is not satisfactory.

The result depicts the system retrieved most of the relevant documents in the collection out of the total relevant documents in the corpus. However, the result of the precision indicates that, the non-relevant documents retrieved are higher than the relevant documents retrived. This is because; documents containing one of query terms but not-relevant are retrieved.

These documents are irrelevant because, the query term found in those documents not express the meaning of the query with respect to other terms found in the query. For example, for query "የአደጋ ጊዜ እርዳታዎች" which express the aid given at accidental time, the system retrieved irrelevant documents such as 'doc216', 'doc245', 'doc254' and 'doc253' because they contains query term 'አደጋ'. However, in these documents the term ''አደጋ' is used to expresses different accidental cases which are not related with aid. The same problem is also revealed for other queries.

On the other hand, in probabilistic model the initial guess of relevant document is based on Boolean expression. Thus, all terms that match one of user queries will be retrieved which increases the number of denominator used for calculating precision, thereby decreasing the percentage of precision..

Therefore, in order to increase the performance of the system, the probabilistic model uses relevance feedback from the users so as to apply query terms reweighting in order to increase the weight of terms found in relevant documents and decrease the weight of terms found in non-relevant documents.

As can be seen from Table 4.6, after users provide feedback on initially retrieved documents as relevant and non-relevant, the average percentage of precision is increased by 45%, recall is decreased by 19%. Thus, the performance of the system increased with an improvement of 25% F-measure. In this stage, the system retrieved documents that are judged as relevant by the user and other documents nearer to the judged relevant documents. However, the documents judged as non-relevant by the user are not retrieved because the weight of the query terms found in those documents is decreased, which results in excluding such documents from retrieval as relevant. The stemming algorithm implemented for the system also lacks controlling some of query word variant. For instance, relevant documents containing variant of query term "ግንባታ" such as "መገት", "ተገነባ", "መገንባት", "ሊገነባ", "እየተገነባ" are not retrieved. Because of the problem of stemmer, there are documents which are relevant but not retrieved by the system there by decreases the recall ability of the system.

Query	Relevant	Retrived	Rel-retri	P	R	F
የአደጋ ጊዜ እርዳታዎች	15	11	11	1.00	0.73	0.85
የኤች አይ ቪ ምርመራ	6	11	6	0.55	1.00	0.71
የመማሪያ ክፍል ግንባታ	10	7	7	1.00	0.70	0.82
ቅርሶች እንክብካቤና ጥበቃ	14	8	7	0.88	0.50	0.64
ጤና ጣቢያ ማስፋፊያ ስራዎች	9	1	1	1.00	0.11	0.20
የእግርኳስ ስልጠና	6	4	3	0.75	0.50	0.60
ቴክኒክናሙያ ማሰልጠኛ ተቋም	5	15	4	0.27	0.80	0.40
የሞትና የአካልጉዳት አደጋ	6	4	5	1.25	0.83	1.00
የወባ በሽታ መከላከልና ቁጥጥር	12	16	12	0.75	1.00	0.86
ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	4	12	3	0.25	0.75	0.38
			Average	0.77	0.69	0.73

Table 4.6: The performance of the system after relevance feedback is given without considering synonym words

Obtaining the value of precision at standard recall levels for each available 10 queries is important so as to show the performance of the system. Thus, interpolation of precision/recall curve is done. Figure 4.7 shows the interpolated recall-precision curve to depict the performance of the designed prototype system before and after relevance feedback is given.

As it is observed from Figue 4.7 the performance of the system register better performance when the user provides relevant feedback. The curve at the upper side of the graph in which recall and precision reaches maximum point indicates the highest performance registered by the system. At recall level, 0.6 the maximum precision registered is 0.77 this represents the average performance registered by the system.

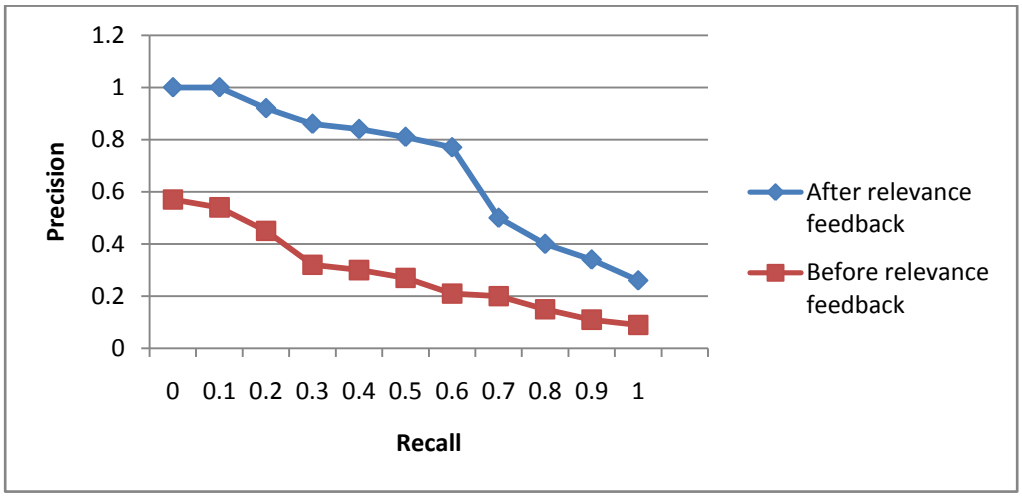

Figure 4.7: Precision/Recall curve before and after relevance feedback

4.3. Test Result on Synonymous and Polysemous

The system is also greatly affected by synonymous and polysemous nature of Amharic words. Considering documents having synonym terms of query word, system performance is evaluated. Table 4.7, shows the result of the system after users provide their relevance feedback.

As the test result in Table 4.7 indicates, as compared to the result obtained without considering synonymous word, the recall ability of the system decreases by 13% when documents having synonym words of query terms are considered. This is because, since synonym terms are not handled in this system, the result of recall highly decreases. There are several relevant documents containing synonym word of query terms. For instance, for the query ‘የመማሪያ ክፍል ግንባታ’’ relevant document ‘doc194’ not retrieved because it expressed by synonym words of query terms called ‘የትምህርት ቤት ማስፋፊያ’.

Query	Relevant	Retrived	Rel-retrie	P	R	F
የአደጋ ጊዜ እርዳታዎች	21	11	11	1.00	0.52	0.69
የኤች አይ ቪ ምርመራ	10	11	6	0.55	0.60	0.57
የመማሪያ ክፍል ግንባታ	15	7	7	1.00	0.47	0.64
ቅርሶች እንክብካቤና ጥበቃ	17	8	7	0.88	0.41	0.56
ጤና ጣቢያ ማስፋፊያ ስራዎች	13	1	1	1.00	0.08	0.14
የእግርኳስ ስልጠና	7	4	3	0.75	0.43	0.55
ቴክኒክናሙያ ማሰልጠኛ ተቋም	5	15	4	0.27	0.80	0.40
የሞትና የአካልጉዳት አደጋ	9	4	5	1.25	0.56	0.77
የወባበሽታመከላከልናቁጥጥር	18	16	12	0.75	0.67	0.71
ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	7	12	3	0.25	0.43	0.32
			Average	0.77	0.50	0.60

Table 4.7: The performance of the system after relevance feedback is given considering synonym words

4.4. Findings and Challenges

The obtained result indicates probabilistic based IR system for Amharic language register encouraging performance that increases in 10% F-measure as compared to the recent IR system developed for Amharic language by Abey [6].

However, there are several challenges faced which limits to register the optimum performance expected from the model in order to outperform the entire IR system developed for Amharic language.

The first challenge comes from the probabilistic model itself. In probabilistic model the initial guess is made based on Boolean expression. This results every documents that matches one of the term in query are retrieved. It can be possible limiting the retrieved documents using threshold; however, from the experiment it is found that if the user enter less than or equal to two queries there is a high probability of having similar weight. This will result retrieving no documents or retrieving all documents containing those terms. In this case, the precision decreases highly and the recall becomes 1.0.

On the other hand, since probabilistic model used Boolean expression for initial guess of relevant document, it does not consider the importance of the document based on the frequency of the terms in the document. For this reason, sometimes those documents having query terms with highest frequency than others could be ranked lately. In this case, users faced with the problem of having to choose the appropriate words that are also used in the relevant documents. Hence, poor result could be displayed when the system retrieve documents after feedback is given.

Additionally, one of the experimentation result shows that, in a very rare case a relevant document may not be retrieved. This is because when the weight of the terms found in documents which are similar with query terms are a sequence of negative (-) and positive (+) result, adding the score may generate a negative result. Since in probabilistic a negative result is considered as non-relevant document, the document will not be retrieved.

The other challenge comes from the problem of synonym and polysemy terms. Like any other information retrieval model, the probabilistic model has not incorporated a mechanism to control synonym or polysemous terms. As discussed in section 2.3.4, Amharic language encompasses full of synonym and polysemous words. In IR system unless there is a mechanism to control those kinds of words, the performance of the system highly decreases because relevant documents containing synonym word with of query term are not retrieved, while irrelevant documents that contains polysemy words are retrieved. For instance, for a query "እርዳታ" meaning they aid, a document contain word "ሰጡ" meaning they give and "ለገሱ" meaning they donate could not be retrieved unless it contain

the query word itself "እርዳታ". The combination of the above results leads to a decrease in the performance of the system in both precision and recall.

Table 4.6 shows, some examples of test queries with their word variants uncontrolled by the stemming algorithm implemented for the system and synonym terms of queries found in relevant documents but not retrieved.

Queries	Synonyms words	Variants words
የአደጋ ጊዜ እርዳታዎች	'ልገሳ', 'ሰጡ', 'አበረከቱ'	'ረዱ', 'በመርዳት', 'ሲረዱ'
የኤች አይ ቪ ምርመራ	'ኤድስ'	'ተመረመሩ', 'እንዲመረመሩ'
የመማሪያ ክፍል ግንባታ	'ታንጸዉ', 'ተሰሩ', 'ትምህርት ቤት' 'ህንጻ'	'ተገነቡ', 'ይገነባሉ', 'በመገንባትላይ', 'አስገንብቶ'
ጤና ጣቢያ ማስፋፊያ ስራዎች	'ሆስፒታል', 'ጤና ኬላ', 'ተግባራት'	
ቅርሶች እንክብካቤና ጥበቃ	'ክትትል' 'እድሳት'	
የሞትና የአካልጉዳት አደጋ	'ህይወቱ አለፈ' 'አረፉ'	
የወባ በሽታ መከላከልና ቁጥጥር	'ህመምተኞች'	'ወባኝክ'
ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ	'ተቋማት' 'ተግማራት' 'ስራ'	

Table 4.8: Test queries with their synonym word and word variant

Finding a large size and standard corpus for Amharic language is also considered as one of the challenge faced in this research. The state of the art in the area of text processing indicates that, there is no any developed standard corpus for Amharic language. Thus, in this research the researcher uses small size corpus extracted from Walta Information Center website [21]. This result not only weakens the performance of the system but also makes it difficult to compare the result obtained with several researches since there is different in test queries, document content and size used for testing.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1. Conclusion

Information retrieval is the mechanism for finding relevant documents from unstructured document collection that satisfies information need of the users [4]. It is obvious that the main goal of information retrieval system is to retrieve relevant information. Several models have been developed considering that goal. However, most of them have the problem of defining uncertainty exists in IR system. One of the model which have the ability to control the uncertainty in IR system is probabilistic model. It incorporates a learning or relevance feedback component that can improve performance when presented with sample relevant documents.

Therefore, in this research an attempt has been made to design and develop a probabilistic based information retrieval system for Amharic language in order to enhance the retrieval performance of Amharic IR system considering the advantage of the probabilistic model.

In designing the IR system, Binary Independent Model (BIM) is selected and implemented. At first step when the search component initiated the system generates the first ranked list of relevant documents according to the initial guess made based on the notion of the model, then the system asked for evidences from the user about the relevancy or non-relevancy of the document. Finally based on the evidence and users relevance feedback the system improves its performance.

System evaluation has been done to discover the extent to which the designed system enhances the performance of Amharic IR system based on the F-measure. As the experimental result show, probabilistic based Amharic IR system register a better performance and score on the average 73% F-measure. This is a promising result to design an applicable IR system if polysemous and synonymous nature of Amharic words is controlled with the help of thesaurus and co-occurrence analysis.

5.2. Recommendation

The designed probabilistic based Amharic IR system is just the first attempt to see the advantage of the model in order to enhance the performance of IR system for Amharic text. Therefore, to obtain the optimum performance expected form the model, this work can be further pursued in several future directions.

- Probabilistic model make the initial guess based on Boolean expression, which inhibit to know important words to represent a document and, accordingly may not retrieve relevant documents that contain large number of terms found in a given query. Hence, there is a need to build hybrid system that uses vector space model to guess relevant documents for user query using non-binary weighting technique and then use probabilistic relevance feedback to improve the performance of the system.

- One of the problems in enhancing the performance of Amharic IR system is the existence of synonyms and polysemy terms in Amharic text. We recommended integrating mechanisms of controlling synonym and polysemy terms in the probabilistic model to enhance precision and recall of the system.

- The stemming algorithm used in this research is the best algorithm developed so far for Amharic language. However, it frequently overstems word variants greatly affecting the performance of the system. Therefore, future word need to consider designing ontology based stemming algorithm that conflates based on meaning understanding.

- Finding a standard corpus and test queries with relevance judgement for testing the designed system is one of the challenges faced in this research. Therefore, future research need to consider the development of standard Amharic corpus that can be used by researcher to evaluate progress made in designing Amharic IR system.

- Sometimes users may need a document which contains combination of query terms by using conjugations words such as 'ና' or'እና' meaning AND,'ወይም' meaning OR and 'ሳይሆን' meaning NOT. This demands an intelligent IR system that reconstructs query terms as per user's requirement. Hence, there is a need to integrate query reconstruction mechanism to the Amharic IR system.

REFERENCES

[1] H.S. Christopher D.Manning, Prabhakar Raghavan, "An Introduction to Information Retrieval", 1st Edition, Cambridge University Press, Cambridge England, 2009.

[2] A. Singhal, "Modern Information Retrieval: A Brief Overview", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 24(4): pp. 35-42, 2001.

[3] T. Saracevic, "Information science: Encyclopedia of Library and Information Science", 3rd Edition, Taylor and Francis publisher, New Jersey, USA, 2009.

[4] B. R., Ribeiro Neto, "Modern Information Retrieval", 2nd Edition, Addison-Wesley-Longman Publishers, New York, USA 1999.

[5] D. Hiemstra, T. Pothoven, M.V. Vliet, and D. Harman, "Behind the Scenes of the Digital Museum of Information Retrieval Research", In Proceedings of the 9th Dutch-Belgian Information Retrieval Workshop, 9(1):99-100, 2009.

[6] Abey Bruck, "Semantic Based Query Expansion Technique for Amharic IR", MSc Thesis, School of Information Science, Addis Ababa University, Ethiopia, 2011.

[7] N. Fuhr, "Probabilistic Models in Information Retrieval", The Computer Journal, 35(3):243-255, 1992.

[8] "Informaiton Retrieval Model", Available at; http://comminfo.rutgers.edu /~aspoerri/InfoCrystal/ Ch_2.htm, Accessed Date; 12/5/2012, 2012.

[9] J. Picard and R. Haenni, "Modeling Information Retrieval with Probabilistic Argumentation Systems", 20th BCS-IRSG Annual Colloquium on IR, 14(7):10-25,1998.

[10] Nlp.stanford.edu, "Probabilistic Information Retrieval", Available at; http://nlp.stanford.edu/IR-book/html/htmledition/probabilistic-information-retrieval-1.html, Accessed Date; 27/1/2012, 2008.

[11] K.S. S.E.Robertson, "Relevance Weighting of Search Terms", Journal of the American Society for Information Science, 27(4):129-146, 1976.

[12] C.J. Rijsbergen, "A Theoretical Basis for the Use of Co-occurrence Data in Information Retrieval", Journal of Documentation, 23(2):106-–119, 1977.

[13] F.D. Commission, "Summary and Statistical Report of the 2007 Population and Housing Census", pp. 1-113, 2008.

[14] Betelihem M., "N-Gram-Based Automatic Indexing for Amharic Text", MSc Thesis, Addis Ababa University School of Information science, Ethiopia, 2002.

[15] Tewodros H., "Amharic text retrieval: An Experiment Using Latent Semantic Indexing (LSI) with Singular Value Decomposition (SVD)" , MSc Thesis, Addis Ababa University, School of Information science, Ethiopia, 2003.

[16] Tesema M. and Solomon A. , "Design and Implementation of Amharic Search Engine", fifth international conference on signal image technology and internet based systems, 3(1):318-325, 2009.

[17] J.B. Teevan, "Bayesian Model for Information Retrieval", Available at; http://www.ai.mit.edu/research/abstracts/abstracts2000/pdf/z-teevan1.pdf, Accessed Date; 29/1/2012, 2000.

[18] V.H. Surajit Chaudhuri, Gautam Das, "Probabilistic Information Retrieval Approach for Ranking of Database Query Results", ACM Transactions on Database systems (TODS), 31(3): 1-43, 2004.

[19] S.E. Xiangji Huang, "Okapi Chinese Text Retrieval Experiments at TREC-6", in Proceedings of TREC'1997, 16(4) 552-569, 2000.

[20] L. Dolamic and J. Savoy, "Indexing and Searching Strategies for the Russian Language", Journal of the American Society for Information Science, 60(24):2540-2547, 2009.

[21] Abiy Z. , Alemayehu W. , Daniel T. , Melese G. and Yilma S. , "Introduction to Research Methods", 1st Edition, Graduate studies and Research Office of Addis Ababa University, 2009.

[22] Walta-Information-Centre, "Local News Papers", Available at; http://www.newspapersites.net/ newspaper/ walta- information-centre. Accessed Date; 11/1/2012, 2012.

[23] M. Lutuz, "Learning Python" , O'Reilly publish, USA ,4th Edition, 2009.

[24] K. D.Lee, "Python Programming Fundamentals" , Springer-verlog press, USA, 1st Edition, 2011.

[25] T. Mandl, "Recent Developments in the Evaluation of Information Retrieval Systems : Moving Towards Diversity and Practical Relevance", the European journal for the information professional, . 32(5) 27-38, 2008.

[26] www.dsoergel.com, "The Scope of IR Utility , Relevance , and IR System Performance", Available at; http:// www.dsoergel.com%2FNewPublications%2FHCIEncyclopediaIRShortEF orDS .pd, Accessed Date; 7/2/2012, 1997.

[27] D. Hiemstra, "Information Retrieval Models", John Wiley and sons publisher, Enschede-Noord, Netherlands , 1st Edition, 2009, 2009.

[28] Ed Greengrass, "Information Retrieval : A Survey by Ed Greengrass", Available at; www.csee.umbc.edu/cadip/readings/IR.report.120600.book.pdf, Accessed Date; 3/12/2011, 2011.

[29] H.R. Turtle and W.B. Croft, "A Comparison of Text Retrieval Models", The Computer Journal, 35(2): 279-290,1992.

[30] S.W. k. Sparck Jones, "A probabilistic Model of Information Retrieval: Development And Comparative Experiments", Information Processing and Management, 36(4): 779-808, 2000.

[31] Ruggeri F, Faltin F and Kenett R, Bayesian Networks, Available at; http://www.eng.tau.ac.il /~bengal/BN.pdf, Accessed Date; 17/3/2012, 2012

[32] M.L. Krieg, "A Tutorial on Bayesian Belief Networks", Available at; http://www.dsto.defence. gov . au, Accessed Date; 10/1/2012, 2012.

[33] Shipeng y, Deng C, Ji-Rong W and Wei-Ying M, "Improving Pseudo-Relevance Feedback in Web page Segmentation", Proceedings of the 12th International Conference on World Wide Web, 1(1) pp. 11 - 18, 2003.

[34] Arnaud F and Renata D, "Rocciho's Relevance feedback Algorithm in Basic Vector Space Comparison and LSI Models", Available at; http://www.mpi-inf.mpg.de/~bast/ir-seminar-ws04/project-proposals/renata_dividino_arnaud_fietzke.pdf, Accessed date: 15/4/2012, 2012

[35] T. Bloor, "The Ethiopic Writing System: a Profile", Journal of the Simplified Spelling Society, 19(1): 30-36, 1995.

[36] M. Bender, "The Ethiopic Writing System", Oxford University Press, London, 1976.

[37] Saba A, "The Application of Information Retrieval Techniques to Amharic Documents on the Web", MSc Thesis, Addis Ababa University School of Information Studies for Africa, Ethiopia, 2001.

[38] Alemayehu ,"Application of Query Expansion for Amharic information retrieval System", MSc Thesis, Addis Ababa University, School of Information Science, Ethiopia, 2002.

[39] Nega A. and P. Willett, "Stemming of Amharic Words for Information Retrieval", in Literary and Linguistic Computing. Oxford University press, Oxford London, 17(1): pp. 1-18, 2002.

[40] C.J. Rijsbergen, "Information Retrieval" , Butterworth-Heinemann publisher, USA, 2nd edition, 1979.

[41] M.J. Gerard Salton, "Introduction to Modern Information Retrieval" , Available at; http://www.sifaka.cs.uiuc.edu/course/410s12/mir.pdf , Accessed Date; 20/2/2012, 2012.

APPENDIXES

Appendix 1: Amharic character set

	ā/ä [a]	u [u]	ī/i [i]	a [a]	ē/e [e/ɛ]	(i)/(ə) [ə]	0 [o/ɔ]
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
[h]	ha	hu	hi	ha	he	h(ə)	ho
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
[l]	le	lu	li	la	le	l(ə)	lo
h/ḥ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
[h]	ha	hu	hi	ha	he	h(ə)	ho
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
[m]	me	mu	mi	ma	me	m(ə)	mo
s/ś	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
[s]	se	su	si	sa	se	s(ə)	so
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
[r]	re	ru	ri	ra	re	r(ə)	ro
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
[s]	se	su	si	sa	se	s(ə)	so
sh/š	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
[ʃ]	ʃe	ʃu	ʃi	ʃa	ʃe	ʃ(ə)	ʃo
k’/q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
[k’]	k’e	k’u	k’i	k’a	k’e	k’(ə)	k’o
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
[b]	be	bu	bi	ba	be	b(ə)	bo
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
[t]	te	tu	ti	ta	te	t(ə)	to
ch/č	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ

	ā/ä [a]	u [u]	ī/i [i]	a [a]	ē/e [e/ɛ]	(i)/(ə) [ə]	o [o/ɔ]
h/ḫ	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
[h]	he	hu	hi	ha	he	h(ə)	ho
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
[w]	we	wu	wi	wa	we	w(ə)	wo
‘/ʿ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
[ʔ]	ʔa	ʔu	ʔi	ʔa	ʔe	ʔ(ə)	ʔo
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
[z]	ze	zu	zi	za	ze	z(ə)	zo
zh/ž	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
[ʒ]	ʒe	ʒu	ʒi	ʒa	ʒe	ʒ(ə)	ʒo
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
[j]	je	ju	ji	ja	je	j(ə)	jo
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
[d]	de	du	di	da	de	d(ə)	do
j/ǧ	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
[ʤ]	ʤe	ʤu	ʤi	ʤa	ʤe	ʤ(ə)	ʤo
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
[g]	ge	gu	gi	ga	ge	g(ə)	go
t’/ṭ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
[t’]	t’e	t’u	t’i	t’a	t’e	t’(ə)	t’o
ch’/č̣	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
[ʧ’]	ʧ’e	ʧ’u	ʧ’i	ʧ’a	ʧ’e	ʧ’(ə)	ʧ’o
p’/ṗ	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ

[ɸ]	ɸe	ɸu	ɸi	ɸa	ɸe	ɸ(ə)	ɸo
h/ḫ	ሃ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
[h]	ha	hu	hi	ha	he	h(ə)	ho
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
[n]	ne	nu	ni	na	ne	n(ə)	no
ny/ñ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
[ɲ]	ɲe	ɲu	ɲi	ɲa	ɲe	ɲ(ə)	ɲo
’/’	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
[ʔ]	(ʔ)a	(ʔ)u	(ʔ)i	(ʔ)a	(ʔ)e	(ʔ)(ə)	(ʔ)o
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
[k]	ke	ku	ki	ka	ke	k(ə)	ko

[p’]	p’e	p’u	p’i	p’a	p’e	p’(ə)	p’o
ts’/ṣ	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
[ts]	ts’e	ts’u	ts’i	ts’a	ts’e	ts’(ə)	ts’o
ts’/ṣ́	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
[ts]	ts’e	ts’u	ts’i	ts’a	ts’e	ts’(ə)	ts’o
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
[f]	fe	fu	fi	fa	fe	f(ə)	fo
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
[p]	pe	pu	pi	pa	pe	p(ə)	po
v	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ
[v]	ve	vu	vi	va	ve	v(ə)	vo

Appendix 2: Amharic punctuation mark list set

፣	።	፤	፥	፦	፧
comma	full stop / period	colon	semi-colon	preface colon	question mark (no longer used)

Appendix 3: Amharic numbering set

፩	፪	፫	፬	፭	፮	፯	፰	፱	፲
1	2	3	4	5	6	7	8	9	10
፳	፴	፵	፶	፷	፸	፹	፺	፻	፼
20	30	40	50	60	70	80	90	100	10000

Appendix 4: Relevance judgment used

Documents	የኢደጋ ጊዜ እርዳታዎች	የኤች አይ ቪ ምርመራ	የመማሪያ ክፍል ማንባታ	ቅርሶች እንክብካቤ ና ጥበቃ	ጤናጣቢያ ማስፋፊያ ስራዎች	የእግርኳስ ስልጠና	ቴክኒክና ሙያ ማሰልጠኛ ተቋም	የሞትናየአካል ጉዳትአደጋ	የወባ በሽታ መከላከልና ቁጥጥር	ድርጅቶች የሚያደርጉት የልማት እንቅስቃሴ
doc001	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc002	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc003	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc004	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc005	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc006	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc007	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc008	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc009	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc010	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc011	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc012	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc013	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc014	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc015	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc016	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc017	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc018	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc019	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc020	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc021	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc022	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc023	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc024	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc025	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc026	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc027	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc028	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc029	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc030	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc031	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc032	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc033	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc034	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc035	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc036	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc037	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc038	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant
doc039	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc040	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc041	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc042	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc043	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc044	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc045	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc046	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc047	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc048	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc049	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc050	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc051	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc052	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc053	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc054	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc055	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc056	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc057	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc058	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc059	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc060	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc061	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc062	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc063	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc064	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc065	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc066	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc067	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc068	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc069	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc070	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc071	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc072	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc073	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc074	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc075	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc076	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc077	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc078	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc079	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc080	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc081	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc082	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc083	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc084	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc085	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc086	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc087	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc088	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc089	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc090	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc091	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc092	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc093	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc094	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc095	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc096	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc097	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc098	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc099	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc100	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc101	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc102	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc103	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc104	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc105	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc106	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc107	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc108	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc109	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc110	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc111	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc112	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc113	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc114	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc115	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc116	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc117	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc118	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc119	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc120	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc121	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc122	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc123	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc124	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc125	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc126	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc127	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc128	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc129	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc130	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc131	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc132	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc133	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc134	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc135	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc136	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc137	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant

doc138	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc139	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc140	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc141	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc142	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc143	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc144	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc145	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc146	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc147	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc148	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc149	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant
doc150	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc151	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc152	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc153	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc154	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc155	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc156	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc157	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc158	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc159	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc160	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc161	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc162	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc163	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc164	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc165	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc166	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc167	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc168	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc169	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc170	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc171	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc172	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc173	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc174	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc175	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc176	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc177	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc178	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc179	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc180	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc181	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc182	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc183	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc184	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc185	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc186	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc187	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc188	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc189	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc190	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc191	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc192	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc193	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc194	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc195	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc196	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc197	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc198	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc199	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc200	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc201	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc202	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc203	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc204	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc205	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc206	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc207	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc208	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc209	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc210	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc211	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc212	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc213	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant
doc214	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc215	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc216	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc217	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc218	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc219	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc220	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc221	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc222	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc223	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc224	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc225	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc226	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc227	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc228	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc229	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc230	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc231	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc232	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc233	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc234	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc235	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc236	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc237	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc238	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc239	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc240	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc241	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc242	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc243	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc244	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc245	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc246	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc247	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc248	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc249	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc250	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc251	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc252	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc253	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc254	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc255	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc256	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc257	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc258	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc259	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc260	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc261	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc262	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc263	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc264	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc265	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc266	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc267	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc268	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc269	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc270	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc271	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc272	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc273	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc274	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc275	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc276	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc277	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

doc278	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc279	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc280	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc281	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc282	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc283	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc284	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc285	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc286	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc287	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc288	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc289	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc290	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc291	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc292	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc293	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc294	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc295	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc296	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant
doc297	non-relevant	non-relevant	non-relevant	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc298	non-relevant	relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc299	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant
doc300	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant	non-relevant

Appendix 5: Probabilistic Amharic IR system python code

- Indexing module

```
import os
import math
import codecs
import string
import re
import glob
import sys
class amhariIndexing:
   def index(self, path):
      def condition1(affix, tempp):
         if len(tempp)> 2:
            if affix.__eq__(u'\u12a8'):
                 return 1
            elif affix.__eq__(u'\u1260'):
                 return 1
            elif affix.__eq__(u'\u129E\u127D'):
                 return 1
            elif affix.__eq__(u'\u1295'):
                 return 1
            elif affix.__eq__(u'\u1275'):
                 return 1
            elif affix.__eq__(u'\u1290\u1275'):
                 return 1
            elif affix.__eq__(u'\u12CE\u127D'):
                 return 1
            elif affix.__eq__(u'\u127D'):
                 return 1
            elif affix.__eq__(u'\u127D\u1295'):
                 return 1
            elif affix.__eq__(u'\u127D\u1201'):
                 return 1
            elif affix.__eq__(u'\u1278\u12CD'):
                 return 1
            elif affix.__eq__(u'\u1271'):
                 return 1
            elif affix.__eq__(u'\u1279'):
                 return 1
            elif affix.__eq__(u'\u12CA'):
                 return 1
            elif affix.__eq__(u'\u12CA\u1275'):
                 return 1
            elif affix.__eq__(u'\u12CD\u12EB\u1295'):
                 return 1
            elif affix.__eq__(u'\u1277\u120D'):
                 return 1
            elif affix.__eq__(u'\u121D'):
```

```
            return 1
        elif affix.__eq__(u'\u1275') or affix.__eq__(u'\u12A0'):
            return 1
        elif affix.__eq__(u'\u129B'):
            return 1
        elif affix.__eq__(u'\u1290\u1275') or affix.__eq__(u'\u12A5'):
            return 1
        elif affix.__eq__(u'\u121D') or affix.__eq__(u'\u12A5'):
            return 1
        elif affix.__eq__(u'\u1208\u1275'):
            return 1
    else:
        return 0

def condition2(affix,tempp):
    if len(tempp) > 2:
        if affix.__eq__(u'\u12A8'):
            if tempp.__eq__(u'\u12A8\u1265\u1275') or tempp.__eq__(u'\u12A8\u1208\u12A8\u1208') or
       tempp.__eq__(u'\u12A8\u1290\u12A8\u1290') or tempp.__eq__(u'\u12A8\u1228\u12A8\u1228') or
       tempp.__eq__(u'\u12A8\u1128\u12A8\u1218') or tempp.__eq__(u'\u12A8\u1230\u12A8\u1230') or
       tempp.__eq__(u'\u12A8\u1270\u12A8\u1270') or tempp.__eq__(u'\u12A8\u1270\u121B') or
       tempp.__eq__(u'\u12A8\u1228\u1295') or tempp.__eq__(u'\u12A8\u1230\u120D') or
       tempp.__eq__(u'\u12A8\u134D\u1270\u129B') or tempp.__eq__(u'\u12A8\u1265\u1275'):
                return tempp
            else:
                tempp = tempp[1: ]
                tempp = preffix(tempp);
                return tempp
        elif affix.__eq__(u'\u1260'):
            if tempp.__eq__(u'\u1260\u123D\u1273') or tempp.__eq__(u'\u1260\u1300\u1275') or
       tempp.__eq__(u'\u1260\u1228\u1260\u1228') or tempp.__eq__(u'\u1260\u1230\u1260\u1230'):
                return tempp
            else:
                tempp = tempp[1: ]
                tempp = preffix(tempp);
                return tempp
        elif affix.__eq__(u'\u129B'):
            if tempp.__eq__(u'\u1260\u123D\u1273'):
                return tempp
            else:
                tempp = tempp[ :-1]
                tempp = preffix(tempp);
                return tempp
        elif affix.__eq__(u'\u1295'):
            if tempp.__eq__(u'\u12D8\u1218\u1295') or tempp.__eq__(u'\u1261\u12F5\u1295') or
       tempp.__eq__(u'\u123D\u134B\u1295') or tempp.__eq__(u'\u12C8\u1308\u1295') or
       tempp.__eq__(u'\u1261\u12F5\u1295') or tempp.__eq__(u'\u123D\u134B\u1295') or
       tempp.__eq__(u'\u12C8\u1308\u1295') or tempp.__eq__(u'\u132D\u1241\u1295') or
       tempp.__eq__(u'\u1205\u133B\u1295') or tempp.__eq__(u'\u12A5\u12CD\u1295') or
       tempp.__eq__(u'\u12A5\u121D\u1295') or tempp.__eq__(u'\u12D8\u1348\u1295'):
                return tempp
            else:
```

```
        tempp = tempp[ :-1]
        tempp=sadisConverter(tempp)
        tempp = suffix(tempp)
        return tempp
 elif affix.__eq__(u'\u1208\u1275'):
    if tempp.__eq__(u'\u12A0\u1208\u1275') or tempp.__eq__(u'\u12A5\u1208\u1275') or
tempp.__eq__(u'\u1201\u1208\u1275'):
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp);
        return tempp
 elif affix.__eq__(u'\u1275'):
    if tempp.__eq__(u'\u12A0\u1263\u1275') or tempp.__eq__(u'\u12A0\u12EB\u1275') or
tempp.__eq__(u'\u1205\u12ED\u12C8\u1275') or tempp.__eq__(u'\u12A0\u122B\u12CA\u1275') or
tempp.__eq__(u'\u1230\u12A3\u1275') or tempp.__eq__(u'\u1218\u1265\u1275') or
tempp.__eq__(u'\u1218\u1230\u1228\u1275') or tempp.__eq__(u'\u12C8\u1245\u1275') or
tempp.__eq__(u'\u1325\u1228\u1275') or tempp.__eq__(u'\u1275\u122D\u134D') or
tempp.__eq__(u'\u1201\u1208\u1275') or tempp.__eq__(u'\u1236\u1235\u1275') or
tempp.__eq__(u'\u12A0\u122B\u1275') or tempp.__eq__(u'\u12A0\u121D\u1235\u1275') or
tempp.__eq__(u'\u1235\u12F5\u1235\u1275') or tempp.__eq__(u'\u1230\u1263\u1275') or
tempp.__eq__(u'\u1235\u121D\u1295\u1275') or tempp.__eq__(u'\u1275\u121D\u1205\u122D\u1275')
or tempp.__eq__(u'\u1325\u1245\u121D\u1275') or tempp.__eq__(u'\u1233\u121D\u1295\u1275') or
tempp.__eq__(u'\u12AD\u122B\u12CA\u1275'):
       return tempp
    else:
        tempp = tempp[ :-1]
        #change to sades
        tempp = suffix(tempp);
        return tempp
 elif affix.__eq__(u'\u1290\u1275'):
    if tempp.__eq__(u'\u12A5\u12CD\u1290\u1275') or tempp.__eq__(u'\u12A5\u121D\u1290\u1275'):
        return tempp
    else:
        tempp = tempp[ :-1]
        #change to sades
        tempp = suffix(tempp);
        return tempp
 elif affix.__eq__(u'\u12CE\u127D'):
    if tempp.__eq__(u'\u1230\u12CE\u127D'):
        str1 = tempp[-1: ]
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp)
        return tempp
 elif affix.__eq__(u'\u129E\u127D'):
    if tempp.__eq__(u'\u12F3\u129E\u127D'):
        tempp = tempp[ :-1]
        ##change to sades
```

```
        return tempp
    else:
        tempp = tempp[ :-1]
        #change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D\u1295'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D\u1201'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1278\u12CD'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1271'):
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1279'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u12CA'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u12CA\u1275'):
    if tempp.__eq__(u'\u12A0\u122B\u12CA\u1275'):
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp);
        return tempp
elif affix.__eq__(u'\u12CD\u12EB\u1295'):
        tempp = tempp[ :-3]
        ##change to sades
        tempp = suffix(tempp)
```

```
            return tempp
        elif affix.__eq__(u'\u1277\u120D'):
            if tempp.__eq__(u'\u121E\u1277\u120D'):
                tempp = tempp[ :-1]
                ##change to sades
                return tempp
            else:
                tempp = tempp[ :-2]
                #change to sades
                tempp = suffix(tempp)
                return tempp
        elif affix.__eq__(u'\u121D'):
            if tempp.__eq__(u'\u1308\u12F3\u121D') or tempp.__eq__(u'\u1230\u120B\u121D') or
       tempp.__eq__(u'\u12A0\u1208\u121D'):
                return tempp
            else:
                tempp = tempp[ :-1]
                #change to sades
                tempp = suffix(tempp);
                return tempp
    else:
        return tempp
def punctuationremove(str):
    if str.endswith('\n'):
        str=str.rstrip('\n')
    if str.startswith(' '):
        str=str.lstrip(' ')
    p=codecs.open("Punctuation.txt",'r', encoding = 'utf-8')
    Punctuation = p.read()
    for pc in Punctuation:
        if str.endswith(pc):
            str=str.rstrip(pc)
        if str.startswith(pc):
            str=str.lstrip(pc)
    for j in Punctuation:
        for i in range(1,len(str)):
            if j in Punctuation and j in str:
                str=str.strip(j)
    return str
def normal(str):
    sim=codecs.open("similarWord.txt",'r', encoding = 'utf-8' )
    similarWord=sim.read()
    li1=[]
    for m in similarWord.split():
        li1.append(m)
    for i in range(1,len(li1),2):
        if str==li1[i] or str==li1[i+1]:
            str=li1[i+1]
    temp=codecs.open("similarChar.txt",'r', encoding = 'utf-8' )
    similarChar=temp.read()
    li=[]
    for j in similarChar.split():
```

```
        li.append(j)
    for i in range(1,len(li),2):
        str=str.replace(li[i],li[i+1])
    return str
def preffix(word):
    if len(word)<3:
        return word
    else:
        while len(word)>=3:
            tempp=word[ :3]
            p=codecs.open("prefix.txt",'r', encoding = 'utf-8' )
            pr = p.read()
            y=pr.split()
            p.close()
            if y.__contains__(tempp):
                if (condition1(tempp,word)):
                    word=condition2(tempp,word)
                    return word
                else:
                    word=word[3: ]
                    preffix(word)
            else:
                tempp=word[ :2]
                if  y.__contains__(tempp):
                    if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                    else:
                        word=word[2: ]
                        preffix(word)
                else:
                    tempp=word[ :1]
                    if  y.__contains__(tempp):
                        if (condition1(tempp,word)):
                            word=condition2(tempp,word)
                            return word
                        else:
                            word=word[1: ]
                            preffix(word)

                    else:
                        return word

    return word
def suffix(word):
    if len(word)<3:
        return word
    else:
        while len(word)>=3:
            tempp=word[len(word)-3: ]
            s=codecs.open("sufix.txt",'r', encoding = 'utf-8' )
            su = s.read()
```

```
            x=su.split()
            s.close()
            if x.__contains__(tempp):
               if (condition1(tempp,word)):
                  word=condition2(tempp,word)
                  return word
               else:
                  word=word[ :len(word)-3]
                  suffix(word)
            else:
               tempp=word[len(word)-2: ]
               if  x.__contains__(tempp):
                  if (condition1(tempp,word)):
                     word=condition2(tempp,word)
                     return word
                  else:
                     word=word[ :len(word)-2]
                     suffix(word)
               else:
                  tempp=word[len(word)-1: ]
                  if  x.__contains__(tempp):
                     if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                     else:
                        word=word[ :len(word)-1]
                        suffix(word)

                  else:
                     return word


      return word
   def fileReader(filename):
      indexterms=[]
      infile=codecs.open(filename, encoding='utf-8')
      lterms=infile.readlines()
      infile.close()
      for str in lterms:
         if str.endswith('\n'):
            str=str.rstrip('\n')
         indexterms.append(str)
      return indexterms
   def docsReader(pst):
      ldocid=[]
      infile=open(pst)
      lterms=infile.readlines()
      infile.close()
      for str in lterms:
         if str.endswith('\n'):
            str=str.rstrip('\n')
         ldocid.append(str)
```

```
    return ldocid
def indexTerm(ldocid):
    fileterms=[]
    List2 = []
    List3 = []
    L7 = []
    List4 = []#store terms in each document in sorted order??
    List22 = []#store unique terms from the document
    indexterm=[]
    cff = {}#store terms and its collection frequency in a dictinay
    tff = {}#store terms and its term frequency in a dictinay
    dff = {}#store terms and its document frequency in a dictinay
    posFile=[]#store docid and tf of terms
    docID = 0#store docid
    L9 = []#L9 store term, docid and tf for all document
    L10=[]#L10 stor the docid and tf
    stop=codecs.open("stopw.txt",'r', encoding = 'utf-8')
    stopw=stop.read()
    # the loop that reads the id of each document togather
    for docid in ldocid:
        docID = docID + 1
        List5 = []
        aList1 = []#store the key of dictionay which is unique index terms as a list
        aList2 = []#store the value(tf)
        aList3 = []#store the tf for further purpose
        res = []
        L8 = []#store term,docid and tf
        #fileterms store terms in each documents
        fileterms=fileReader(docid)
        for term in fileterms:
            List1 = []#store every index in every docs terms as a list
            L6 = []#have terms
            indexterms=[]#indexterms[] store each terms in the documents that are stemmed and punc removed
            for index in term.split():
                if index.isalpha():
                    stop_words=codecs.open("stopw.txt",'r',encoding = 'utf-8')
                    n=stop_words.read()
                    s=n.split()
                    stop=0
                    for i in range(0,len(s)):
                        if suffix(preffix(punctuationremove(index)))in s[i]:
                            stop=1
                    if stop==0:
                        indexterms.append(suffix(preffix(punctuationremove(index))))
                for k in indexterms:
                    List1.append(k)
                List4 = sorted(List4 + List1)
                List22 = list(set(List4))
                List5 = sorted(List5 + List1)
    #term frequency count loop
        for tf in List5:
            try:
```

```
            tff[tf] = tff[tf] + 1
        except:
            tff[tf] = 1
        tfff = sorted(tff.items())
    for keyValue in range(len(tfff)):
        aList2.append(tfff[keyValue][0])#index terms
        aList1.append(tfff[keyValue][1])#tf of terms
    for kk in aList1:
        aList3.append(kk)#aList3 store tf for futher process purpose
        posFile.append((docID, kk))
    ListDict = dict(zip(aList2,aList3))
    for key in ListDict:
        L6.append(key)#have terms
        L8.append((key, docID, ListDict[key]))#store term,docid and tf for a single doc
    L7 = sorted(L7 + L6)
    tff = {}
    List3 = sorted(List3 + List2)
    L9 = sorted(list(L9 + L8))#L9 store term, docid and tf for all document
#the loop count the collection frequency of terms
for cf in List4:
    try:
        cff[cf] = cff[cf] + 1
    except:
        cff[cf] = 1
    cfreq = sorted(cff.items())
#the loop count the document frequency of terms
for df in L7:
    try:
        dff[df] = dff[df] + 1
    except:
        dff[df] = 1
    dfreq2 = sorted(dff.items())
y=[]#store terms by extracting from cf
for kk2 in range(len(cfreq)):
    y.append(cfreq[kk2][0])

TDFCF = []#store index term df and cf as a single list
for k1 in range(len(dfreq2) and len(cfreq)):
    TDFCF.append((dfreq2[k1][0], dfreq2[k1][1], cfreq[k1][1]))
L33 = []
ID = 0
L44=[]
for j in ldocid:
    ID = ID + 1
    f=codecs.open(j, 'r', encoding = 'utf-8')
    f2=f.read()
    dic={}
    for term in y:
        if term not in dic:
            found=f2.find(term)
            while found >-1:
                if term not in dic:
```

```
                    dic[term]=found
                    found=f2.find(term, found+1)
                else:
                    dic[term]=((str(dic[term]))+ "    "+(str(found)))
                    found=f2.find(term, found+1)
            for k in range(len(L9)):
                if term == L9[k][0] and ID == L9[k][1]:
                    L10.append((term, ID, L9[k][2], dic[term]))
                    L33.append((term,ID))
                    L44=sorted(L10)
                else: continue
        f.close()
        L11=[]#store term, df and cf
        for i in range(len(TDFCF)):
            L22 = []
            L11.append((normal(TDFCF[i][0]), TDFCF[i][1], TDFCF[i][2]))
            for j in range(len(L33)):
                if TDFCF[i][0] == L33[j][0]:
                    L22.append(L33[j][0])
                else: continue
            L11.append(tuple(L22))
        fiLe=codecs.open("collectionfrequency.txt", 'w', encoding = 'utf-8')
        fiLe.write(str("Term")+ "\t" + str("CF") + "\t" + "\n")
        for i in range(0, len(L11), 2):
            fiLe.write(L11[i][0]  + "\t" + str(L11[i][2]) + "\t")
            fiLe.write("\n")
        fiLe.close()
        fiLe=codecs.open("vocabularyFile.txt", 'w', encoding = 'utf-8')
        fiLe.write(str("Term") + "\t" + str("DF") + "\t" + str("CF") + "\t" + "\n")
        for i in range(0, len(L11), 2):
            fiLe.write(L11[i][0] + "\t" + str(L11[i][1]) + "\t" + str(L11[i][2]) + "\t")
            fiLe.write("\n")
        fiLe.close()
        fiLe=codecs.open("postingFile.txt", 'w', encoding = 'utf-8')
        #fiLe.write(str("docID") + "\t" + str("TF") + "\n")
        for key in range(len(L44)):
            print >> fiLe,(L44[key][1]),"\t" ,
            print >> fiLe,(L44[key][2]),"\t" ,
            print >> fiLe,(L44[key][3])
        fiLe.close()

    def test(indexlist, ldocid):
        totDocs = int(len([fiLe for fiLe in os.listdir("./corpus") if os.path.isfile(os.path.join("./corpus", fiLe))]))
    return test(indexTerm(docsReader(path)), docsReader(path))
ob=amhariIndexing()
f1=open("documentsID.txt",'w')
for i in os.listdir('C:\Users\Amanuel\Desktop\probability prototype2\corpus'):
    docs="corpus"
    docs=docs + '/' + i
    f1.write(docs)
    f1.write("\n")
f1.close()
```

ob.index("documentsID.txt")

- Searching Module

```
import os
import sys
import math
import codecs
import string
print "\n\t\t\tእንኳን ወደ አማርኛ የመረጃ ማእከል በደህና መጡ!"
print "....................................................................................."
print
print "\n\t\t\tመረጃ ፍለጋ ሊዘገይ ስለሚችል እባኮን ትንሽ ይጠበቁ"
print "....................................................................................."
print
def relevancefeedback(UserQuery,vector,flag):
   while flag<3:
      L1,L2,L01,L02=[],[],[],[]
      N = int(len([fiLe for fiLe in os.listdir("./corpus") if os.path.isfile(os.path.join("./corpus", fiLe))]))
      querylist=[]
      userInput=UserQuery.split()
      for i in userInput:
         querylist.append(punctuationremove(suffix(preffix(i))))
      Relevancedoc=[]
      termination=int(raw_input("\n\nየቀረቡት መረጃዎች አጥጋቢ ናቸዉ? ከሆኑ (1) ይጫኑ ካልሆኑ (0) ይጫኑ:"))
      if termination ==1:
         print "እናመሰግናለን ደህናይሁኑ"
         sys.exit()
      else:
         cont=1
         dh=int(raw_input("\n\nምንያህሎቹ መረጃዎች አስፈላጊ ናቸዉ? :"))
         print "የዶኩመንትቶቹን መለያ ቁጥር እዚ ያስገቡ!"
         while cont <= dh:
            R=int(raw_input("===> " +str(cont)+ unicode(" ኛ :", 'utf-8')))
            Relevancedoc.append(R)
            Relevancedocument=sorted(Relevancedoc)
            cont=cont+1
         ch=int(raw_input("\n\nምንያህሎቹ መረጃዎች የማያስፈልጉ ናቸዉ? :"))
         contt=1
         Ireldocs=[]
         while contt <= ch:
            IR=int(raw_input("===> " +str(contt)+ unicode(" ኛ :", 'utf-8')))
            Ireldocs.append(IR)
            Irelevantdocument=sorted(Ireldocs)
            contt=contt+1
         NN=len(Irelevantdocument)
         check=sorted(Relevancedocument+Irelevantdocument)
         provector1=[]
         for n in range(len(querylist)):
            for j in range(1, N+1):
```

```
        for t in range(len(vector)):
            if j==vector[t][1]:
                if querylist[n]==vector[t][0]:
                    provector1.append((vector[t][0],j,1))
                    temp1=sorted(provector1)
                else:
                    pass
rel=[]

for g in range(len(Relevancedocument)):
    for k in range(len(temp1)):
        if Relevancedocument[g]==temp1[k][1]:
            rel.append((temp1[k][0],temp1[k][1],1,1))

cont=0
rel2=[]
for i in range(len(querylist)):
    cont=0
    for j in range(len(rel)):
        for k in range(len(Relevancedocument)):
            if querylist[i]==rel[j][0] and rel[j][1]==Relevancedocument[k]:
                if rel[j][3]==1 and rel[j][1]==k and rel[j][2]==1:
                cont=cont+1
    rel2.append((querylist[i],cont))
count=0
forweightt=[]
for i in range(len(querylist)):
    count=0
    for j in range(len(temp1)):
        if querylist[i]==temp1[j][0]:
            for k in range(len(check)):
                if check[k]==temp1[j][1]:
                    count=count+1
    forweightt.append((querylist[i],count))
weightt=[]
R=len(Relevancedocument)
B=R+NN
for i in range(len(forweightt)):
    mat1=float((rel2[i][1]+ 0.5))
    mat01=float((B)-(R))
    mat02=float(forweightt[i][1])
    mat03=(mat01)-(mat02)
    mat04=(mat1)+(mat03)
    mat2=float((rel2[i][1]) + 0.5)
    mat3=float((R-(rel2[i][1]))+0.5)
    mat4=float(((forweightt[i][1])- (rel2[i][1])) + 0.5)
    sim1=(mat04*mat2)
    sim2=(mat3*mat4)
    sim=float(sim1/sim2)
    simm=math.log((sim),10)
    weightt.append(simm)
countt=1
```

```
rankk=[]
countt=1
scoree=[]
scoree2=[]
for i in range(len(querylist)):
   rankk.append((querylist[i],countt))
   countt=countt+1
for i in range(len(querylist)):
   for j in range(len(provector1)):
      if querylist[i]==provector1[j][0]:
         for m in range(1,(N+1)):
            if provector1[j][1]==m:
               scoree.append((m,rankk[i][1]))
               scoree2=sorted(scoree)

            else:
               pass
addd=0
scoreee=[]

for k in range(len(weightt)):
   for j in range(1,(N+1)):
      for i in range(len(scoree2)):
         if scoree2[i][0]==j:
            addd=(addd)+(weightt[(scoree2[i][1])-1])
            scoreee.append((scoree2[i][0],addd))
      addd=0
   break
Ran=[]
if len(userInput)>0:
   if len(userInput)>=1:
      finalRess={}
      finalRess=dict(scoreee)
      Rankin=[]
      Rankingg=[]
      Rankingg=finalRess.keys()
      f11=open("documentsID.txt", 'r')
      f22=f11.readlines()
      DocLL=[]
      for i in f22:
         recordd=i.split()
         DocLL.append(recordd)
      DocNN=1
      DocL22=[]
      pathListt=[]
      for i in DocLL:
         DocL22.append((DocNN,i[0]))
         DocNN=DocNN+1
      docDictt=dict(DocL22)
      chekpointt=0
      for i in range(0,len(Rankingg)):
         if finalRess[Rankingg[i]] !=0:
```

```
            chekpointt=1
        for i in range(len(Rankingg)):
            for j in range(i+1,len(Rankingg)):
                if finalRess[Rankingg[i]]<finalRess[Rankingg[j]]:
                    rankk=Rankingg[i]
                    Rankingg[i]=Rankingg[j]
                    Rankingg[j]=rankk

        if chekpointt==1:
            print "\n\t\tባስገቡት መጠይቅ መሰረት የተገኙት መረጃዎች  እንደሚከተለው  ቀርበዋል! "
            print"                                    "
            print "ቅደም ተከተል==========================የመረጃዉ መለያ
ቁጥር============================የመረጃዎቹ መመሳሰል ልኬት"
            print
            for i in range(len(Rankingg)):
                if finalRess[Rankingg[i]]>=1.5:
                    pathListt.append(docDictt[Rankingg[i]])
                    print
"_____________________________________________________________________________________"
                    print "",i+1, ".....", docDictt[Rankingg[i]], ".....", finalRess[Rankingg[i]]
                    print
"_____________________________________________________________________________________"
                    f11=codecs.open(docDictt[Rankingg[i]],'r', encoding="utf-8")
                    print
                    print f11.read(50), "..."
                    print
                    f11.close()
            docs = './'
            ch=int(raw_input("\n\nስንት መረጃ ማየት ይፈልጋሉ? : "))
            count=1
            while count <= ch:
                docs = './'
                pathF = raw_input("\nየትኛውን ዶኪውመንት መመልከት ይፈልጋሉ? : ")
                if pathF in pathListt:
                    docs = docs + '/' + pathF
                    showfile = codecs.open(docs,'r', encoding="utf-8")
                    for line in showfile:
                        print line,
                    showfile.close()
                    pathF = ""
                count=count+1
            flag=flag+1
            relevancefeedback(UserQuery,vector,flag)
        else:
            print "ይቅርታ! ከጠየቁት መጠይቅ ጋር የሚመሳሰል መረጃ አልተገኘም።"
        print "ከዚ ዙር ቦሃላ ስይስተሙ መረጃን የመስጠት አቅሙ እየቀነሰ ስለሚሄድ እንደገና መጠይቆን አስተካክለዉ ቢመለሱ ይመከራል!!"
        loop=raw_input("\n\nመጠይቆን መቀጠል ይፈልጋሉ? ከፈለጉ 1 ቁጥርን ይጫኑ፤  መጠይቆን ከጨረሱ ሌላ የፊደል ቁልፉን ይጫኑ!
")
        if loop == '1':
            flag=0
            relevancefeedback(UserQuery,vector,flag)
        else:
```

```
                print "\n\t\tእናመሰግናለን!  ደህና ይሁኑ!!"
                sys.exit()
            else:
                print "ያስገቡት መጠይቅ ተፈላጊ የሆኑ መረጃዎችን ለማግኘት አያስችልም።"
                readIndexData()
        else:
            print "ምንም ዓይነት መጠይቅ አላስገቡም፤ እባክዎ መጠይቆን እንደገና ያስገቡት! "
            readIndexData()
def condition1(affix, tempp):
    if len(tempp)> 2:
        if affix.__eq__(u'\u12a8'):
            return 1
        elif affix.__eq__(u'\u1260'):
            return 1
        elif affix.__eq__(u'\u129E\u127D'):
            return 1
        elif affix.__eq__(u'\u1295'):
            return 1
        elif affix.__eq__(u'\u1275'):
            return 1
        elif affix.__eq__(u'\u1290\u1275'):
            return 1
        elif affix.__eq__(u'\u12CE\u127D'):
            return 1
        elif affix.__eq__(u'\u127D'):
            return 1
        elif affix.__eq__(u'\u127D\u1295'):
            return 1
        elif affix.__eq__(u'\u127D\u1201'):
            return 1
        elif affix.__eq__(u'\u1278\u12CD'):
            return 1
        elif affix.__eq__(u'\u1271'):
            return 1
        elif affix.__eq__(u'\u1279'):
            return 1
        elif affix.__eq__(u'\u12CA'):
            return 1
        elif affix.__eq__(u'\u12CA\u1275'):
            return 1
        elif affix.__eq__(u'\u12CD\u12EB\u1295'):
            return 1
        elif affix.__eq__(u'\u1277\u120D'):
            return 1
        elif affix.__eq__(u'\u121D'):
            return 1
        elif affix.__eq__(u'\u1275') or affix.__eq__(u'\u12A0'):
            return 1
        elif affix.__eq__(u'\u129B'):
            return 1
        elif affix.__eq__(u'\u1290\u1275') or affix.__eq__(u'\u12A5'):
            return 1
```

```
    elif affix.__eq__(u'\u121D') or affix.__eq__(u'\u12A5'):
        return 1
    elif affix.__eq__(u'\u1208\u1275'):
        return 1
  else:
    return 0

def condition2(affix,tempp):
  if len(tempp) > 2:
    if affix.__eq__(u'\u12A8'):
      if tempp.__eq__(u'\u12A8\u1265\u1275') or tempp.__eq__(u'\u12A8\u1208\u12A8\u1208') or
        tempp.__eq__(u'\u12A8\u1290\u12A8\u1290') or tempp.__eq__(u'\u12A8\u1228\u12A8\u1228') or
        tempp.__eq__(u'\u12A8\u1128\u12A8\u1218') or tempp.__eq__(u'\u12A8\u1230\u12A8\u1230') or
        tempp.__eq__(u'\u12A8\u1270\u12A8\u1270') or tempp.__eq__(u'\u12A8\u1270\u121B') or
        tempp.__eq__(u'\u12A8\u1228\u1295') or tempp.__eq__(u'\u12A8\u1230\u120D') or
        tempp.__eq__(u'\u12A8\u134D\u1270\u129B') or tempp.__eq__(u'\u12A8\u1265\u1275'):
         return tempp
      else:
         tempp = tempp[1: ]
         tempp = preffix(tempp);
         return tempp
    elif affix.__eq__(u'\u1260'):
      if tempp.__eq__(u'\u1260\u123D\u1273') or tempp.__eq__(u'\u1260\u1300\u1275') or
        tempp.__eq__(u'\u1260\u1228\u1260\u1228') or tempp.__eq__(u'\u1260\u1230\u1260\u1230'):
         return tempp
      else:
         tempp = tempp[1: ]
         tempp = preffix(tempp);
         return tempp
    elif affix.__eq__(u'\u129B'):
      if tempp.__eq__(u'\u1260\u123D\u1273'):
         return tempp
      else:
         tempp = tempp[ :-1]
         tempp = preffix(tempp);
         return tempp
    elif affix.__eq__(u'\u1295'):
      if tempp.__eq__(u'\u12D8\u1218\u1295') or tempp.__eq__(u'\u1261\u12F5\u1295') or
        tempp.__eq__(u'\u123D\u134B\u1295') or tempp.__eq__(u'\u12C8\u1308\u1295') or
        tempp.__eq__(u'\u1261\u12F5\u1295') or tempp.__eq__(u'\u123D\u134B\u1295') or
        tempp.__eq__(u'\u12C8\u1308\u1295') or tempp.__eq__(u'\u132D\u1241\u1295') or
        tempp.__eq__(u'\u1205\u133B\u1295') or tempp.__eq__(u'\u12A5\u12CD\u1295') or
        tempp.__eq__(u'\u12A5\u121D\u1295') or tempp.__eq__(u'\u12D8\u1348\u1295'):
         return tempp
      else:
         tempp = tempp[ :-1]
         tempp=sadisConverter(tempp)
         tempp = suffix(tempp)
         return tempp
    elif affix.__eq__(u'\u1208\u1275'):
      if tempp.__eq__(u'\u12A0\u1208\u1275') or tempp.__eq__(u'\u12A5\u1208\u1275') or
        tempp.__eq__(u'\u1201\u1208\u1275'):
```

```
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp);
        return tempp
elif affix.__eq__(u'\u1275'):
    if tempp.__eq__(u'\u12A0\u1263\u1275') or tempp.__eq__(u'\u12A0\u12EB\u1275') or
      tempp.__eq__(u'\u1205\u12ED\u12C8\u1275') or tempp.__eq__(u'\u12A0\u122B\u12CA\u1275') or
      tempp.__eq__(u'\u1230\u12A3\u1275') or tempp.__eq__(u'\u1218\u1265\u1275') or
      tempp.__eq__(u'\u1218\u1230\u1228\u1275') or tempp.__eq__(u'\u12C8\u1245\u1275') or
      tempp.__eq__(u'\u1325\u1228\u1275') or tempp.__eq__(u'\u1275\u122D\u134D') or
      tempp.__eq__(u'\u1201\u1208\u1275') or tempp.__eq__(u'\u1236\u1235\u1275') or
      tempp.__eq__(u'\u12A0\u122B\u1275') or tempp.__eq__(u'\u12A0\u121D\u1235\u1275') or
      tempp.__eq__(u'\u1235\u12F5\u1235\u1275') or tempp.__eq__(u'\u1230\u1263\u1275') or
      tempp.__eq__(u'\u1235\u121D\u1295\u1275') or tempp.__eq__(u'\u1275\u121D\u1205\u122D\u1275')
      or tempp.__eq__(u'\u1325\u1245\u121D\u1275') or tempp.__eq__(u'\u1233\u121D\u1295\u1275') or
      tempp.__eq__(u'\u12AD\u122B\u12CA\u1275'):
      return tempp
    else:
      tempp = tempp[ :-1]
      #change to sades
      tempp = suffix(tempp);
      return tempp
elif affix.__eq__(u'\u1290\u1275'):
    if tempp.__eq__(u'\u12A5\u12CD\u1290\u1275') or tempp.__eq__(u'\u12A5\u121D\u1290\u1275'):
      return tempp
    else:
      tempp = tempp[ :-1]
      #change to sades
      tempp = suffix(tempp);
      return tempp
elif affix.__eq__(u'\u12CE\u127D'):
    if tempp.__eq__(u'\u1230\u12CE\u127D'):
      str1 = tempp[-1: ]
      return tempp
    else:
      tempp = tempp[ :-2]
      #change to sades
      tempp = suffix(tempp)
      return tempp
elif affix.__eq__(u'\u129E\u127D'):
    if tempp.__eq__(u'\u12F3\u129E\u127D'):
      tempp = tempp[ :-1]
      ##change to sades
      return tempp
    else:
      tempp = tempp[ :-1]
      #change to sades
      tempp = suffix(tempp)
      return tempp
elif affix.__eq__(u'\u127D'):
```

```
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D\u1295'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u127D\u1201'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1278\u12CD'):
        tempp = tempp[ :-2]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1271'):
        ##change to sades
       tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1279'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u12CA'):
        tempp = tempp[ :-1]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u12CA\u1275'):
    if tempp.__eq__(u'\u12A0\u122B\u12CA\u1275'):
        return tempp
    else:
        tempp = tempp[ :-2]
        #change to sades
        tempp = suffix(tempp);
        return tempp
elif affix.__eq__(u'\u12CD\u12EB\u1295'):
        tempp = tempp[ :-3]
        ##change to sades
        tempp = suffix(tempp)
        return tempp
elif affix.__eq__(u'\u1277\u120D'):
    if tempp.__eq__(u'\u121E\u1277\u120D'):
        tempp = tempp[ :-1]
        ##change to sades
        return tempp
    else:
```

```
            tempp = tempp[ :-2]
            #change to sades
            tempp = suffix(tempp)
            return tempp
        elif affix.__eq__(u'\u121D'):
          if tempp.__eq__(u'\u1308\u12F3\u121D') or tempp.__eq__(u'\u1230\u120B\u121D') or
            tempp.__eq__(u'\u12A0\u1208\u121D'):
            return tempp
          else:
            tempp = tempp[ :-1]
            #change to sades
            tempp = suffix(tempp);
            return tempp
    else:
      return tempp
def punctuationremove(str):
    if str.endswith('\n'):
      str=str.rstrip('\n')
    if str.startswith(' '):
      str=str.lstrip(' ')
    p=codecs.open("Punctuation.txt",'r', encoding = 'utf-8')
    Punctuation = p.read()
    for pc in Punctuation:
      if str.endswith(pc):
        str=str.rstrip(pc)
      if str.startswith(pc):
        str=str.lstrip(pc)
    for j in Punctuation:
      for i in range(1,len(str)):
        if j in Punctuation and j in str:
          str=str.strip(j)
    return str
def normal(str):
    sim=codecs.open("similarWord.txt",'r', encoding = 'utf-8' )
    similarWord=sim.read()
    li1=[]
    for m in similarWord.split():
      li1.append(m)
    for i in range(1,len(li1),2):
      if str==li1[i] or str==li1[i+1]:
        str=li1[i+1]
    temp=codecs.open("similarChar.txt",'r', encoding = 'utf-8' )
    similarChar=temp.read()
    li=[]
    for j in similarChar.split():
      li.append(j)
    for i in range(1,len(li),2):
      str=str.replace(li[i],li[i+1])
    return str
def preffix(word):
    if len(word)<3:
      return word
```

```
    else:
        while len(word)>=3:
            tempp=word[ :3]
            p=codecs.open("prefix.txt",'r', encoding = 'utf-8' )
            pr = p.read()
            y=pr.split()
            p.close()
            if y.__contains__(tempp):
                if (condition1(tempp,word)):
                    word=condition2(tempp,word)
                    return word
                else:
                    word=word[3: ]
                    preffix(word)
            else:
                tempp=word[ :2]
                if  y.__contains__(tempp):
                    if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                    else:
                        word=word[2: ]
                        preffix(word)
                else:
                    tempp=word[ :1]
                    if  y.__contains__(tempp):
                        if (condition1(tempp,word)):
                            word=condition2(tempp,word)
                            return word
                        else:
                            word=word[1: ]
                            preffix(word)

                    else:
                        return word

    return word
def suffix(word):
    if len(word)<3:
        return word
    else:
        while len(word)>=3:
            tempp=word[len(word)-3: ]
            s=codecs.open("sufix.txt",'r', encoding = 'utf-8' )
            su = s.read()
            x=su.split()
            s.close()
            if x.__contains__(tempp):
                if (condition1(tempp,word)):
                    word=condition2(tempp,word)
                    return word
                else:
```

```
                word=word[ :len(word)-3]
                suffix(word)
            else:
                tempp=word[len(word)-2: ]
                if  x.__contains__(tempp):
                    if (condition1(tempp,word)):
                        word=condition2(tempp,word)
                        return word
                    else:
                        word=word[ :len(word)-2]
                        suffix(word)
                else:
                    tempp=word[len(word)-1: ]
                    if  x.__contains__(tempp):
                        if (condition1(tempp,word)):
                            word=condition2(tempp,word)
                            return word
                        else:
                            word=word[ :len(word)-1]
                            suffix(word)

                    else:
                        return word


    return word
def readIndexData():
    flag=0
    L1,L2,L01,L02=[],[],[],[]
    N = int(len([fiLe for fiLe in os.listdir("./corpus") if os.path.isfile(os.path.join("./corpus", fiLe))]))
    a=codecs.open("vocabularyFile.txt", 'r', encoding = 'utf-8')
    b=a.readlines()
    for j in b:
        c=j.split()
        f=c[0],int(c[1])
        L01.append(f)
    for i in L01:
        L1.append((i[0],i[1]))
    a.close()
    n=codecs.open("postingFile.txt", 'r', encoding = 'utf-8')
    m=n.readlines()
    for i in m:
        s=i.split()
        f2=int(s[0]),int(s[1])
        L02.append(f2)
    for i in L02:
        L2.append((i[0],i[1]))
    n.close()
    L3,L6,vector=[],[],[]
    d,h,g,a,m,c=0,0,0,0,0,0
    while c < len(L1):
        L=[]
```

```
        L30=[]
        for h in range(L1[d][1]):
            L.append((L1[m][0], L2[a][0]))
            a = a + 1
        m = m + 1
        d = d + 1
        c=c+1
        vector = vector + L
querylist=[]
provector=[]
UserQuery=raw_input("እባኮን የሚፈልጉትን ፋይል ለማግኘት መጠይቆን ያስገቡ!:- ")
print "....."
userInput=UserQuery.split()
for i in userInput:
    querylist.append(punctuationremove(suffix(preffix(i))))
for n in range(len(querylist)):
    for j in range(1, N+1):
        for t in range(len(vector)):
            if j==vector[t][1]:
                if querylist[n]==vector[t][0]:
                    provector.append((vector[t][0],j,1))
                else:
                    pass
count=0
forweight=[]
for i in range(len(querylist)):
    count=0
    for j in range(len(provector)):
        if querylist[i]==provector[j][0]:
            count=count+1
    forweight.append((querylist[i],count))
weight=[]
for i in range(len(forweight)):
    sim1 = float(((N)-(forweight[i][1])) + 0.5)
    sim2=float((forweight[i][1]) + 0.5)
    sim=float(sim1/sim2)
    simm=math.log((sim),10)
    weight.append(simm)
count=1
rank=[]
count=1
score=[]
score2=[]
for i in range(len(querylist)):
    rank.append((querylist[i],count))
    count=count+1
for i in range(len(querylist)):
    for j in range(len(provector)):
        if querylist[i]==provector[j][0]:
            for m in range(1,(N+1)):
                if provector[j][1]==m:
                    score.append((m,rank[i][1]))
```

```
            score2=sorted(score)

        else:
            pass
add=0
scoree=[]

for k in range(len(weight)):
    for j in range(1,(N+1)):
        for i in range(len(score2)):
            if score2[i][0]==j:
                add=(add)+(weight[(score2[i][1])-1])
                scoree.append((score2[i][0],add))
        add=0
    break
if len(userInput)>0 and len(userInput)<10:
    if len(userInput)>=1 and len(userInput)<10:
        finalRes={}
        finalRes=dict(scoree)
        Ranking=[]
        Ranking=finalRes.keys()
        f1=open("documentsID.txt", 'r')
        f2=f1.readlines()
        DocL=[]
        for i in f2:
            record=i.split()
            DocL.append(record)
        DocN=1
        DocL2=[]
        pathList=[]
        for i in DocL:
            DocL2.append((DocN,i[0]))
            DocN=DocN+1
        docDict=dict(DocL2)
        chekpoint=0
        for i in range(0,len(Ranking)):
            if finalRes[Ranking[i]] !=0:
                chekpoint=1
        for i in range(len(Ranking)):
            for j in range(i+1,len(Ranking)):
                if finalRes[Ranking[i]]<finalRes[Ranking[j]]:
                    rank=Ranking[i]
                    Ranking[i]=Ranking[j]
                    Ranking[j]=rank
        if chekpoint==1:
            print "\n\t\tባስገቡት መጠይቅ መሰረት የተገኙት መረጃዎች እንደሚከተለው ቀርበዋል! "
            print"                                    "
            print "ቅደም ተከተል==========================የመረጃዉ መለያ
          ቁጥር============================የመረጃዎቹ መመሳሰል ልኬት"
            print
            for i in range(len(Ranking)):
                if finalRes[Ranking[i]]>1:
```

```
                pathList.append(docDict[Ranking[i]])
                print "_____"
                print "",i+1, ".....", docDict[Ranking[i]], ".....", finalRes[Ranking[i]]
                print "_____"
                f=codecs.open(docDict[Ranking[i]],'r', encoding="utf-8")
                print
                print f.read(50), "..."
                print
                f.close()
            docs = './'
            ch=int(raw_input("\n\nስንት መረጃ ማየት ይፈልጋሉ? : "))
            count=1
            while count <= ch:
               docs = './'
               pathF = raw_input("\nየትኛውን ዶኪውመንት መመልከት ይፈልጋሉ? : ")
               if pathF in pathList:
                  docs = docs + '/' + pathF
                  showfile = codecs.open(docs,'r', encoding="utf-8")
                  for line in showfile:
                     print line,
                  showfile.close()
                  pathF = ""
               count=count+1
            relevancefeedback(UserQuery,vector,flag)
         else:
            print "ይቅርታ! ከጠየቁት መጠይቅ ጋር የሚመሳሰል መረጃ አልተገኘም።"
         loop=raw_input("\n\nማሰተካከል ይፈልጋሉ? ከፈለጉ 1 ቁጥርን ይጫኑ፤  መጠይቆን ከጨረሱ ሌላ የፊደል ቁልፉን ይጫኑ! ")
         if loop == '1':
            readIndexData()
         else:
            print "\n\t\tእናመሰግናለን!  ደህና ይሁኑ!!"
            sys.exit()

      else:
         print "ያስገቡት መጠይቅ ተፈላጊ የሆኑ መረጃዎችን ለማግኘት አያስችልም።"
         readIndexData()
   else:
      print "ምንም ዓይነት መጠይቅ አላስገቡም፤ ወይም መጠይቆ ከሚፈቀደዉ መጠን በላይ ነዉ! እባክዎ መጠይቆን እንደገና ያስገቡት! "
      readIndexData()
readIndexData()
```