Addis Ababa University
College of Natural Sciences
School of Information Science

Designing Amharic Definitive Question Answering

By
WONDWOSSEN TESHOME

June, 2013

Addis Ababa University
College of Natural Sciences
School of Information Science

Designing Amharic Definitive Question Answering

WONDWOSSEN TESHOME

A Thesis Submitted to the College of Natural Sciences of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Science

June, 2013

Addis Ababa University
College of Natural Sciences
School of Information Science

Designing Amharic Definitive Question Answering

WONDWOSSEN TESHOME

Name and signature of members of the examining Board

Name	**Title**	**Signature**	**Date**
Henock Lulseged (M.Sc)	Chairperson	_____	_____
Martha Yifiru (Ph.D)	Advisor	_____	_____
Million Meshesha (Ph.D)	Examiner	_____	_____

Dedication

to

My Mother

Acknowledgment

It would not have been possible to write this master's thesis without the help and support of the kind people around me. First and foremost, I would like to take this opportunity to express my deepest gratitude toward my supervisor, Martha Yifiru (Ph.D), for her commitment, patience and encouragement throughout the entire process of this thesis.

Extraordinary thanks go to Million Meshesha (Ph.D) who taught me undergraduate and postgraduate courses with exceptional excellence and introduced me to the fascinating field of question answering.

I would like to extend my special thanks to my mother who always gives me her heartfelt love and encouragement. A bunch of thanks go to my sister, Mulu Teshome.

I must sincerely thank to my friends: Fantahun Aberra, Muse Belew, Tesfamichael Habte, Getaneh Kassa, Michael Mengistu, Gedion Assefa, Samuel Teshome, and Biniam Asnake who are always on my side. I am lucky to have you all.

I would like to thank Tessema Mindaye, Seid Yimam, Henock Sahlu, and Yalemfkir Girma for providing me resources and supporting ideas.

Last but not least, many thanks to all the staff of School of Information Science, A.A.U.

Abstract

The amount of available information is becoming very huge, especially with the Web proliferation. The problem faced by the user is not the lack of documents or information but is the lack of time to find a short and precise answer among the variety of available documents. Search engines offer a lot of links toward web pages, but are not able to provide an exact answer; instead return ranked documents based on relevance measure with the posed query from users. Thus, a new need is emerged: the possibility of obtaining a brief and concise answer. Providing a brief and concise answer is the main goal of Question Answering systems.

Though there are studies towards developing question – answering system for factoid questions, there is no research conducted to develop a definition question answering system for Amharic and we couldn't compare our result with any other efforts in the topic. In this study, an attempt is made to design Amharic question answering for definitive questions.

Definition QA systems in other languages have been extensively researched and have shown reasonable outcomes.

The proposed Question Answering approach in this study deals with Amharic definition question by applying surface text pattern method. This method considers two main steps. First, it applies a pattern to discover a set of definition-related text patterns from the Amharic legal corpus. Then, using these patterns, it extracts a collection of concept-description pairs from a target document file, and applies the definition extraction to return answer to a given question.

The research achieved nugget precision of 85.6 %, nugget recall of 73% and F-measure of 78.8%. Usage of surface patterns is effective to answer Amharic definition questions. Definiendum extraction from users question and extracting concept-descriptions from corpora are the major challenges of this study. Further sequence mining algorithm can be experimented to extract concept-description relationship from the corpus.

Table of Contents

List of Tables

List of Figures

List of Equations

List of Algorithms

List of Sample Codes

List of Acronyms

HP	Hard Pattern
IR	Information retrieval
MUC	Message Understanding Conference
NLP	Natural Language Processing
NP	Nugget Precision
NR	Nugget Recall
QA	Question Answering
SP	Soft Pattern
TREC	Text REtrieval Conference

CHAPTER ONE

INTRODUCTION

"The field of Question Answering has grown out of dissatisfaction with Information Retrieval being merely document retrieval"

JOHN M. PRAGER

1.1 Background

Document retrieval systems have become part of our daily lives, mostly in using Internet search engines and tools. Although document retrieval systems do a great job in finding relevant documents, given a set of keywords, there are circumstances where we want more specific information need. The conventional Information Retrieval systems do not really perform the task of information retrieval; they in fact aim at only document retrieval. A survey showed 85% of internet users using search engines and search services attempt to find specific information (Liu, 2007). The survey also indicated that users are not satisfied with the performance of the current search services due to majority of search engines developed well are only able to return ranked lists of related documents, but they don't deliver exact answers, so users have to extract the demanded answer from the large volumes of information (Liu, 2007). The virtue of question answering systems is that they allow the user to state his or her information need in a more specific and as a natural language question, and that they do not return full documents which have to be skimmed by the user to determine whether they contain an answer, but short text extract, or phrases or even precise answers.

Textual question answering systems date back to the 1960s ORACLE (Phillips, 1960), and PROTOSYNTHEX (Simmons et.al, 1963). Until the early 1990s, there were few research efforts in the area. In recent years however, question answering witnesses a true renaissance. The re-emerging interest in textual question answering is largely due to the Text REtrieval Conference (TREC) initiative, which has featured a textual question answering track since 1999 (Voorhees, 2001). At TREC, participating groups evaluate and compare their question answering systems with respect to some standard set of questions. This allows for an objective comparison of

question answering techniques and the rapid interchange of ideas to further the research in that area.

Question-Answering subject has been focused for a few decades in different forms, and currently still is a challenging topic that is actively researched. Recent successes have been reported in a series of Question-Answering evaluations that started from 1999 as part of the TREC and also have been encourage by the Message Understanding Conference (MUC) (Hirschman, 2001). At present, the most outstanding systems are able to answer more than two thirds of factual questions in that evaluation (Hirschman, 2001).

Finding textual answers to open-domain questions (Hirschman, 2001) in large text collections, such as AQUAINT[1], is a challenging problem. Unlike Information Retrieval (IR) which aims at providing documents satisfying users information needs expressed in the form of a query, Question Answering (QA) aims at providing users with short text units that answer specific well formed natural language questions. There are an infinite number of questions users may ask a QA system. There are defined evaluation methods at TREC for the following three types question answer (Saggion, 2004):

- Factoid questions which require a single fact as answer. These include question such as "Who is the President of Ethiopia?" and "What lays blue eggs?"
- List questions which require multiple facts to be returned in answer to a question. Examples are "Name 22 cities that have a subway system" and "List 16 companies that manufacture tractors"
- Definition questions, such as "What is aspirin?", which require textual answers that cover essential (e.g., aspirin is a drug) as well as non-essential (e.g., aspirin is a blood thinner) descriptions of the definiendum (i.e., the term to be defined).

There are a number of QA systems that exist for public use in different languages including English, Chinese, Arabic, and so on (Figueroa, 2010). But there is no a single public Amharic question answer system, though electronic Amharic documents are increasing.

[1] AQUAINT consists of over 1 million texts from the New York Times, the AP newswire, and the English portion of the Xinhua newswire totaling approximately 3.2 gigabytes of data.

Designing of QA is a challenging task as its solutions draw from related fields such as Information retrieval (IR), information extraction (IE), and Natural Language Processing (NLP).

1.2 Statement of the Problem

The number of Amharic documents on the Web is increasing as many newspaper publishers started providing their services electronically. People were relying on IR systems to satisfy their information needs but it has been criticized for lack of delivering "readymade" information to the user so that Question Answering systems emerge as the best solution to get the required information to the user with the help of information extraction techniques.

Attempts have been made to develop QA system by Nico (2005) for English language, Haque (2010) for Bangla language, Huang (2004) for Chinese language, and Triqui et.al (2008) for Arabic language. The aforementioned QA systems cannot be used directly for Amharic language because of the fact that question analysis, question focus and answer extraction components are dependent to the language.

Seid Yimam (2009) tried to develop the first prototype Amharic QA for Factoid Questions. Seid recommended an extend research to work on other question types such as list, definition, and other non-factoid questions as it would be beneficial for wider applications where only a piece of information is not sought.

This study is; therefore, aim to explore on **definition questions** for Amharic Question Answering**.** This is possible because of the expressive power of natural language to represent the content. The user provides the query in the form of natural language text and users query are compared with the documents to extract the required definition.

To fill the above mentioned gap on Amharic question answering tool, this research address the following research questions:

- What are the language dependent components of question answering?
- How concept – description relationships are extracted from the Amharic document based on the available information retrieval techniques?

- Which question answer technique is effective for designing definitional Amharic QA system?
- To what degree the performance of Amharic QA for definition system is achieved?

1.3 Objective of the study

1.3.1 General objective

The general objective of this research is to discover a technique to answer Amharic definition questions

1.3.2 Specific Objectives

In order to achieve the general objective of this study, the following specific objectives are formulated:

- To review previously conducted literatures on question answering for conceptual understanding of principles, theories, approaches and algorithms.
- To study the general structure of Amharic statements related to definition question types.
- To prepare Amharic legal corpus and queries that is used to measure the performance of the prototype system.
- To design a general architecture for Amharic definition question Answering.
- To design algorithm suitable to identify definiendum from user natural language question
- To develop a prototype and evaluate the performance of the system.

1.4 Scope and Limitation of the Study

Naturally, Question Answering is a very complex and rigorous task which needs understanding of natural language processing techniques. Full-fledged QA systems require a number of natural language processing tools. There are no NLP tools for Amharic language that are publicly available for integration. Having these limitations in mind, the scope of this study is limited to exploring Amharic definition questions of closed-domain definition questions types, specifically

from Amharic law documents and training materials related to law are considered due to the fact that law documents includes definitions and there is no standard Amharic corpus to experiment on open-domain definitions. In this study due emphasis is given for the two areas of QA: question processing and definition extraction.

The limitation of this study are the question answering is on closed domain and there is no a standard corpus to measure the system performance. Categorization of answer as vital and non-vital nugget requires human judgment.

1.5 Methodology of the Study

The term methodology is derived from a Greek word, denoting the practice of analyzing different methods, implying a set or system of methods, principles and rules for regulating a given discipline (Berndtsson, 2008). As explained in Kothari (2004) research methodology is a way to systematically answer the research problem. It may be understood as a science of studying how research is done scientifically. Therefore, the following methods, techniques and tools have been used with the aim of achieving the general objective of this research.

1.5.1 Literature Review

Extensive literatures from journal articles, books, conference papers and the Internet have been reviewed for understanding concepts related to question answering with a due emphasis to definition question answering. Further review was also conducted to be acquainted with the previous research on Amharic question answering for factoid question. Then, the best techniques and tools were analyzed, selected, adopted and/or modified with the intention of developing a system. As the system is developed for definitive Amharic documents, characteristics and challenges of the Amharic language was also studied.

1.5.2 Dataset Collection

To evaluate the performance of the proposed system, Amharic corpora related to law are collected from the Web and the concerned judiciary organ. In studying and analyzing Question and Answer patterns, besides studying Amharic grammar books, from the legal documents

definitive questions are by different individuals to have better coverage of question particles. In addition, the questionnaires are used to test the performance of the prototype.

1.5.3 Implementation Tool

To develop the prototype, Python programming language is used as it is an ideal language to process text that supports Unicode and the researcher is familiar with the language. The other reason we selected Python is, it has rich implementations of the widely used NLP algorithms and Python program is simple to develop patterns wherever necessary.

Python is a powerful language and has the right combination of performance and features that make writing programs in Python both fun and easy (Swaroop, 2012).

1.5.4 Testing Procedures

Generally speaking, there is no single metric that supplies a definitive and complete view of the performance of definition QA systems (Figueroa, 2010) due to the fact that different systems or components are designed to meet distinct requirements, stressing different aspects of their output. The most broadly used metrics: recall, precision and F(β)-Score are used as evaluation metrics of the system.

F**(**β**)-**Score has been used regularly for assessing definition QA systems in the TREC track since 2003 (Voorhees, 2003). F**(**β**)-**Score measurement balances the precision and recall of a system by making a judgment about its output with respect to a manually generated ground truth.

There is no limit to the number of nuggets (important concept) per definiendum, and each nugget is associated with a category. In this assessment, the ground truth gives a hierarchy of nuggets, which consists of “vital" nuggets (must be in the description of the concept) and “non-vital" nuggets (not necessary). These labels are assigned by the legal assessor in agreement with the requirements and the objectives of each particular definition QA system.

TREC evaluation standard is adopted and used to evaluate the performance of the system towards correctness in identifying the correct target word and answers. The evaluation was done mainly on the effectiveness so as to see whether the system provides correct answer

based on pattern based answer selection techniques. The evaluation is done on sample Amharic legal corpuses; the questions were constructed by volunteer individuals who are not expertise of law but the assessor who identifies vital and non-vital nuggets is a legal expert.

The assessment takes into consideration the number of non-whitespace characters in the entire answer string, number of vital nuggets retrieved, and number of non-vital nuggets retrieved. The length-based measure used a system an allowance of 100 (non-white-space) characters for each correct concept it retrieves. The precision score is set to one if the response is no longer than this allowance. If the response is longer than the allowance, the precision score is downgraded using the function precision is equal to (Figueroa, 2010):$1 - \frac{length-allowance}{length}$.

The formula used to calculate the F-measure in TREC competition is given in Equation 1.4. The β value (arbitrary) of indicates that recall is considered β times more important than precision, an arbitrary value set for the purposes of the evaluation.

Evaluation of *DefAmharicQA* is mainly for accuracy and recall of answers. Precision for definition QA is calculated as the number of correct concepts retrieved to the total number of concepts retrieved, but it is challenging since the correct value for the denominator is unknown. A trial evaluation at TREC showed that assessors found enumerating *all* concepts represented in a response to be so difficult as to be impractical. Therefore, we can use length as a (crude) approximation to precision. The length-based measure for definition QA uses about an allowance of 100 - 110 (non-white-space) characters for each correct concept it retrieves. The precision score is set to one if the response is no longer than this allowance. If the response is longer than the allowance, the precision score is downgraded. The function precision can be equated as:

$$NP = 1 - \frac{length - allowance}{length};$$

Equation 1.1: Nugget Precision

if the length is greater than the allowance otherwise NP=1. Where length is the number of total non-whitespace characters in the definition, allowance is the number of non-whitespaces

that ranges between 100 and 110. The system is penalized for not retrieving very important (vital) concepts, and penalized for retrieving items that are not on the assessor's answer list at all, but neither penalized nor rewarded for retrieving a non-vital concept.

Note: allowance = 100 * (number of vital + non-vital nuggets returned)

Equation 1.2: Allowance formula

To measure recall is a straightforward; it is the ratio of the number of correct concepts retrieved to the number of concepts in the assessor's list.

(i.e. $NR = \frac{Number\ of\ vital\ nuggets\ returned}{Number\ of\ vital\ nuggets}$)

Equation 1.3 : Nugget Recall

The final score for a response was computed using the F-measure, a function of both recall (R) and precision (P). The general version of the F-measure is:

$$F = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R}$$

Equation 1.4 : Nugget F-measure

Where β value (arbitrary) indicates that recall is considered β times more important than precision, an arbitrary value set for the purposes of the evaluation.

If β = 1, then equal emphasis is given for both precision and recall; if the value of β *is* greater than 1, implies more weight is given to recall than precision and if the value of β is less than 1 the system is more of precision oriented than recall.

The performance of *DefAmharicQA* is evaluated towards correct answers. The evaluation is mainly on the effectiveness of the system and we have evaluated the system towards correct answer based on pattern based answer selection techniques. The evaluation is done on sample Amharic legal corpuses; the questions are constructed by volunteer individuals who are not expertise of law but the assessor who prepared vital and non-vital nuggets is a legal expert.

Twenty questions are constructed to evaluate the overall performance of the system. The assessor created, for each question, a list of acceptable information nuggets (piece of text) for the twenty questions. Some nuggets are considered vital (i.e., a piece of information that should be part of the definition) while others are considered not vital (non-essential). The assessor takes each system response and marks all vital and non-vital nuggets contained in it. A score for each question consists of nugget-recall (NR) and nugget-precision (NP). As the classification of vital and non-vital nuggets is accomplished manually the assessor decided the number questions.

1.6 Significance of the Study

When users want to find out about certain information from a large collection of documents, the most widely used tools are the search engines. They pose a query to a search engine system, and the search engine returns a number of pages related to the query. Generally, the documents returned are ranked based on keywords matching instead of the relevance to the query. The users have to read a lot of pages to organize the information they wanted by themselves. The above procedure is time consuming, and the information acquired is not focused.

The number of digital documents that are written in Amharic language and Ethiopic script is increasing. However, there is no tool that can be used to search this increasing number of Amharic documents on the Web. And a number of researches have been conducted to develop local search engine. Cognizant of this fact, the techniques used in this system can be applied and integrated with local search engine for retrieving relevant definitions information of users.

The major significance of the system is more convenient to inexperienced users with a flexible access to information, allowing them writing a query in natural language that ease communication and obtaining not a set of documents that contain the answer, but the concise answer as opposed to information retrieval that list ranked documents.

Moreover, it can be used by other researchers and component of other systems such as machine Translation.

1.7 Organization of the Thesis

This thesis is organized into six chapters. The first chapter discusses the background of the study and statement of the problem. It also presents general and specific objectives of the study, methodology of the study, scope and limitation of the research and significance of the study.

Chapter two address literature reviews on question answering and essential elements of QA. Moreover, brief review techniques related to QA, general framework of QA, types of QA, related global and local researches are discussed.

Chapter three describes language specific issues of the Amharic language, Amharic punctuation marks, Amharic question construction and challenges in Amharic QA.

In chapter four, the proposed architecture of Amharic definition question answer system is discussed.

Chapter five briefly explains the detail implementation and the performance measure of the system are discussed. The core components of the system, the functional operation module and the specific sub-component of each module are briefly discussed in this Chapter. Moreover, in the aforementioned chapter the evaluation measures that are used for measuring the performance of the system are presented.

Finally, based on the findings of the study, conclusion and recommendations of the research are stated in chapter six.

CHAPTER TWO

LITERATURE REVIEW

Today the Web is the largest resource of knowledge and, therefore, sometimes this makes it difficult to find precise information. Current search engines can only return ranked snippets containing the effective answers to a query user. But, they cannot return exact answers. Question answering systems present the solution to obtain effective and exact answers to a user question asked in natural language instead of keywords query.

2.1 Overview of Question Answering

The discipline of QA is concerned with the retrieval of accurate answers to natural language questions from a corpus (Schlaefer, 2007). Often, in QA systems, the Web is utilized as a large and redundant knowledge source. Local text corpora are used in expert systems for restricted domains and in evaluations to ensure the comparability of the results. QA systems differ from document retrieval systems in that the question may be posed in natural language and the answers are precise and to the point.

Question Answering is a dream since the first investigation in to artificial intelligence has been to converse with a machine in natural language to get answer to questions (Maybury, 2004). Simply question answering, aims to provide natural language response to natural language queries. More comprehensively:

> *Question Answering (QA) is an interactive human computer process that encompasses understanding the user information need, typically expressed in a natural language query; retrieving relevant document, data, or knowledge from selected sources; extracting, qualifying and prioritizing available answers from these resources; and presenting and explaining responses in an effective manner* (Maybury, 2004).

As is evident from the above definition, QA relies upon many component technologies, including but not limited to natural language processing, information retrieval, and explanation generation. QA promises an important new way of information access for all, a natural step beyond the key word query and document retrieval characteristics of today's web search.

2.1.1 Disciplines Related to Question Answering

Question answering is challenging, in part, because it lies at the intersection of several scientific fields including natural language processing (understanding and generating natural language text), information retrieval (query formulation, document analysis, relevancy feedback), and human computer interaction (interface design, user modeling). Figure 2.1 illustrates the relation of the three areas and their intersection in systems that support question answering. Several additional scientific discipline that may support question answering are not shown, for example knowledge representation and reasoning for question and answer analysis, or recommended technology to find preferred answers, or multimodal information processing to help extract answers from audio or video sources, or information visualization for results display (Maybury, 2004).

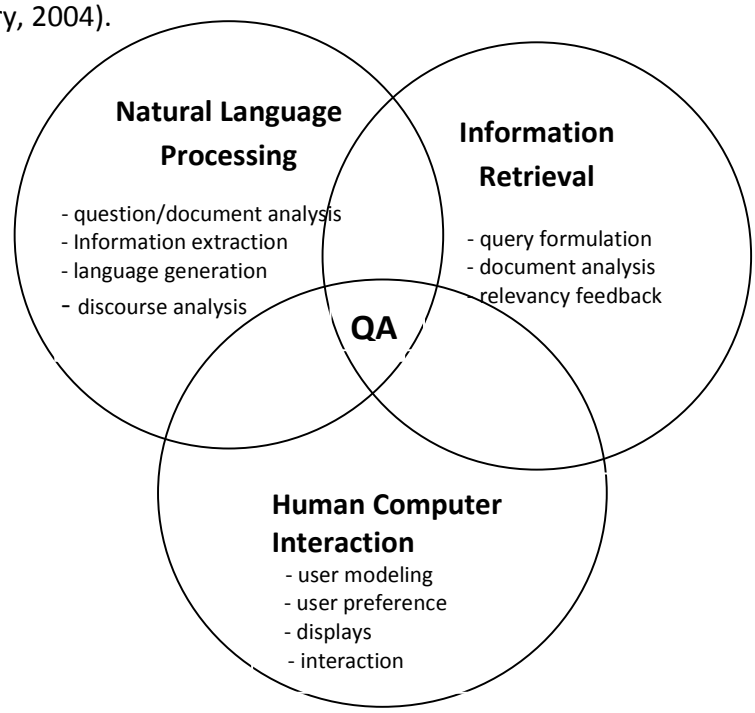

Figure 2.1 Relationship of Question Answering to Other Disciplines

For constructing an effective QA system, which allows a user to ask a question in natural language and receive an answer quickly and succinctly, the more advanced technologies are required that generally combine related techniques of more established tasks; Information Retrieval (IR), Information Extraction (IE) and Natural Language Processing (LNP) (Aunimo, 2007; Hu, 2006).

2.1.1.1 Information Retrieval

Information Retrieval (IR), taken to be the retrieval of relevant documents in response to a user query, has been an active research area since the mid-1950s. IR systems return documents, not answers, and users are left to extract answers from the documents themselves. The traditional information retrieval can be considered to be similar to a web search engine, such as Google, although the original IR engines predate the existence of the web, searching locally-stored collections of documents.

An IR engine takes as input a query expressed in the engine's query syntax, which can be as simple as a "bag of words" or as complicated as that of a system such as INQUERY, which allows the user to query on phrases, sets of synonyms and keywords in strict order over windows of text (James et.al,2003). As output, an IR engine provides a ranked list of documents drawn from the collection it has previously indexed that are relevant to the user's query, for some definition of relevance.

IR is mainly relevant to question answering for two reasons (Hirschman, 2001). First, IR techniques have been extended to return not just relevant documents, but relevant passages within documents. The size of these passages can be steadily reduced so that in the limiting case, what is extracted is, effectively, just the answer to a question. Thus, question answering can be thought of as passage retrieval in the limit. Second, the IR community has, over the years, developed an extremely thorough methodology for evaluation, the most well-known current exemplars of which are the annual TREC, run by the US National Institute of Standards and Technology. It is from this methodology and community that the recent question

answering evaluation developed, which in turn has stimulated much of the current interest in question answering.

2.1.1.2 Information Extraction

The other strand of research that has fed into the current TREC question answering track is Information Extraction (IE), as it was initially known, message understanding. IE can be defined as the automatic extraction of structured information from unstructured or semi-structured documents by filling predefined templates (Andrew, 2005). In the current context, IE templates can be viewed as expressing a question and a filled template as containing an answer. Thus, IE may be viewed as a limited form of question answering in which the questions are static and the data from which the questions are to be answered are an arbitrarily large dynamic collection of texts.

IE is a new technology enabling relevant content to be extracted from textual information available in electronic format. IE essentially builds on natural language processing and computational linguistics, but it is also closely related to the well established area of information retrieval and it is, as a method of searching for information in some ways similar to Question Answering (Hu, 2006). The IE community devised its own evaluation exercise, the Message Understanding Conferences (MUC), which runs between 1987 and 1998, the last MUC-7 was held in 1998. The termination of the MUC exercises, coupled with the desire to continue to push language understanding technology in novel directions via open evaluation exercises, were enabling conditions for the current TREC question answering evaluation. Generally, the process of IE has three major parts (Figueroa, 2010). First, the system **extracts individual "facts"** from the text of a document through local text analysis. Second, it **integrates these facts**, producing larger facts or new facts (through Inference). As a final step after the facts are integrated, the pertinent facts are translated into the required output format.

2.1.1.3 Natural Language Processing

No matter what kind of knowledge and purpose a **QA** system corpora contains, the common issue a designer face is how users communicate with it efficiently (Jianan, 2006). Since users

normally tend to be human agents, the most optimized choice of getting access to the corpus is querying with natural language, which indicates the direction of system characteristic. So to answer a question, a system must be capable of analyzing the natural language question, perhaps in the context of some ongoing interaction; it has to find one or more answers by consulting the corpus. A series of problem produced in the implementation because of the special features of human language and it widely recurs to NLP techniques (Harabagiu, 2008).

While user intent is clearly relevant to question answering, determining that intent is not a straightforward task. The ambiguity of user queries is always an issue. In the end, a system can only guess at what the user meant to ask. How a system interprets a question depends on multiple factors, including the kind of linguistic or statistical processing employed, the structure of the information sources (databases, free text, etc.), and the domain of knowledge. Past approaches to question answering have dealt with these considerations and others in order to develop functional **Natural Language Question Answer (NLQA)** systems (Stacey, 2001).

2.2 Definition Question Answering

There are an infinite number of questions users may ask and the questions can be categorized as **Factoid**, **List**, **Definition**, **Reason**, **Purpose**, and **Procedure** questions. We give emphasis to definition question answering since this research deals with such kind of questions.

Factoid questions require a single fact as answer (Saggion, 2004). These include question such as "Who is the President of Ethiopia?" and "What is the capital city of Addis Ababa?"

List questions were originally a simple extension of the factoid questions (Andrew, 2005). Instead of requiring a single exact answer list questions simply required a fixed sized, unordered set of answers, for example "Name 5 schools of Addis Ababa University".

Definition Question Answering

Certain types of question cannot be answered by a single exact answer. For example questions such as *"*What is data*?"* and *"Who* is Abebech Gobena?*"* do not have a single short answer. Answers to such questions should resemble entries in encyclopedias or biographical dictionaries telling the user all the important information about the subject of the question.

This type of question is usually referred to as a **definition question**. While definition questions can be natural language questions there are only a few ways in which they can phrased and most are of the form *"Who/What is/was X?"*. In these questions *X* is the ***definiendum*** often referred to as the 'target' of the question and is the thing, be it person, organization, object or event, which the user is asking the system to define.

Embedding the target in a question seems artificial. Users of electronic encyclopedias' would not expect to have to enter a full question to find an article but would usually enter just the name of the thing they were interested in. While more natural to the user it actually complicates the problem for researchers designing definitional QA systems. With full questions it is easier to discern if the target is a person or some other entity allowing a definition for a person to be constructed differently to those for an organization or generic name, such as aspirin. By taking only the target as input there is no obvious sign of what type of thing the target is (i.e. no words like *who*) and as such all targets are likely to be treated the same.

As currently the only accepted test sets for definitional question answering are those used in the TREC evaluations (from 2003 onwards) we assume the same scenario for guiding the production of a definition answers.

The currently accepted evaluation methodology for definitional questions focuses on the inclusion in the definition of given **nuggets** of information. A nugget is a single atomic piece of information about the current target (Andrew, 2005). For example, when asked to define *"ኃይሌ ገብረሰላሴ" (Haile Gebresilasie)* systems should, according to the TREC supplied answer key, include in a definition the following facts: *የሩጫ ጀግና (*Hero Athlet*)፣ የረጅም ርቀት አትሌት (*long distance runner*)፣ እና ባለ ሃብት (*investor*)*. Notice that this does not include a lot of facts that is expected in a full definition of a person. There is no mention of when or where he was born; given that he is a hero to whom he runs for; and on which type of investment he is engaged. Whilst it may be preferable to update the answer keys so as to more accurately reflect the information usually contained in encyclopedia or biographical dictionary entries.

While definitional question answering systems tend to adopt a similar structure to systems designed to answer factoid questions there is often no equivalence to the question analysis component due to the little information contained in the question (Andrew, 2005). Any processing of the question that does take place happens within the document retrieval process in an attempt to maintain a useable search space for the extraction of information nuggets.

The following section briefly describe a component of a definitional QA system called Nugget Extraction which is not found in any other types of QA system (Figueroa, 2010).

Nugget Extraction

Whatever the approach to finding relevant sentences the next stage in most definitional QA systems is to cluster, rank, and simplify the sentences to present a short concise definition. A number of systems make use of indicative patterns either to select highly relevant sentences in their entirety or to extract short phrases of information. (Robert Gaizauskas, 2004) Look for part-of-speech based patterns to determine relevant facts about the targets. For instance, the pattern TARGET, WD VBD is used to extract information about people and matches phrases such as *"Aaron Copland, who composed...".* Xu, (2003) uses similar pattern based approaches as well as considering appositives and copula constructions to select relevant phrases such as *"George Bush, the US President..."*.

A number of systems (Echihabi, 2003) have mined clue words from online bibliographies and dictionaries to allow them to find and highly rank sentences about the target which contain these words. Echihabi, (2003) built a list of 6,640 words which occur at least five times in biographies and which occur more frequently in biographies than standard text. This list includes terms such as; Nobel, knighted, studied, travelled, Composer, edited, and Physicist.

Sentences, or phrases, selected using these (and other) approaches are then ranked usually based upon the features they contain (i.e. the number of clue words or the precision of indicative patterns they match). As the current evaluation metric is based partly on the size of the resulting definition ranked sentences are usually clustered and a single representative example from each cluster is used to build the definition (Andrew, 2005). A common approach to clustering in Xu (2003) is simply the word overlap between sentences with using a 70%

overlap to determine that two sentences contain the same information and using 50% overlap in Robert Gaizauskas (2004).

2.3 Approaches to Question Answering

A practical question answering system takes a question posed in natural language as input, accesses a knowledge base, and returns an answer, and all stages of this process are executed automatically, without any human intervention (except for posing the question, of course).

In this section early QA systems that were developed before the 1970's and their general ideas and shortcomings are discussed. Three types of QA systems are distinguished (Monz, 2003): **data base-oriented**, **text based**, and **inference based**. For each type of system a number of implementations are discussed below.

2.3.1 Data Base oriented systems

Database-oriented question answering systems use a traditional data base to store the facts which can be questioned. The database can be queried by natural language questions which are translated into a database language query, e.g., SQL. These types of systems are often referred to as **front-end systems**, because they do not address the problem of answer extraction, but leave this to standard database techniques. BASEBALL[2] and LUNAR[3] are examples of data base oriented QA systems (Monz, 2003).

2.3.2 Text Corpus-Based QA

Text-based systems do not assume the data to be pre-formatted. The data used to answer questions is plain machine readable text. Text-based systems have to analyze both, the question as well as the data, to find an appropriate answer in the text corpus (Monz, 2003).
Textual QA systems are information systems that receive as input a natural language question, search for the answer from a large database of unstructured text and finally return a text string containing the exact answer to the question (Aunimo, 2007). The large database of

[2] answers English questions about the scores, teams, locations, and dates of baseball games
[3] enable lunar geologists to conveniently access, compare, and evaluate the chemical analysis data

unstructured text may consist of newspaper text, of user manuals concerning the products of a company or of documents in the WWW. Sometimes textual QA is also called **corpus-based QA**. Textual QA systems may be either **open** (also called general) domain systems or **closed** (also called restricted) domain systems (Monz, 2003). Open domain systems take as input all kinds of questions. Closed domain QA systems restrict themselves to a specialized domain, such as the law domain or a company's products. The experiments presented in this thesis deal with closed domain question answering of law domain, but the methods presented could as well be used in a open domain system.

In practice, rare systems are purely textual, as it makes sense to store already extracted answers to frequently asked questions and to compare new questions for similarity with the old ones as well as to use structured data in parallel with unstructured data if available. There are QA systems that are solely based on comparing the new incoming question to previously asked questions. These systems direct all new questions (that is all questions that are detected to be very dissimilar form the previous ones) to a human who provides the answer. This type of systems is especially good for cases where questions with the same semantic contents tend to be asked often. When using the documents from the WWW as a database for finding answers, not only the unstructured text but also the structured parts in the documents are very useful. Answering questions from structured or semi-structured data naturally requires different techniques than answering questions based only on unstructured text. QA systems that are based on structured or semi-structured data are based on traditional work on natural language interfaces to relational databases. ORACLE[4] and PROTOSYNTHEX[5] can be exemplified as text based oriented QA systems (Monz, 2003).

2.3.3 Inference Based Systems

Similar to data based-oriented systems, most inference-based systems require the data to be preformatted. Although this is not an essential requirement, it eases the process of inference drawing. The focus of inference-based systems is to infer relationships that are not explicitly

[4] developed by Phillips in 1960
[5] attempts to answer questions from an encyclopedia

stated between entries in the knowledge base on the one hand, and the question and the knowledge base on the other hand (Monz, 2003).

2.4 General QA System Architecture (Document Retrieval)

Textual QA systems may have several different types of architecture. In this section the generic architecture that is common to almost all textual QA systems is presented. The core part has stayed the same for several years and it is typically composed of a **question processor**, a **document retriever, document analyzer** and an **answer processor** (Aunimo, 2007) as depicted below in figure 2.2.

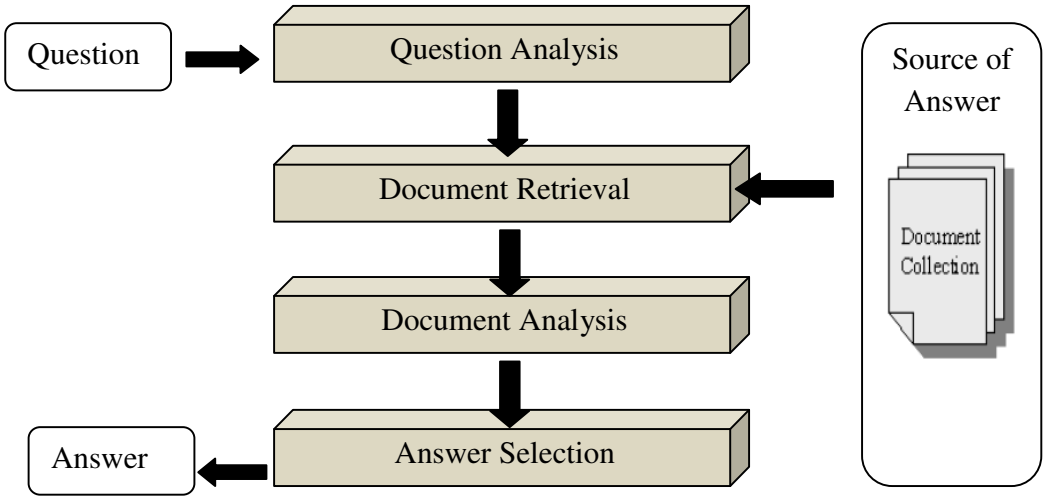

Figure 2.2: Generic Question Answer Architecture

2.4.1 Question Analysis

The first stage of processing in any QA system is question analysis. In general the input to this first stage of processing is a natural language question. However, it should be noted that some systems simplify the analysis component either by using only a subset of natural language both in terms of syntax and vocabulary and by some form of restricted input method. For example, many natural language interfaces to databases have some restrictions on the language that can be used and some question answering systems, such as SHAPAQA which is developed by Bucholz and Daelemans in 2001 require the information need to be specified explicitly through a form style interface. Another issue that question analysis components have to deal with is

that of context. If each question is entirely self-contained and independent from any other question then the system can process it. If, however, the question is part of a longer sequence of questions (often referred to as context questions or a question scenario) then knowledge such as the answer or scope of previous questions may need to be considered, as may anaphora in the question which requires processing to find the correct antecedent before the question can be answered. This is the direction QA research seems to be taking with the questions in the TREC 2004 QA evaluation being divided into sets of questions about a given target entity (Dang, 2011).

The role of the question analysis component is to produce as output a representation of the question which can be used by the rest of the system to determine the answer to a question. Often this actually involves producing multiple output representations for different parts of the system. For example, question analysis components may supply the document retrieval component with a correctly formed IR query generated from the question whilst passing the answer extraction component of semantic representation of the expected answer.

2.4.2 Document Retrieval

Although, we have already been mentioned in section 2.3.2, the question analysis component is usually responsible for constructing an appropriate IR query; this does not mean that the document retrieval component consists solely of using the query to pass on relevant documents to the answer extraction component. In fact the document retrieval component can be quite complex involving numerous strategies for retrieving both documents and associated data as well as determining the amount and structure of the text to be passed to the final component in the system pipeline (Andrew, 2005).

Full syntactic and semantic parsing can be a time consuming process but one which many QA systems rely on for answer extraction. The time required to produce full syntactic or semantic representations often limits their use within real-time question answering (Andrew, 2005). One solution to this, at least for QA over closed collections, would be to pre-parse the whole collection storing the resulting parse trees and semantic representations (Gonzalez, 2006). Of

course, other less intensive and time consuming processing can also be carried out in advance. In fact there is no fundamental reason why all question independent processing (tokenization, Parts of Speech tagging, named entity recognition (NER) ...) cannot be carried out in advance, as long as the time to retrieve the data is not longer than the time required to generate it as this would defeat the main use of pre-processing – faster question answering.

The main role of the document retrieval stage is, however, to retrieve a subset of the entire collection which is processed in detail by the answer extraction component (Andrew, 2005). The main issues arising at this point are which IR paradigm to adopt (ranked or Boolean) and the volume as well as structure of text to retrieve.

Many QA systems make use of ranked IR engines, such as Okapi (Robertson, 1999), although a number of researchers have suggested that boolean retrieval is better suited to the problems of question answering (Harabagiu, 2000). Query formulation for boolean IR approaches is inherently more complex as due care and consideration has to be given to how to rank or restrict an unordered list of relevant documents, something performed automatically when using a ranked IR engine. The main advantage of boolean systems is that their behavior is easier to inspect as the matching process is inherently transparent with little or no interaction between separate search terms.

The document retrieval component is responsible for balancing trade-off between coverage and answer extraction accuracy (Andrew, 2005). Of course the total amount of text passed to the answer extraction component can be structured in a number of different ways. If the answer extraction component works better with a small amount of text then it is likely that passing a few full documents gives worse performance than passing a larger number of short relevant passages making up the same volume of text. This is because the second approach is likely to contain more answer instances than the first (assuming that on average the answer to a question appears at most once in any given document – a reasonable but unsubstantiated assumption). A number of papers have reported numerous ways of segmenting full documents to provide passage selection in an attempt to increase the coverage while keeping the volume

of text to a minimum. Monz (2003) proposes an approach to passage selection based not around fixed passage sizes but rather around the smallest logical text unit to contain the question terms.

2.4.3 Document Analysis

The document analysis component searches through the documents returned by the retrieval component to identify phrases that are of the appropriate type, as specified by the question analysis component. To this end, a named-entity recognizer is used to assign semantic types to phrases in the top documents (Aunimo, 2007). The set of named entities includes person names, organization, dates, locations, temporal and spatial distances, etc. If a phrase is of the appropriate type, it has to be linked to the information need expressed by the question, in order to consider it a potential answer, or candidate answer. Linking a candidate answer to the question is a non-trivial process, and there are a number of ways to do this (Monz, 2003). For certain types of questions, parse trees or parse dependency graphs can be used to determine whether the phrase occurs in the right syntactic position. For instance the answer phrase to the question "*ለብላሽርዚያ በሽታ መድሃኒት ያገኘው ማን ነው*?", can be expressed as the subject of a relative clause "*ለብላሽርዚያ በሽታ መድሃኒት ያገኘው ዶ/ር አክሊሉ ለማ* . . .". In some cases these syntactic relationships can also be approximated by **patterns**, although pattern matching lose some of the flexibility of using a deeper analysis like parsing. For other types of questions, on the other hand, pattern matching is a simple and effective means to find answers. Consider the question "**ደራሲ ስብሃት ገብርእግዚአብሔር የተወለደው መቼ ነው**?", and the text snippet ". ... **ስብሃት** (1936– 2012)." Here, a candidate answer can simply be identified by applying a pattern such as NAME (YEAR_BIRTH-YEAR_ DEATH); pattern matching can lead to a well-performing question answering system (Monz, 2003).

Sometimes, linking a phrase to a question is much more difficult, and involves complex reasoning on the lexical definition of a word. The following example is taken from (Harabagiu, 2008). Consider the question "Where did Bill Gates go to college?" and the text excerpt in "... Bill Gates, Harvard dropout and founder of Microsoft…", which contains the answer phrase Harvard. The fact that Bill Gates has attended Harvard can be intuitively inferred from the noun

dropout. However, drawing this inference automatically can be very difficult. Machine readable dictionaries, such as WORDNET (Miller, 1995), do contain more information about the meaning of the word dropout. The WORDNET entry for dropout is someone who quits school before graduation, but this leaves us with another sub problem. We have to draw the inference that the verb quit presupposes a prior phase of attending, which unfortunately cannot be extracted from WORDNET. This example just illustrates that many inferences that are intuitively rather easy are often hard to automate.

If it is not possible to establish an explicit link between a phrase of the appropriate type and the question, be it via the syntactic structure, pattern matching, or lexical chaining, then linear proximity is often used as a fallback strategy to link the phrase to the question (Greenwood , 2005). As a proximity restriction it is often required that the candidate answer phrase occurs in the same sentence as some of the query terms, or in the preceding or following sentence.

The document analysis component passes on the list of candidate answers to the answer selection component, together with the way in which each candidate answers was linked to the question based on proximity constraints.

2.4.4 Answer Extraction

The answers extracted for textual QA are short text snippets, typically named entities or numeric or temporal expressions. The main approach is the pattern matching based approach (Nico, 2005).

Processing a question in order to find relevant documents or passages is itself a difficult task. It should be remembered that most of the work required to actually answer a previously unseen question takes place within the answer extraction component (Andrew, 2005).
Depending on the approach followed for answer extraction, this component have to perform a number of processing steps to transform the documents passed to it from the document retrieval process into a representation from which the answers can be located and extracted.
Most answer extraction components rely on the relevant documents being subjected to a number of standard text processing techniques to provide a richer representation than just the

words as they appear in the documents (Andrew, 2005). This usually includes tokenization, sentence splitting, POS tagging and named entity recognition. Depending upon the exact approach taken to answer extraction, other techniques can be applied using the output from these simple techniques.

For example, surface matching text patterns (Andrew, 2005) usually require no further processing of documents. These approaches simply extract answers from the surface structure of the retrieved documents by relying on a fairly extensive list of surface patterns. For instance, questions for which the answer is a birth date can be answered by extracting answers using patterns such as the following (Hovy, 2001):

<NAME> (<ANSWER> -)
<NAME> was born on <ANSWER>,
<NAME> was born <ANSWER>
Whilst assembling extensive lists of such patterns can be time consuming and difficult, once assembled they can be exceptionally accurate for such a simplistic approach. One system (Soubbotin, 2001) which used this approach as its main way of answering questions was in fact the best performing system in the TREC 2001 QA evaluation, correctly answering 69.1% of the questions.

Surface matching text patterns operate surprisingly well using little in the way of NLP techniques other than basic named entity recognition. Semantic type extraction systems, however, require more detailed processing of the documents under consideration. Answer extraction components, such as that described by Gaizauskas (2004) operate simply by extracting the most frequently occurring entity of the expected answer type. This requires being able to recognize each entity of the expected answer type in free text, resulting in more complex entity recognition than is required by the surface matching text patterns, usually going well beyond simply named entity detection. While such systems require more detailed processing than the surface matching text patterns they stop short of requiring deep linguistic processing such as full syntactic or semantic parsing of relevant documents.

QA systems such as FALCON (Moldovan, 2002) and QA-LaSIE (Greenwood, 2002) make use of deep linguistic processing requiring syntactic and semantic parsing in order to infer connections between questions and possibly relevant documents.

2.5 Question Answering using Text Mining Approach

Text mining approach of QA describes a method for definition question answering based on the use of surface text patterns. The method is specially suited to answering definition questions and acronym's descriptions and it considers two main steps (Claudia, 2005). First, it applies a sequence-mining algorithm to discover a set of definition-related text patterns from the Web. Then, using these patterns, it extracts a collection of concept-description pairs from a target document database, and applies the sequence-mining algorithm to determine the most adequate answer to a given question.

In the first step, the method applies a mining algorithm in order to discover a set of definition-related text patterns from the Web. These lexical patterns allow associating persons with their positions, and acronyms with their descriptions.

In the second step, the method applies the patterns over a target document collection in order to answer the specified questions. Ravichandran et.al, (2002) apply the patterns over a set of "relevant" passages, and trust that the best (high-precision) patterns allow identifying the answer. In contrast, Claudia et.al, (2012) applies all discovered patterns to the entire target document collection and constructs a "general catalog". Then, when a question arrives, it mines the definition catalog in order to determine the best answer for the given question. In this way, the answer extraction does not depend on a passage retrieval system and takes advantage on the redundancy of the entire collection.

Figure 2.3 shows the general scheme of the method (Claudia et.al, 2012). It consists of two main modules; one focuses on the discovery of definition patterns and the other one on the answer extraction. The two main modules are discussed in the following subsection.

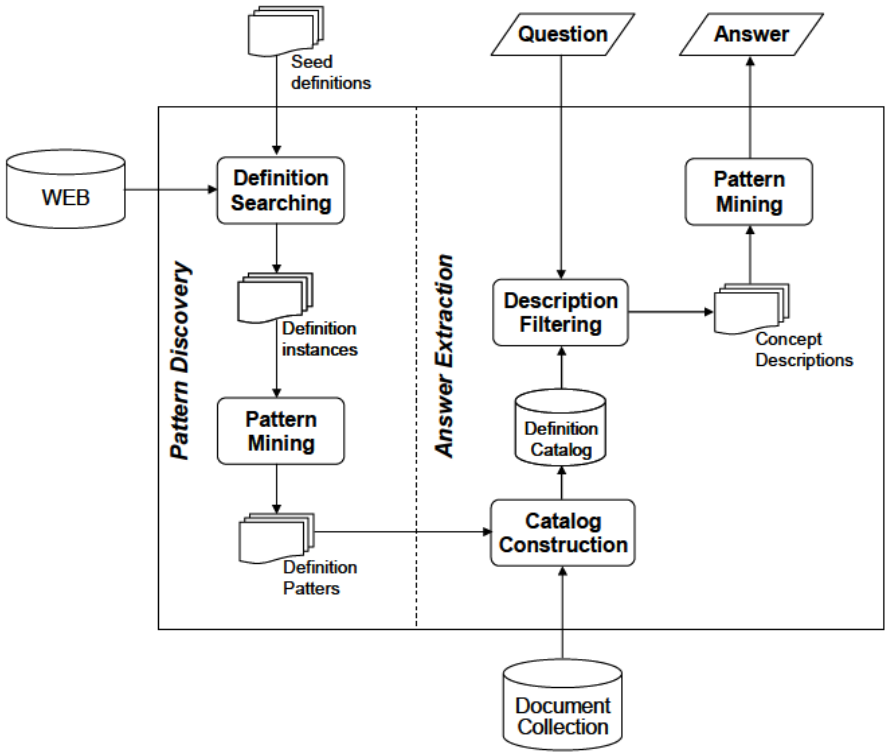

Figure 2.3: General Architecture of QA using Text Mining (Claudia et.al, 2012)

2.5.1 Pattern Discovery

There are certain stylistic conventions frequently used by authors to introduce new concepts in a text. Several QA approaches exploit these conventions by means of a set of lexical patterns (Claudia et.al, 2012). Unfortunately, there are so many ways in which concepts are described in natural language that it is difficult to come up with a complete set of linguistics patterns to solve the problem. In addition, these patterns depend on the text domain, writing style and language.

In order to solve these difficulties a very general method for pattern discovery are used. The method captures the definition conventions through their repetition. It considers two main subtasks (Claudia et.al, 2012).

Definition searching

This task is triggered by a small set of empirically defined concept-description pairs. The pairs are used to retrieve a number of usage examples from the Web. Each usage example represents a definition instance. To be relevant, a definition instance must contain the concept and its description in one single phrase.

Pattern mining

It is divided in three main steps: data preparation, data mining and pattern filtering. The purpose of the data preparation phase is to normalize the input data. In this case, it transforms all definition instances into the same format, using special tags for the concepts and their descriptions.

In the data mining phase, a sequence mining algorithm is used to obtain all maximal frequent sequences–of words and punctuation marks–from the set of definition instances. The sequences express lexicographic patterns highly related to concept definitions. Finally, the pattern-filtering phase allows choosing the more discriminative patterns. It selects the patterns satisfying the following general regular expressions (Chua, 2007):

<left-frontier-string> DESCRIPTION <center-string> CONCEPT < right-frontier-string>

<left-frontier-string> CONCEPT <center-string>DESCRIPTION < right-frontier-string>

Types of Pattern Matching

As noted by Chua (2007), there are two types of pattern matching namely hard pattern matching and soft pattern matching

In hard pattern matching (HM) the alignment between a snippet and a lexical pattern is done slot-by-slot against test sentences and does not deal with the language variation (Rosso, 2010). Most approaches applicable to definition sentence retrieval use hard pattern matching while these patterns are highly precise, they are often poor in recall because of language variations. Many systems create patterns manually Harabagiu (2008) employed 38 manually constructed definition patterns for plain text news articles; Liu (2003) mined topic-specific definitions using hand-crafted rules from Web pages.

Virtually all definitional QA systems that employ manual patterns or automatic rule induction algorithms are hard pattern matching systems, as their patterns perform slot-by-slot matching. But hard pattern matching has two drawbacks of using such generalized pattern rules for extracting definitions (Chua et.al, 2007): *Inflexibility in Matching* (fail to match when there are even small variations between the training instances and the test text, such as extra or missing tokens) and *Inconsistent Weighting of Patterns (*treat each definitional pattern with the same level of importance*)*.

To circumvent the above problems, an alternative pattern generation and matching technique, ***soft pattern matching***, are proposed (Chua et.al, 2007). The manually constructed lexico-syntactic rules are too rigid to cover the notion of extended definitions.

Soft Pattern matching (probabilistic lexico-syntactic patterns): treat definition patterns as sequences of lexical and syntactic tokens. Therefore, pattern matching can be considered as probabilistic generation of test sequences based on training sequences. Lexico-syntactic patterns, which are either manually constructed or machine learned, are often represented and matched as regular expressions.

Soft pattern matching can be considered probabilistic generation of test sequences based on training sequences (Chua et.al, 2007). After preprocessing, each definition sentence is converted into a pattern instance that consists of left and right token sequences relative to the search target. At the training phase, token sequences are aligned and represented as a single vector *P:* $<S_1,S_2, \ldots, S_m>$ that combines information over all of the training sequences, where S_i represents the i^{th} slot (or position) left or right of the search target and contains tokens appearing in that position. Unlike hard matching patterns that generalize training instances into pattern rules, soft matching patterns model each token's distribution statistics in each slot over all training instances. Given a test sequence *T* with length n*:*$<t_1, t_2 \ldots\ldots t_n>$ where t_i is the token corresponding to the i^{th} slot, soft pattern models are used to model the generative probability of sequence *T*, given training sequences represented in vector *P.*

2.5.2 Answer Extraction

This second module handles the extraction of the answer for a given definition question. It is also based on a text mining approach. Its purpose is to find the more adequate description for a requested concept from an automatically constructed definition catalog.

Because the definition patterns guide the construction of the definition catalog, it contains a huge diversity of information, including incomplete and incorrect descriptions for many concepts. However, it is expected that the correct information is more abundant than the incorrect one. This expectation supports the idea of using a text mining technique to distinguish between the adequate and the improbable answers to a given question. This module considers the following steps (Claudia et.al, 2012):

Catalog construction: In this phase, the definition patterns discovered in the previous stage (i.e., in the pattern discovery module) are applied over the target document collection. The result is a set of matched segments that presumably contain a concept and its description. The definition catalog is created gathering all matched segments.

Description filtering: Given a specific question, this procedure extracts from the definition catalog all descriptions corresponding to the requested concept. As mentioned above, these "presumable" descriptions may include incomplete and incorrect information. However, it is expected that many of them may contain the required answer as a substring,

Answer mining: This process aims to detect a single answer to the given question from the set of extracted descriptions. It is divided in to three main phases (Chua et.al, 2007): data preparation, data mining and answer ranking.

The data preparation phase focuses on homogenizing the descriptions related to the requested concept. The main action is to convert these descriptions to a lower case format. In the data mining phase, a sequence mining algorithm is used to obtain all maximal frequent word sequences from the set of descriptions.

Each sequence indicates a candidate answer to the given question. Then, in the answer raking phase, each candidate answer is evaluated according to the frequency of occurrence of its subsequences. The idea is that a candidate answer assembled from frequent subsequences has

more probability of being the correct answer than one formed by rare ones. Therefore, the sequence with the greatest ranking score is selected as the correct answer.

2.6 Related Research Works

Many QA systems have been researched extensively and obtained significant improvement in performance, but the questions are limited in a sense that they can be answered from the information contained within a fixed size corpus. However, the Web is apparently an ideal source of answers to a large variety of questions due to the tremendous amount of information available online for technology favored language like English. Thus, recently QA researchers have explored uses of the Web and tried to port existing QA techniques to a much larger context, WWW.

2.6.1 Global Research Works

An attempt is already done to develop QA system for different languages worldwide including English, Chinese, Indian languages.

2.6.1.1 Question Answering for English language

Many QA systems are publicly accessible on the Web, such as the following online systems: START[6], QuASM[7], SiteQ/E[8] , IONAUT[9], AskJeeves[10], LCC-Web[11], AnswerBus[12], Encarta[13]

START (SynTactic Analysis using Reversible Transformations)

Among above, as representative such as, START (SynTactic Analysis using Reversible Transformations), the world's first Web-based question answering system, has been on-line and

[6] http://www.ai.mit.edu/projects/infolab
[7] http://ciir.cs.umass.edu/ re-u2/
[8] http://ressell.postech.ac.kr/ pinesnow/siteqeng/
[9] http://www.ionaut.com:8400/
[10] http://www.ask.com/
[11] http://www.languagecomputer.com/demos/
[12] http://misshoover.si.umich.edu/ zzheng/qa-new/
[13] http://encarta.msn.com/

continuously operating since December, 1993 (Mohammad, 2008). It was developed by Boris Katz at MIT's Artificial Intelligence Laboratory and a key technique called "natural language annotation" was used in a pre-compiled knowledge base to answer questions (Katz, 1997).

AnswerBus is an open-domain question answering system based on sentence level Web information retrieval. It accepts users' natural-language questions in Multi-lingual and extracts possible answers from the Web. It can respond to users' questions within several seconds. It can respond to users' questions within several seconds.

LCC (Language Computer Corporation)

LCC's QA system (Harabagiu, 2003) was developed by Language Computer Corporation (LCC), and it was the best system in TREC-QA 2002, TREC-QA 2003 and TREC-QA 2004. LCC succeeded because of combining the strengths of Information Extraction techniques with the vastness of axiomatic knowledge representations derived from WordNet to justifying answers that are extracted. It relied a suite of sophisticated tools including Named Entity Recognizer (NE), Syntactic Parser, Logic Form Transformer, Word Sense Desambiguator, Lexical Chainer and Logic Prover, etc. so that it achieved 70-80% accuracy for factoid questions at TREC2003-2004.

However, the above QA systems are still not ready to return the exact answers for the questions. In recent years, some web-based English fully-automated QA systems to return exact answers for user's question have been developed which relies on multiple search-engines (Kwok, 2001). It is based on a novel architecture that combines information retrieval ideas with statistical natural language processing, questions parsing and formulation, answer extraction and voting. The system works creating a specific search that is a reformulation of the question, and extracting the answer from summaries in the search page based on information from its question classifier.

AskMSR

Open-domain Web QA system, **AskMSR**, (Brill, 2002) that applies simple rewriting query to the snippets returned by Google and a set of 15 handcrafted semantic filters to achieve a striking accuracy: Mean Reciprocal Rank (MRR) of 0.507 up to the retrieving 100 snippets. AskMSR,

developed by Microsoft Research, is a relatively simple web-based QA system that makes use of the redundancy in the web rather than applying linguistically sophisticated techniques (Nico, 2005). AskMSR process and answer questions in four steps. At first, it generates rewrites of the question that may occur in a declarative answer. E.g. the question "Where is the Louvre Museum located?" is rephrased into "the Louvre Museum is located in". The **Ephyra** system also generates query reformulations by applying reformulation rules that can be specified in resource files. Secondly, AskMSR queries a web search engine for these rewrites, fetches the snippets from the search results and breaks them down into 1-, 2- and 3-grams. These n-grams are then judged by comparing their type to the expected answer type of the question. E.g. if the question asks for a number ("How many …?"), then the score of an n-gram that contains digits is boosted.

Ephyra

Ephyra's answer type filter follows a similar yet more general approach. In addition, it assigns probabilities to answer type matches and it looks up proper names in named entity lists.
Ephyra question answering engine, which aims to be a modular and extensible framework that allows integrating multiple approaches to question answering in one system (Nico, 2005). Ephyra follows a pattern-learning approach. It automatically learns text patterns that can be applied to text passages for answer extraction. The system can be trained on question-answer pairs, using conventional web search engines to fetch passages suitable for pattern extraction. Nico proposed an approach to question interpretation that abstracts from the original formulation of the question, which helps to improve both precision and recall of answer extraction. In addition, Ephyra deploys knowledge annotation to support frequent question classes and questions that are hard to address with generic methods (e.g. definition questions, queries for the weather). Answers to such questions are extracted directly from structured web sites or web services.

AnsweFinder

The work in (Greenwood, 2005) investigates a number of novel techniques for open-domain question answering. Investigated techniques include: manually and automatically constructed question analyzers, document retrieval specifically for question answering, semantic type answer extraction, and answer extraction via automatically acquired surface matching text patterns. The aforementioned work investigated approaches on Factoid and Definitional QA techniques. The novel techniques in the paper are combined to create two end-to-end question answering systems which allow answers to be found quickly: AnswerFinder (developed by the Centre for Language Technology at Macquarie University) and Varro. Varro builds definitions for terms such as “what is aspirin?” and “define golden age”. Both systems allow users to find answers to their questions using web documents retrieved by Google. Together, these two systems demonstrate that the techniques developed in the paper can be successfully used to provide quick and effective open-domain question answering.

2.6.1.2 Question Answer System for the Bangla Language

Bangla is one of the top 10 most widely spoken languages of the world with over 200 millions speakers, however, having such a vast speaker base the language lacks many of the basic language processing resources and tools that are already available for other languages (Haque, 2010). The motivation of the researchers is, there are no known initiatives for a digital Bangla QA system and tried to grasp this opportunity and propose a basic QA system for Bangla. Then Haque (2010) explored the possibility of a cross-language QA environment where a user would ask a question in Bangla but the system would generate an answer from texts in a language other than Bangla and translate it back to Bangla for the user.

The researchers proposed a framework and the interface is able to take in a Bangla question in a transliterated form and query an Internet search engine that works with English texts. The researchers proposed a transliteration and table look-up based implementation as an interface for a digital Bangla QA scenario but limited the domain to only certain varieties of medical questions. The reason behind choosing the domain was that medical terms in the Bangla

language sound pretty close to their English counterparts. Finally, proved a method to use finite state transducers to translate the medical terms written in transliterated Bangla to their original English spellings. A table look-up approach is used to translate the question and generate the complete English question. They achieved 37% correct translations for the medical terms. With the transliteration module and the table look-up method combined they were able to translate 53% of the questions correctly from Bangla to its equivalent English versions.

2.6.1.3 Chinese Question Answer System

Since Chinese text retrieval has just been developed lately and there are so many various specific characteristics in Chinese language, the research of Chinese QA systems using natural language was developed later than that of western countries and Japan.

The developers have faced the great difficulty of constructing practical Chinese QA systems that use syntactic and semantic information. Thus, the techniques of statistical retrieval and shallow language analysis are chiefly used in QA systems for the reasons mentioned above. On the web-based QA, Huang (2004) attempted to represent Chinese text by its characteristics, and tried to convert the Chinese text into ERE (E: entity, R: relation) relation data lists and then to answer the question through ERE relation model.

2.6.1.4 Arabic Definition Question Answering System

Trigui et.al, (2008) proposed an Arabic definitional Question Answering system based on a pattern approach to identify exact and accurate definitions about organization using Web resources. They experimented using 2000 snippets returned by Google search engine and Wikipedia Arabic version and a set of 50 organization definition questions.

The system does not use any sophisticated syntactic or semantic techniques. The system provides effective and exact answers to definition questions expressed in Arabic language from Web resources. It is based on an approach which employs a little linguistic analysis and no language understanding capability. It identifies candidate definitions by using a set of lexical

patterns, filters these candidate definitions by using heuristic rules and ranks them by using a statistical approach.

Two evaluation experiments have been carried out on the system. 50 definition questions are used for both experiments. The first experiment was based on Google as a Web resource and has obtained an MRR equal to 0.70 and a rate of questions answered by the first answer equal to 54%, while the second experiment was based on Google coupled with Wikipedia as Web resources. In this experiment, they obtained an MRR equal to 0.81 and a rate of questions answered by the first answer equal to 64%.

2.6.2 Local Research Work

An innovative research on Amharic Question Answer for factoid questions was conducted by Seid (2009).

History students or story seekers, tourists, online customers, organization information desk users, service provider's users such as hospitals prefer an exact answer to their query rather than paged details of a document as of search engines. This specific problem motivated the researcher to study and investigate the possibilities of QA system development for Amharic language.

Seid tried to solve problem in finding precise answers for Amharic factoid questions. Novel technique are used to develop to determine the question types, possible question focuses, and expected answer types as well as to generate proper Information Retrieval query, based on Amharic language specific issue investigations.

As to the performance evaluation of the system, the rule based question classification module classifies about 89% of the question correctly. The document retrieval component shows greater coverage of relevant document retrieval (97%) while the sentence based retrieval has the least (93%). The gazetteer based answer selection using a paragraph answer selection technique answers 72% of the questions correctly. The file based answer selection technique exhibits a recall of 90.9%. The pattern based answer selection technique has better accuracy for

person names using paragraph based answer selection technique while the sentence based answer selection technique has outperformed in numeric and date question types.

Seid recommend to developing automatic named entity recognizer, Incorporating a parser and part of speech tagger, Developing Amharic WordNet, Enhancing the Amharic stemmer, Incorporating Machine learning and statistical Question classifications. Moreover Seid recommended extending the research on other question types.

The aforementioned work is original and contributes in paving the way to Amharic QA.

CHAPTER THREE

AMHARIC LANGUAGE

Ethiopia is a linguistically diverse country where more than 80 languages are used in day-to-day communication. Although many languages are spoken in Ethiopia, Amharic (**አማርኛ**) is dominant language spoken as a mother tongue by a considerable part of the population and it is the most commonly learned second language throughout the country. Amharic is a Semitic language spoken in many parts of Ethiopia. It is the official working language of the Federal Democratic Republic of Ethiopia and thus has official status nationwide (Tessema et.al, 2009). It is also the official or working language of several of the states/regions within the federal system, including Amhara and the multi-ethnic Southern Nations, Nationalities and Peoples (SNNP) region. Amharic is written using a writing system called *fidel* (**ፊደል**) or *abugida* (***አቡጊዳ***), adapted from the one used for the Ge'ez language (Tessema et.al, 2009).

3.1. Amharic Grammatical Arrangement

The Amharic language has been declared to have word categories as **ስም** (noun), **ግስ** (verb), **ቅፅል** (adjective), **ተውሳከ ግስ** (Adverb), **መስተዋድድ** (preposition), and **ተውላጠ ስም** (pronoun) (Baye, 2000).

Noun: a word is categorized as a noun, if it can be pluralized by adding the suffix ***ኦች/ዎች*** and used as nominating objects like person, animal, and so on (Getahun, 1989).

Verb: any word which can be placed at the end of a sentence and which can accept suffixes as /**ህ**/,/**ሁ**/,/**ሽ**/, etc. and is used to indicate masculine, feminine, and plurality is classified as a verb.

Adverb: it can be used to qualify a verb by adding extra idea on the sentence. The Amharic adverbs are limited in number and include **ትናንት**, **ገና**, **ዛሬ**, **ቶሎ**, **ክፉኛ**, **እንደገና**…

Adjective: any word that qualifies a noun or an adverb, which actually comes before a noun (e.g. **ጎበዝ ሸፋር** and after an adverb (**በጣም ጎበዝ**). Other specific property of adjectives is, when pluralized, it repeats the previous letter of the last letter for the word (e.g. **አሮጅ**--› **አጫሮጅ**).

Preposition: preposition is a word which can be placed before a noun and perform adverbial operations related to place, time, cause and so on; which can't accept any suffix or prefix; and which is never used to create a new word. It includes **ከ፣ ለ፣ ወደ፣ ስለ፣ እንደ**…

Pronoun: this category further can be divided as deictic specifier, which includes **ይህ**, **ያ**, **እሱ**, **እስዋ**, **እኔ**, **አንተ**, **አንች**…; quantitative specifier, which includes **አንድ**, **አንዳንድ**, **ብዙ**, **ጥቂት**, **በጣም**…; and possession specifier such as **የእኔ**, **የአንተ**, **የእሱ**…

In question answering, part of speech tagging has immense advantages to extract answers. Understanding the structure of a sentence whereby a given noun can be easily indicated to facilitate extracting the correct answer. The Amharic basic sentence is constructed from noun phrase and verb phrase (**ስማዊ ሀረግ + ግሳዊ ሀረግ**) (Baye, 2000).

Sentence= Noun_Phrase + Verb_Phrase.

For example, the sentence: "**ጥቂት ጎበዝ ተማሪዎች ባለፈው ሳምንት ከአክሊሉ ለማ ፋውንዴሽን ሽልማት አገኙ**" (In English, "Last week few brilliant students has got prize from Aklilu Lemma foundation") has noun phrase "**ጥቂት ጎበዝ ተማሪዎች** " (Some brilliant students) and verb phrase "**ባለፈው ሳምንት ከአክሊሉ ለማ ፋውንዴሽን ሽልማት አገኙ**" (Last week got prize from Aklilu Lemma foundation).

3.2. Amharic Punctuation Marks and Numerals

The Amharic writing system consists of as many as ten punctuation marks in addition to the characters (Baye, 2000). However, only few of them are practically used, especially in computer-written system. The word-separator (**ሁለት ነጥብ**), two square dots arranged like colon,(**:**), and sentence-separator, four square dots arranged in a square pattern,(**።**), are the

basic punctuation marks in Amharic writing system. **ሁለት ነጥብ** is oddly used more in hand written practices today than in modern typesetting. Its place is almost completely taken over by space (Bender, 1976).

Lists in Amharic text are separated by an equivalent of comma, '**ነጠላ ሰረዝ**' (') followed by ASCII space and '**ድርብ ሰረዝ**' (፤), which is the equivalent of semi-colon, may also be found in use as a list separator. In addition to these, the writing system has borrowed some punctuation marks from foreign languages. For example, exclamation mark and question marks. The question mark, '?', is used to ask definitive and other questions are used in the language (Bender, 1976).

The Amharic number system consists of twenty (20) single characters. They represent numbers one to ten, multiples of ten (twenty to ninety), hundred, and thousand. These characters are derived from Greek letters (Bender, 1976), and in order to make them look like the Amharic characters the symbols are modified by adding a horizontal stroke above and below. The system has no place value and there is no symbol representing the number zero (0). In addition, the number system does not use commas or decimal points. These situations make arithmetic computation using this system very complicated (Bender, 1976). Both Amharic and Western numerals are in use today. Though the Amharic has long since been retired to a reserved use primarily for calendar dates and demarcation of sections in literature, while Western numerals are used everywhere else following western practices.

3.3. Sentences in Amharic

A sentence, in every language, is a group(s) of word(s) that comply with the grammatical arrangement of the language and capable of conveying meaningful message to the audience. A sentence in Amharic can be a **statement** which is used to declare, explain, or discuss an issue; an **interrogative** sentence which can be used for questioning; **exclamatory** and **imperative** (Baye, 2000)**.** Statement (**አረፍተ-ነገር**) can have the noun phrase and verb phrase combinations. The noun phrase and the verb phrase further divided to different particles such as other sub noun phrase and verb phrase, noun, adjectives, specifier and so on. Similarly, the

interrogative sentences have the same structure with little rearrangements and introduction of question particles. Questions are raised for different purposes to know something unknown, or to assure something that is known.

3.4. Question Particles (Interrogative particles)

A sentence that questions about the subject, the complement, or the action the verb specifies, is called an interrogative sentence or question (Atelach, 2002).

Questions are constructed with the help of question particles (also known as interrogative words) and a question mark (?) which is placed at the end of the question. The question mark, by itself kept at the end of the statement, indicates that the sentence is a question. In English, the interrogative words (WH words) **who, what, where, when, why, how …** are used to construct a question (Atelach, 2002). In Amharic, there are a number of interrogative words that help in constructing a question. These are (Baye, 2000):

ማን (Who), **ለማን** (to whom), **ማነው** (Who is he), **እነማን** (Who for plural),
ማናማን(who and who),
ማንኛው (Who for maculine), **ማንኛይቱ** (Who for feminine), **ማንኛዋ** (Who for feminine),
ጥቀስ (list), **ግለፅ** (explain), **አጠራራ**(brief)
የት (Where), **የቱ** (Which one), **የቷ** (Which one for feminine), **የቲቱ** (Which one for feminine), **የቶቹ** (Which one for plural), **ወዴት** (to where), **የትኛዋ** (Which one for feminine), **የትኞቹ** (Which one for plural)
ምን (What)
ስለምን (about what), **ለምን** (Why),
መች (when), **መቼ** (when), **እስከመች** (up to when),
ስንት (how many/much)
ወይስ (orelse), **ምረጥ** (choose), **ወይ** (or), **እንዴት** (how)

Consider the following Amharic questions, though the questions are all about the same topic the expected answer types are different base on the question particles (Atelach, 2002).

1. **ካሳ መቼ መጣ**?/ When did Kassa come?
2. **ካሳ መጥቷልወይ** ? / Did Kassa come?
3. **ካሳ መጣእንዴ**? / Did Kassa come?

In the first question the interrogator knows that it is Kassa who came, but inquires about the time of his arrival. The second question interrogator knows that Kassa would come but does not know whether he came. The "**ወይ**" in the end implicates that the answer to the question is either yes or no. The last question appears to be the same in idea as the second one, but it does not assure that the answer would be a yes or no like that of the second one.

"**እንዴ**" appears at the end of questions, as "**ወይ**" does. "**እንዴ**" indicates that the interrogator has some prior knowledge about the subject matter at hand.

3.5. Question and Answer Formation

In English, most of the time, the interrogative words occur at the beginning of a sentence (Seid, 2009). For example, in the sentence "**Who** is the president of Ethiopia?", the interrogative word **who** clearly indicates that it is a question sentence. The statement form of the question is "**X** is the president of Ethiopia". In Amharic, however, the interrogative words are put at the end of the sentence more often than as the beginning. Consider the question "**ከአፍሪካ በቆዳ ስፋት ትልቁ ሀገር ማነው**?" (Which country is the largest from Africa), the interrogative word, **ማነው** is placed near the end of the sentence. The statement form of the question could be "**ከአፍሪካ በቆዳ ስፋ ተልቁ ሀገር** X **ነው**". Amharic language is flexible to present a single question in different ways and the focuses remain intact.

Answers arae formed using some of the question terms and the expected answer or just the expected answer only. If a question is the type of yes/no, the answer can also be just yes/no, with possible explanations. If a question is the type of place name, the answer is just the name

of the place or a statement with the place name. For example, the question "**የኢትዮጵያ ዋና ከተማ ማን ይባላል**?" might have an answer "**የኢትዮጵያ ዋና ከተማ አዲስ አበባ ይባላል**" or just a short answer "**አዲስ አበባ**".

Linguists put the formation of Amharic questions and answers differently. Accordingly, questions can be raised about some action or condition, about the performer of an action, about the agent to perform the action or time and place, about the cause of the action or aim of the action, how the action is performed or techniques used to perform the action, and so on (Atelach, 2002). In fact all of these types of questions occur in the phrase, so that the question focus is a phrase. Let us take an example "**ካሳ ከእናቱ ጋር በጠዋት ቁርሱን በላ።**" (Kassa ate his breakfast early morning with his mom). Here we have two noun phrases (**ካሳ** and **ቁርሱን**) and two prepositional phrases (**ከእናቱ ጋር** and **በጠዋት**). We have also the verb phrase that is constructed from the noun phrases and the prepositional phrases which is **ከእናቱ ጋር በጠዋት ቁርሱን በላ**. So questions arise on these five phrases. On the noun phrase we can ask questions like "**ማን ከእናቱ ጋር በጠዋት ቁርሱን በላ**?" and "**ካሳ ከእናቱ ጋር በጠዋት ምን በላ**?". So the question particles (question pronoun) **ማን** and **ምን** are placed on the place of the subject and object respectively.

Similarly, questions can be asked as "**ካሳ ከማን ጋር በጠዋት ቁርሱን በላ**?" and "**ካሳ ከእናቱ ጋር መቼ ቁርሱን በላ**?". Here the questions are about the prepositional phrases. Finally, questions can be raised about the verb phrase as "**ካሳ ምን አደረገ**?"

Unlike to English language, most Amharic definitive questions start by the definiendum. The following are examples of Amharic definitive question:

- **የህግ ትርጉም ምንድነው**?
- **ኃይሌ ገብረስላሴ ማን ነው**?
- **እገዳ ምንድ ነው**?
- **ሪኮንስትራክቲቭ ህግ ማለት ምን ማለት ነው**?

3.6. Challenges in Amharic Question and Answer

Through the adaptation process from Ge'ez language and other factors the Amharic writing system got some problems.

The first problem is the presence of character variants (*fidels)* in the language's writing system. These *fidels* (alphabets) have the same pronunciation but different symbols. Although these different *fidels* give each word different meaning in Ge'ez, in Amharic language they have been used interchangeably. These fidels are **አ** and **ዐ**, **ፀ** and **ጸ**, **ሰ** and **ሠ**, **ሀ**, **ሐ**, **ኃ**, **ኀ**. For example, the word "sun" can be written as, **ጸሀይ**, **ጸሓይ**, **ፀኃይ**, **ፀሓይ**, etc all mean the same, although they are written differently. Several solutions have been proposed. One of them is to choose one letter (e.g. one of **ሀ**, **ሐ**, **ሃ** or **ኀ**) and eliminate the unnecessary ones. Such challenge is handled by normalization of the definiendum characters both the definiendum and the Amharic legal documents.

Another very interesting challenge of QA in Amharic language is identifying definiendum from the question. In English identifying definiendum is not a problem as they are capitalized in most of the time. There is no capitalization in Amharic so that a proper name/ definiendum are written similarly with other parts of speech. To identify definiendums from the questions we developed an algorithm that extracts it from the Amharic definitive questions.

Statements in Amharic have unique punctuation mark, which is **አራት ነጥብ** (።), to separate from other statements. The problem occurs on the writing style of different bodies, where using two colons (::) in place of ። is very difficult to demarcate statements as these symbols have different Unicode representation. The other problem is that, there is a practice not to use the punctuation marks. Hence, document normalization is made to bring the document to same standard.

Chapter FOUR

DESIGN OF *DefAmharicQA*

In this study surface text patterns are used to define a concept. The study takes advantage of some stylistic conventions frequently used by writers to introduce new concepts. These conventions include some typographic elements that can be expressed by a set of lexical patterns and the patterns are created manually.

4.1 Components of *DefAmharicQA*

From a general viewpoint, the system is composed of the following components: Indexing and definition searching. Through the Indexing component there are two subcomponents preprocessing of the corpus and extraction of all potential definitions from the corpus are extracted using patterns. The definition searching component comprises of question analysis and retrieving the appropriate definition

DefAmharicQA starts by analyzing the Amharic legal corpus, the “Indexing” module handles language dependent issues of the documents plus extracts a collection of concept-description pairs from the Amharic legal corpus; the “Question Analysis” module extract the definiendum from the natural language question. In the module “Definition Extraction”, the lexical patterns are used to identify the candidate definitions from the selected snippets (i.e., a set of lexical patterns are manually constructed as there is no large Amharic legal corpus to automatically create the patterns). Figure 4.1 depicts the general architecture of *DefAmharicQA.*

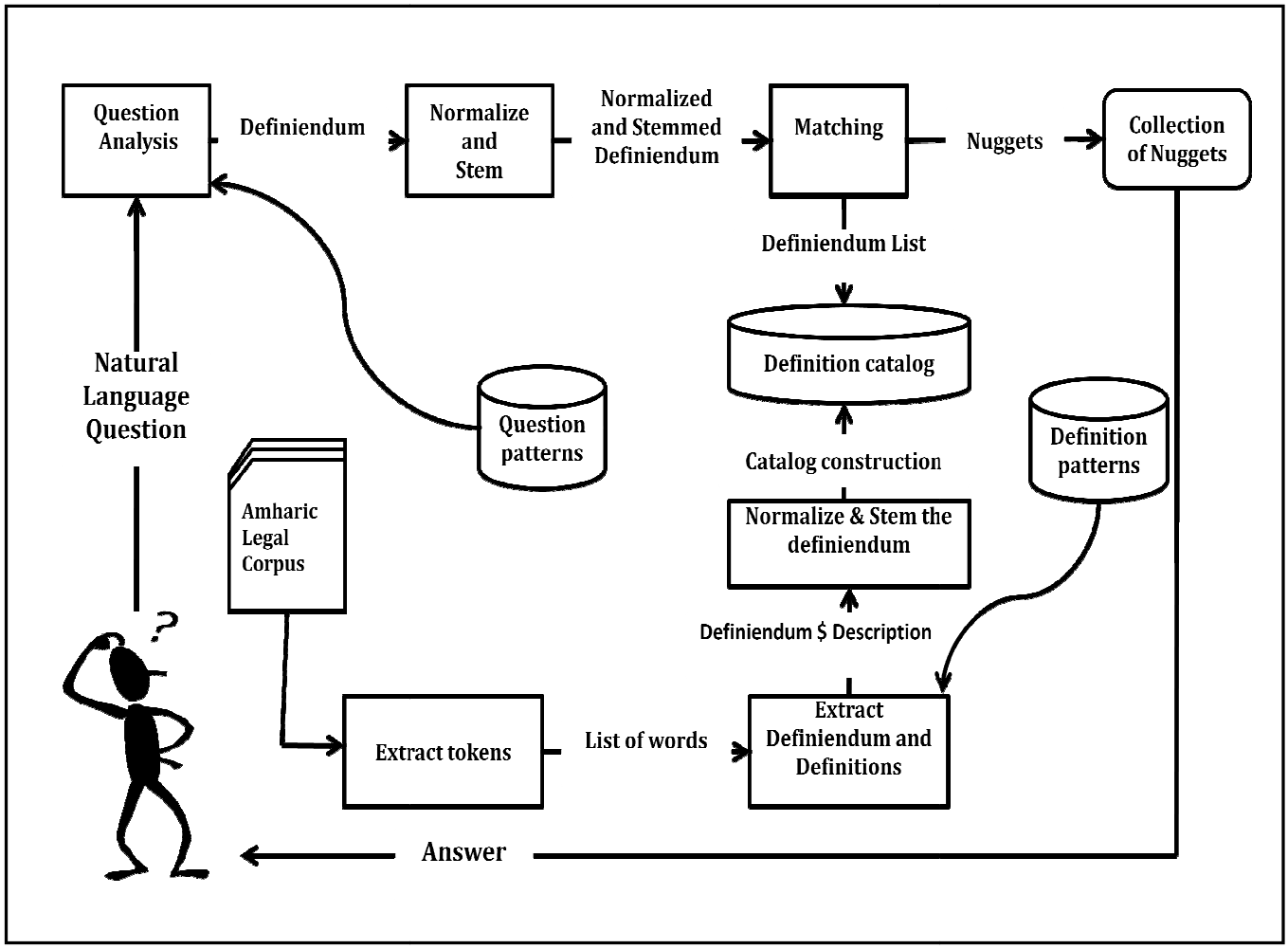

Figure 4.1: Architecture of *DefAmharicQA*

4.2 Document Preprocessing

For the purpose of this thesis, Amharic legal corpuses are collected from Ministry of Justice, Federal Supreme Court and website[14]. The document consists of proclamations, laws, directives, and training manuals. For testing purpose, documents are preprocessed to evaluate the prototype. The preprocessing step includes sentence extractor, character normalization, sentence/ paragraph tokenization, stemming, and synonym. We discuss the key tasks under document processing. The detail steps of document preprocessing are depicted on Figure 4.2.

[14] http://www.abyssinialaw.com/

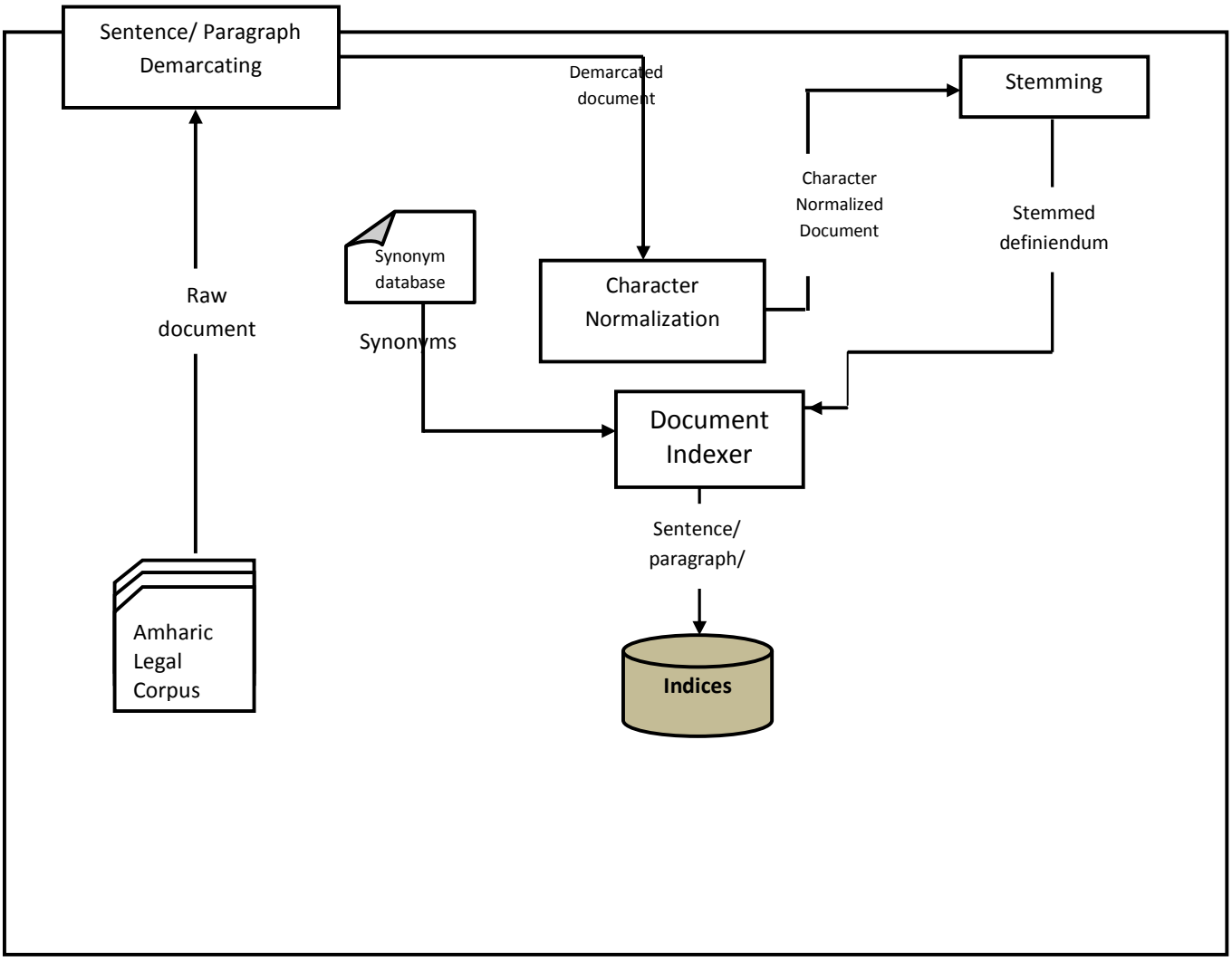

Figure 4.2: Document preprocessing subsystem of *DefAmharicQA*

Sentence: Paragraph in Amharic usually starts with new line and in some documents it is accompanied by indentation. A special symbol $ (dollar sign) is used to demarcate a paragraph in this thesis. Amharic statements are usually separated one from the other by a special Amharic punctuation mark **። (አራት ነጥብ)** analogous to the English full stop or period.

Character Normalization: One of the problems in Amharic writing is the variation of alphabets (*fidels*) used with the same pronunciation. Although in the Ge'ez language, these different symbols give each word different meanings, in the Amharic language they have been used

interchangeably. These *fidels* (alphabets) have the same pronunciation but different symbols (Mindaye et.al, 2010). These different *fidels* can be used interchangeably without meaning change. The *fidels* are አ and ዐ, ጸ and ፀ, ሰ and ሠ and ሀ, ሐ, and ኀ. For example, the word "sun" can be written as, ጸሀይ, ጸሓይ, ፀኃይ, ፀሓይ, etc … all mean the same, although they are written differently. Consider, the use of all of the three different symbols ሀ, ኀ, and ሐ became unnecessary, since all these are now given the same meaning. Likewise, both ሰ and ሠ have the same meaning, and it holds true for ፀ and ጸ (Bender, 1976). Consider another examples, one may write ሀይማኖት, ሓይማኖት, ሐይማኖት, ኀይማኖት which means 'religion'; the words ንጉስ and ንጉሥ (to mean 'king') are the same and do not change the meaning of the word. Similarly, the words ፀደይ and ጸደይ (to mean 'spring') have the same meaning. Therefore, such different writing systems should be solved in the automatic text processing including their orders of the language as they may decrease the performance of *DefAmharicQA*. Tessema has developed an analyzer for normalizing documents to a specific form of a letter such as ሰ and ሠ to ሰ and ሀ, ኀ, and ሐ to ሀ and ጸ and ፀ to ፀ as well as their orders (ሠ, ሡ, ሢ, etc. to ሰ, ሱ, ሲ, etc.) (Tessema, 2007). In addition to the above normalizations, (Seid, 2009) investigated and found that some other orders of the letters should also be normalized. For example ሀ, ኀ, ሐ, ሃ, ኃ, and ሓ should be normalized to ሀ. Similarly the characters ቸ, ቼ; ሸ, ሼ; የ, ዬ; አ, ኣ, ዐ, ዓ; ው, ዉ; ኝ, ኚ; ች, ቺ; ሯ, ጨ, and so on should be normalized to one form as they are being used interchangeably in documents. The algorithm presented in Algorithm 4.1 is used to accomplish this task (Zelalem, 2001).

For each word in a definiendum list

For each character in a word

If the character is any one of ሃ፣ ሐ፣ ኀ,or ኃ any other order thereof then

Replace it by ሀ continue

Else if the character is any one of

ኣ, ዐ, or ዓ any other order thereof then

Replace it by አ continue

Else if it is ሠ or any other order thereof

Replace it by ሰ continue

Else if it is ፀ

Replace it by ጸ· continue

Else if it is ፀ or any other order thereof

Replace it by ጸ continue

End if

End for

End for

Algorithm 4.1: Algorithm for character normalization

Stemming: Stemming (i.e., grouping words that share the same morphological root), is also one technique to handle word variants. For this thesis work, stemmer algorithm of Nega (2002) is used and the stemmer stem inflectional morphology, which stems a word for tense, number, case, gender and case differences.

Synonym indexing: WordNet has been recognized as a valuable resource in the human language technology and knowledge processing communities but there is no Amharic WordNet

so far (Tessema et.al, 2009). Therefore, synonym indexing is important in the absence of WordNet to language. Amharic question and answer synonym words are prepared based on the usage similarity of definition question and answers.

Indexing: Indexing is the last stage of document pre-processing. Once the document contents are normalized using the previous techniques, it is indexed. Sentence indexing is done on the document based on the sentence punctuation mark, Amharic full stop.

4.3 Question Analysis/ Processing

When users submit their question to search engines it suffices to forward only the key words; unlike search engine queries, question answer queries needs extra information included and have a greater importance to retrieve appropriate documents and to extract the exact answer. But most of the time natural language questions that sought definition of a word don't include extra clues which makes definitive question answer challenging.

Question Analysis module is a vital component of *DefAmharicQA*. The result of this module is the identification of the definiendum. The first stage of processing in any QA system is question analysis (Greenwood, 2005). The question analysis component is usually responsible for generating an IR query from the natural language question which can then be used by the definition extraction module. The question analysis module is responsible to interpret the question in such a way that it would be suitable to the definition extraction components; but any mistakes made at this point are likely to miss the definiendum and any further processing of the question leads to wrong definition. If the question processing module incorrectly determine the definiendum it is highly unlikely that the system returns a correct answer.

4.4 Definition Searching

The searching definition module is initiated after accepting a natural language question from the user and then preprocessing is applied on the question. There are two components in the searching definition module: identifying the definiendum and answering.

Definiendum identification is the core component of the whole system even if most Amharic definition questions put the definiendum at the beginning; we developed an algorithm to extract the definiendum using patterns. To return the answer for the definition question, the indexed file which contains concept-description pair is opened and search for a match between the concept and definiendum. If the definiendum is found in the dictionary catalog it display the answer otherwise it display an error message.

CHAPTER FIVE

IMPLEMENTATION AND EXPERIMENT of *DefAmharicQA*

Generally *DefAmharicQA* has two main components: Indexing and definition searching. Through the Indexing component there are two subcomponents preprocessing of the corpus and extraction of all potential definitions from the corpus using patterns. The definition searching component comprises of question analysis and retrieving the appropriate definition.

5.1. Indexing and Definition Identification

The Amharic legal corpus is thoroughly studied and advantage is taken on the stylistic nature of authors to introduce new concept. There are some words that usually exist right after the definiendum words (post definiendum words): **ማለት**, **የሚባለው**, **ትርጉም**, **ትርጉሙ**, **ሲተረጎም**, **ሲባል**, and **፡-**.

The first step in the implementation of *DefAmharicQA* is document preprocessing. After document processing each file is read from the Amharic legal corpus, tokenized and all the tokens are saved in a file as list of tokens. Each token are read and whenever there is a match between the read token and words in post definiendum the word/ phrase before the post definiendum are considered as a concept/ definiendum. Then on the potential definiendum preprocessing is applied and indexed with its description, the sentence, separated by dollar ($) sign (i.e. <concept>$<description>) in a file (definition catalog). Therefore, two separate indices are created; the first is index consisting of tokens and the second indices are candidate definitions/ sentences based on the already identified signal words. To be relevant, a definition instance must contain the concept and its description in one single sentence.

5.1.1. Document Preprocessing

Document preprocessing is a procedure which may include many text operations: Lexical analysis of the text with the objective of treating digits, hyphens, and punctuation marks;

stemming of the remaining words with the objective of removing affixes (i.e., prefixes and suffixes) and allowing the retrieval of documents containing syntactic variations of query terms.

Cognizant of the above, the Amharic legal corpuses require preprocessing which includes normalization and stemming to make it ready for *DefAmharicQA* system and the different pre-processing tasks that are performed on the text before it is indexed.

In this study normalization refers to character and document level normalizations

Character Normalization: As pointed out in section 4.2, the presence of several Amharic alphabets (*fidels*) with the same sound in the Amharic writing system creates a problem in text retrieval. The same word written using different symbols (letters) with the same sound are considered as different words by the retrieval system. In this research, choosing one letter from the group of letters with the same sound and replace the remaining ones is taken as a solution to the problem. Therefore, if a character is any of ኀ, ሐ, ኻ, ኍ, and ሓ (all with the sound 'h') is normalized to ሀ. Also the different orders of ሐ and ኍ are changed to their corresponding equivalent orders of ሀ. Similarly, all orders of ሠ (with the sound 's') are changed to their corresponding equivalent orders of ሰ, all order of ዐ (with the sound 'a') are changed to their corresponding equivalent orders of አ, all orders of ፀ (with the sound 'tse') are changed to their corresponding equivalent orders of ጸ.

Document Normalization: Document normalization for character standardization helps to make a standard document that can match queries regardless of differences that occur during writing and reading; this help in matching documents otherwise may left unmatched.

The algorithm works as follows. Read character by character and if a character is in one of the list (**አኡኢኤእኦሰሱሲሳሴስሶጸጹጺጻጼጽጾወሀሁሂሃሄሆ**) do nothing as it is doesn't require normalization; but if a character is one of variations of the above list replace the character by the next immediate character as shown in the implementation Listing 5.1.

Listing 5.1 : Normalization Code

```
def normalizer(query):
   normquery=[]
   lis=u'አኡኢኤእአሰሱሲሳሴስሶጸጹጺጻጼጽጸዉሀሁሂሄህሆ'
norm=u'ዐአዑኡዒኢዓአኣኤዔእዖኦእየአሠሰሡሱሢሲሣሳሤሴሥስሦሶፀጸፁጹፂጺፃጻፄጼፅጽፆጸውዉሐ
ሀሑሁሒሂሃሀሓሀሔሄሕህሖሆኅህኑሁኒሂኃህኔሄኅህኖሆኻሀኹሁኺሂኻሀኼሄኽህኾሆ00'
   for a in range(len(query)):
      str1=query[a]
      temp=''
      for b in range(len(str1)):
         for x in range(len(lis)):
            flag=0
            if str1[b]==lis[x]:
               temp+=str1[b]
               flag=1
               break
         if flag==0:
   for y in range(len(norm)):
               if str1[b]==norm[y]: # if the character is in the list
                  temp+=norm[y+1] # use the standard character we set
                  break
            else:
               temp+=str1[b]
      normquery.append(temp)
f.close()
   return normwords
```

Stemming: There are two schools of thoughts on the application of stemming in general information retrieval and for question answer systems in particular. The two thought are categorized whether they are in favor or against the application of stemming in QA systems because of performance reasons (increase/ decrease). Cognizant of this fact we performed the experiment with and without applying stemming on our document and query.

Amharic is a morphologically rich language where up to 120 words can be conflated to a single stem (Tessema, 2007). This clearly shows that stemming has a profound effect on the retrieval process. A morphological variant of a word can just differ in tense, case, plurality, etc or can have a whole different meaning or class. Stem is part of the word that never changes even when morphologically inflected. For example, *walk* is the stem for the words *walk*, *walks*, *walking*, and *walked*. ***ሰበር*** in Amharic is the stem for the words ***ሰበርኩ***, ***ሰበርሽ*** , ***ሰበርን*** , ***ሰበርኣችሁ***, etc.

Amharic can create few hundreds of words by attaching different affixes to a stem[15]. Stemming can be applied on both derivational morphology[16] and inflectional morphology or on either of the two. Derivational morphology usually results in a change in class of a word which in turn may result in some loss of semantic (i.e. change in meaning). In Amharic language, some prefixes and combination of some prefixes and suffixes create different meaning when they are applied on the given stem. This kind of semantic loss is not usually witnessed during inflectional morphology, which usually involves grammatical features such as, singular/plural, and tense. Some of them are the following:

Number: **-ኦች** like in **ሰዎች → ሰው + ኦች**

-ዎች like in **በሬዎች → በሬ + ዎች**

Tense: **ይ-** like in **የ ሰበር → ይ+ ሰበር** future

-ኡ like in **ሰበሩ → ሰበር+ - ኡ** past and plural

Stemming helps to reduce distinct words to their common grammatical root. Stemming algorithm proposed by Nega Alemayehu (Nega, 2002) is used in this study. All words are not

[15] **አምድ** (Amharic meaning)
[16] **ስነ ምእላድ** (Amharic meaning)

stemmed except the definiendum. Unlike to the previous ; proper nouns, dates, and numerals are indexed in Tessema (2007) except with some prefixes and suffixes such as ዐ, ለ, የ, ከ, ና, ን, ም, እነ, etc. Listing 5.2 shows implementation of stemming code used for prefix removal; if the definiendum starts by one of the characters 'የከለዐ' it consider as prefix and discard the character.

Listing 5.2: Stemming code to remove prefixes

```
def stem(stopfree):
    stemw=[]
    temp=''
    pre=u'የከለዐ'
    flagpre=0
#prefix
    for i in range(len(stopfree)):
        if (len(stopfree[i]))>2:
            for k in range(len(pre)):
                if stopfree[i][0]==pre[k]:
                    flagpre=1
                    break
        if flagpre==0:
            temp=stopfree[i]
            stemw.append(temp)
            temp=''
        else:
            for j in range(len(stopfree[i])):
                if j>0:
                    temp+=stopfree[i][j]
            stemw.append(temp)
            temp=''
            flagpre=0
        return stemw
```

Listing 5.3 shows the implementation of the stemming code used for suffixes removal if the length of the definiendum is greater than or equal to three. Check the definiendum if it ends with the character '**ች**', if so and the character preceding '**ች**' is the character '**ም**' remove the suffix '**ምች**' otherwise remove the suffix '**ች**' and convert the remaining last character to its sixth order. If the last character of the definiendum is the seventh order replace by its sixth order (*sades*).

Listing 5.3 : Stemming code to remove suffixes

```
def stem2(stemw):
   stemw1=[]
   stemw2=[]
   suf=u'ዎች'
   temp=''
   flag=0 # for 'ሳድስ'
   flagsuf1=0 # for ች
   flagsuf2=0 # for ዎች
# suffix
   for i in range(len(stemw)):
      if (len(stemw[i]))>=3:
         if stemw[i][len(stemw[i])-1]==suf[1]:
            flagsuf1=1
            flag=1
         if stemw[i][len(stemw[i])-2]==suf[0]:
            flagsuf2=1
            flag=0
      if flagsuf1==0 and flagsuf2==0:
         temp=stemw[i]
         stemw1.append(temp)
         temp=''
      if flagsuf1==1:
         for j in range(len(stemw[i])):
            if flagsuf2==1:
               if j<(len(stemw[i])-2):
                  temp+=stemw[i][j]
            else:
               if j<(len(stemw[i])-1):
                  temp+=stemw[i][j]
         stemw1.append(temp)
         temp=''
         flagsuf1=0
         flagsuf2=0
      str1=stemw1[i]
      if flag==1:
         lis=u'ማከትኝውርልድህይቅምስብችንእዉዝጥ'
         sad=u'ንግኮክቶትጭኝዎውሮርሎልዶድሆህዮይቆቅሞምሶስቦብቾችኖንኡእዎዉዞዝጦጥ'
         temp=''
         check=0
         for b in range(len(str1)):
            for x in range(len(lis)):
               check=0
               if str1[b]==lis[x] and b==(len(str1)-1)    …
         stemw2.append(temp)
         flag=0
      else:
         stemw2.append(str1)
 return stemw2
```

Sentence and punctuation mark normalization: The collected Amharic legal documents used different types of punctuation marks to represent space between words. Some documents don't have any symbol to demarcate the end of sentences hence words like **ነው·** and **ናቸው·** are used to identify the end of a sentence. All colons (**፡ ሁለት ነጥብ**) between words are replaced by space and group of colons (፡፡) is replaced by the Amharic full stop (።).

5.1.2. Definition Identification

The definition extraction module identifies and extracts definition sentences from the relevant document set. Since it is all-important to provide enough contexts to ensure the readability of the final answer, definition QA systems prefer sentences to nuggets. Definition type questions require word meaning, term definition, and description of term and so on. So as to give ample information for the user we extract a full sentence as answer for users natural language query.

Extracting definitions is a very important process in definitional QA systems and there are a number of definition extraction approaches as we discussed in the literature review and among them hard pattern matching is used to identify definitions with their descriptions. HM alignment between a snippet and a lexical pattern is done slot-by-slot.

While preprocessing is conducted, from the Amharic legal corpus the maximum possible concept-dentition is extracted and saved in a file (definition catalog). All definitions in the documents are searched based on concept - description pairs by applying definition patterns prepared manually by analyzing the stylistic usage of authors to introduce new concept.
The following patterns are prepared; in addition definition signal words at the end of the sentence like **ነው·**, **ናቸው·** and **ይተርጎማል** are used to identify the concept- description pairs. Table 5.1 shows definition patterns authors usually use to introduce new concept.

Table 5.1: Hard Definition Patterns

‹definiendum› (**ማለት፣ የሚባለው**)
‹definiendum› (**ወይም**) * (**በመባል የሚታወቀው፣የሚባለው**)
‹definiendum› (**ማለት፣ የሚባለው፣ በተለምዶ፣በልምድ**) * (**በመባል የሚታወቀው፣ይባላል**) * (**ነው፣ናቸው**)
‹definiendum› (**የሚገልፀው፣ተብሎ የሚገለፀው**) * **ነው**
‹definiendum› (**ትርጉሙ፣ትርጉማቸው**) * (**ነው፣ናቸው**)
የ‹definiendum› (**ትርጉም**) * (**ነው**)
‹definiendum› (**ማን፣እነማን፣ማን፣ምንድን**)* (**ነው፣ናቸው**)

5.1.3. Searching Definition

The searching definition module starts by accepting a natural language question from the user. Then any punctuation mark is removed from the question, stemming is applied, and character normalization is performed. There are two components in the searching definition module: identifying the definiendum and answering.

Definiendum identification is the vital component of the whole system as it is a decisive factor to answer the question correctly. Though most Amharic definition questions positioned the definiendum at the beginning, the algorithm formulated and depicted in Algorithm 5.4 extract the definiendum from the question and advantage of the stylistic presentation of users' question is taken. The identified definiendum is captured by a variable.

To return the answer for the definition question, the indexed file which contains concept-description pair is opened and search for a match between the concept and definiendum. If the definiendum is found in the dictionary catalog it displays the answer; otherwise it displays a message "No answer".

The next sections discuss question analysis, question particle identification, and definiendum extraction.

Question Analysis

Given a natural language question as input, the overall function of the question processing module is to process and analyze the question, and to create some representation of the definiendum. The query for search engines and QA systems are different in such a way that queries in QA should be more specific to locate relevant documents.

The question-answering process in *DefAmharicQA* like that of most other question-answering systems begins with a question analysis phase that attempts to determine what the question is asking for and how to approach answering it. Question analysis receives unstructured natural language question as input and identifies syntactic and semantic elements of the question, which are encoded as structured information that is later used by the other components of *DefAmharicQA*. Nearly all of *DefAmharicQA's* components depend in some way on the information produced by question analysis.

Even though structurally simpler than the more general class of fact-seeking questions, definition questions have a few properties which complicate their processing as there are only few available keyword(s) in the question.

Question particle identification: Despite dealing with a wide variety of natural language inputs, there are some distinctive structures utilized by users for indicating definition queries. The question particles are used for different types of question formulation. Table 5.2 shows list of a few of the most common definition query logs.

Table 5.2: Definition question particles

Question particles	Question type
<definiendum> ማለት ምን ማለት ነው? What does <definiendum> mean?	Term/expression
<definiendum> ማን ነው?/ <definiendum> ማን ናቸው? Who is/are <definiendum>?	Person name, place, organization
<definiendum> ምንድነው?/ <definiendum> ምንድን ናቸው? What is/are <definiendum>?	Term/expression
የ <definiendum> ትርጉም ምንድነው? What is the meaning of <definiendum>?	Term/expression
ተርጉም <definiendum> Define <definiendum>	Term/expression

Unlike to what these five cases might suggest, the categorization of definition questions and extraction of the definiendum from poor construction of question cannot be seen as an easy task.

Table 5.2 clearly shows that questions are analyzed first for the presence of possible question particles. Questions with none of the question particles are classified as can't be answered and no answer is returned. Otherwise, it first check the question if it has the specific question particles that has clear indication on the question type such as ማለት, የሚባለው, ምንድነው, ምንድናቸው ትርጉም, ተርጉም which always indicate looking for a certain description. If the definition question contains ማን, ማናቸው, and ማን ናቸው indicates the question focus on such as person, place and organization type of question types.

Query Generation/ Definiendum Extraction

The goal of query generation is to construct one or more queries in the query syntax of the document retrieval engine. Most if not all, question analysis components create an IR query by starting with a bag-of- words approach; all the words in the question ignoring case and order are used to construct a query (Greenwood, 2005).

The major problem we face when dealing with definition questions is that little or no information is provided other than the target itself which makes it difficult to construct rich IR queries that locates relevant documents. For instance submitting the target ሃይሌ ገብረስላሴ as two terms to an IR engine not only retrieve relevant documents but also documents about the famous film producer ሃይሌ ገሪማ and other people with the name ሃይሌ could be retrieved by such a query.

As question answer systems attempt to define a given target, it is likely that relevant documents contain the definiendum in the text and so one possible approach to improving the retrieval and hence the quality of the resulting passages, is to use the target as a phrase when searching a text collection (i.e. in a relevant document both terms must appear next to each other, ignoring stopwords, and in the same order as in the query). For a simple example such as ሃይሌ ገብረስላሴ this approach has the desired effect dramatically reducing the number of documents retrieved from the collection while retaining those which are actually about the target.

Users' natural language questions are not submitted to the indexed file as it is. It requires some modifications that enhance the possibility of matching relevant concept from the dictionary catalog. Query generation sub module is very critical; otherwise incorrect queries might result in returning irrelevant description and that lead to extract incorrect definition answer. The query generation sub module includes stemming, character normalization, and stopwords removal and synonym expansion as we don't have Amharic WordNet. We discussed the important procedures we followed to generate the definiendum as follows.

Similar to document preprocessing stage, the first duty in query generation is to remove punctuation mark and apply normalization. Character normalization is applied in to the query term that helps to make standardize the characters in to the same format that we used in the document preprocessing module. If character normalization is not handled at query generation stage of question processing, it reduces recall significantly as there could be unmatched documents with the query.

The other task in query generation is number normalization. Number normalization is done in the question processing module to match various representations of numbers in a document; specifically all Ge'ez numerals found in the query arevalso be expanded in to the Arabic number format as the Amharic legal documents used both types of numbering so that the different variations of numbers are included to the query (query expansion).

To sum up, suppose the question is about specific term. The query term may contain regular expressions such as "([definiendum] |ማለት | ምንድነው)", "(የ[definiendum] ትርጉም)", "(ትርጉም+ [definiendum])" etc. Therefore documents aren matched also based on the regular expressions besides the normal query terms. We have identified a number of words that provide signal for the position of definiendum from construction of natural language query and we classify them based on their position in the query as shown below.

List one contain the tokens ማለት የሚባለው ትርጉም

List two contain tokens ትርጉም ፍቻ አብራራ ዐብራራ ዓብራራ አብራራ ግለፅ ግለጽ ማለት የሚባለው በተለምዶ በልምድ ምን ምንድን ማን እነማን ትርጉሙ ትርጉማቸው የሚገልፀው

Algorithm 5.4 is used to identify definiendums from user's natural language questions. The algorithm works based on the number of tokens in user question. If the question contains only one word, it is considered as the definiendum, if the question contain two words and if the first token is found in list one take the second token as the definiendum; if the second token is found in list two, take the first token as the definiendum otherwise consider both tokens as definiendum.

Algorithm 5.4: Algorithm for definiendum extraction

For i=1 to n do

If the number of token is one

Take the token as definiendum

If the number of tokens are two

If the 1st token is in List one

Take the second token as definiendum

If the 2nd token is in List two

Take the first token as definiendum

Else

Consider both tokens as definiendum

If the number of tokens are three

If the 1st token is in List one and the 3rd token is in List two

Take the 2nd token as definiendum

If the 1st token is not in list one and the 2nd token is not in list two but the 3rd token is in list two

Take the first and the second tokens as definiendum

If the first token is in list one but the 2nd and the 3rd tokens are not in list two

Take the first two tokens as definiendum

If the 1st token is not one of the tokens in list one but the2nd token is in list two

Take the first token as definiendum

If the number of tokens in the query are greater than or equal to four

Do the same as token=3 except the 3rd if clause is modified to:

If the first token is in list one the 2nd and the 3rd tokens are not in list two but the nth token is in list two

Take the first two tokens as definiendum

End if

End for

5.2 Performance Measure

This study undertakes experiments to design Amharic question answer for definitive questions. The focuses of the experiment are effectiveness with due emphasis on correctness, completeness and exactness by computing recall and precision of the system.

We evaluated the question processing module to what degree it identifies the definiendum (target word) as it is a decisive factor for our system to get appropriate definitions for users natural language query. We have evaluated the performance of our system with the documents that have been distributed for different people for formulating definition questions with and without applying indexing on the legal corpus. Moreover we evaluated our system with the effect of stemming on the performance of our system. The actual performance of our system is evaluated by the fraction of correctly extracted definition from the corpus (precision).

For the experimentation Toshiba laptop with specification Intel® Dual-Core™ CPU T2390 @ 1.86 GHz 1.87 GHz, 2GB RAM, 250 GB Hard Disk and Windows® 7 professional edition operating system is used. The widely used text processor software, Python v2.7 is used to implement the algorithm we have designed to develop the system.

5.2.1 Dataset preparation

We have collected about twenty Ethiopian government proclamations, five law related directives and four training manuals a total of one hundred seventy five pages related to law.

We have selected the area, law, because it comprises of many definitions which are suitable for the purpose of designing a definitive question answer system.

5.2.2 Question preparation

As mentioned in the above section, for the evaluation purpose we have prepared the Amharic legal document and provide to ten individuals so as to formulate two definition questions each from the document we provided to them. We have collected twenty definition questions only because preparing vital and non-vital nuggets for each question require much effort of the

assessor. Whence, the twenty definition question formulated by volunteer participants from different walks of life and professions are used to test our system performance.

5.2.3 Answer judgment

As it holds true for many other information retrieval tasks, legitimate differences in opinion about relevance are an inevitable fact of evaluating definition questions. Such systems are designed to satisfy real world information needs, and users inevitably disagree on which definitions are important or relevant. These disagreements manifest as scoring variations in an evaluation setting. The important issue, however, is the degree to which variations in judgments affect conclusions that can be drawn in a relative evaluation. The vital/non-vital/no distinction on definition answers is one major source of differences in judgment. To alleviate such variations on evaluating answers we have contacted a domain expert, to be the assessor of our system and she identified the correct and incorrect answers.

Definition QA systems are penalized for not retrieving vital concepts, and penalized for retrieving items that are not on the assessor's concept list at all, but should be neither penalized nor rewarded for retrieving a non-vital concept.

5.2.4 Experiment One: Normalization and Performance

Evaluation of *DefAmharicQA* is mainly for accuracy and recall of answers. Precision for definition QA is calculated as the number of correct concepts retrieved to the total number of concepts retrieved.

The performance of *DefAmharicQA* has been evaluated before and after document normalization and stemming. This evaluation shows us the significant effect of document normalization for performance. Out of the 20 questions the system doesn't retrieve a single vital nugget for seven questions. Consider Table 5.3 for comparison.

Q. No.	# vital nuggets retrieved	# non-vital nuggets retrieved	# vital nuggets from assessor's list	Allowance	Length (# of non-whitespaces)	Length - Allowance	Precision	Recall	F-measure
1	0	0	1	0	55	55	0.000	0.000	0.000
2	1	1	2	200	194	-6	1.000	0.500	0.667
3	0	0	2	50	12	-38	1.000	0.000	0.000
4	0	0	2	0	126	126	0.000	0.000	0.000
5	1	1	2	200	218	18	1.000	0.500	0.667
6	1	0	2	100	40	-60	1.000	0.500	0.667
7	2	1	2	300	85	-215	1.000	1.000	1.000
8	0	0	2	78	53	-25	1.000	0.000	0.000
9	2	0	3	200	72	-128	1.000	0.667	0.800
10	1	0	1	100	72	-28	1.000	1.000	1.000
11	2	0	2	200	167	-33	1.000	1.000	1.000
12	0	1	1	100	20	-80	1.000	0.000	0.000
13	1	0	2	100	177	77	0.565	0.500	0.531
14	0	0	2	0	139	139	0.000	0.000	0.000
15	1	0	1	100	102	2	0.980	1.000	0.990
16	1	1	2	200	175	-25	1.000	0.500	0.667
17	2	0	2	200	58	-142	1.000	1.000	1.000
18	0	0	2	0	71	71	1.000	0.000	0.000
19	2	1	3	300	150	-150	1.000	0.667	0.800
20	0	0	1	0	10	10	0.000	0.000	0.000
						Weighted Average	**0.78**	**0.44**	**0.49**

Table 5.3: Performance of DefAmharicQA before normalization

Q. No.	# vital nuggets retrieved	# non-vital nuggets retrieved	# vital nuggets from assessor's list	Allowance	Length (# of non-whitespaces)	Length - Allowance	Precision	Recall	F-measure
1	0	1	1	100	106	6	0.943	0.000	0.000
2	1	1	2	200	194	-6	1.000	0.500	0.667
3	0	0	2	50	16	-34	1.000	0.000	0.000
4	0	0	2	0	126	126	0.000	0.000	0.000
5	2	1	2	300	218	-82	1.000	1.000	1.000
6	1	0	2	100	40	-60	1.000	0.500	0.667
7	2	1	2	300	85	-215	1.000	1.000	1.000
8	0	0	2	78	108	30	0.722	0.000	0.000
9	2	0	3	200	72	-128	1.000	0.667	0.800
10	1	0	1	100	72	-28	1.000	1.000	1.000
11	2	0	2	200	167	-33	1.000	1.000	1.000
12	0	1	1	100	20	-80	1.000	0.000	0.000
13	1	0	2	100	177	77	0.565	0.500	0.531
14	0	1	2	100	255	155	0.392	0.000	0.000
15	1	0	1	100	102	2	0.980	1.000	0.990
16	1	1	2	200	175	-25	1.000	0.500	0.667
17	2	0	2	200	58	-142	1.000	1.000	1.000
18	2	0	2	200	83	-117	1.000	1.000	1.000
19	2	1	3	300	150	-150	1.000	0.667	0.800
20	0	0	1	0	10	10	0.000	0.000	0.000
						Weighted Average	**0.83**	**0.52**	**0.56**

Table 5.4: Performance of DefAmharicQA after normalization

Table 5.3 and Table 5.4 depict the performance of DefAmharicQA before normalization and after normalization is conducted respectively. The comparison is shown in Table 5.5.

Table 5.5: Summary of Normalization Vs Performance

Before normalization			**After normalization**		
Precision	Recall	F-measure	Precision	Recall	F-measure
0.78	0.44	0.49	0.83	0.52	0.56

From the above table it is inferred that document normalization has a significant effect on precision and recall of *DefAmharicQA*. In some questions, though the question processing module identifies the definiendum properly from the natural language questions couldn't match with the appropriate target word from the document because of the orthography differences. For instance the question "**ህግ ማለት ምን ማለት ነው**?" didn't get the appropriate definition even if the question processing module identify "**ሕግ**" as definiendum but the definition extraction module couldn't extract definition and the system respond No answer. As a result recall is penalized and precision reduced as compared to the normalized one.

5.2.5 Evaluation of definiendum identification

DefAmharicQA's question processing module has been evaluated for identification of the definiendum correctly as improper identification of the definiendum leads to incorrect result that affect the performance of the system. A hard pattern rule based technique is used to identify the definiendum. Other than the twenty performance testing questions, seventeen questions are used to evaluate the query processing module, of which the module identifies fifteen (88.24%) definiendums properly but it wrongly identifies two (11.76%) as definiendums.

5.2.6 Experiment two: Effect of Stemming on Effectiveness

The performance of *DefAmharicQA* has been evaluated before and after stemming. This evaluation shows us the impact of stemming on performance. Consider Table 5.6 for comparison.

Table 5.6: Effect of Stemming on Performance

Before stemming			**After stemming**		
Precision	Recall	F-measure	Precision	Recall	F-measure
0.83	0.52	0.56	0.856	0.730	0.788

As can be depicted from the Table 5.6, stemming increases the performance of *DefAmharicQA*.

5.2.7 Evaluation of *DefAmharicQA*

The final system performance experiment is conducted under the assumption that both nugget precision and nugget recalls are equally important ($i.e\ \beta = 1$).

The human assessor prepared all vital and non-vital nuggets of answers for each question manually. The system performance is tested based on the retrieved and not-retrieved number of nuggets for each question and cross checked with the assessor list of the ground truths.

As clearly stated in equation 1.1 allowances is calculated based on the returned number of vital and non-vital nuggets times hundred. (i.e. allowance is always a multiple of 100 based on the assumption that a nugget approximately have 100 non-whitespace characters). The non-white space characters of each nugget are counted using the built-in word count function of Microsoft word and recorded in the table below. Nugget precision is one if the returned non-white space characters are less than the allowance of that question otherwise it is calculated as shown in Equation 1.4. Accordingly nugget precision, nugget recall and F-measure are calculated.

The calculated average evaluation metrics are: nugget precision = 85.6%%, nugget recall = 73 % and F-measure = 78.8%

Table 5.7 clearly shows that normalized and stemmed based answering Amharic definition question technique outperforms the others (i.e. without normalization and stemming).

Table 5.7: Performance of *DefAmharicQA* for each questions

Q. No.	# vital nuggets retrieved	# non-vital nuggets retrieved	# vital nuggets from assessor's list	Allowance	Length (# of non-whitespaces)	Length -Allowance	Weight X Precision	Weight X Recall	Weighted F-measure
1	1	1	1	200	106	-94	1.000	1.000	1.000
2	1	1	2	200	194	-6	2.000	1.000	1.333
3	0	0	2	0	16	16	0.000	0.000	0.000
4	2	0	2	200	126	-74	2.000	2.000	2.000
5	2	1	2	300	218	-82	2.000	2.000	2.000
6	1	0	2	100	40	-60	2.000	1.000	1.333
7	2	1	2	300	85	-215	2.000	2.000	2.000
8	2	0	2	200	74	-126	2.000	2.000	2.000
9	2	0	3	200	72	-128	3.000	2.000	2.400
10	1	0	1	100	72	-28	1.000	1.000	1.000
11	2	0	2	200	167	-33	2.000	2.000	2.000
12	0	0	1	0	20	20	0.000	0.000	0.000
13	1	0	2	100	177	77	1.130	1.000	1.061
14	2	0	2	200	255	55	1.569	2.000	1.758
15	1	0	1	100	102	2	0.980	1.000	0.990
16	1	1	2	200	175	-25	2.000	1.000	1.333
17	2	0	2	200	58	-142	2.000	2.000	2.000
18	2	0	2	200	83	-117	2.000	2.000	2.000
19	2	1	3	300	150	-150	3.000	2.000	2.400
20	0	0	1	0	10	10	0.000	0.000	0.000
						Weighted Average	**0.856**	**0.730**	**0.788**

Table 5.8: Summary of performance of *DefAmharicQA*

Correct Answer	Wrong answer	No answer	Precision	Recall	F-measure ($\beta = 1$)
(85%)	(15%)	(0%)	0.856	0.730	0.788

5.2.8 Findings and Challenges

In this research, we attempted to design Amharic question answering for definitive questions. Experimental result shows that normalization and stemming increase the effectiveness of the system significantly. On the average, 78.8% F-measure is achieved.

Out of the 20 questions in the test collection 17 questions (85%) are answered correctly. Questions on which *DefAmharicQA* failed to answer correctly are found that two of the wrong answers are traced back to problems with definiendum extraction. If the definiendum is not correctly identified, then all subsequent modules have little chance of providing relevant nuggets. Moreover, *DefAmharicQA* failed to answer one question.

The DefAmharicQA system attains a precision of 85.6%, a recall of 73.0% and an F-measure of 78.8%. The evaluation is done directly by ten native Amharic speakers who presented two questions each to the system and judged by the assessors, legal expert, for the correctness of the answers (i.e. vital/ non-vital).

Whilst it is true that the techniques introduced in this study have performed well it is clear that there were three questions which the system unable to answer correctly due to the fact that the question analysis module missed the appropriate target word. Identification of the definiendum (target word) is a very challenging task and any error on identification of the target word penalizes the whole system. One of the questions the system did not answer correctly is "ቤተሰብ ማን ማንን ያጠቃልላል?" and it returned proclamation number only; due to the fact that the concept - description extractor component extract irrelevant description from the corpus. Lack of accessibility of linguistic resources such as Amharic

corpora basic NLP tools (tokenizers, stemmer and Amharic WordNet) make the whole process challenging. The related challenge we encountered is identifying and handling phrasal definiendums.

A definition question is not a fully formed question, like a factoid question; rather it is just the name of the thing the user wishes to define. For example *"ሕግ"* and *"የመረጃ መረብ"* are both definition questions. Just like factoid questions the target of the definition has to be analyzed to enable relevant documents to be located. With no guiding words, such as *ማን*, *የት*, and *መቼ*, in the targets to guide the analysis of the definiendum is a challenging task.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

The continuous growth and diversification of text information became a challenge to handle and make use of it, affecting both the designers and the users of information processing systems. The development of systems that assist users in finding relevant pieces of information across large text collections is a key task, because they transform a static set of stored text files into accessible and searchable knowledge. Researchers have proved that to a great extent, definition questions have become especially interesting in recent years, because of its number of submissions to search engines, about 25% of queries in real search engine logs are definition queries.

These days, it is very true that electronic Amharic documents are produced significantly. This research is a continuation of other attempts to develop a question answer system for the Amharic language. The ultimate objective of this research is to design Amharic question answer for definitive questions (*DefAmharicQA*).

6.1 Conclusion

A definition question answer system for Amharic language is developed. This research work attempted to identify the basic language specific issues in definition question answering.

Amharic legal documents are collected and questions are prepared to evaluate the performance of the system. The document is preprocessed which includes normalization and stemming to make it ready for indexing. The question analysis component which is the vital component of the system identifies the definiendum of the query posed by users to the system. The definition extractor is also another decisive component of the system.

For this study hard pattern matching is used to extract concept – description relations from the Amharic legal corpus and to identify definiendum from user's natural language question.

In this research we proposed a definitional Question Answering system for Amharic language. This system answers legal definition questions expressed in Amharic language for a closed legal

corpus. It is based on an approach which employs a little linguistic analysis and language understanding capability. *DefAmharicQA* identifies candidate definitions by using a set of lexical patterns.

DefAmharicQA system is evaluated using Amharic legal documents and 20 definitive questions. The evaluation shows that obtained results are very encouraging. DefAmharicQA succeeded in generating pertinent definitions for seventeen out of 20 definition questions. The system provided answer sets for 17 of the 20 questions (85%). For these 20 questions, 10 (50%) questions contained all the vital nuggets of the assessor list. The system achieved nugget recall of 73.0%, nugget precision of 85.6% and F-measure of 78.8%. We found extracting concept-description relationship from the corpora is the major language dependent component of the system in addition to query analysis.

Application of surface text pattern makes the system effective. From the research we inferred normalization and stemming increases the performance of Amharic question answering for definitive questions.

However, the system's effectiveness is highly affected by identification of the definiendum word/ phrase(s) and extraction of definitions from the corpus. The major reasons are definition and answer patterns are prepared manually, inappropriate descriptions are extracted for a concept from the corpus and the question analysis

6.2 Recommendation

Although, this research attempted to develop a closed domain definition question answer system with a promising F-measure value. The following issues must be dealt with to increase the effectiveness and efficiency of the system.

1. **Sequence mining algorithm**: An automatic sequence mining algorithms can be used to extract concept-description relationship from large corpuses.
2. **Amharic spell checker**: integrating Amharic spelling checker helps to improve performance as users usually make mistake while writing the query.

3. **Standard corpus**: having standard corpus to measure the performance of definition question answer system is essential.
4. Develop a definition QA system for other Ethiopic languages
5. Integrating with Amharic search engine
6. **Amharic WordNet**: the development of Amharic WordNet is very essential that includes word synonym, hyponym, and antonym so as to increase the recall ability of the system.
7. **Named Entity Extractor**: target extraction is a key, non-trivial capability critical to the success of a definition question answer system. Similarly, database lookup works only if the relevant definiendum are identified and indexed while preprocessing the corpus. Both of these issues point to the need for a more robust named-entity extractor, capable of handling specialized names (e.g., "**እሳቱ ተሰማ**", "**የሸዋወርቅ**", "**አትጠገብ**",…).
8. **NLP tools**: The improvement of the used NLP tools is another area needing much work: to improve the syntactical parser and, specially, the semantic analyzer (which is a quite open problem in the NLP community).
9. **Stemmer**: To enhance the Amharic definitive question answering system performance, further studies should be conducted to design an effective stemmer that can stem infixes for Amharic words since it is rich in morphology.
10. **Query expansion**: Future researches can also improve the performance of the system by exploring and integrating query expansion techniques.
11. **Domain experts**: to reduce the effect of judgment on categorization of vital and non-vital nuggets team of experts has to be participated in the evaluation. In this study only one legal expert is participated.
12. **Open-domain**: This research is conducted on closed domain corpora hence the same research can be extended to open domain Amharic definition questions.
13. Amharic question answering can be regarded as at its infant stage, therefore researchers can develop on other types (how, why, list etc) of QA to assist users so as to get precise and concise answer for their query.

References

A. Krogh, M. B. (1994). Hidden Markov Models in Computational Biology - Applications to Protein Modeling. *J. Molecular. Biology* 13 No.3, pp.27-39

Abdessamad Echihabi, U. H. (2003). Multiple-Engine Question Answering in TextMap. *Proceedings of the 12th Text REtrieval Conference.*

Alexander Panossian, G. W. (2007). Knowledge Bases in Medicine: a review. *Journal of Ethno pharmacology, Bulletin of the Medical Library Association* (2), 183-212.

Andrew, G. M. (2005). *Open-Domain Question Answering.* PhD Thesis, University of Sheffield, Sheffield.

Antonio Juarez-Gonzalez, A. T.-V.-C. (2006). INAOE at CLEF 2006: Experiments in Spanish question answering. *Working Notes for the CLEF 2006 Workshop.* Alicante, Spain.

Atelach, Alemu. (2002). *Automaic Sentence Parsing for Amharic Text an Experiment using Probabilistic Context Free Grammars. MSC thesis*, Addis Ababa University.

Aunimo, L. (2007). *Methods for Answer Extraction in Textual Question Answering.* University of Helsinki, Computer Science, Helsinki.

Baeza-Yates, B. R.-N. (1999). *Modern Information Retrieval.* Addison Wesley.

Baye, Yimam. (2000). *የአማርኛ ሰዋሰው፡ ት.መ.ማ.ማ.ድ.*

Bender, M. L. (1976). *The Ethiopian Writing System .* LOndon: Oxford University Press.

Bernardo Magnini, S. R. (2004). The Multiple Language Question Answering Track at CLEF 2003. In J. G. Carol Peters (Ed.), *Comparative Evaluation of Multilingual Information Access Systems. Fourth Workshop of the Cross-Language Evaluation Forum, CLEF 2003*, *3237.* Trondheim, Norway.

Brill, E. D. (2002). An Analysis of the AskMSR Question Answering System. *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing.*

Carlos Amaral, D. L. (May 2004). Design and implementation of a semantic search engine for Portuguese. *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, (pp. 247-250). Lisbon, Portugal.

Chua, H. C.-Y.-S. (2007, April). Soft Pattern Matching Models for Definitional Question Answering. *ACM Transactions on Information Systems , 25*.

Claudia Denicia-Carral, M. M.-y.-G.-P. (2012). *A Text Mining Approach for Definition Question Answering.* Spain: Language Technologies Group, Computer Science Department.

Cody Kwok, O. E. (2001). Scaling Question Answering to the Web . *10th international conference on World Wide Web*, (pp. 150-161). Hong Kong,.

Croft, X. L. (2002). Evaluating Question-Answering Techniques in Chinese. *The 10 th Text Retrieval Conference.* NIST .

Dan Moldovan, S. H. (2002). LCC Tools for Question Answering. *Proceedings of the 11th Text REtrieval Conference.*

Dan Moldovan, S. H. (2000). The Structure and Performance of an Open-domain Question Answering System. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, (pp. 563-570). Hon Kong.

Dang, E. M. (2011). Overview of the TREC 2005 question answering track. *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005).* Maryland, United States.

Doan-Nguyen Hai, L. K. (2004). The Problem of Precision in Restricted-Domain Question Answering. Some Proposed Methods of Improvement. *the ACL 2004 Workshop on Question Answering in Restricted Domains*, (pp. PP.8-15.). Barcelona, Spain.

Dominique Laurent, P. S. (2006, September). Cross lingual question answering using QRISTAL. *CLEF 2006* (Working Notes for the CLEF 2006 Workshop).

Drabek, E. F. (2000). Using Co-occurence Statistics as an Information. *Proceedings of Second Chinese Language Processing Workshop*, (pp. 22-28). Hong Kong.

Eduard Hovy, L. G.-Y. (2001). Towards Semantics-Based Answer Pinpointing. *Proceedings of the DARPA Human Language Technology Conference (HLT).* San Diego , CA.

Figueira, H. M. (2007). Priberam's Question Answering System in a Cross- Language Environment., *4730*, pp. 300-309.

Figueroa, A. G. (2010). *Finding Answers to Definition Questions on the Web.* A dissertation submitted to the Philosophy Faculty of Saarland University in partial fulfilment of the requirements for the degree of Doctor of Philosophy, University of Saarland, Department of Computational Linguistics and Phonetics.

Getahun, A. (1989). *የአማርኛ ሰዋሰው በቀላል አቀራረብ.*

Green, W. C. (1961). BASEBALL: An automatic question answer. *the western Joint Computer Conference*, (pp. 219-224).

Greenwood. (2002). *Proceedings of the 11th Text REtrieval (TREC 2002 Q&A System).* The University of Sheffield.

Greenwood, A. (2005). *Open-Domain Question Answering.* PhD Thesis, University of Sheffield, Sheffield.

Greenwood, M. A. (2005). *AnswerFinder: Question Answering from your Desktop.* University of Sheffield, Department of Computer Science , Regent Court, Portobello Road, Sheffield.

Hang Cui, M.-Y. K.-S. (2005). Generic Soft Pattern Models for Definitional Question Answering. *Text Retrieval Conference* . Singapore.

Haque, N. (2010). *A prototype framework for a Bangla question answering system using translation based on transliteration and table look-up as an interface for the medical domain.* MSc Thesis, University of Malta, Department of Computer Science and Artificial Intelligence.

Harabagiu, S. M. (2003). Answer Mining by Combining Extraction Techniques with Abductive Reasoning,. *In Proceedings of the TREC-2003 Conference, Language Computer*, (p. 375).

HARABAGIU, S. M. (2005). Employing two question answering systems in TREC-2005. *TREC.*

Harabagiu, T. S. (2008). *Advances in Open Domain Question Answering* (Vol. 32). The Netherlands: Springer.

Harman, E. M. (2001). In E. M. Harman (Ed.), *The Tenth Text REtrieval Conference* (pp. 500-250). NIST Special Publication, Department of Commerce, National Institute of Standards and Technology.

Hatcher, O. G. (2005). *Lucene in Action.* Bruce Park Avenue, Greenwich, CT 06830: Manning Publications Co.

Hirschman, L. &. (2001). Natural Language Question Answering: The View From Here. *Natural Language Engineering, 7(4).*

Hovy, D. R. (2002). Learning surface text patterns for a question answering system. *the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, (pp. 41-47). Philadelphia.

Hu, H. (2006). *A Study on Question Answering System Using Integrated Retrieval Method.* PhD Thesis, The University of Tokushima, Graduate School of Engineering, Tokushima.

Huang, G. T. (2004). Chinese question-answering system. *Journal of Computer Science and Technology , 19*.

Hui Yang, H. C.-Y.-S. (2003). QUALIFIER in TREC-12 QA Main Task. *In Proceedings of the 12th Text REtrieval Conference.*

Jianan, L. (2006). *An Intelligent FAQ Answering System Using a Combination of Statistic and Semantic IR Techniques.* Master Thesis, Mälardalen University, Computer Science and Electronics , Stocholm.

Jinxi Xu, A. L. (2003). Answering Definitional Questions. *TREC2003 QA at BBN.*

Kanaan, G. A.-S. (2009). A new question answering system for the Arabic language. . *American J Applied Sci. , 6*, 797-805.

Katz, B. (1997). From sentence processing to information access on the World Wide Web. *In Natural Language Processing for the World Wide Web* (pp. 77-94). AAAI Spring Symposium.

Kothari, C. (2004). *Research Methodology: Methods and Techniques.* India: New Age International Publishers.

Kwok, C. E. (2001). Scaling Question Answering to theWeb. *Proceedings of the 10th ACM World Wide Web conference.*

Lampert, A. (2004). *A Quick Introduction to Question Answering.* Australia.

Lin, B. K. (April 2004). Selectively using relations to improve precision in question answering. *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering.*

LIU, B. C. (2003). Mining topic-specific concepts and definitions on theWeb. *WWW*, (pp. 251–260).

Liu, J. (2007). An Intelligent FAQ Answering System Using a Combination of Statistic and Semantic IR Techniques. Department of Computer Science and Electronics at Mälardalen University.

M. Berndtsson, J. H. (2008). *Thesis Projects: A Guide for Students in Computer Science and Information Systems.* London: Springer-Verlag Limited.

Maria Varges, V. a. (2004). AQUA, A Knowledge-Based Architecture for a Question Answering System . *Tech Report Kmi-o4-15.* England: Knowledge media institute Milton Keynes.

Mark A. Greenwood, I. R. (2002). *Proceedings of the 11th Text REtrieval (TREC 2002 Q&A System).* The University of Sheffield.

Matthew, B. K. (2004). What Works Better for Question Answering: Stemming or Morphological Query Expansion? *In Proceedings of the Information Retrieval for Question Answering (IR4QA) SIGIR 2004*, (p. 7). Sheffield, England.

Maybury, M. T. (2004). *New Directions in Question Answering.* California: AAAI Press.

MCCALLUM, A. F. (2000). Maximum Entropy Markov Models for information extraction and segmentation . *ICML*, (pp. 591–598).

Michael Fleischman, E. H. (2003). *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics,.*

Michael Fleischman, E. H. (2003). Offline strategies for online question answering: Answering questions before they are asked. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics.*

Miller, G. A. (1995). *WordNet: A Lexical Database.* Communications of the ACM.

Mindaye et.al, M. S. (2010). The Need for Amharic WordNet.

Mitchell P. Marcus, B. S. (1993). Building a large annotated corpus of English:The penn treebank. In *Computational Linguistics* (p. 19(2):313{330).

Mohammad Reza Kangavari, S. G. (2008). Information Retrieval : Improving Question Answering Systems by Query Reformulation and Answer Validation. *World Academy of Science, Engineering and Technology*, *48.*

Moldovan, S. H. (2002). LCC Tools for Question Answering. *Proceedings of the 11th Text REtrieval Conference.*

Monz, C. (2003). *From Document Retrieval to Question Answering.* INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION. Amsterdam: ILLC.

Nega Alemayehu, (2002) Stemming of Amharic Words for Information Retrieval. *In Literary and Linguistic Computing. 17 No.1*, pp. 1-17. Oxford: Oxford University press.

Nico, S. (2005). *Pattern Learning and Knowledge Annotation for Question Answering.* PhD Thesis, Carnegie Mellon University, USA.

Omar Trigui, L. H. (2008). *DefArabicQA: Arabic Definition Question Answering System.* University of Sfax.

Phillips. (1960). *A question-answering routine, Artificial Intelligence Project.* Memo.

R. Simmons, S. K. (1963). Indexing and dependency logic for Answering English Questions. United States.

Radev, D. (2001). Answering what-is questions by virtual annotation. *Proceedings of the First International Conference on Human Language Technology Research* (pp. 1–5). Morristown, NJ: Association for Computational Linguistics.

Ravichandran D., a. H. (2002). Learning Surface Text Patterns for a Question Answering System. *Proceedings of the ACL-2002 Conferenc.* Philadelphia, USA.

Robert Gaizauskas, M. A. (2004). TREC 2004 Q&A Experiments. *The University of Sheffield's.*

ROSENFELD, R. (2000). Two decades of statistical language modeling: Where do we go from here. In R. ROSENFELD (Ed.), *IEEE 88*, *8.*

Rosso, O. T. (2010). An Automatic Definition Extraction in Arabic Language. *Text Retrieval Conference*, (pp. 240–247).

Roth, X. L. (2002). Learning Question Classifiers. *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02).* Taipei, Taiwan.

Saggion, M. A. (2004). *A Pattern Based Approach to Answering Factoid, List and Definition Questions.* University of Shefield, Department of Computer Science, Shefield.

Sanda Harabagiu, D. M. (2000). FALCON: Boosting Knowledge for Answer Engines. *the 9th Text REtrieval Conference.*

Sanda M. Harabagiu, D. I. (2001). The role of lexico-semantic feedback in open-domain textual questionanswering. *The 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, (pp. 274-281).

Schlaefer, N. (2007). Deploying Semantic Resources for Open Domain Question Answering. Germany: Universit¨at Karlsruhe (TH).

Schlaikjer, S. B.-G. (2003). A Hybrid Approach for Answering Definitional Questions. *In Proceedings of the 12th Text REtrieval Conference pp.75-84*

Seid, Yimam. (2009). *Amharic Question Answering System for factoid Questions.* MSc Thesis, Addis Ababa University, Department of Computer Science, Addis Ababa.

Skounakis, M. C. (2003). Hierarchical Hidden Markov Models for information extraction. *International Joint Conference on Artificial Intelligence*, *18th*, pp. 427–433.

Soubbotin, M. M. (2001). Patterns of Potential Answer Expressions as Clues to the Right Answers. *Proceedings of the 10th Text REtrieval Conference.*

Stacey M. Bailey, B. (2001). Providing QuestionCoverage Through Answer type Classification in Natural language Question Answering *(NLQA) Systems.* Georgetown University, Graduate School of Arts and Sciences, Washington, DC.

START. (2008). *The START Natural Language System. The START Natural Language Question Answering System.* Masachusets Institute of T echnology, Computer Science. .

Stephen E. Robertson, S. W.-B. (1999). Okapi. *the 8th Text REtrieval Conference.* TREC-8.

Swaroop. (2012). *A Byte of Python.* Swaroop C H.

Tessema, M. (2007). *Design and implementation of Amharic Search Engine.* A Thesis Submitted to the School of Graduate Studies of the Addis Ababa University in partial fulfillment for the Degree of Master of Science in Computer Science, Addis Ababa University, Addis Ababa.

Tessema, Mindaye (2009). The Need for Amharic WordNet.

Valentin Jijkoun, M. d. (2004). Information extraction for question answering: Improving recall through syntactic patterns. *Proceedings of the Computational Linguistics (COLING 2004).*

Voorhees, E. M. (2002). Overview of the TREC 2002 Question Answering Track. *Proceedings of the 11th Text REtrieval Conference.*

Voorhees, E. (2001). Natural Language Engineering. *The TREC question answering track*, (pp. 361-378).

Voorhees, E. (2003). Overview of the TREC 2003 question asnwering track. *In the proceedibgs of the Tewelfth Text REtrieval Conference.*

Wang, M. (2006). *A Survey of Answer Extraction Techniques in Factoid Question Answering.* Carnegie Mellon University, School of Computer Science. Association for Computational Linguistics.

Weischedel, J. X. (2003). Answering Definitional Questions. *In Proceedings of the 12th Text REtrieval Conference.*

Wesley Hildebrandt, B. K. (2008). Answering Definition Questions Using Multiple Knowledge Sources. Cambridge: MIT Computer Science and Artificial Intelligence Laboratory.

Willett, N. A. (2002). The Effectiveness of Stemming for Information Retrieval in Amharic. *Electronic Library and Information Systems , 37*, 254-259.

XU, J. W. (2004). Evaluation of an extraction-based approach to answering definitional questions. *Proceedings of SIGIR*, (pp. 418–424).

Zelalem, S. (2001). *Automatic Classification of Amharic News Items: The Case of Ethiopian News Agency.* MSc Thesis, Addis Ababa University, School of Information Studies for Africa, Addis Ababa.

Annex

Source Code:

```
# -*- coding: utf-8 -*-
import string
import codecs
import re
import os

#------------------------------------------------------------------------
# opening files
#------------------------------------------------------------------------
def openfiles(filename):
   try:
      f=codecs.open(filename,'r',encoding='utf-8')
   except IOError:
      print('ፋይሉን መክፈት አልተቻለም %s')%(filename)
   return f
#---------------------------
# definiendum
def definiendum(f):
   Pd=[]
   Def=[]
   global Disc
   D=[]
   L=[]
   words=[]
   try:
      ff=codecs.open('pdefn.txt','r',encoding='utf-8')
```

```
except IOError:
   print('ፋይሉን መክፈት አልተቻለም %s')%(filename)
for line in ff:
   if not line.strip():
      continue
   else:
      D=line.split(None)
      for a in range(len(D)):
         Pd.append(D[a])
ff.close()
for line in f:
   if not line.strip():
      continue
   else:
      L=line.split(None)
      for a in range(len(L)):
         words.append(L[a])
   t=''
   te=[]
   for i in range(len(words)):
      if i==1:
         if words[i] in Pd:
            t+=(words[i-1])
            Disc.append(line)
      if i==2:
         if words[i] in Pd:
            t+=(words[i-2])
            t+=(' ')
            t+=(words[i-1])
```

```
                Disc.append(line)
            if t!='':
                Def.append(t)
            t=''
        words=[]
    '''for i in range(len(Disc)):
        print (Disc[i])'''
    return Def

#--------------------------------------------------------------------------
# remove punctuation marks, symbols
#--------------------------------------------------------------------------
def puncremover(words):
    freewords=[]
    myre=re.compile(ur'[\ufeff\u1363\u1364\u1361\u1362\u201d\u2019÷›››‹“”]+',re.UNICODE)
    for a in range(len(words)):
        str1=''
        for b in range(len(words[a])):
            if words[a][b] in string.punctuation:
                continue
            if words[a][b] in string.digits:
                continue
            else:
                str1+=words[a][b]
        str1=myre.sub('',str1)
        if str1!='':
            freewords.append(str1)
    return freewords
```

```
# stemming (remove prefixes)
#--------------------------------------------------------------------------
def stem(stopfree):
   stemw=[]
   temp=''
   pre=u'የከለበ'
   flagpre=0
#prefix
   for i in range(len(stopfree)):
      if (len(stopfree[i]))>2:
         for k in range(len(pre)):
            if stopfree[i][0]==pre[k]:
               flagpre=1
               break
      if flagpre==0:
         temp=stopfree[i]
         stemw.append(temp)
         temp=''
      else:
         for j in range(len(stopfree[i])):
            if j>0:
               temp+=stopfree[i][j]
         stemw.append(temp)
         temp=''
         flagpre=0
   print ('wwwwwww')
   return stemw
```

```
#-------------------------------------------------------------------------
# stemming (remove suffixes)
#-------------------------------------------------------------------------
def stem2(stemw):
    stemw1=[]
    stemw2=[]
    suf=u'ዎች'
    temp=''
    flag=0 # for 'sadis'
    flagsuf1=0 # for ች
    flagsuf2=0 # for ዎች
# suffix
    for i in range(len(stemw)):
        if (len(stemw[i]))>3:
            if stemw[i][len(stemw[i])-1]==suf[1]:
                flagsuf1=1
                flag=1
            if stemw[i][len(stemw[i])-2]==suf[0]:
                flagsuf2=1
                flag=0
        if flagsuf1==0 and flagsuf2==0:
            temp=stemw[i]
            stemw1.append(temp)
            temp=''
        if flagsuf1==1:
            for j in range(len(stemw[i])):
                if flagsuf2==1:
                    if j<(len(stemw[i])-2):
                        temp+=stemw[i][j]
```

```
        else:
            if j<(len(stemw[i])-1):
                temp+=stemw[i][j]
    stemw1.append(temp)
    temp=''
    flagsuf1=0
    flagsuf2=0
str1=stemw1[i]
if flag==1:
    lis=u'ማከትኝውርልድህይቅምስብችንእዉዝጥ'
    sad=u'ንማኮከቶትኞኝዎውሮርሎልዶድሆህዮይቆቅሞምሶስቦብኞችኖንኡእዎዘዝጦጥ'
    temp=''
    check=0
    for b in range(len(str1)):
        for x in range(len(lis)):
            check=0
            if str1[b]==lis[x] and b==(len(str1)-1):
                temp+=str1[b]
                check=1
                break
        if check==0:
            for y in range(len(sad)):
                if str1[b]==sad[y]and b==(len(str1)-1):
                    temp+=sad[y+1]
                    break
            else:
                temp+=str1[b]

    stemw2.append(temp)
```

```
            flag=0
        else:
            stemw2.append(str1)
    return stemw2
Disc=[]
pos=[]
posst=[]
docname=[]
docname=os.listdir("D:\corpuss")
corp="D:\corpuss\ "
cop=corp.strip(None)
for i in range(len(docname)):
    filename=cop+docname[i]
    f=openfiles(filename)
    Def=definiendum(f)
    freewords=puncremover(Def)
    normwords=normalizer(freewords)
    stemw=stem(normwords)
    stemed=stem2(stemw)
    ind=codecs.open('sindex.txt','a',encoding='utf-8')
    for i in range(len(Def)):
        ind.write(stemed[i])
        ind.write('$')
        ind.write(Disc[i])
    ind.close()
    Disc=[]
for i in range(len(normwords)):
    print normwords(i)
```

Searching concept-description

```
# -*- coding: utf-8 -*-
import string
import codecs
import re
import os

def openfiles(filename):
   try:
      f=codecs.open(filename,'r',encoding='utf-8')
   except IOError:
      print('ፋይሉን መክፈት አልተቻለም %s')%(filename)
   return f
def usertext():
   utext=''
   utext=raw_input('እባክዎ ጥያቄዎን (እንዲተረጎምልዎ የሚፈልጉትን ቃል ያስገቡ)...')
   return utext
#--------------------------------------------------------------------------
# rempve punctuation marks, symbols and numbers
#--------------------------------------------------------------------------
def puncremover(query):
   freequery=[]
   myre=re.compile(ur'[\ufeff\u1363\u1364\u1361\u1362\u201d\u2019÷›››‹“”]+',re.UNICODE)
   for a in range(len(query)):
      str1=''
      for b in range(len(query[a])):
         if query[a][b] in string.punctuation:
            continue
         if query[a][b] in string.digits:
```

```
            continue
         else:
            str1+=query[a][b]
      str1=myre.sub('',str1)
      if str1!='':
         freequery.append(str1)
   return freequery

#-------------------------------------------------------------------------
# the function below normalizes users query based on the list of similar
# characters stored in list 'norm'.
#-------------------------------------------------------------------------
def normalizer(query):
   normquery=[]
   lis=u'አኡኢኤእኦሰሱሲሳሴስሶጸጹጺጻጼጽጾዉሀሁሂሄህሆ'

norm=u'ዐአዑኡዒኢዓኣአዔኤዕእዖኦሠሡሢሣሤሥሦሰፀፁጺጻጼጽፆጸውዉሐሀሑሁሒሂሄህሓሀሔሄሕሀሖሆኅኁኂኃኄኅኆሆኸኹኺኻኼኽኾሆ00'
   for a in range(len(query)):
      str1=query[a]
      temp=''
      for b in range(len(str1)):
         for x in range(len(lis)):
            flag=0
            if str1[b]==lis[x]:
               temp+=str1[b]
               flag=1
               break
         if flag==0:
```

```
            for y in range(len(norm)):
                if str1[b]==norm[y]: # if the character is in the list
                    temp+=norm[y+1] # use the standard character we set
                    break
            else:
                temp+=str1[b]

        normquery.append(temp)
    return normquery

def extract(normtext):
    display=''
    D=[] #to store definiendum and definition form the index file
    p=[] # to store possible pre definiendum words
    po=[] #to store possible post definiendum words
    Def=[] # to separately store definiendum from index file
    Disc=[] # to separately store definition from index file
    defn='' # to store extracted definiendum for users query
    try:
        ff=codecs.open('preq.txt','r',encoding='utf-8')
    except IOError:
        print('ፋይሉን መክፈት አልተቻለም %s')%(filename)
    for line in ff:
        if not line.strip():
            continue
        else:
            p=line.split(None)
    ff.close()
```

```
try:
   fff=codecs.open('postq.txt','r',encoding='utf-8')
except IOError:
   print('ፋይሉን መክፈት አልተቻለም %s')%(filename)
for line in fff:
   if not line.strip():
      continue
   else:
      po=line.split(None)
fff.close()
'''for i in range(len(po)):

   print(po[i])
   print (',,,,,,,,,,,,,,')'''
if len(normtext)==1:
   defn=normtext[0]
elif len(normtext)==2:
   if normtext[0] in p:
      defn=normtext[1]
   elif normtext[1] in po:
      defn=normtext[0]
   else:
      defn+=normtext[0]
      defn+=' '
      defn+=normtext[1]
         elif len(normtext)==3:
   if normtext[0] in p and normtext[2] in po:
      defn=normtext[1]
   elif not(normtext[0] in po) and normtext[2] in po and not(normtext[1] in po):
```

```
            defn+=normtext[0]
            defn+=' '
            defn+=normtext[1]
        elif normtext[0] in po and not(normtext[1] in po) and not(normtext[2] in po):
            defn+=normtext[1]
            defn+=' '
            defn+=normtext[2]
        elif normtext[1] in po and not(normtext[0] in po):
            defn=normtext[0]
    elif len(normtext)>3:
        if normtext[0] in p and normtext[2] in po:
            defn=normtext[1]
        elif not(normtext[0] in po) and normtext[2] in po and not(normtext[1] in po):
            defn+=normtext[0]
            defn+=' '
            defn+=normtext[1]
        elif normtext[0] in po and not(normtext[1] in po) and not(normtext[2] in po) and
normtext[3] in po:
            defn+=normtext[1]
            defn+=' '
            defn+=normtext[2]
        elif normtext[1] in po and not(normtext[0] in po):
            defn=normtext[0]

    f=openfiles('sindex.txt')
    for line in f:
        D=line.split('$')
        Def.append(D[0])
        Disc.append(D[1])
```

```
    for i in range(len(Def)):
        if defn==Def[i]:
            display=Disc[i]
            break
    return display
#-----------------------------------------------------------------
# The program starts here
#-----------------------------------------------------------------
terms=[]
print('-------------------------------------------------')
print('እንኳን ወደ ህግ ትርጉም ሰጪ ሲስተም በደህና መጡ!!')
choice=10 # to manage users' input
while choice!=0:
    y=0
    while y<5: # A loop to control exception.
        try:
            choice=int(raw_input('ለመጠቀም አንድን(1) ያስገቡ ለመውጣት ዜሮን(0) ያስገቡ...'))
            print('-------------------------------------------------')
            if choice>1 or choice<0:
                y+=1
                if y==5:
                    print'የተሳሳተ ቁጥር በተደጋጋሚ አስገብተዋል ... ደህና ይሁኑ'
                    break
                else:
                    print'የተሳሳተ ቁጥር አስገብተዋል'
                    continue
            y=6
        except ValueError:
            y+=1
```

```
        if y==5:
            print'የተሳሳተ ምርጫ በተደጋጋሚ አስገብተዋል ... ደህና ይሁኑ'
            break
        else:
            print'የተሳሳተ ፊደል አስገብተዋል'
            continue
    if y==5:
        break # Exit application if wrong choice entered 5 times.
    if choice==0:
        print'ደህና ይሁኑ!!'
        break
    else:
        # calling functions
        utext=usertext()
        terms=utext.split(None)
        puncfree=puncremover(terms)
        normtext=normalizer(puncfree)
        '''for i in range (len (normtext)):
            print (normtext[i])'''
        display=extract(normtext)
        if display=='':
            print '\nየተሳሳተ ጥያቄ አስገብትዋል ወይም ጥያቄ አላስገቡም \n'
        else:
            print(display)
```

Declaration

I declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source materials used for this thesis have been duly acknowledged.

_____

Wondwossen Teshome

Date: 14th June 2013

This thesis has been submitted for examination with my approval as University advisor.

_____

Martha Yifiru (Ph.D)

Date: 14th June 2013