ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

LETEYEQ (ልጠየቅ)-A Web Based Amharic Question Answering System for Factoid Questions Using Machine Learning Approach

By: Desalegn Abebaw Zeleke

Advisor: Mulugeta Libsie (PhD)

A THESIS SUBMITTED TO

THE SCHOOL OF GRADUATE STUDEIS OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE

March 2013

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

LETEYEQ (ልጠየቅ)- A Web Based Amharic Question Answering System for Factoid Questions Using Machine Learning Approach

By: Desalegn Abebaw Zeleke

ADVISOR: Mulugeta Libsie (PhD)

APPROVED BY

EXAMINING BOARD:

1. Dr. Mulugeta Libsie, Advisor _____
2. _____ _____
3. _____ _____

Dedication

To my teachers, who gave me lessons in life and in school from very childhood to this level. It is because of you that I am who I am today; I owe you a lot.

Acknowledgements

My **God**, I know my words wouldn‟t add to your grace, but thank you for being with me during the desperate and joyful times. **Saint Virgin Mary,** I know how you helped me in all my life, thankyou. I would like to thank my thesis advisor **Dr. Mulugeta Libsie** for your guidance and encouragement in making this work a reality. **Seid Muhie,** it wouldn‟t be possible for me to accomplish this thesis work without your unconditional support in explaining how your first Amharic question answering system, TETEYEQ, was made. **Tessema Mindaye,** thankyou for your support in explaining me how your Habesha Search Engine works. **My class mates,** I had the wonderful time of my entire school life because of the very friendly and helping atmosphere you offered me. **Ethiopian News Agency (ENA)**, thank you for your cooperation in supplying me the news corpus. My gratitude finally goes to my family in general for encouraging me and to my sisters **Meskerem Abebaw** and **Elshaday Bayu** in particular for giving me your hands in manually preparing training questions.

Table of Contents

List of Tables

List of Figures

Acronyms

AWBQA	Amharic Web Based Question Answering
AEP	Answer Extraction Pattern
AI	Artificial Intelligence
API	Application Programming Interface
DF	Document Frequency
IDF	Inverse Document Frequency
IR	Information Retrieval
LIM	Language Identification Module
ML	Machine Learning
NE	Named Entity
NER	Named Entity Recognizer
NLP	Natural Language Processing
OCR	Optical Character Recognition
POS	Part Of Speech
QA	Question Answering
SPI	Service Provider Interface
SVM	Support Vector Machine
TF	Term Frequency
TREC	Text REtrieval Conference
UDHR	Universal Declaration for Human Rights
URL	Universal Resource Locator

Abstract

When users need for a certain fact and try requesting search engines for it, they get back a bunch of addresses and snippets which are „related" to their need and it is up to the users to decide which address to choose expecting that the requested fact could be found there. Opening the address could present the user with lots of pages of information and it is the user's duty to go through the information and extract the actual fact. But given a collection of documents, a Question Answering system attempts to retrieve correct answers to questions posed in natural language. Hence question answering relieves users from the task of digging the information from related pages.

There are different types of questions like definition, list, acronyms, true/false, and factoid types. Most of the question answering systems have three major components, question analysis, document (passage) retrieval, and answer extraction.

For languages like English, many question answering systems are available which are designed in different approaches. But in the case of Amharic, Seid Muhie's [4] TETEYEQ is a pioneer work designed to answer Amharic factoid questions. It aims to answer four kinds of Amharic factoid questions namely the „Person", „Place", „Time", and „Quantity" question types. It was designed by employing a rule based approach in question analysis component by manually writing rules to classify questions into one of these four question types resulting in an accuracy of classifying 86.9% of the questions correctly. It was also using manually collected Amharic documents as a search space and the reported overall system performance was 72%.

We have designed a similar system for answering the four kinds of Amharic factoid questions using a machine learning approach than the rule based one by employing the known machine learning based classification algorithm, support vector machine (SVM). By doing so, we attained an accuracy of 94.2% in question classification which outperforms the rule based question classification in TETEYEQ.

We have integrated a web crawler by customizing the open source JSpider crawler to automatically gather Amharic documents from the web in preparing the search space. The downloaded Amharic documents are then indexed by the open source tool, Lucene indexer, to facilitate the document retrieval process. Hence, our system has two major parts, the search engine part (crawler and indexer) and the question answering part. Besides, our system is designed to be a web based system for interacting with the end users on the web.

By employing the machine learning algorithm in question classification and adopting answer extraction techniques used in TETEYEQ, we have achieved 77% overall system performance which is better than that of TETEYEQ's.

In the absence of basic natural language processing (NLP) tools like part of speech (POS) tagger and named entity recognizer (NER) for the Amharic language, both TETEYEQ and our system have achieved a considerable performance which would be boosted up by the addition of such NLP tools in the future.

Key Words: Amharic Factoid Question Answering, SVM based question classification, Answer Extraction, Web based Amharic Question Answering.

Chapter One - Web Based Amharic Question Answering System

1.1 Introduction

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) and computational Linguistics, with an emphasis on the role of knowledge representations, which is for representations of our knowledge of the world in order to understand human language with computers. NLP is the use of computers to process written and spoken language for some practical, useful purpose such as: to translate languages, to carry on conversations with machines, so as to get advice about and so on. The goal of NLP is to design and build software that will analyze, understand, and generate languages that humans use naturally, so that eventually we will be able to address our computer as though we were addressing another person.

Some of the major tasks in NLP include: Optical Character Recognition (OCR), Speech Recognition, Part-of-Speech Tagging and Question Answering.

Question Answering (QA) is the task of automatically answering a question posed in natural language. QA research attempts to deal with a wide range of question types including: fact, list, definition, How, Why, hypothetical, semantically constrained, and cross-lingual questions. Search collections vary from small local document collections, to internal organization documents, to compiled newswire reports, to the World Wide Web [2].

The traditional search engines have a shortcoming for the concise and complete retrieval of information. Commonly, search engines return the relevant, even most of the times irrelevant links or document lists to the search keywords which are excessive that users need more efforts to acquire the needed information, may be after reading a number of pages for a longer time. Users prefer to have a tool for asking natural language like questions and expect precise answers for their questions rather than having list of web addresses which contain information related to the question. This cannot be done in traditional search engines. Therefore, the need for a Question-Answering tool is vital.

A common strategy in the design of Question Answering is to divide the entire task into a pipeline of quasi-independent tasks, such as question analysis, document retrieval, and answer extraction.

An effort has been made to design a Question Answering tool in many languages. The popular web based Question Answering System for English Language is available online [3]. A similar effort has been made to design an Amharic Question Answering System for factoid questions by Seid Muhie [4]. This work was done using a rule based approach to analyze natural language questions.

Prior implementations of language processing tasks typically involved the direct hand coding of large sets of rules. Modern approaches to NLP are grounded in Machine Learning (ML). This approach uses learning algorithms often grounded in statistical inference to automatically learn such rules through the analysis of large *corpora* of typical real world examples as a training set.

Systems based on machine-learning algorithms have many advantages over hand-produced rules [1]:

- The learning procedures used during machine learning automatically focus on the most common case, whereas when writing rules by hand it is often not obvious at all where the effort should be directed.
- Automatic learning procedures can make use of statistical inference algorithms to produce models that are robust to unfamiliar input and to erroneous input whereas it would be extremely difficult and error-prone to do so in the rule-based approach.
- Systems based on automatically learning the rules can be made more accurate simply by supplying more input data. However, the rule based can only be made more accurate by increasing the complexity of the rules, which is a much more difficult task.

1.2 Statement of the Problem

In a web based question answering system, questions are given in a search engine like interface in natural language form and the question type and the expected answer type should be identified by the question analyzer component. Then, the web documents having a potential answer to the question will be retrieved by the document retrieval component. This component is working like search engines to return the web addresses which have the potential to contain an answer to the given question. Finally, the answer extractor component will perform the extraction of appropriate answers from the candidates returned by the document retrieval component.

A similar research has been conducted by Seid Muhie [4]. The researcher used hand crafted rules to identify question and answer types. As it was the first effort made to design an Amharic question answering tool, this work was mainly designed to take pre formatted Amharic documents as a source.

Questions of type ***where, who, when, which, yes/no, true/false, name-of, etc.*** are kinds of factoid questions. As an example, "what is a Virus?" is a definition question whereas "**Is Amharic the national language of Ethiopia?**" is a true/false question [6]. Factoid questions need very brief and short lined answers. Users expect very brief answers for factoid questions. For the question "What is the name of the capital city of Ethiopia?", the user is seeking for a fact (the name of a city) therefore, the system"s response to the user is expected to be "Addis Ababa" than supplying a list of available web sites with a content related to the question (like the way search engines do).

In this research, we will investigate a machine learning approach to Amharic question answering. The advantage of a machine learning approach is that it is more adaptable, robust, flexible, and maintainable. There is no need for a human to manually engineer a set of handcrafted rules and continuously improve or maintain the set of rules. Rather, the computer will be trained with a large amount of training set and learn how to identify question types and expected answer types.

Beyond the above advantages, the question answering system designed with a machine learning approach has showed a competitive accuracy with the one designed with rule based technique for English language as in the case of [5]. It is also suggested in the future work section of [4] that despite the requirement of large training set, the machine learning approach is believed to show better performance in Amharic Question Answering.

The first Amharic Question Answering system designed by Seid Muhie [4] was experimented on manually collected and pre-formatted documents from the web. Hence, the system was a desktop application designed to take offline documents rather than directly applying question answering on the web itself.

Due to the richness of the web and the increase in availability of Amharic documents on the web, it would be more feasible to design a web based question answering tool for Amharic factoid questions than to design an offline based desktop application. This research is targeted towards designing a web based question answering tool for Amharic factoid questions to help people find out exact answers for their factoid questions by being on the web.

The first question answering tool for Amharic factoid questions [4] was a great effort made towards serving users with their needs for a fact written in Amharic documents (offline). There is also an Amharic search engine developed by Tessema Mindaye [7] and also there are freely available open source web crawlers, like JSpider to download web documents from a specified address, which can be used to build up our repository that we will use for document retrieval. By improving the performance of the question answering tool (with the employment of a machine learning approach to its design) and the modification or reconstruction of an Amharic search engine to suit our need, it will be a great contribution to users seeking for a fact from Amharic online documents.

1.3 Objectives

The general objective of this research work will be to develop a web based Amharic Question Answering System for factoid questions using a machine learning approach.

The specific objectives of this research work will be:

a. To analyze question and answer patterns.

b. To study machine learning algorithms suitable for the system.

c. To Train the machine with Amharic factoid question so that the machine would learn to analyze questions.

d. To develop an algorithm for web based Amharic Factoid Question Answering system.

e. To develop a suitable architecture of web based Amharic factoid question answering.

f. To develop a prototype for the new system.

g. Investigate the efficiency of our system against the existing question answering system in [4].

1.4 Scope and Limitation

Naturally, Question answering is a very complex and rigorous task which needs understanding of natural language techniques. A full-fledged QA system will require a number of Natural language processing tools such as Sentences parser, chunker, Part of Speech (POS) tagger, Stemmer, Named Entity Recognizer (NER) and so on [4]. Even though some of the NLP tools have been developed by some researchers, they are not publicly available for integrating with our new system. Having these limitations in mind, our scope will be:

a. Question pattern analyzing for factoid questions by following a machine learning approach (preparing a large training set of Amharic questions for this task).

b. Amharic Documents downlading using a suitable search engine.

c. Answer pattern analyzing.

d. Answering only "ማን","የት","ስንት" and "መቼ" type of Amharic questions.

Most of the Amharic NLP tools are done as part of academic demand by MSc students and are not publicly available. A full-fledged QA system needs to have Named Entity Recognizer, Part of speech tagger, stemmer and so on. In addition, the web based question answering tool will require a search engine. An Amharic search engine is designed by different researchers earlier. If the search engines and the above mentioned language tools are not easily and openly available for the research purpose, we will use open source crawlers.

1.5 Methodology

The major task in this research is collecting/preparing huge amount of training set in the form of factoid questions to train the machine in the question analysis phase. For this purpose, appropriate machine learning algorithms and tools would be employed.

We will conduct literature review on QA systems done using machine learning approach in other languages like English. In addition, open source Machine learning based natural language toolkits will be considered for the system. The first web based English question answering system, START [3], can be a good reference in developing our System. In preparing the training data set, as it requires manual collection of Amharic factoid questions, we will look for the support of other people.

In comparing the performance of the new system, we will use the already developed question answering system for Amharic [4] as the only candidate reference. Therefore, the performance of our system will be measured against the performance of this rule based system developed earlier.

The web based Amharic question answering system using machine learning approach will have so many components and developmental stages. At one hand there will be different tasks related to linguistics and on the other hand it will have different computational tasks. The linguistic related task comprises of studying Amharic grammatical structure, tokenizing, Named Entity recognizing, Question Pattern Analysis, Answer Pattern Analysis, Answer Ranking, Answer Extraction and so on. Computational related task includes selecting suitable programming language, developing algorithms for the different linguistic tasks, methodology and techniques employed in searching Amharic texts, selecting better machine learning algorithms for training the question analyzer and so on. The system will have three major steps: (i) question analysis, (ii) document retrieval with a search engine component, and (iii) answer extraction. The question analysis tool will be designed using a machine learning approach.

The system‟s performance will be measured and compared against the performance of the Amharic question answering system designed using rule based approach which was designed by Seid Muhie [4]. The machine learning approach usually needs large amount of training set than the rule based approach. Due to this, a big effort will be exerted in manually preparing the training and test sets for the system.

1.6 Application of Results

Question answering systems are the futures of search engines. The web based Amharic question answering system will be used to serve end users in finding short answers for Amharic factoid questions from documents found on the web. The system can also be customized to serve as a

help desk for users seeking to get service from a certain company by simply narrowing down the search space into the company‟s web site designed for help purposes.

1.7 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents literature review in which different concepts related to the thesis are presented. Chapter 3 is about works somehow related to our work which are done by other researchers in Amharic and other languages. Chapter 4 deals with the design of our system, from the general architecture of our system to discussions of basic components and their interaction in the system. Chapter 5 is about the detail design (implementation) of the system. It discusses the algorithms we used in achieving the goal of every component in the system supported with real examples. Chapter 6 deals with the experiments done in every component and the results achieved together with explanations of how such results happen. Chapter 7 winds up our work by presenting a conclusion and future works recommendation for the improvement of the system.

Chapter Two - Literature Review

2.1 Information Retrieval (IR) and Question Answering (QA)

Information Retrieval is the area of study concerned with searching for documents, for information within documents, and for metadata about documents, as well as that of searching structured storage, relational databases, and the World Wide Web. The most common applications of Information Retrieval are web search engines [28].

Search engines like Google provide an interface for users to type their query and reply to their information needs by taking into consideration every term (word) supplied by the user. The main task of such search engines is to provide ranked web addresses, together with a statement (statements) from the pages that match with the user query terms. Here, the user needs to choose relevant addresses and visit them for getting the actual information required. Such IR systems are not suitable for users seeking short and precise answers without being confronted to a collection of web addresses which might contain the expected answers to their questions.

Question Answering Systems are designed to take users‟ questions expressed in their natural language and reply with expected answers. Here the user is free to ask questions as if he/she is communicating with a human being. Natural language Questions like “What is the capital city of Ethiopia?” would be answered by Question Answering Systems concisely than that of the IR systems.

In information retrieval, queries tend to be more general and lengthy while in QA, the queries should be specific and shorter in number of query words. Responses from IR systems are collection of web addresses that are expected to satisfy user‟s need whereas, in QA systems, the response is precise information which is believed to be the answer for the user‟s question.

Despite these differences, most QA systems are backed by IR systems. Question Answering systems contain Information Retrieval systems during the process of retrieving relevant documents which are believed to contain information related to the user‟s need.

There are different Question Answering Systems which are designed to answer factoid questions, reading comprehension questions, list, why, and how questions. Recent successes have been reported in a series of question-answering evaluations that started in 1999 as part of the Text Retrieval Conference (TREC) [28].

Question answering has many applications. We can subdivide these applications based on the source of the answers. The sources can be structured data (databases), semi-structured data (for example, comment fields in databases) or free text. We can further distinguish among search over a fixed set of collections, as used in TREC (particularly useful for evaluation); search over

the Web, search over a collection or book, e.g., an encyclopedia or search over a single text, as done for reading comprehension evaluations. We can also distinguish between domain-independent question answering systems (systems designed to answer general questions in all domains) and domain specific systems (systems designed to answer questions generated only within a certain domain like medicine, chemistry, and so on).

A Question Answering System has been designed for languages like English, Arabic, Chinese and the likes. A system designed to answer questions raised in one language cannot help in answering questions from other languages since a question answering system needs a language dependent processing.

2.2 Search Engines

Search Engines, including the Amharic Search Engine [7], have three major components; The Crawler, Indexer, and Query Engine components.

The Web Crawler is a component used to interact with the World Wide Web for crawling (downloading) web pages into a local repository (Pages Directory) and the Indexer is a component used to index downloaded web pages and store the index into an index directory so as to make them ready for use by the index searcher. The query engine gives the user an interface to enter queries and returns the final results composed of the relevant documents related to the given query.

2.2.1 Crawler

Web crawlers are programs that exploit the graph structure of the Web to move from page to page [20].

Web crawling is the core component of search engines to prepare the source from which the engine searches for related pages to users‟ queries. Search engines frequently use web crawlers to collect information about what is available on public web pages. Their primary purpose is to collect data so that when Internet surfers enter a search term on their site, they can quickly provide the surfer with relevant web sites. Linguists may use a web crawler to perform a textual analysis; that is, they may explore the Internet to determine what words are commonly used today. Market researchers may use a web crawler to determine and assess trends in a given market.

Web based Question answering systems, like ours, also highly rely on the functionality of web crawlers.

A Web crawler is one type of software agent that is used to visit web pages on behalf of people. It does in the same way as we browse the web just by starting from a certain initial page and following links in that page to go to other related pages.

In general, it starts with a list of URLs to visit, called the *seeds*. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the *crawl frontier*. URLs from the frontier are recursively visited according to a set of policies [34].

While navigating from one page to the other, we can perform additional tasks. For language dependent crawling, we can incorporate a method to identify whether the contents in the just visited site are of a certain language or not and decide to download it or discard it. This task is handled by a separate program other than the web crawlers, called language (script) identifiers.

The method of language (script) identification, while crawling, adds an overhead to the crawling task and potentially degrades the speed of crawling. Another alternative is to supply seed URLs to the crawler and let the crawler download whatever pages it visits following these seeds and do the identification later on or at the same time to be done on the repository where the pages are downloaded and stored.

Downloaded and filtered documents should be managed in a certain way to facilitate future searching. This is the duty of indexers. The downloaded pages will be used by the indexer component so that each term in the web pages gets indexed and stored in an index file for facilitating the upcoming document retrieval activity.

For enriching our web based Amharic question answering system, the crawling process should always be running in the background and keep on downloading new web pages. Whereas, the language (script) identification of downloaded pages and indexing can be performed periodically to filter pages with the Amharic contents and add the filtered pages into the existing index file so that the new pages can be considered as a source in the coming document retrievals.

2.2.2 Indexer

The Indexer component is the component that creates the representation of the document collection that was collected by the crawler. For a language dependent search engine, crawled pages will pass through the language‟s analyzer before indexing. The analyzer includes components for tokenizing, stop words removing, stemming, and other language dependent tasks. By the indexer, words in every crawled page are stored in an index file in a format suitable for searching by the query engine component when the user poses a query to the search engine.

2.2.3 Query Engine

The query engine provides the interface between the search index, the user, and the web. The query engine retrieves from the search index information about potentially relevant web pages that match the keywords in the user query, and in the second step a ranking of the results is produced, from the most relevant to the least relevant one.

Once the result is produced, the query engine sends the results list to the search interface, which displays the results on the user's screen. The user interface provides the look and feel of the search engine, allowing the user to submit queries, browse the results, and click on chosen web pages for further browsing.

2.3 Lucene

Lucene is an open source, Jakarta project used to build and search indexes. Lucene can index any text-based information we like and then find it later based on various search criteria. Although Lucene only works with text, there are other add-ons to Lucene that allow us to index Word documents, PDF files, XML, or HTML pages. It provides a basic framework that we can use to build full-featured search into our application. Lucene is specifically an Application Programming Interface (API), not an application. This means that all the hard parts have been done, but the easy programming has been left to us to develop towards our application needs. The following explanation is partly adopted from the Lucene tutorial in [33].

Let's take a look at the key classes that are useful to build a search engine.

- ***Document*** - The *Document* class represents a document in Lucene. We index *Document* objects and get *Document* objects back when we do a search.
- ***Field*** - The *Field* class represents a section of a *Document*. The *Field* object will contain a name for the section and the actual data. For example, („URL", „www.aau.edu.et") is one *Field* object for a *Document* representing the Addis Ababa University web page.
- ***Analyzer*** - The *Analyzer* class is an abstract class that is used to provide an interface that will take a *Document* and turn it into tokens that can be indexed. There are several useful implementations of this class but the most commonly used is the *StandardAnalyzer* class.
- ***IndexWriter*** - The *IndexWriter* class is used to create and maintain indexes. We pump data into the Index, and then do searches on the Index to get results out. To build the Index, we use an *IndexWriter* object. Document objects are stored in the Index, but they have to be put into the Index at some point. We have to select what data to enter in, and convert them into Documents. We read in each data file (from the downloaded and filtered pages), instantiate a Document for it, break down the data into chunks (like url, title, and content of the page) and store the chunks in the Document as *Field* objects (a

name/value pair). When we are done building a Document, we write it to the Index using the IndexWriter.

- ***IndexSearcher*** - The *IndexSearcher* class is used to search through an index which was created by the *IndexWriter* .
- ***QueryParser*** - The *QueryParser* class is used to build a parser that can search through an index. Queries can be quite complicated, so Lucene includes a tool to help generate Query objects, called a *QueryParser*. The QueryParser takes a query string, much like what we would put into an Internet search engine, and generates a Query object.
- ***Query*** - The *Query* class is an abstract class that contains the search criteria created by the *QueryParser*. The search itself is a *Query* object, which we pass into IndexSearcher.search(). IndexSearcher.search() returns a *Hits* object, which contains a Vector of *Document* objects.
- ***Hits*** - The *Hits* class contains the *Document* objects that are returned by running the *Query* object against the index.

Generally, Lucene handles the indexing, searching and retrieving, but it doesn't handle:

- managing the process (instantiating the objects and hooking them together, both for indexing and for searching)
- selecting the data files
- parsing the data files
- getting the search string from the user
- displaying the search results to the user

2.4 General Architecture of Question Answering Systems

Functionally, most question answering systems today can be decomposed into four major components as **question analysis, document retrieval, passage retrieval, and answer extraction** [8]. The question analysis component classifies user questions by the expected semantic type of the answer, e.g., the expected answer type of "*When did Emperor Tewodros came to power?*" is time. In addition, it is responsible for formulating one or more queries targeted at a particular document retriever; these queries are used to find a set of potentially relevant documents from the corpus. From these documents, the passage retrieval component selects a handful of paragraph-sized fragments. In some systems, however, document and passage retrieval are performed simultaneously. Finally, the answer extraction component

searches the passages for the answer to the question, for example, by finding named-entities that match the expected answer type.

Figure 2.1 shows a pipelined architecture of a question answering system taken from [12]. The Figure outlines the major components in a Question Answering system. The functionalities of these components will be discussed next.

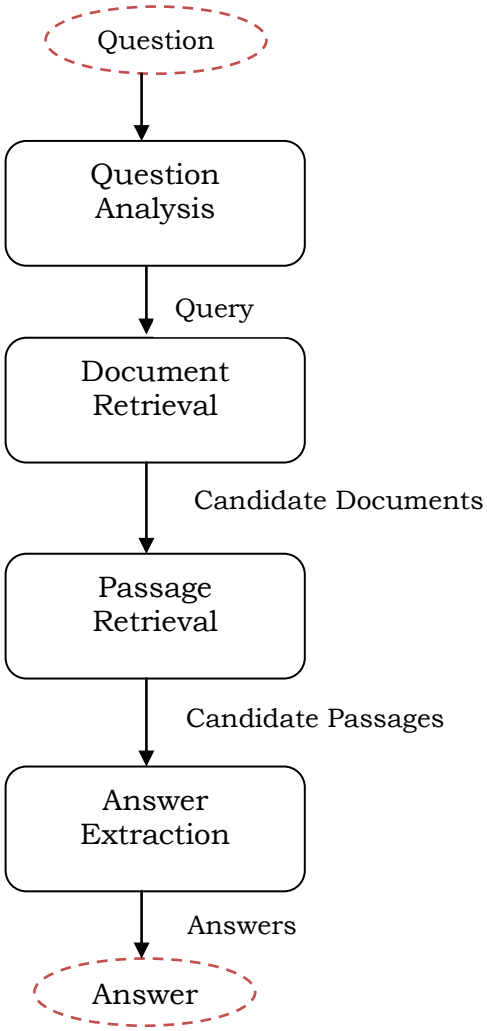

Figure 2.1: General architecture of question answering systems

Let‟s have a look at the basic question answering components in their order of use.

2.4.1 Question Analysis

This is the first component in which the system starts to analyze the natural language question posed by the user right from the input window in a web browser. The natural language question input by the user needs to be analyzed into whatever form or forms are needed by subsequent parts of the system. Question Analysis is the most important component of question answering systems. The following three key tasks are performed in the question analysis stage:

- Automatic Question Classification
- Illustration of the expected answer type from the type of question identified.
- Construction of proper query for the IR component of the QA system (i.e., for the document retrieval).

Failing to analyze the question will make the entire system to fail to answer the question correctly as the document retrieval will end up in retrieving wrong documents and the wrong identification of the question type will result in a wrong illustration of expected answer type.

In the coming topics, we will discuss the above sub tasks done in the question analysis component of the question answering system, question classification, expected answer types, and query construction.

2.4.1.1 Question Classification

Questions can be broadly classified as Questions of type Place, Person (Entity), Term Definition, Quantity, Listing, Explanation, True/False, Time, Choice, and so on. Person, Time, Place, and Quantity question types are the focus of this study. Based on the question type, an expected answer type will be identified. Therefore, the question analysis component of a QA system will play a great role in determining question types and identifying answer types.

Question Classification is concerned with assigning semantic classes to questions. This semantic classification can be used to reduce the search space of possible answers, i.e., if we can determine that the question **"Who is the Ethiopian Prime Minister?"** belongs to the semantic category Person, then we only need to look for instances of type Person as possible answers.

In [4], the question classification was done by writing hand crafted rules (regular expressions) so as to classify the given question in to one of the above question types.

Question Classification Approaches

According to [9], the approaches to question classification can be classified into the following three main groups: rule-based, machine learning based, and language modeling based.

In the rule based approach, hand-written grammar rules and a set of regular expressions are employed to parse a question and to determine the answer type. With this approach the researches have faced the following limitations:

- Hand-writing classification rule is a difficult and time-consuming process.
- Hand-written rules have limited coverage and are fairly complicated to broaden the scope of answer categories to include more detailed ones.
- In order to adopt a new taxonomy, many previously prepared rules have to be modified or completely rewritten.

Considering these limitations, the majority of systems that use hand-written rules are bound to use a limited number of question type categories. Consequently, question category information is limited to its use, which as previously described, influences the performance of the whole QA system.

In the machine learning approach, expert knowledge is replaced by a sufficiently large set of labeled questions. Using this collection, a classifier is trained in a supervised manner. Possible choices of classifiers include but are not limited to: Neural Network, Naive Bayes, Decision Tree and Support Vector Machines. The machine learning approach addresses many limitations of the rule-based method, which were presented above. The advantages include:

- Short creation time.
- Classifier is created automatically; no classification rules need to be provided by hand.
- Broader coverage; can be obtained by providing new training examples.
- If required, the classifier can be flexibly reconstructed (retrained) to fit to a new taxonomy.

At present, the results achieved using the machine learning approach represent a state of the art in question classification [9].

In the case of language model based question classification, one language model for every class of questions is prepared based on the training data set. To classify a question, the probability of

generating it is calculated for each class based on its language model, and the highest probability determines the classification.

In the question classification task, a language model is prepared for each category C of sample questions. When a new question Q comes, the probability P(Q|C) for each C is calculated and the one with the highest probability is picked as the class for the new question Q.

The language model based question classification takes advantage of flexibility over the rule based question classification as in the rule based, it needs to modify hard coded rules to handle new cases while the language model can be automatically maintained [29].

Support Vector Machines (SVM) Classifier

Support Vector Machines (SVM) were developed by Vapnik [21] in 1995 based on the Structural Risk Minimization principle from statistical learning theory.

SVM is a supervised learning algorithm typically used for classification problems like prediction, text categorization, handwritten character recognition, image classification, facial expression classification, and so on. This means, if we have some sets of things classified but we know nothing about how we classified it, or we don't know the rules used for classification and when a new data comes, SVM can predict which set the data should belong to. The function which will be used to separate the two classes is induced from available training examples [22].

SVM is basically a binary classifier. It takes a set of input data and predicts, for each given input, which of two possible classes forms the output, making it a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

There are other supervised learning algorithms like Naïve Bayes, Neural Networks, and Decision Trees. But for text classification purposes, SVM is proved to give better performance [11, 24, 37].

2.4.1.2 Expected Answer Types

The expected answer types are similar to the question types but these are the types expected from the candidate answers for the given question. The type for the candidate answers is formulated before the candidate answers themselves are generated by the system. Once the type for the

expected answers is determined, it will help in filtering the candidate answers in the Answer Extraction component. But with the absence of the expected answer type, the system may end up generating wrong answers.

2.4.1.3 Query Construction

Other task of the Question Analysis component in question answering system is formulating appropriate query from the user‟s natural language questions so that it would be sent to the document retrieval component of the system. Query formulation can be done by stemming, by expanding the query based on the expected answer type or based on expansion of words in the question.

The stemming approach is a process in which query words are broken down into their simplest form with all prefixes, infixes and suffixes stripped to find the stem of each word and perform the document retrieval by the stem word.

There are arguments on the importance of applying a stemmer in Question Answering Systems regarding its role in increasing the performance of such systems [12]. There was an attempt to design an Amharic stemmer in [13] which was evaluated on Amharic text from two domains, news articles and a classic fiction text. It was shown to have an accuracy of 60% for the old fashioned fiction text and 75% for the news articles. But it is not publicly available. Tessema Mindaye [7] has adopted an Amharic stemming algorithm done in [30]. For every stemming need in our work, we have used this implementation.

Query expansion based on the expected answer type is to create a semantically based index of documents based on the expected answer type. This means, a given document‟s paragraph, or even sentence will be analyzed semantically so that it will be labeled as location name, person name, numerical, etc. as its expected answer type. This technique, while efficient, is difficult to produce since determining the semantic group of a document such as Person Name, Location, Measurement, and so on is tedious as it needs extraordinary analysis.

The other approach, query formulation based upon question words, is to expand the query based on words in the question. It includes expanding queries based on word synonyms. In this approach an English language QA query expansion could be possible to use WordNet for a word possible synonym expansion as is used in [14]. since there is no Amharic WordNet available, we will not use this method of query construction.

It might be reasonable to employ more than one of the above methods in query formulation to increase the performance of the system. But speed is a major consideration when designing a question answering system, and research indicates that users will only wait a maximum of ten

seconds for a page to load [8]. Therefore, in our system, we will be employing the query formulation based on stemming for query construction.

2.4.2 Document Retrieval

Document retrieval is the task of identifying potential documents or passages which have a potential to contain an answer for the given question. The query input to this component is given from the question analysis component. A question answering system heavily depends on the effectiveness of a document retrieval system as a means of providing documents which are likely to contain an answer to a user‟s question. The document retrieval component presents ranked documents so that the answer extraction component will act up on it in a later stage. The addresses of the retrieved documents which are believed to contain the answer will be passed to the next component, the passage retrieval component, which is responsible to retrieve the passages which are believed to contain the answer.

For retrieving candidate documents related to the posed question, in the first place, we need to have a repository of documents from which we perform the retrieval and in the second place, we need to have a certain searching mechanism for performing the fetching and ranking of related documents.

2.4.3 Passage Retrieval

This component is initiated immediately after the document retrieval is over. The passage retrieval selects relevant passages of text from the candidate documents retrieved and indexes them according to relevancy.

Passage retrieval, which aims to find the text excerpts that may contain the exact answer of the given question, has long been studied in IR and recently has been an important component in QA systems. The passage retrieval module is usually added as an intermediate stage between the document retrieval module and answer extraction module. It can facilitate quick answer extraction and improve the efficiency of answer finding by users [15].

Furthermore, passages themselves form a very natural unit of response for question answering systems. A research showed that users prefer passages over exact phrase answers in a real-world setting because paragraph-sized chunks provide context which may help users to pick the appropriate answer in cases when the question itself seems to be ambiguous [16].

Researchers have undertaken a quantitative evaluation of different passage retrieval algorithms used in some of the best question answering systems selected by TREC [16]. According to this research, the performance differences between various passage retrieval algorithms vary with the

choice of document retrieval, which suggests significant interactions between document retrieval and passage retrieval.

2.4.4 Answer Extraction

The Answer Extraction (AE) component identifies candidate answers from the relevant passage set and extracts the answer(s) most likely to respond to the user question [17]. Given the top N relevant passages from the passage retrieval component, the answer extraction component performs detailed analysis and pin-points the answer to the question. Usually, the answer extraction component produces a list of answer candidates and ranks them according to some scoring functions [18].

Named Entity Recognizer (NER) is a technique that is used to label different entities such as Person Name, Place, Number, dates and times, and so on in a document. NER has been used for information extraction. Current text-based question answering systems usually contain a NER as a core component. The rationale of incorporating a NER in a QA system is that many fact-based answers to questions are entities that can be detected by a NER that will considerably reduce the task of finding an answer. NER basically can be used in the final stage of answer extraction module to filter out sentences that might not contain the expected answer types. Suppose the question type is place and an excerpt contains no expected answer type (i.e., Place), then NER will remove that sentence considering it as irrelevant and will not be considered for answer extraction [19].

A QA system typically uses both taxonomy of expected answers and taxonomy of named entities produced by its NER to identify which named entities are relevant to the question. Hence, the answer extraction component will check if there is a match between these taxonomies to determine the correct answer [19].

Chapter Three - Related Work

3.1 TETEYEQ (Amharic Question Answering for Factoid Questions)

The first question answering tool for Amharic factoid questions, TETEYEQ [4], was a great effort made towards serving users with their needs for a fact written in Amharic offline documents. Figure 3.1 shows a snapshot of TETEYEQ‟s interface replying to the user‟s question “የኢትዮጵያ ጠቅላይ ሚኒስትር ማን ይባላሉ ? ”. In this research, a large number of Amharic corpuses (approximately 15600 news articles) were collected from the Web and Ethiopian newspapers. The corpuses need to pass through a document pre-processing phase to make them have a pre determined format. This pre-processing could be done for documents already collected and expected to be used for answering all the natural language questions to be raised by users. This could be suitable in designing domain specific question answering systems having a predetermined knowledgebase.

But this doesn‟t make sense when thinking about the entire web as a knowledgebase for the question answering system. Therefore, in our research, as we are aiming to use the web as our knowledgebase, we don‟t need to undertake a pre-processing task.

The preprocessed documents were then indexed using the Lucene indexer after being analyzed by the Amharic analyzer (for tokenizing, removing stop words, character normalization, and stemming purposes).

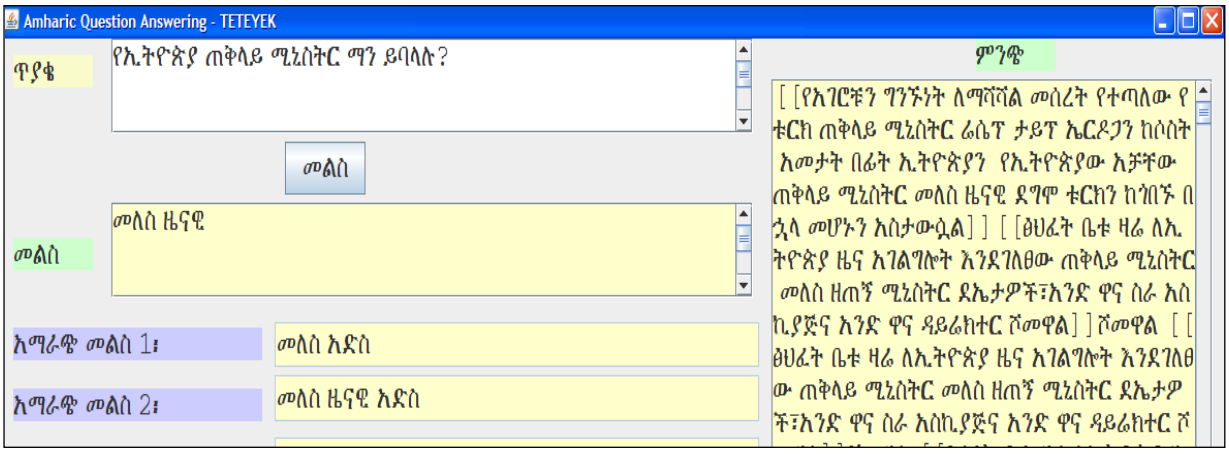

Figure 3.1: TETEYEQ – Amharic Question Answering System

User generated questions are received by the system and a rule based question type identification carried out to determine what type of question the user is asking and what type of answer is being expected from that question. But both the question type and expected answer types

identification was done using the help of manually crafted regular expressions (rules) which are not easy to develop and are most of the time rigid to entertain other types of questions if a need arises, in the future, for the system to deal with other question types other than the four factoid types („person", „time", „place", and „quantity").

Our system is employing an automatic question identification approach by using a statistical text (question) classification algorithm known as Support Vector Machine (SVM). With this approach, we have used a collection of manually prepared questions, categorized into the four question types to train the system so that in future, it will be able to identify user posed questions into one of these four question types. This approach doesn"t require writing hand crafted rules and is also easy to scale up when a need to deal with other question types arises in the future. Despite its difficulty in preparing training data sets for this purpose, it was quite easy and is a state of the art approach in many text classification researches resulting in a good classification performance [11].

Based on the question type identified, a proper query is then generated to be used for the document retrieval component. The same Amharic analyzer employed in preparing document index is used here too for getting a better result out of the document retrieval.

Techniques similar to this system are used in our system with a little modification for query generation and document retrieval. Hence, the basic difference is in question type classification and answer extraction techniques that we believe to bring a better performance and add flexibility to our system in handling other types of questions in the future; when compared to the previous one.

3.2 START Natural Language Question Answering System

The first web based English question answering system, START [3], has been developed by Boris Katz and his associates of the InfoLab Group at the MIT Computer Science and Artificial Intelligence Laboratory. It is a web based question answering system restricted on few domains like places (e.g., cities, countries, lakes, coordinates, weather, maps, demographics, political and economic systems), movies (e.g., titles, actors, directors), people (e.g., birth dates, biographies), and dictionary definitions.

START answers natural language questions by presenting components of text and multimedia information drawn from a set of information resources that are hosted locally or accessed remotely through the Internet. These resources contain structured, semi-structured and unstructured information. START targets high precision in its question answering, and in large part, START"s ability to respond to questions derives from its use of natural language annotations as a mechanism by which questions are matched to candidate answers. When new

information resources are incorporated for use by START, natural language annotations are often composed manually.

Annotations are computer-analyzable collections of natural language sentences and phrases that describe the contents of various information segments. The texts and other multimedia contents in the knowledge base are all annotated resources. These natural language annotations make it possible to index text and non-text resources which cannot be analyzed by other similar systems.

A user can query the system in English language only. The query is analyzed in the same way as ways used to create the knowledge base. The query‟s analyzed form is matched against the knowledge base to retrieve stored knowledge. The system will then produce an English response. Other linguistic tasks like synonymy, hyponymy, are also used in the matching process.

Since it came on-line in December, 1993, START has engaged in exchanges with hundreds of thousands of users all over the world, supplying them with useful knowledge. The system provides users with answers in the form of a paragraph that often contains multimedia fragments such as pictures and audio clips. Figure 3.2 shows a snapshot of START‟s web based question answering system user interface taking an English question “Who is the prime minister of Ethiopia?” and its reply is shown in Figure 3.4.

Figure 3.2: A snapshot of START's user interface

The START developers believe that paragraph-sized chunks form the most suitable unit of response to a user question, because complementing the short answer with additional contextual information may help with interpretation and analysis for the end user.

While its way of answer presentation seems to be a detailed one, its domain specificity and a labour intensive annotation of every material in the knowledge base is a limitation.

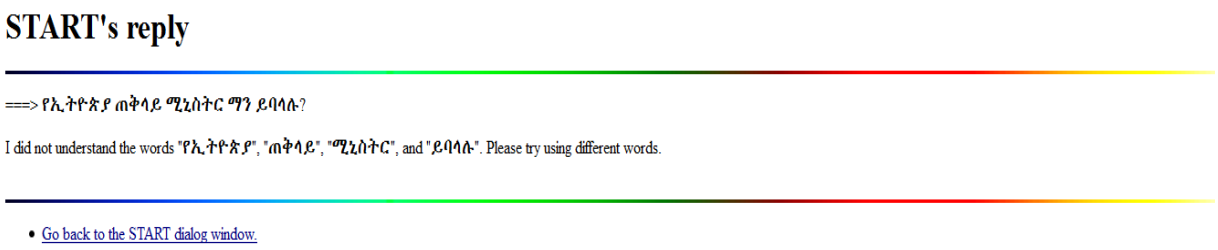

Figure 3.3: START's reply to an Amharic question

As it is designed for only English language, it doesn't give answers for questions in Amharic or other languages. For example, for the question "የኢትዮጵያ ጠቅላይ ሚኒስትር ማን ይባላሉ?" it doesn't understand every one of the words in the question as it is shown in Figure 3.3.

But for the same question given in English as "Who is the prime Minister of Ethiopia", as is shown in Figure 3.2, it displays a table showing the officials of the country Ethiopia as a whole. Figure 3.4 shows part of the START's reply for this question.

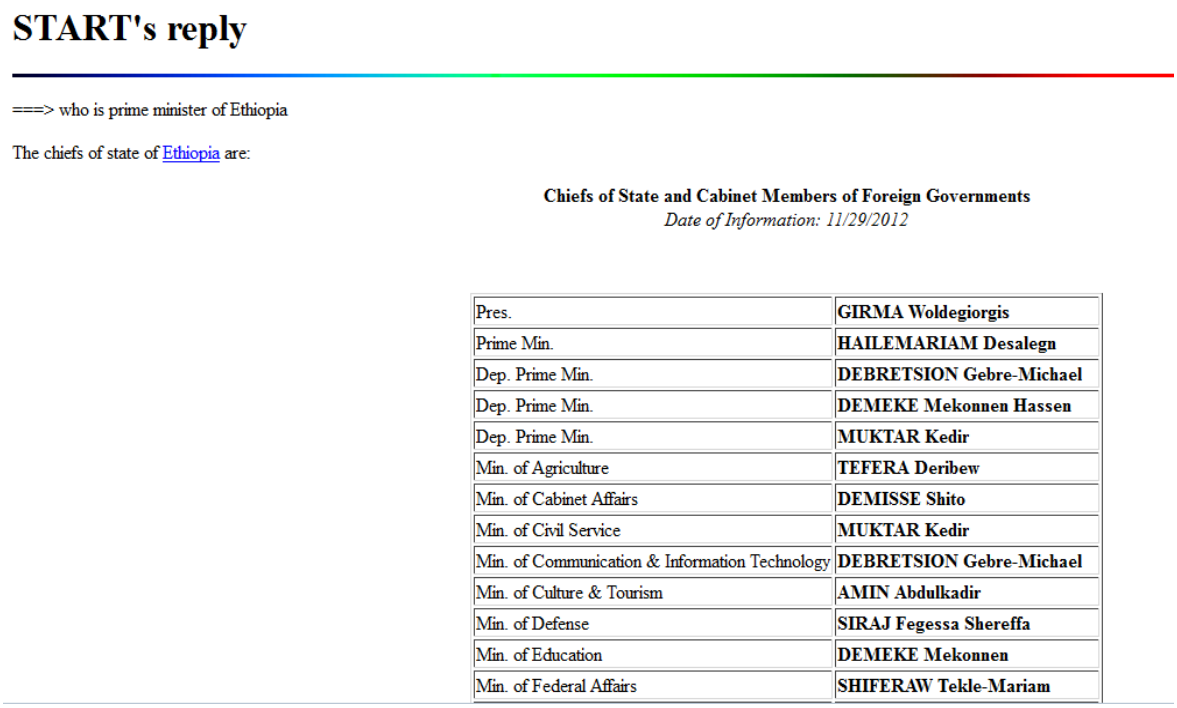

Figure 3.4: START's reply to an English question

3.3 Open Domain Factoid Question Answering System

Open domain factoid question answering system [32] is a desktop system by Patanaik and Sarkar designed to answer natural language factoid questions for English.

In the research, they have followed the pipelined architecture of a question answering system similar to the one in [12]. But, in the document retrieval component, they have designed a way to use both local corpus and a search result from existing search engines, Google and Yahoo, as a knowledgebase. The system supports document retrieval from Google and Yahoo via their public search engine application programming interfaces (APIs).

In the question analysis component, they have employed the Naïve Bayes and SVM classification algorithms for question classification.

In the document retrieval they have used a local corpus and also they make real time document retrieval by programmatically sending a query to Google and Yahoo and use the returned results as an additional resource in document retrieval. This approach incurs an additional overhead resulting in a delayed return time specially in an intermittent Internet connection.

They have also performed a passage extraction to pick the answer holding sentence out of it. They assumed that all the information required to produce an answer exists in a single sentence. But this assumption contradicts to the idea of the researcher in [26] claiming that, for open ended factoid questions, the answer terms might be dispersed over the entire document.

Figure 3.5: A snapshot of a sample run of Open Domain QA System

Then, they employed a named entity recognizer (NER) on the identified sentences to extract potential answers. The final answer returned to the user could be a phrase or a sentence believed to contain the answer. Figure 3.5 [32], for example, shows a snapshot of the developed system answering a question **"Who performed the first human heart transplant?".**

Figure 3.5 shows five answers to the posed factoid question. All of the returned answers are of phrase type except the third one. It also shows the source of the answer and a rank assigned to the answer.

The researchers have also experimented the performance of Naïve Bayes (with partitioned feature and bag of words feature) and Support Vector Machine (with bag of words feature and weighted feature set) classifiers for question classification. The experiment shows the following results:

Naïve Bayes Classifier (using Bag of Words feature set) gives 64% accuracy, Naïve Bayes Classifier (using Partitioned feature set) gives 69% accuracy, while Support Vector Machine Classifier using Bag of Words feature set brings 78% accuracy, and finally the Support Vector Machine Classifier using Weighted feature set results in 85% accuracy.

3.4 Language Independent Question Classification

The researchers in [10] came up with a language independent method for question classification which they have tested to work on English, Italian and Spanish languages with a classification accuracy of 88.92%.

Their approach exploits lexical features and the Internet to train the Support Vector Machine (SVM) classifier. The SVM learning scenario considers as attribute information prefixes of words in combination with attributes whose values are obtained from the Internet. These Internet based attributes are targeted to extract evidence of the possible semantic class of the question.

The procedure for gathering attributes from the web is by using a set of heuristics to extract from the question a word w, or set of words, that will complement the queries submitted for the search. They then go to the Google search engine and submit queries using the word w in combination with all the possible classes in focus. They then count the number of results returned by Google for each query and normalize them by their sum. The resultant numbers are the values for the attributes used by the learning algorithm.

They claim the classification method to be language independent since the features used as attributes in the learning task can be extracted from the questions in a fully automated manner without using semantic or syntactic information.

They believe that this method can be successfully applied to other languages, such as Romanian, French, Portuguese, and Catalan that share the morphologic characteristics of the three languages used in the experiment, English, Italian, and Spanish.

But, as there is no full fledged Amharic Search Engine available on the web and even Google is not replying well for Amharic queries as compared to the languages they have experimented with, we can not expect the system to achieve a good classification performance for classifying Amharic questions.

3.5 Question Classification Using Support Vector Machines

The researchers in [11] have experimented with five different machine learning approaches for automatically classifying questions. The machine learning algorithms used are Nearest Neighbors (NN), Naïve Bayes (NB), Decision Tree (DT), Sparse Network of Winnows (SNoW), and Support Vector Machines (SVM). They perform the test using bag-of-words and bag-of-ngrams of features in the questions. In the experiment of classifying questions in to 6 coarse grained and 50 fine grained question categories, the SVM algorithm yields a better classification performance than the other types in both fine grained and coarse grained categories. They have also reported their observation about employing different kernels to tune the performance of SVM classifier and witnessed that the SVM based on linear kernel returns similar results as the results acquired by using SVM based on polynomial kernel or other kernels.

3.6 Summary

Natural language question answering systems are studied in different approaches for many languages. We have tried to discuss about some of the works somehow related to our work. In Section 3.1, we have seen how TETEYEQ [4], the first Amharic factoid question answering system, works. It was designed by employing hand crafted rules in question classification and it was aimed for use as a desktop application. But our system is designed to employ machine learning algorithms in classifying questions and also is meant to be used in the web. Hence, we have added a web crawling component so as to acquire web documents for our search space.

The first web based English question answering system, START [3], answers natural language English questions by presenting text and multimedia information. It uses natural language annotation technique to label the information they have and when a certain question, with a matching annotation to the stored information comes, then they will present the information back to the user. The manual annotation makes it labour intensive and also it is designed to answer natural language questions of the English language only. Our system is also a web based one which aims at answering Amharic natural language questions. Instead of using a labour intensive

annotation to resources, we tried to automatically classify questions and retrieve documents related to the questions using the known open source Lucene searcher.

An open domain question answering system [32] is discussed in Section 3.3. It is designed for answering English natural language questions. They have tried to use a local document source as well as a real time search result acquired from Google and Yahoo search engines. Such methods of document retrieval depend on the availability of a high speed Internet connection. They have also used a named entity recognizer (NER) for English language in extracting answers. This simplifies the answer extraction process, but with the absence of NER tool for Amharic language, we have tried to use a list of person and place names and regular expressions for extracting answers terms for the person, place, time, and quantity types.

Researchers in [10] have developed a language independent method for question classification and tested it on the English, Italian, and Spanish languages. They have used a heuristic method of pushing the query terms into the Google search engine and performing certain computations on the returned results to determine the question type. This method also is dependent on the availability of a fast Internet connectivity.

The research in [11] compares different machine learning algorithms in the task of automatic question classification for question answering systems. Their testimony declares that support vector machine (SVM) algorithm outperforms other four known machine learning algorithms. Hence, we stick in employing the SVM classifier in our research too.

Chapter Four - Design of WBAQA (ልጠየቅ)

Our QA system has been given a name **ልጠየቅ (Ask Me)**. **በላ ልበልሀ (Bela Libeliha)** is an ancient Ethiopian traditional case law, where two people appear before a judge and they used to ask questions for each other in a poem. When the trial starts, infront of the judges, the plaintiff says **ተጠየቅ (Be questioned)** to the defendant and the defendant, after getting himself ready, replies by saying **ልጠየቅ (Ask Me)**. Then the trial continues where both parties are asking questions (natural language questions) to each other and finally, the one who is able to reason out the most wins the trial. Likewise, our system, after all the necessary data preparations, is saying **ልጠየቅ (Ask Me)** to the end user. Hence, the system is named after this historic term.

The web based Amharic question answering system, as its name indicates, is a natural language question answering system expected to respond to end users‟ Amharic factoid questions posed to the system like the way end users are querying human beings in natural language questions.

Therefore, in the first place, the system needs to understand what the end user is meant to look for. Analyzing the users‟ request by itself is not enough. The request (question) needs to be formulated into an equivalent query that can be understood by the system in retrieving documents related to what is being asked.

Passage retrieval is an intermediate process in some question answering systems between document retrieval and answer extraction. Potential passages are selected from the returned documents so that the answer extraction effort will be minimized. But it doesn‟t always guarantee to get a better result than extracting answers directly from retrieved documents. This is basically dependent on the distribution of answer terms in the document. If it is believed that answer terms are commonly found in a single passage, the passage retrieval becomes efficient. But for open ended factoid questions, the answer terms might be dispersed over the entire document. Hence, dealing with passage retrieval doesn‟t work in our case [26].

Knowing the key words to retrieve documents from the document repository is the next step. But the document repository‟s richness has the major effect in the likelihood of finding the documents in the repository. The repository can be a small (or constant) collection of documents collected from the web or offline materials. This results in two situations: First, if a matching document is not found in the repository, given a particular set of query terms, it can always be determined that the same response will be generated in the future for similar queries. Second, the repository size has a direct relationship in the likelihood of finding related documents for a given query; the larger the number of documents in the repository, the higher the likelihood of finding related documents for a given query term(s).

Figure 4.1 shows the entire architecture of the web based Amharic question answering system. The detail explanation about each component in the system comes next.

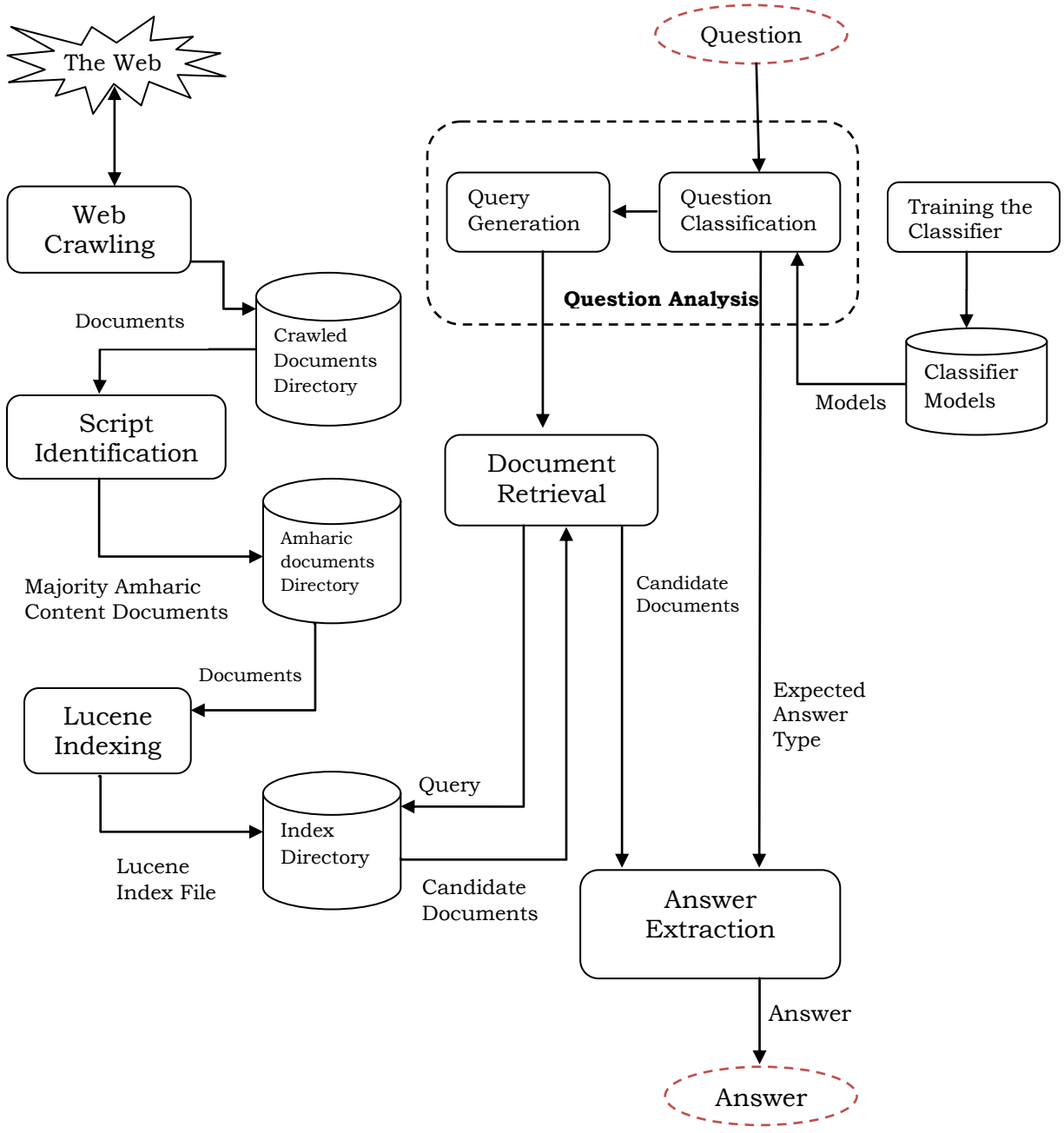

Figure 4.1: Architecture of the Web based Amharic Question Answering System

4.1 Web Crawling

The objective of web crawlers is to retrieve web pages and download them or their representations to a local repository. The repository might be used for different tasks like the search engine as a source of searching documents for users‟ queries, mostly after passing through an indexing process.

The crawling process is initiated by giving initial addresses (seed URLs) to the crawler to start for crawling. The content in the seed URL will get downloaded and external links in the seed page will be followed for another download to take place. The external links might be links to other pages in the same site or in external sites. The visiting and downloading process goes like this until all the links are exhausted or some other stopping conditions are reached. The crawling order can also be different like first all the seed URLs are downloaded and the links in them will be downloaded. This order is called breadth first crawling order. It gives priority for the most important seed URLs before giving a visit to the external links. And the other one is, taking one seed URL, downloading it and directly going to links in the page before other seed URLs called depth first crawling order.

Crawlers are categorized as language specific and generic crawlers based on the specificity on a certain natural language or to any natural language at all [7]. Our crawler is a generic one as it is meant for crawling all the pages found in the given URL without checking whether the page is with Amharic content or not. For example, one of our seed URLs, the Ethiopian News Agency (www.ena.gov.et), has news in English in addition to the Amharic ones, but the crawler will download all the pages without checking the language. Also, not all the pages fetched following external links in the seed URLs result in an Amharic content page.

Therefore, a certain categorization task must be called after downloading a particular page so that non-Amharic content pages should be discarded and pages with most of Amharic content and written with Unicode encoding should be stored in the repository for further processing. As more Internet standard protocols designate Unicode as the default encoding, there will undoubtedly be a significant shift toward the use of Unicode on web pages [36]. That is why we preferred to work on Unicode encoding. Also, known Amharic news sites (like: www.ethiopianreporter.com, www.waltainfo.com, www.ena.gov.et, etc.) are using this encoding. The language (script) identification is the task of the language (script) identifier component. As is shown in Figure 4.1, the language (script) identifier takes in the crawled web pages from the repository and puts the pages with Amharic majority web pages into another repository that is to be used by the Lucene indexer.

For the language (script) identification task, with certain modifications, an open source tool called Language Identification Module (LIM) developed by Japanese researchers is used to classify contents in a given page as to Amharic majority content pages or not. Based on this tool, pages containing 60% or more Amharic contents and written in Unicode encoding are treated as

relevant else non-relevant for the storing process [35]. A proportion of 60% is chosen so as to include pages with a multilingual content but with a majority of Amharic texts in the page. Relevant documents then are stored in the repository and used for subsequent tasks.

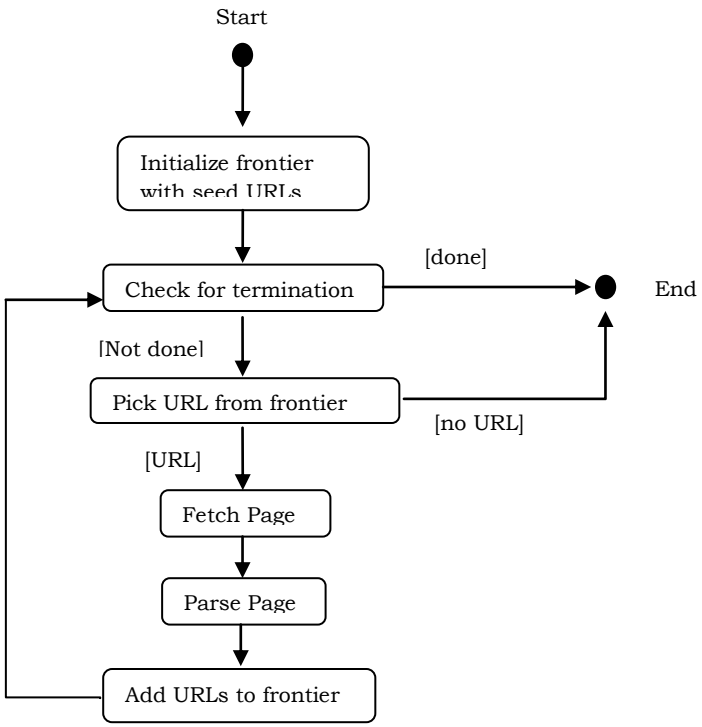

Figure 4.2: Flow of basic sequential crawler

Figure 4.2 is a flow chart showing the activities in a general crawling process [20]. In the Figure, the frontier is the *to-do list* of a crawler that contains the URLs of unvisited pages. First, the selected seed URLs will be stored in the frontier. The frontier works in a queue fashion executing URLs in a First In First Out (FIFO) order for the links to be visited. The top in the queue are the seed URLs and every time when a new link is found in the process of fetching a page, the link will be added to the frontier list at the bottom. This ensures a breadth- first crawling mechanism.

Every time, when a new link is found, it should first be checked for its existence in the queue to avoid repetition before adding it to the queue.

4.2 Script Identification

As is explained above, fetching a page doesn‟t mean it to be added to the final repository. It goes through the categorization process to determine its importance for our task based on whether the Amharic contents in the page is of 60% and more of the entire content or not plus the encoding used is Unicode. This identification is done with the help of the Language Identification Module (LIM) trained with full Amharic contents with Unicode encoding.

The LIM takes in a page from the crawled pages directory and identifies the scripts‟ proportion used in writing texts in the page. Then, pages considered as relevant in our case, with the above criteria, will be stored in the Amharic pages directory to be used for the indexing task.

The identification task doesn‟t always help us purely identify Amharic documents. As the identification is based on the characters and encodings used in the documents, it cannot exclude documents written in other Ethiopian languages (like Tigrigna) using similar characters and encodings. For this issue, we have another module used after the identification process to check the availability of common Amharic language stop words in the documents and store the document to the final repository if it contains these stop words or reject if it doesn‟t. For this purpose, a list of frequently appearing Amharic stop words is collected from [38].

4.3 Indexing

Indexing is the process of converting documents in the repository into a highly efficient cross-reference lookup (index). For this task, we have used the Lucene library. In Lucene indexing, the index stores statistics about terms in order to make term-based search more efficient. Lucene's index falls into the family of indexes known as an inverted index. This is because it can list, for a term, the documents that contain it. This is the inverse of the natural relationship, in which documents list terms. Lucene index contains a sequence of documents, where a document is a sequence of fields and a field is a named sequence of terms called strings.

The Indexer extracts a text from the downloaded documents. But this text passes through the Amharic analyzer before it is stored in the Index file. Figure 4.3 shows the processes in the Amharic analyzer. Description of the flow of activities in the Figure is follows.

- The character streams will be *tokenized* into words by taking white space and Amharic punctuation marks as word demarcations. Also, known Amharic alternative characters will be normalized into a unique character.
- If there is a word expressed in shorter forms (abbreviations), it will be replaced by its expanded form.

- Then, the words will be checked for *stop-words*. If the word is a stop-word or if it is a variant of a stop-word, it will be removed.
- The remaining non-stop-words will be *stemmed* to reduce them to their common form.

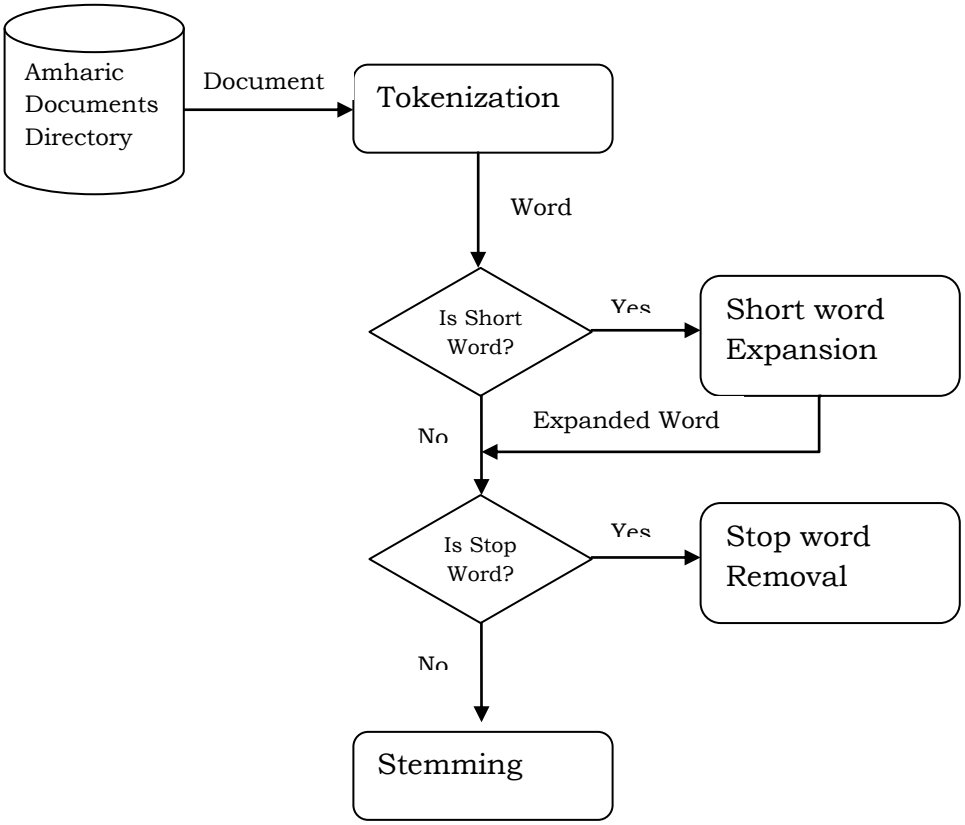

Figure 4.3: Flow of processes in Amharic Analyzer

The final result from Figure 4.3, the stemmed word, is added into an index, a structure that is appropriate for fast searching and further processing. The resultant file is called Lucene index file and it cannot be directly viewed in text editors. But there is a Lucene index toolbox called *Luke,* freely distributed under Apache software license, which is useful to open the Lucene index file and see the indexed terms, the number of documents containing the term, and the frequency of the term in the documents containing it.

As the Lucene does the indexing for us, given the necessary preprocessing modules for tokenizing, abbreviation expansion, stop-words removal, and stemming for Amharic language, our task is to integrate these Amharic language processing modules into that of Lucene so that the document retrieval component of the question answering system can go safely.

So far, we have been going through the preparation of the data repository from which the document retrieval component can search for related documents for the query terms generated from the natural language question posed by the user.

Next, we will deal with the other components of the system starting from the analysis of the users‟ question to the system‟s answer finally expected to be delivered in response to the question.

4.4 Training the classifer

Support Vector Machines were initially designed for binary classification (classification into one of the two classes). How to use it in multi-class classification is still a research issue. Different approaches have been proposed on how to use the binary classifier for multi-class classification tasks. A "one-against-one" way of combining binary classifiers is a more suitable way for practical use [23].

Hence, in our training, we have generated a total of six classification models in the "one-against-one" terminology (i.e. n(n-1)/2 models where n is the number of classes; n=4 in our case). That is, models for „person‟ & „time‟, „person‟ & „place‟, „person‟ & „quantity‟, „time‟ & „place‟, „time‟ & „quantity‟, and „place‟ & „quantity‟ are developed and stored for use in testing the classifier.

SVM, as a statistical approach, needs large amount of training data so as to train the machine so that it would make accurate classifications for the unseen test data.

In this case, we have prepared 1200 questions from the four classes of factoid questions („person‟, „time‟, „place‟, and „quantity‟) for training the classifier.

The Windows version of the SVM classifier, SVM_Light[1], developed by Thorsten Joachims, is used for our question classification task. It is freely available for research purpose and can be smoothly integrated with our system.

The more training data prepared for training the classifier, the more accurate the classifier becomes. But there is a trade-off for the accuracy; it increases computation time to classify the test question into one of the classes. In our case, the accuracy is a priority with the expense of the computational time as the accuracy of the classifier has a vital role in the question classification, and other subsequent components of the system like the Answer extraction component. Hence failing to accurately classify might end up in returning a wrong or no result to the given question.

[1] Freely available at `http://svmlight.joachims.org/`

If it turns out that the type of question doesn‟t fall in any of the question types (person, time, place, and quantity), then the system decides that the question is not of Amharic factoid question type and returns a “ይቅርታ ለጥያቄዎ መልስ አልተገኘም” message to notify the user for this effect and halt the question answering process at this stage.

To use the SVM_Light classifier, the training and testing data should be converted into a format suitable for the SVM algorithm. The conversion process and other classification details are presented in detail in the next chapter.

4.5 Question Analysis

This is the first component of the system which directly interacts with the end user. When the user poses Amharic factoid question to the system, the question analysis component takes in the user query and passes it to its sub components. The two major sub components are automatic question classification, and query generation.

4.5.1 Automatic Question Classification

One of the major differences of our system with that of Seid Muhie‟s [4] is the question classification approach. In [4], a serious of hand-crafted rules were used to classify user‟s questions into the factoid question types of „person‟, „time‟, „place‟, and „quantity‟. But, in our case, the classification is done using a statistical approach by getting use of the very known and efficient text classification algorithm called the Support Vector Machine (SVM).

The classification models developed in Section 4.4 will come in to use at this stage to classify new questions posed by the users. The six classification models will be used to classify the user question and then for the total effect, the class having the largest number of appearance after executing the six classifications will be the class assigned for the given question.

Details on how the training and testing procedures are done on the SVM classifier will be presented in Chapter 5.

4.5.2 Query Generation

This is the component which transforms the natural language question into the one suitable for searching the terms in the question from the Lucene index file. The generated query hence is the sole input to the forthcoming document retrieval component.

During the generation phase, after two steps, the given question goes through similar steps as the crawled pages went in their way to the indexing (similar to Section 4.3). These steps are:

- Question particles like "ማን ይባላል?", "ምን ያህል ነው?" will be removed from the question because it doesn‟t worth for searching. This is done by checking the existence of such terms in the question and replacing them with an empty character.
- Lucene‟s built in query parser is used to parse the user query. The parser needs a language analyzer as a parameter. Hence, the Amharic analyzer will come in to use at this point. The following steps illustrate the processes in the Amharic analyzer.
- The character streams will be *tokenized* in to words by taking white space and Amharic punctuation marks as word demarcations.
- If there is a word expressed in shorter forms (abbreviations), it will be replaced by its expanded form.
- Then, the words will be checked for *stop-words*. If the word is a stop-word or if it is a variant of a stop-word, it will be removed.
- The remaining non-stop-words will be *stemmed* to reduce them to their common form.

The reason why the query is analyzed with the same analyzer (Amharic Analyzer) in which the crawled documents are analyzed is mainly to have the same kind of processing being applied on the text in the index and in the query so as to get the best out of the searching process.

The generated query is then passed to the document retrieval component for searching documents related to the query generated and believed to contain candidate answers.

4.6 Document Retrieval

The document retrieval component is responsible for fetching documents which are related to the generated query in the previous component. The quality of this component has a direct relationship with the overall Question Answering System‟s performance [25]. As is mentioned earlier in this chapter, the searching of documents having maximum number of terms matching to the query generated is done in the index level. That is, as the crawled documents are analyzed, indexed and stored in a Lucene index file, and the generated query is similarly analyzed to have compatible format as the indexed document, the searching becomes straight forward.

The searching is solely performed by the Lucene‟s IndexSearcher class by passing the path of the index directory for documents to the constructor of this class and passing the analyzed query as an argument to the search method of this class. As a result, the set of related documents to the given query will be returned as Hits.

This may work fine for other Information retrieval tasks like the search engines where their primary goal is to provide the user with related documents containing a term (or terms) from the search query. But, for the question answering task, hits returned from the document retrieval component are not guaranteed to hold candidate answers for the user question. Because, the Lucene‟s index searching only cares about the match (similarity) between query terms and indexed documents. But it is likely that a top ranked document, a document containing maximum number of query terms when compared to other documents, might not contain candidate answers in it.

Therefore, the documents returned from the Lucene‟s search method are subjected to further inspection by the subsequent component, answer extraction component.

4.7 Answer Extraction

This is a fundamental component that a question answering system differs from the ordinary search engines. Its main objective is to extract terms, which are believed to be answers to the question posed by the user, from the hits returned by the Lucene‟s index searcher.

Different factoid question answering systems employ different methods of extracting an answer from candidate documents or passages. Most of the methods are pattern matching technique, semantic parser, temporal context identifier, and logical prover [27].

The answer extraction methods based on pattern matching form the most simple and commonly used group of answer extraction methods. The most complex part of the method lies in the way the patterns are created or generated, whereas the patterns themselves are simple. Answer extraction methods based on pattern matching follow the tradition of information extraction methods.

Answer extraction patterns (AEPs) may consist of several types of units, such as punctuation marks, capitalization patterns, plain words, part-of-speech (POS) tags, tags describing syntactic function, named entities (NEs) and temporal and numeric expressions.

The logical prover has access to knowledge of five different types: extended WordNet axioms, ontological axioms, linguistic axioms, a semantic calculus and temporal reasoning axioms.

Besides using the above ways, there are other methods. One of the methods is the very simple proximity based method. It is based on the proximity between the query terms and candidate answer terms. These candidate answer terms are identified from the documents based on some pattern matching technique.

In our system, we follow the pattern matching approach together with the proximity method.

Here, we follow the following steps to generate the likely answer:

- Acquire the expected answer type for the posed question. This is primarily determined in the question analysis component as discussed in Section 4.4. It is straight forward that questions of type „person" are expecting an answer of same type, questions of type „time" are expecting an answer of type „date/time", and so on. Determining the expected answer type is useful for us to make use of different pattern matching methods accordingly.
- Extract a candidate answer from the top n hits returned by the document retrieval component. For identifying candidate answer terms from the hits returned, using a Named Entity Recognizer (NER) module is of great importance as is witnessed in many other question answering systems for other languages [24]. Unfortunately, we couldn't find NER developed for Amharic language. Hence, we use a pattern matching approach for picking date/time and quantity/numeric terms from the document and for the person and place terms in a document we use a gazetteer (list of names of persons and places) matching technique by supplying the rules and list of gazetteers manually. The regular expressions and gazetteers are adopted from Seid Muhie's work in [4]. This is done in light of the expected answer type acquired in the above step.
- Compute the proximity of extracted candidate answers in the document with the query terms in the document. This is done for top n Hits returned.
- Filter the hits based on their relevancy. The relevancy of a document is decided based on the number of query terms found in the document.
- Pick the n likely answer(s) from the relevant documents above. A document with a maximum number of query terms and with a smallest distance between these terms & the candidate answers is more likely to have the correct answer.

Finally, the list of likely answers will be returned back to the user as a reply to the question posed at the very first step.

It is not a surprise that after all these sequential interaction of the question answering components one after the other, the final result may end up in returning "no answer" to the user. This happens because of the absence of relevant documents in the repository, because the query terms are found very dispersed in „relevant" documents, or absence of candidate answer terms in the „relevant" documents.

Chapter Five - Implementation of Web Based Amharic Question Answering

5.1 Web based Question Answering

The title "Web based Question Answering" implies for answering users‟ natural language questions being online (in the web). The architecture and basic explanations about the components of the system is covered in the previous chapter. Here we will look into the details of the design mainly discussing about the algorithms employed in the system and the rationale behind selecting the algorithms.

The overall system has two major parts, the search engine part and the question answering part.

Next, we shall discuss about the system‟s development environment and then we will have a detail view of the sub parts in the system and discuss how these parts are formulated with the following order: Web crawler, indexer, question analysis, document retrieval, and answer extraction.

5.2 Development environment

We have prepared a prototype which can basically take user‟s natural language query and after going through all the mentioned processes, deliver an answer in the way users prefer to see just like two persons interacting with each other where the one is asking for a fact and the other is replying. Therefore, the system is developed and tested in a system with Intel Core 2 Duo CPU of 2.1 GHZ speed, a 2 GB RAM, and a Windows 7 operating system. Furthermore, the system is dependent on an Internet connection for regularly performing a web crawling and the system itself is a web based system where end users access it online.

The entire system is designed with the Java programming language to take the language‟s advantages of machine independency and its ease in interacting with other open source tools like the SVM_Light tool, JSpider crawler, and the Language Identification Module (LIM) we use for text classification, web crawling, and identifying Amharic web pages respectively. Moreover, the Lucene Application Programming Interface (API) is fully designed by Java programming language.

For crawling the web, we have used the JSpider crawler. It is an open source project, written entirely in Java (J2SE). It is an implementation of a highly flexible, configurable web robot engine.

After the web crawling task, we need to classify a given web page as relevant or irrelevant to our study based on the page‟s proportion of Amharic contents written in Unicode encoding style. This major duty wouldn‟t have been easily accomplished without the help of the freely available Java based Language Identification Module (LIM) tool developed by Japanese researchers in their project called Language Observatory Project (LOP)[2] .

Another important tool we use is SVM_Light which is the implementation of the Support Vector Machine (SVM) algorithm for a statistical text classification task that we have used in the question analysis component for classifying questions into one of the question types.

Next, let‟s go for the details of implementing every component and sub components one by one.

5.3 Crawling

We have used the Java implemented, open source web crawler called JSpider crawler with a little modification on the settings.

As is shown in Figure 5.1 [31], the basic parts in JSpider crawler are:

- The engine core
- A model, describing all spidered resources, sites, etc…
- An event model notifying of what‟s going on
- Components implementing a SPI (Service Provider Interface)

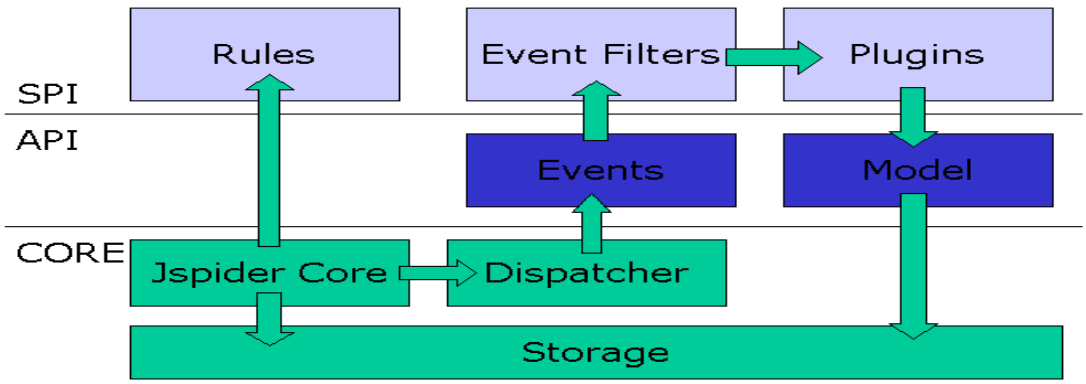

Figure 5.1: Basic components of the JSpider web crawler

[2] http://gii2.nagaokaut.ac.jp

Let‟s see the Jspider components in brief.

a) SPI components

There are several types of SPI components that can be used by JSpider.

The most important ones are Rules, Plugins, and Event Filters.

Rules

Rules decide which resources should be fetched and/or processed by JSpider. By constructing a set of rules on a global or per-website basis, we can define JSpider‟s behaviour and scope.

There are a lot of rule implementations that come with JSpider, and we can also develop our own.

For example, we can write the following global level rule to decide about the spidering (fetching) of resources:

jspider.rules.spider.count=1

*jspider.rules.spider.1.class=net.javacoding.jspider.mod.rule.**OnlyHttpProtocolRule***

This makes sure that we only spider resources with the **http://** protocol in front of the URL and ignore others.

Plugins

Plugins are components that have access to the data structures exposed by JSpider, and are notified of certain events happening. They can then take appropriate actions. These actions can be anything:

- Writing a report file
- Displaying a message on the console
- Writing a fetched resource to disk
- Sending a mail to someone, etc…

By implementing our own plugin, we can add functionality to JSpider. We could, for instance, construct a configuration in which JSpider tests a certain site for 404 errors (link errors), and send an e-mail with all error links to the webmaster. Another usage would be to mirror a website on our local disk: for this purpose, we would enable a plugin that writes every fetched resource into a file on our hard disk.

Event Filters

Event Filters can select the events that have to be handled by the system as a whole or a particular plugin.

b) API components

The JSpider API consists of object model and an event system.

Object model

The object model represents everything that JSpider encounters while spidering. It can be Sites, Resources (URLs), Content, Cookies, etc…

This model can be accessed from within the components in order to look up data, write a report, calculate statistics, etc… The model is backed by the storage component.

Event system

The event system is a group of event classes that will be used to notify all interested plugins of certain events happening during the spidering progress.

There are three types of events in JSpider:

- *Engine events* like: spidering started, stopped, configuration chosen, …
- *Spidering events* like*:* site discovered, resource spidered, fetch error, …
- *Monitoring events* like: give information about the spidering progress and the thread pool occupation.

c) Engine core

The main part of JSpider is the engine core that implements the most basic functionality delivered by JSpider.

All the above components‟ configurations are kept in properties files that we can modify for our own use. In our case, we have made changes on the proxy configurations to specify the proxy server we are using („cache.aau.edu.et, port: 8080‟) in the global level (default) JSpider configuration file.

We are using Version 0.5 of the JSpider crawler released for Windows version. The JSpider is started from the command prompt via a startup script. After making important configurations like the proxy server and the number of threads to be initiated, we type the command “jspider”

followed by the seed URL we want to crawl and "download" command at the end. A sample snapshot of the crawling is shown in Figure 5.2 by supplying a sample seed URL www.ethiopianreporter.com to start crawling the website of the Ethiopian Reporter newspaper.

Figure 5.2: Starting the JSpider web crawler

After we have downloaded pages from the selected seed URLs, we need to identify the pages with Amharic majority content using the language identification module (LIM) and further excluding pages which are written with other Ethiopian languages using the same characters as Amharic language (languages like Tigrigna). This is done by checking the availability of commonly appearing Amharic language stop words in the identified documents to decide whether to include the document into the final document repository or not. The list of commonly appearing Amharic stop words is collected from [38].

5.4 Indexing

For creating the Lucene index file, we need to understand the following basic classes of Lucene.

- ***Document*** - The *Document* class represents a document in Lucene. We index *Document* objects and get *Document* objects back when we do a search. For example, the following is the code we used to instantiate a lucene document object with a name „doct".

  Document doct = **new** Document();

- ***Field*** - The *Field* class represents a section of a *Document*. The *Field* object will contain a name for the section and the actual data. A Document object is a collection of Field objects (name/value pairs). So, for each file to be indexed, we instantiate a Document, then populate it with Fields.

  In our case, the most important Fields are the <"title"><"title of the page">, <"url"><"url of the page">, and <"content"><"content in the page">. It is also possible to add other Fields like the date modified, author, and so on. The code to add title, and URL, to the document „doct" is:

  ```
  Field ttl= Field.Text("title","Official site of Addis Ababa University");
  ttl.setBoost(2);
  doct.add(ttl);
  doct.add(Field.Keyword("url","www.aau.edu.et"));
  ```

There is an option to give emphasis for particular fields so that while searching, a match found in these fields shall be given a priority. In our system, the *title* Field should get a priority hence, when we do searching, if a match is found between the query terms and the title field, the document will be given a higher rank. This is done by giving a value for the *setBoost* property of the title Field as *ttl.setBoost(2)* as is written above.

- ***Analyzer*** - The *Analyzer* class is an abstract class. The most commonly used implementation of this abstract class is the *StandardAnalyzer* class. But it is designed for English Language. So, for our case we need to have an analyzer for Amharic language. We are using the Amharic Analyzer developed in [4] by making suitable improvements of the work in [7]. The Analyzer"s job is to do preprocessing tasks on the files before they are indexed. The tasks include tokenizing, short words expansion, stop-words removal, and stemming. The details on the AmharicAnalyzer will be presented later.
- ***IndexWriter*** - The *IndexWriter* class is used to create and maintain indexes. This class needs an appropriate analyzer to be passed in order to create an object from it. So, we need to use the AmharicAnalyzer in this case.

  Figure 5.3 shows the algorithm used to create the lucene index file by taking documents from the crawled Amharic pages repository.

Step 1: While the repository has files, take a file from the Amhraric pages repository

Step 2: Create a lucene ***Document*** object to hold the file

Step 3: Add ***Fields*** and their values to the lucene ***Document*** object

Step 4: Create lucene ***IndexWriter*** object by using the Amharic analyzer and a path in which the lucene index file is to be created

Step 5: Add the lucene ***Document*** object to the ***indexWriter*** object.

Step 6: Go back to Step 1

Figure 5.3: Algorithm to create a Lucene index file

The Amharic pages repository was the one created in the document identification component. The algorithm in Figure 5.3 will be applied for all the files in this repository. An object is instanitated from the lucene ***Document*** class to hold every file. The different sections of the file like its title, URL, and content are then added to the object together with their values as lucene

Fields. An ***IndexWriter*** object is then instanciated by passing the ***AmharicAnalyzer*** object and a directory for the lucene index file to be created. The Amharic analyzer is used for tokenizing every word in the file, filtering stop words, and stemming non stop words to be indexed. Finally, the lucene ***Document*** object with its necessary ***Fields*** added will be written by the ***IndexWriter*** object into the lucene index file.

5.4.1 The Amharic Analyzer

The Amharic analyzer is the principal component in making the Lucene index by undertaking fundamental preprocessing tasks on the files before they actually get indexed. The following are the tasks performed in the Amharic analyzer in their order of use:

a) Tokenizing

The Amharic documents in the repository are tokenized before going to the indexing process. Tokenization breaks the stream of characters into raw terms or tokens. This process detects word boundaries of a written text. In Amharic, words can be separated using a white space, tab, Amharic punctuation marks, or by new line. Therefore, by tokenization, we identify separate terms from the document.

In the process of tokenization, alternative Amharic characters found in the current token will be normalized to a unique character. The known Amharic alternative alphabets are then replaced by a normal one throughout the documents in the repository. The normalization is similar to the one in [4].

Example: characters „ሀ“, „ኀ“, and „ሐ“, are normalized into „ሀ“. „ሰ“ & „ሠ“ are normalized into „ሰ“.

b) Short words expansion

Amharic nouns formed by combining two separate words are commonly written in short forms. But, for a better search result, they need to be written in a unified form so that enquiries using the expanded form can find the shortened forms. The short form writing is done by use of a forward slash (“/”) or a dot (“.”).

Therefore, short words like “ም/ሊቀመንበር”, “ት/ቤት”, and “ቤ.ክርስቲያን” are expanded into the words “ምክትል ሊቀመንበር”, “ትምህርት ቤት”, and “ቤተ ክርስቲያን” respectively through the process of short words expansion.

c) Stop words removal

Stop words are commonly occurring words in a sentence. As stop words have no document discrimination value for information retrieval purpose, they usually are chopped out in most of

IR systems. There is no comprehensive list of all Amharic stop words. But we use the list generated in [4] by taking the list of stop words identified in [7] with a little modification plus a new list added to include question particles like ማን (who), መቼ (when), የት (where), and ስንት (how much).

d) Stemming

Stemming is a technique whereby morphological variants are reduced to a single stem. It is language dependent and should be tailored for each language since languages have a varying degree of differences in their morphological properties.

As Amharic is a morphologically rich language, designing a perfect stemmer is not an easy task. So far, different efforts have been made to come up with a full fledged stemmer for Amharic language.

For the question answering task, we should keep proper nouns, dates, and numerals un-stemmed because these are the terms with a potential to bear an answer. So, for this purpose, the stemming algorithm developed in [4] has been employed instead of the one in [7].

All these preprocessing tasks done in the Amharic analyzer make documents ready to be indexed by the Lucene‟s index writer and stored in an index file. The very important point here is, we need to make sure that the same Amharic analyzer must be employed while querying the index searcher.

The Lucene index files can be viewed by a tool called *Luke.* As Lucene employs inverted indexing, which shows list of documents containing a term instead of list of terms contained in a document; Figure 5.4 is a snapshot of a sample index file showing some of the documents containing the stemmed term “ደረጅ” which is the stem of the terms “ደረጃ”, “ደረጃና”, “ደረጃዎች”, and the likes. As we can see, there are 93 indexed documents containing this term displayed in decreasing order of the query score in each document. This score is computed based on the term frequency (tf) and inverted document frequency (idf) of the query term (“ደረጅ”) in each document.

The Figure also shows the lucene ***Document Fields*** added during indexing. These ***Fields*** are the *content, localurl, titile, and url* as shown in order.

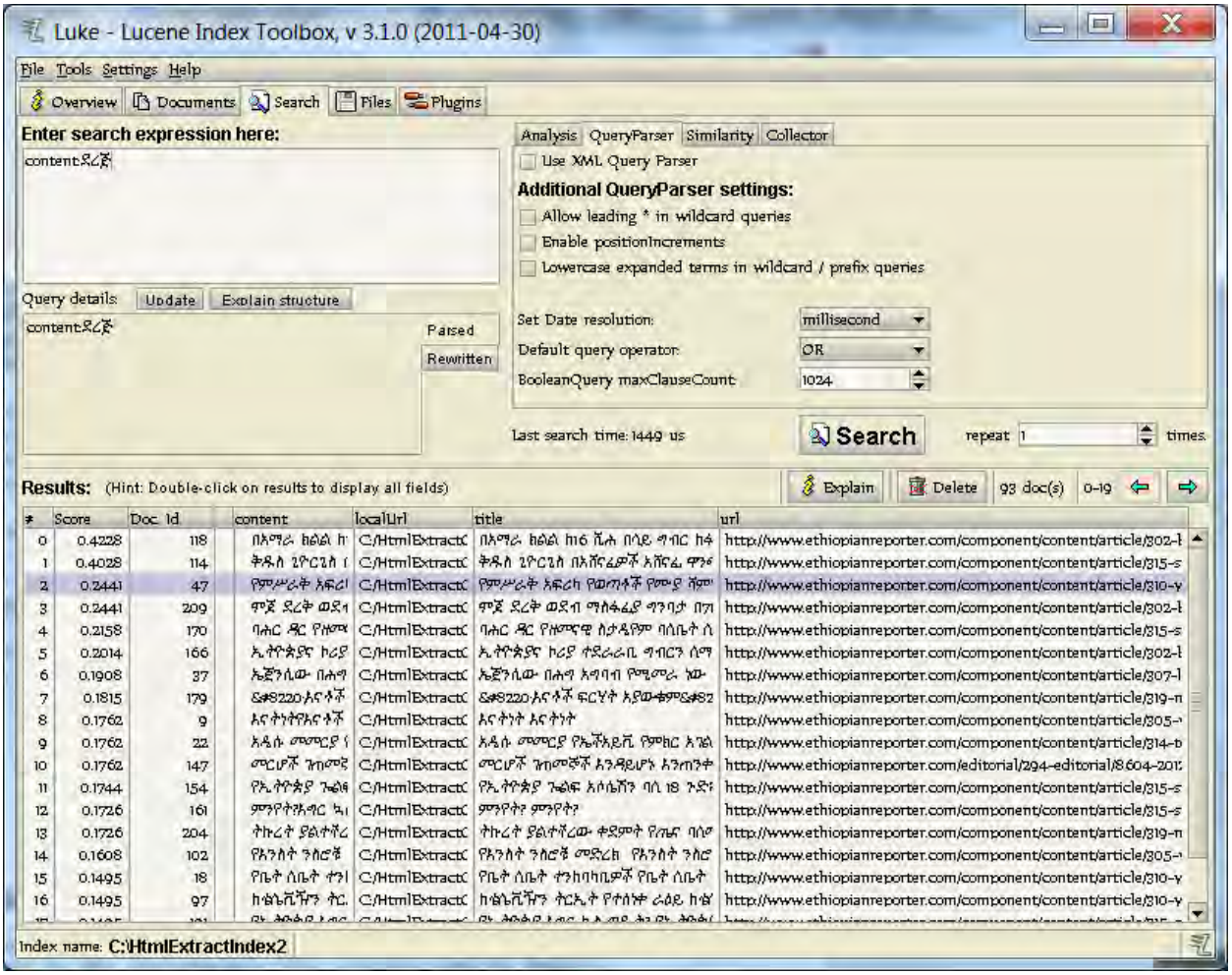

Figure 5.4: A Lucene Index file as seen in the Luke

5.5 Question Analysis

In this component, the user query is taken as an input and passes through two major and critical sub components called statistical question classification and query generation from which the subsequent components of the system are taking their inputs. Let"s go to the detail discussion on their implementation.

5.5.1 Statistical Question Classification

Classification is a supervised way of teaching machines with sufficiently enough amounts of training samples so that the machine can learn to classify test samples in an accurate manner. For question classification, we have prepared training data from the four question classes (person,

time, place, and quantity) to train the classifier. The classifier‟s accuracy will be presented in the Experiments chapter. But for now, let‟s take a look at the steps in training and testing the SVM_Light classifier.

a) Training the classifier

Steps:

1. Preparing the training set. Sample questions from the four question types, with a due consideration on the different ways of making questions, are prepared in this step by putting a single question in a file. For example, the following are sample questions from each type:

   -› የትግራይ ክልል ትምህርት ቢሮ ኃላፊ ማን ይባላሉ? **(person)**

   -› የኢትዮጵያ አየር መንገድ የማርኬቲንግ ትምህርት ቤት መቼ ተቋቋመ? **(time)**

   -› 19ኛውን የዓለም እግር ኳስ ዋንጫ ውድድር ያስተናገደችው አገር ማን ትባላለች? **(place)**

   -› በአፍሪካ በሳንባ ነቀርሳ ከተጠቁ ሰዎች መካከል ምን ያህሉ በኤችአይቪ/ኤድስ የተያዙ ናቸው? **(quantity)**

   Much of the questions are made from a news corpus that we have collected from the Ethiopian News Agency (ENA). The news itself was prepared in different categories as “sports”, “science & technology”, “accident”, “law & justice”, “education”, “health”, “politics”, “foreign relations”, “economy”, “culture & tourism”, “social”, and “environment” categories. Therefore, in preparing training questions that represent different aspects, the news corpus was a good source.

   As is discussed in the previous chapter, SVM can only work for numeric data. So, to use the SVM_Light classifier and prepare a classifier model, we need to convert these training samples into the input format the SVM requires. For doing so, the next steps are important.

   For a simplified discussion, we will consider a model prepared to classify questions into either „person‟ or „time‟(i.e. „person‟ vs „time‟classifier).

2. Create category vocabulary. This is a vocabulary made in each class by taking every term in all the questions of that class type. The format in the vocabulary is:

   [The term] [count (frequency) of the term in all documents of that class].

Here, the minimum frequency of a term is 1 (i.e., it at least appears once in one of the documents in that class). For example, Table 5.1 shows some vocabularies taken from „person‟ category:

Table 5.1: Sample vocabularies in category 'person'

Term	Frequency
የጋምቤላ	1
ዲን	2
ክልል	6
ኃይል	2
የመሬት	1
ቦርድ	1

In Table 5.1, the term “ክልል‟ appears 6 times whereas the terms “ዲን” and “ኃይል‟ appear twice and the terms “የጋምቤላ”, “የመሬት‟, and “ቦርድ‟ appear only once.

3. Create a global vocabulary. This is done by taking all the terms available in training questions found in either or both „person‟ & „time‟ classes and computing the occurrences of the terms. A global vocabulary is prepared as in the following format:

   [The term] [its number of appearances in „person‟ class documents] [its number of appearances in „time‟ class documents].

   Here, for classification purpose, we treat the first class („person‟) as positive and the second one („time‟) as negative.

Table 5.2: Sample global vocabularies in person and time classes

Term	Person frequency	Time frequency
ዲን	2	0
ክልል	6	2
ቀን	0	11
ተቋቋመ	0	2

Here in Table 5.2, the term “ዲን‟ appears twice in „person‟ and never in „time‟ class documents whereas the term “ክልል” appears 6 times in „person‟ and twice in „time‟ class documents.

4. Compute term frequency of every term. The term frequency of a term is the number of times the term appears in a document. So, for every term in all documents of both classes, we computed term frequencies and kept it in one file per document. Most often, terms have a chance of appearing only once in a question document.

5. Assign an id for every term and compute the terms‟ document frequencies in training sets of both classes. Assigning a numeric (integer) id is mandatory as the classification algorithm only entertains numbers for the classification. For this task, we can take the global vocabulary created in step 3 and a directory containing all question documents from both classes. Table 5.3 shows a sample output of this process:

Table 5.3: Terms, term ids, and document frequencies

Term	Term id	Total frequency
የጋምቤላ	70	1
ዲን	71	2
ክልል	72	8
ጋይል	73	2
የመሬት	74	2

The term ids are given starting from 1 for the first term in the global vocabulary and keeps on adding 1 until the last term.

6. Identify the best K features (terms) from each class which have an influence in classifying questions to either of the classes. The selected best features (terms) from each class are written into a file. This selection of best terms of one class type is done with respect to the other class being in consideration for the binary classification.

   Next, a collection of the best terms from both „person‟ and „time‟ classes are written into a file. Finally, the best terms will be assigned a new id starting from 1 to the count of best terms, the number of appearances of these best terms in the entire document collection will be counted, and written next to the terms id in another file.

7. Compute weights for each best term. The weight is calculated using the known formula as:

   $$Weight = tf * idf$$

   where,

tf (term frequency) is the number of times the term appears in a document.

Also, idf (inverse-document frequency) is formulated as:

idf=N/df

where,

N is the total number of documents in the training set and

df is the number of documents in which the term appears.

For normalizing the weight values, we use the logarithm function on *idf*. Therefore, the final weight value becomes:

*Weight= tf *log(idf)*

In this phase, the classes „person" and „time" are symbolized by 1 and -1 respectively and the terms identified as best terms of „person" with respect to „time" and vice versa are subject to the weight computation.

The class labels (1 or -1), term ids, and computed term weights form the weight file.

Finally, the last thing to do is to sort the weight file based on their class label and feature (term) ids and store as for example „SwtWhovsWhen". This is what is required to train the SVM_Light classifier.

Let"s have a look at a sample from the sorted weight file:

1 **2**:0.32248058871954294 **3**:0.1747801156704992 **54**:0.32248058871954294

Here, 1 is the label for class „person", 2 is the feature id of one of the best features, and 0.32248058871954294 is the weight computed by tf*idf values above. The same is true for the features with ids 3 and 54, their weights are shown next to the ids after a colon.

Training of the classifier ends up in creating the classification model when we write the following command to the SVM_Light tool:

svm_learn SwtWhovsWhen TrainWhovsWhen.model

Where, ***svm_learn*** is an SVM key word, ***SwtWhovsWhen*** is the sorted weight file created above, and ***TrainWhovsWhen.model*** is the classifier model created. We will use this model when we actually test the classifier to determine the question type of a user supplied query.

All the steps followed to construct the classification model for classes „person" and „time" are employed for other types. Totally, we have prepared 6 classification models for the four question types („person", „time", „place", and „quantity"). This is done iteratively by changing the parameters to form a new classifier model. It is one of the recommended ways of combining binary classifiers to form a multi-class classifier, called „one-against-one" method [23].

Let"s go through the steps we have passed in actually classifying the user"s question given to the question analysis component.

b) Testing the classifier

So far, the SVM based classifier is trained and classification models are built. Now is time to see how a given user question is decided to fall into one of the four factoid question types („person", „time", „place", and „quantity").

Steps:

i. Write the received question into a file.

ii. Compute term frequencies of terms in the question file.

iii. Using the term frequencies in step ii, compute weights for the question terms with respect to the term document frequencies created for all the possible class combinations in training the classifier (in step 6). And sort the weights too.

iv. By employing the generated classifier models from the training phase and the sorted weights in step iii, call the SVM"s classify method as:

svm_classify test.data TrainWhovsWhen.model test.result

where, ***svm_classify*** is an SVM command, ***test.data*** is the sorted weight data with respect to possible class combinations, ***TrainWhovsWhen.model*** is the classifier model developed at the end of the training phase, and ***test.result*** is a file to hold the final classification result.

These steps need to be repeatedly performed for all possible class combinations (six class combinations in our case for the four classes) and a separate result file should be kept for each iteration.

The classification result is a numeric value that shows to which question type the classifier classifies the given question. As is expressed in the training phase, the classes are represented in numbers. For example, the classification result of the question : "የኢትዮጵያ ዋና ከተማ ማን ይባላል?" is shown in Table 5.4.

Table 5.4: Classification results by SVM_Light

Classes involved	Classification Model	Result
Person vs Time	TrainWhovsWhen	0.35049575
Person vs Place	TrainWhovsWhere	-0.4494484
Person vs Quantity	TrainWhovsHowM	1.0758087
Time vs Place	TrainWhenvsWhere	-0.68862095
Time vs Quantity	TrainWhenvsHowM	0.75988091
Place vs Quantity	TrainWherevsHowM	0.96303829

Considering Table 5.4, when we see a positive result, it means the classification is into the first class, if negative, it is to the second class. Values close to -1 and 1 are indicating the classification is done with higher accuracy whereas, values near to 0 indicate classifications of poor accuracy resulting in classification into an „unknown" class type.

In our case we have taken results more than 0.5 and less than -0.5 as decisive ones to conclude the class type. The 0.5 and -0.5 are selected because, in the SVM classifier, absolute value of the classification result tells us the confidence of the classifier to put the given question into one of the classes. Hence, we need to have at least 50% of confidence in classifying.

After considering the six results, we have computed the frequency of occurrence of each class out of the six values and the one with higher frequency is supposed to be the question type for the given question. Otherwise, if there is no one class with highest frequency, the question type is decided as „unknown". This is the result expected when the question is out of Amharic factoid question types („person", „time", „place", and „quantity").

5.5.2 Query Generation

Once the question type is decided, the next step in question analysis is to regenerate the user's question in a form suitable to the subsequent components and to ensure the likelihood of finding a correct answer from the entire system. Let's go through the steps by assuming the user's question is "የአፍሪካ ሕብረት ጽ/ቤት የት ይገኛል?"

i. Remove the question mark and question particles like "ማን ይባላል ?" , "ምን ያህል ነው? ", and "የት ይገኛል?" because, these terms do not contribute to the task of document retrieval. This is quite similar to the stop words removal process we do for the crawled documents before indexing.

   We have removed these terms by preparing a list of such terms, checking the appearance of any of these terms in the question, and then removing them from the question if they

do exist. Therefore, after this step, the sample question "የአፍሪካ ሕብረት ጽ/ቤት የት ይገኛል?" becomes: "የአፍሪካ ሕብረት ጽ/ቤት".

ii. Parsing the query terms left after the question particles are removed in the above step. This is done by Lucene"s built in *QueryParser* class. The *QueryParser* turns readable expression into query instance. We instantiate an object from this class by passing the Amharic analyzer as one of the arguments and calling the *parse* method by sending the remaining query terms.

As is described in the previous chapter, the Amharic analyzer does the following important jobs:

- Tokenizing the query terms by using white spaces, tabs, new line markers and Amharic punctuation marks.
- Expanding terms written in short form. Abbreviation terms in Amharic are often abbreviated by "/" or ".". Therefore, we replaced these terms in the query by their expanded equivalent form by preparing such a list for the known Amharic abbreviations.
- Removing the stop words, if they do exist in the remaining query terms.
- Stemming query terms. The remaining non-stop words from the query terms will be stemmed to reduce them into their common form.

As a result of this step, the sample question becomes: "አፍሪከ ህብረት ጽህፈት ቤት" as all the query terms are stemmed into their stem form.

The most important thing is the Amharic analyzer we have employed here is the one that we have used in preprocessing crawled documents before they went through the indexing process so that both the query generated and the indexed documents are having similar view after going through the same Amharic analyzer.

The resultant query after all these steps becomes the one used for document retrieval by the Lucene"s index searcher as it is the final step in question analysis component. But the retrieval process needs to be initiated only if the resultant query is not null.

5.6 Document Retrieval

As Lucene"s IndexWriter is to pushing documents into an index file, IndexSearcher is to popping documents out of the index file. To get an IndexSearcher, we simply instantiate an IndexSearcher with a single argument that tells Lucene where to find an existing index. The argument is either of these two:

- A string containing a path to the file,
- A Lucene Directory object (mentioning the directory we have created to store the lucene index file while using IndexWriter; „C:/newindex“ for example.)

So, the IndexSeracher is instantiated as:

```
File indexDir = new File("C:/newindex");

Directory fsDir = FSDirectory.getDirectory(indexDir, false);

IndexSearcher is = new IndexSearcher(fsDir);
```

Then we feed the parsed Query to the IndexSearcher.search(). The return is a Hits object, which is a collection of Document objects for documents that matched the search parameters. The Hits object also includes a score for each Document, indicating how well it matched.

```
Hits hits = is.search(query);
```

is.search(Query) returns a "Hits" object, which is like a Vector, containing a ranked list of Lucene Document objects. These are the same Document objects we fed into the IndexWriter, but specifically the ones that matched our search.

By this, the document retrieval from the Lucene index is over. Now, we have the documents related to the user query and are found in the Hits object. It would have been enough for a search engine to return these Hits back to the user.

5.7 Answer Extraction

As explained in the previous chapter, pattern based matching technique combined with a proximity measurement between query terms and candidate answers is our approach in extracting answers from candidate documents.

Steps:

i. Acquire the expected answer type from the question analysis sub component and employ different pattern matching techniques according to the types.

ii. Extract candidate answer terms by applying the pattern matching technique for the question type in focus. If the expected answer type is „time“ or „quantity“, we have a regular expression to filter out candidate answer terms matching with the „time“ or „quantity“ expression from candidate documents. And, if the expected answer type is „person“ or „place“, we extract candidate answer terms matching with the list of person names or place names from the candidate documents. We do this for all candidate

documents in the Hit and put all these extracted candidate answer terms into a hash map for future processing.

iii. Filter the Hits based on their relevancy. Relevancy is decided based on the number of query terms found in the document. We used the criteria in Figure 5.5 for relevancy of documents.

```
Boolean relevantdoc=false;
if(qwordcount==QueryTerms)
        relevantdoc=true;
else if (QueryTerms<3&&qwordcount!=QueryTerms)
        relevantdoc=false;
else if(QueryTerms<=6&&qwordcount>=QueryTerms/2)
        relevantdoc=true;
else if(QueryTerms>=7&&qwordcount>=QueryTerms/3)
        relevantdoc=true;
else
        relevantdoc= false;
```

Figure 5.5: Algorithm to determine document relevancy

`QueryTerms` is the number of terms in the query, and `qwordcount` is the number of query terms found in the document.

If `qwordcount` is less than 3 terms, all of them must be found in a document so as to consider the document as relevant. If `qwordcount` is between 3 and 6, at least half of the query terms must be found in a document to be considered as relevant. If the terms in the query are 7 or more, at least, 1/3 of the terms must be found in the document to be relevant. Irrespective of the number of terms in the query, if all of the query terms are found in a document, the document is considered relevant.

iv. Compute proximity of candidate answer terms within a document (extracted in Step ii) to the matching query terms found in the document. We then pick the nearest term as a selected answer from the document in focus. This is done for all the top n Hits returned from the document retrieval component. Figure 5.6 shows the answer selection algorithm used to pick selected answer from relevant documents.

For all relevant documents

Step 1: Identify the query terms with a minimum distance to other query terms in the document.

Step 2: Compute distance (number of characters) between each candidate answer and selected query terms in *Step 1.*

Step 3: Pick the candidate answer term with the smallest distance as the ***selected answer*** from the document in focus.

Step 4: Give weight to the ***selected answer*** based on the number of query terms found in the document as:

```
If(qwordcount>(3/4*QTerms))
        weight=qwordcount*10
else if(qwordcount>(2/3*QTerms))
        weight=qwordcount*8
else if(qwordcount>=(1/2*QTerms))
        weight=qwordcount*4
else if(qwordcount<(1/3*QTerms))
        weight=0
```

Step 5: Put the ***selected answer*** and its ***weight*** into a hash map.

Figure 5.6: Answer selection algorithm

For all the documents considered relevant (decided in step iii), the answer selection algorithm in Figure 5.6 starts by identifying query terms (which might appear more than once in a document) which are close to other query terms in the document. This is useful for selecting the query terms in the document that will be used as a benchmark from which the distance for candidate answer terms is measured. Specially, in a relevant document having query terms appearing more than once, it would be difficult to measure the closeness of the candidate answer terms to such a query term unless one of the repeated query terms is chosen for this task. This is done by checking the proximity of the repeated query term to other query terms and the one with highest proximity will be picked to represent the repeated query term.

In Step 2 of Figure 5.6, the distance between the selected query terms and candidate answer terms will be measured. And then, in Step 3, the candidate answer with the least distance will be labeled as the selected answer for the current relevant document.

To simplify ranking of answers selected from the relevant documents, in Step 4, we compute a weight for each relevant document‟s selected answer based on how much of the terms in the query are present in the relevant document. The selected answer and its weight will then be stored in a hash map to be used in the next step.

v. Pick the top n ***selected answers*** with the highest ***weight*** from the hash map (constructed in Step iv). This is the list of answers finally to be returned to the user. Here, we picked the top 5 answers in our case.

Chapter Six - Experiment

6.1 Crawling Evaluation

We have used the JSpider crawler for crawling and downloading web pages from the given seed URLs. As an example, let"s see its performance in crawling the website of Ethiopian Radio and Television Agency (www.erta.gov.et).

The crawling was running in an ordinary PC in the computer lab (2.6 GHz speed, 3 GB RAM) and the Internet connection was the usual intermittent connection.

The JSpider"s final statistics after running for about 55 minutes reveals the following:

2969 – Visited URLs of which:

2188 – Parsed URLs

781 – Parse Ignored URLs

The crawler has an efficiency of crawling and downloading about 40 pages per minute.

6.2 Language Identification Module (LIM) Evaluation

The Language Identification Module (LIM) is trained with the universal declaration for Human rights (UDHR) document prepared in Amharic with UTF-8 Unicode encoding and other purely Amharic content pages.

The LIM, by default, is trained with 322 world languages. Once after we have trained it with the above document for Amharic Language, the LIM is able to identify documents quite perfectly and return results with the list of languages (scripts) which are related to the contents in the test document, expressed in percentage as shown in Table 6.1.

Table 6.1: Part of the Language Identification Module Identification output

Language	Script	Encoding	Total	Matched	Percent
Amharic	Ethiopic	UTF8	2490	1780	71.49%
English	Latin	Latin1	2490	146	5.86%
Guarani	Latin	Latin1	2490	144	5.78%

Table 6.1 shows the top three rows of the identification output for a certain document having the majority of its content written in Amharic. The test document under consideration has a match of

1780 characters out of the total 2490 found in the document (71.49%) to Amharic, 146 characters out of the total 2490 characters (5.86%) to English, 144 characters out of the 2490 (5.78%) match to characters in Guarani language, and so on. This is due to the similarity of the characters found in the document and the characters found in the training document which was used to train the language (script) identifier for these languages. Therefore, its performance is quite dependable as it is merely based on the content of the test document.

We have tested the identifier for 6257 html pages with crawled and downloaded from Addis Admas and Ethiopin Reporter Amharic news sites. It has identified with 100% accuracy.

But, given web pages with majority of its contents written in other Ethiopian languages which use the same character set as Amharic, the identifier will also classify them as an Amharic one since they use the same Geez alphabets. This might not be a major problem for us since the Language (script) identification is done after the pages are downloaded by supplying seed URLs which are believed to contain majority of Amharic contents only.

Even if the supplied seed URLs have a document written in Tigrigna or other Ethiopian languages using the same character set as Amharic, it would be filtered out by using the most commonly used stop words list in Amharic to further deny documents written in other languages from being identified as Amharic.

6.3 Question Classification Evaluation

The SVM based automatic question classifier was used to classify user questions to one of the four Amharic factoid question types („person", „time", „place", and „quantity"). As we have used the one-against-one implementation of the binary classifier to classify into the four classes, we need a total of 6 classifiers trained and tested (i.e., n (n-1)/2 classifiers) in a single experiment, where n=number of classes, hence 4.

While evaluating the performance of the question classification, we have followed a 10-fold cross validation technique. In this technique, ten separate experiments are done by reserving one tenth of the data (120 questions) for testing and using the rest of the data (1080 questions) for training the classifier in each experiment. We have used 6 classifier models in a single experiment, resulting in a total of 60 models in the ten experiments.

By supplying 30 test questions from each class, we have recorded the performance of the classifier in each experiment and then took the average of the ten experiments to report the actual performance of the question classifier in each of the four question types.

While training the classifier models in each experiment, the SVM_Light classifier"s learning procedure has identified the available support vectors, the error, recall, and precision of the classifier models being developed.

Table 6.2 shows the average performance of these classifiers to classify any given question into one of the two classes used to construct the classifier model, as is displayed in the summary of the SVM_Light‟s learning procedure.

Table 6.2: Average Performance of classifier models

Classifier Name	**Number of Support Vectors Found**	**Error (in %)**
Person Vs Time	344	1.85
Person Vs Place	339	6.50
Person Vs Quantity	334	1.28
Time Vs Place	324	2.11
Time Vs Quantity	358	2.81
Place Vs Quantity	322	1.55

As it is shown in Table 6.2, the largest error observed is 6.50% which appears in the „Person Vs Place‟ classifier. This is due to the similarity of „person‟ and „place‟ type questions found in the training set. In formulating questions of type „person‟ and „place‟, there are key terms we use in both types as we used to ask questions like:

“What is the capital city of Ethiopia?” (“የኢትዮጵያ ዋና ከተማ ማን ይባላል?”) – Place type, and

“Who is the Minister of Ministry of Health?”(“የጤና ጥበቃ ሚኒስቴር ሚኒስትር ማን ይባላሉ?”)- Person.

We mostly use the term “ማን” in both cases. That is the main Support Vector which is found in training examples of both classes and making this much error.

While testing the SVM classifier models, we have used 30 test questions known to be in each of the question types but not used to train the classifiers in each experiment. The average test results are shown in Table 6.3.

Table 6.3: Question classification average test results

Question Type	**Correctly Classified**	**Wrongly Classified**
Person	28 (93.3%)	2 (6.7%)
Time	29 (97.0%)	1 (3.0%)
Place	28 (93.3%)	2 (6.7%)
Quantity	28 (93.3%)	2 (6.7%)

As Table 6.3 shows, 28 out of 30 Person, Place, and Quantity questions are classified in their correct classes while 29 out of 30 Time type questions are classified in their correct class. From this, we can say that we have acquired a classification performance from the SVM classifier

giving 94.2% accuracy on average. Sample questions and classification results from one of the experiments are attached in Appendix A.

We have also performed the 10-fold cross validation to check the performance of Seid Muhie‟s [4] rule based question classifier by employing the test questions we used to test our classifier. It has shown a good classification performance on Person, Time, and Quantity question types but worst in correctly classifying Place question types. On average, the rule based question classifier has classified 86.9% of the questions correctly. Besides the advantages of employing machine learning based classifier over the rule based one, we observed that the classification performance of the rule based classifier is less than that of the SVM based classifer.

6.4 Document Retrieval Evaluation

The document retrieval component can return all the documents which contain any of the query terms as long as the query terms are written in the same spelling as that of the terms found in the indexed documents. We have proved this by cross checking the documents which are returned by the document retrieval component against the original documents containing the query terms.

The Amharic stemmer that we have used in the indexing could decrease the document retrieval‟s performance because of the way it stems certain words like (“የበራ”, “በሬ”, & “በር”), (“ታክሲ” & “ታክስ”), and (“ቢጋ” & “በግ”). For example, the stemmer gives a stem word “ታክስ” for both “ታክሲ” and “ታክስ”. So, when we use one of such terms in the query, the document retrieval module would have returned documents containing other terms with similar stem as the term we used in the query.

But not all of the documents retrieved are relevant for our case. The decision on relevancy of a retrieved document is presented in section 5.7 which decides a retrieved document as relevant or not based on the number of query terms found in the document, the more query terms it contains, the more likely to be relevant.

Also, as document retrieval is done next to the question classification task, the above problem created by the stemmer is not likely to happen because the given query should be classified into one of the question types before invoking the document retrieval component. Hence, for example, giving single term queries using terms like the above ones would end up in returning a “ይቅርታ ለጥያቄዎ መልስ አልተገኘም” message and the system halts before calling the document retrieval component as it couldn‟t classify the question into one of the four question types. Figure 6.1 shows the screen shot of the system answering a single term question “ታክስ”.

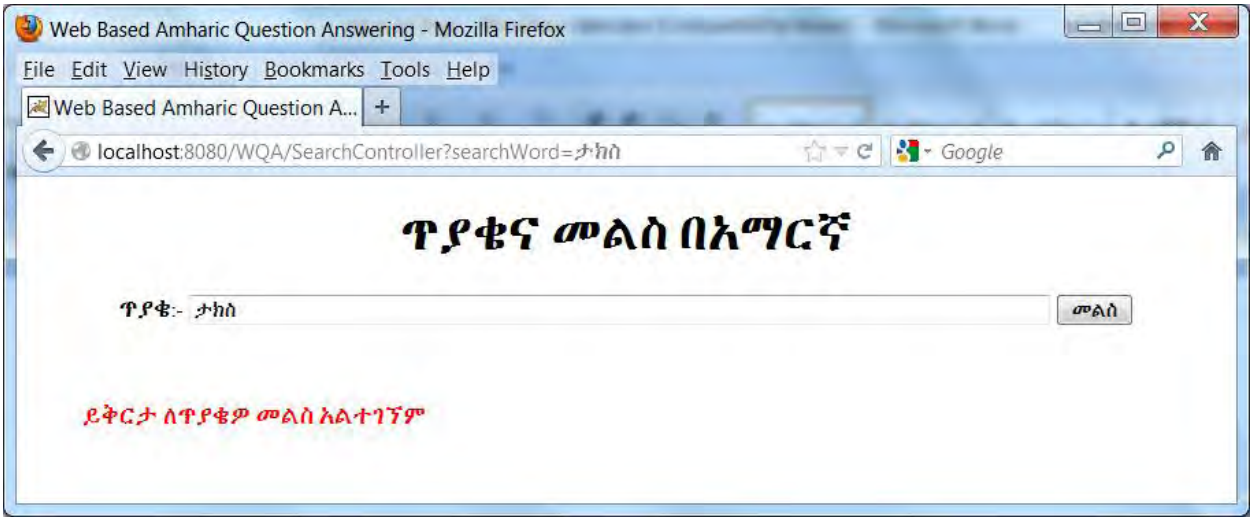

Figure 6.1: Screen shot of answering single term questions

Therefore, the document retrieval component is returning all documents relevant to our case and we pick the top 5 relevant documents for further processing with the answer extraction module. But, as we don‟t have used synonyms of the query terms, here relevancy of a document doesn‟t take into consideration the documents containing synonym terms.

6.5 Answer Extraction Evaluation

The answer extraction component is the one responsible to extract candidate answer terms from the relevant documents, and select the best answers out of the candidate answer terms in each of the top 5 relevant documents.

For “Person” and “Place” question types, we have used a list of person names and a list of place names respectively and for “Time” and “Quantity” question types we have used a regular expression to extract “Time” and “Quantity” terms from the relevant documents.

Let‟s see first the performance of the above two answer extraction approaches used to return candidate answers and then we will evaluate the efficiency of the best answer selection method.

We used the common performance evaluation concepts in information retrieval, precision and recall. Precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved.

Therefore, in the answer extraction, precision is the fraction of extracted candidate answer terms that are relevant to the question type in focus. It is formulated as:

$$\text{Precision} = \frac{\text{Extracted AND Relevant}}{\text{Extracted}}$$

Recall is the fraction of the extracted candidate answer terms that are relevant to the question type that are successfully extracted. For example, when the question type is „Person", it shows how much of the relevant person names are successfully extracted from the candidate documents. Hence, it is formulated as:

$$\text{Recall} = \frac{\text{Extracted AND Relevant}}{\text{Relevant}}$$

Table 6.4 shows evaluation result of candidate answers extraction For "Person" and "Place" question types.

Table 6.4: Precision and Recall of answer extraction for Person and Place Question Types

Question Type	**Precision**	**Recall**
Person	55.7%	77.5%
Place	62.8%	68.3%

As Table 6.4 shows, the precision and recall of extraction of answer terms for "Person" types is 55.7% and 77.5% respectively. While "Place" types show 62.8% precision and 68.3% recall. This low performance is due to the following reasons:

- Ambiguity of Amharic nouns as they can also be used as verbs and adjectives. For example, person names like "አበራ", "ሙሉ", "መሰረት", and "ጉዳይ" can also be used as verbs and adjectives. Person names are also interchangeably used as place names like school names, town names and so on. For example, the screen shot of the system in Figure 6.2 shows that the correct answer selection is affected due to the selection of the term "በላይ", which is not actually a name in the context of the document; in the document, the term "በላይ" appears in the sentence "የብራና ፅሁፎች የሚፃፉት ከዋና መስመሩ **በላይ** በተደራረቡ ምልክቶች አማካይነት ነው". This wouldn"t have been a problem in the presence of a part of speech (POS) tagger.

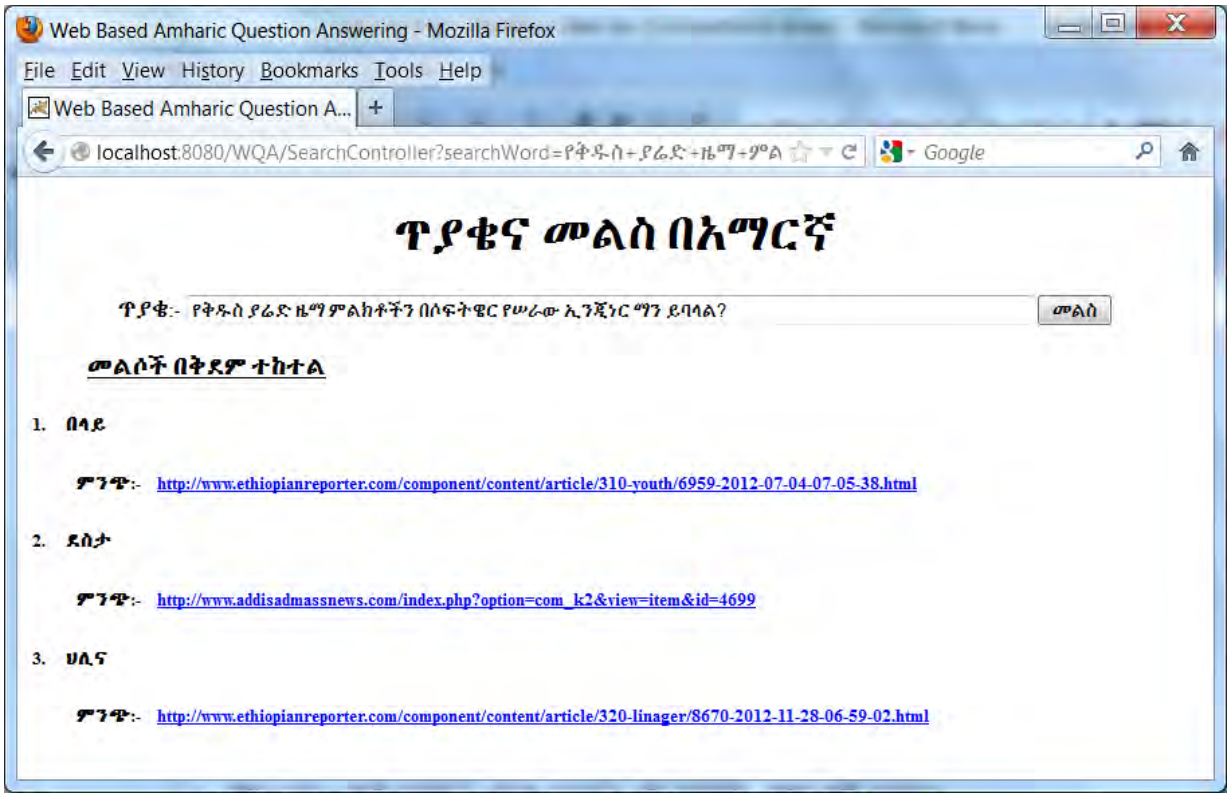

Figure 6.2: Screen shot of a wrong answer due to ambiguous names

While calculating the distance for the extracted answer terms, the system has the following results showing extracted candidate answers together with their distance from query terms in the document in focus:

{ስራው =2507, መስፍን አማዴ =391, አብነት =1569, መሰረት =1301, መስፍን =177, ይሁን =3051, በላይ =59, ታሪክ =1917, ድረስ =3197, ስነ =1539, አዲስ አበባ =281}

So, as the system only takes a single answer from a relevant document in focus, the correct answer "መስፍን" would have been picked as an answer if the term "በላይ" were not in the list.

- The person names list we have prepared mostly contains Ethiopian nouns. Hence, a document containing person names used in foreign countries wouldn't be extracted. For example, Figure 6.3 shows a screen shot of the system giving a wrong result due to the absence of foreign names in the names list we used in the system.

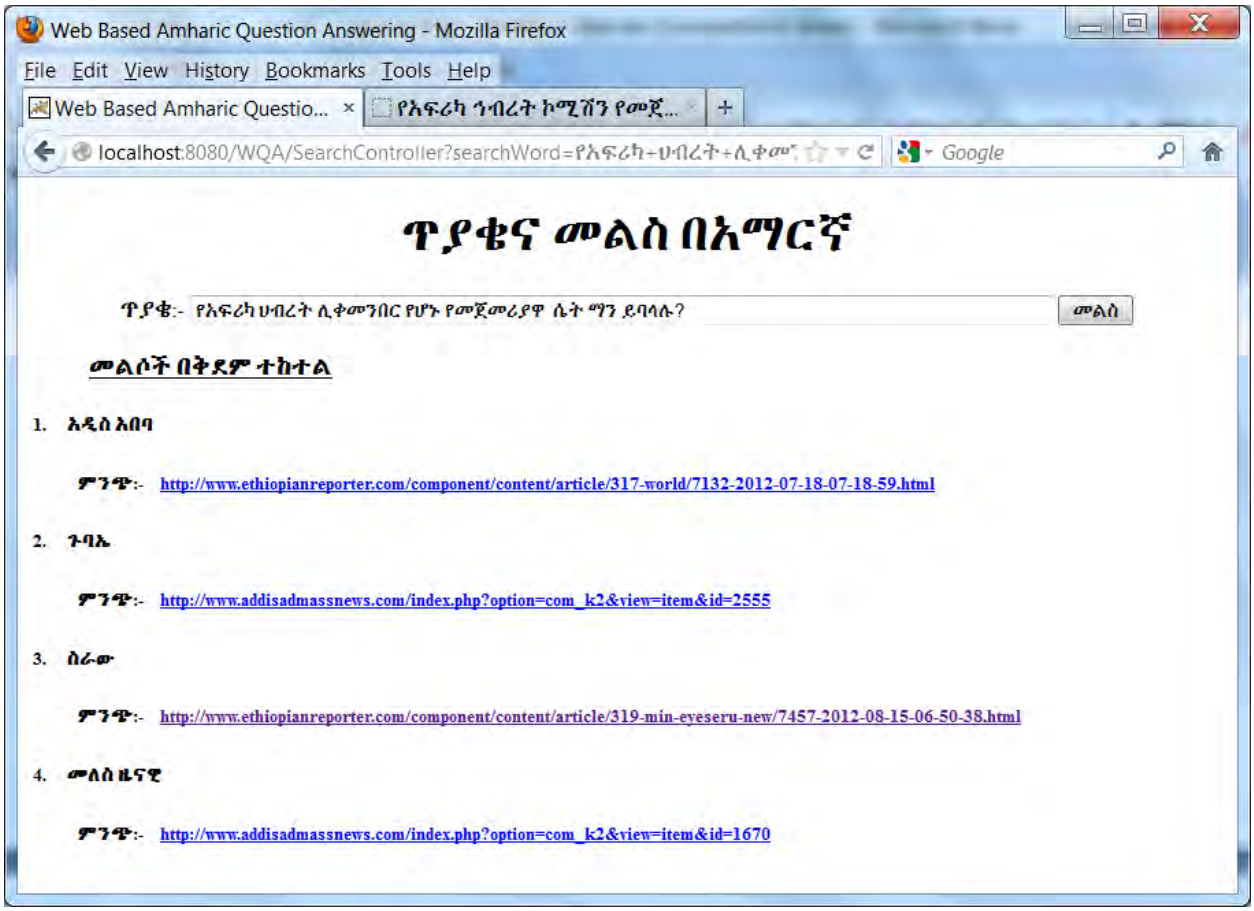

Figure 6.3: Screen shot of a wrong answer due to absence of foreign names

Here, the top ranked relevant document has the following two sentences which are used to determine the selected answer:

"በአራተኛው የምርጫ ውድድር ለማሸነፍ የበቁት **ዶ/ር ዙማ** ለአፍሪካ ኅብረት ኮሚሽን የመጀመሪያዋ ሴት ሊቀመንበር ሆነዋል። **በአዲስ አበባ** በተካሄደው የአፍሪካ ኅብረት የመሪዎች ጉባዔ፣ እሑድ ሐምሌ 8 ቀን 2004 ዓ.ም. ለሰኞ አጥቢያ በተደረገው ዝግ ስብሰባ ለሊቀመንበርነት ያሸነፉት **ዶ/ር ዙማ**፣ ትናንት ሐምሌ 10 ቀን 2004 ዓ.ም. ቃለ መሐላ ፈጽመዋል።"

Including the non Ethiopian name "**ዙማ**" in the names list would result in the 1st ranked answer to be "**ዙማ**" than "**አዲስ አበባ**". But, it is not easy to include all non Ethiopian names in the names list.

For the "Time" and "Quantity" types of questions, the answer terms extraction performance is shown in Table 6.5.

Table 6.5: Precision and Recall of answer extraction for Time and Quantity Question Types

Question Type	Precision	Recall
Time	71.1%	78.9%
Quantity	59.7%	87.3%

As is shown in Table 6.5, the precision values are 71.1% and 59.7% for Time and Quantity types respectively. This is due to the interchangeability of numeric terms like years in the regular expressions used to extract candidate answer terms of Time and Quantity types.

In general, on average, extraction of candidate answer terms from relevant documents has a precision of 62.3% and a recall of 78%.

The final task of answer extraction component is selecting the best answer from candidate answer terms in each relevant document. This task would be highly affected by the performance of the candidate answer terms extraction task since the nearest candidate answer to the query terms would be selected as the best answer as discussed in Section 5.7.

We have used 100 test questions (25 from each type) prepared from the indexed news documents to test the answer selection performance of the system. We found out that the answer selection task has the accuracy of 77% on average for the four question types. While our system returns top 5 answers to the given question, most of the correctly answered questions we have used in the test have answers in the first three ranks. Figures 6.4, 6.5, and 6.6 show screen shots of the system returning the correct answer in the 1st, 2nd, and 3rd ranks respectively. The 100 test questions, their expected correct answers, the list of returned answers, and the ranks of correct answers returned by the system are shown in Appendix B.

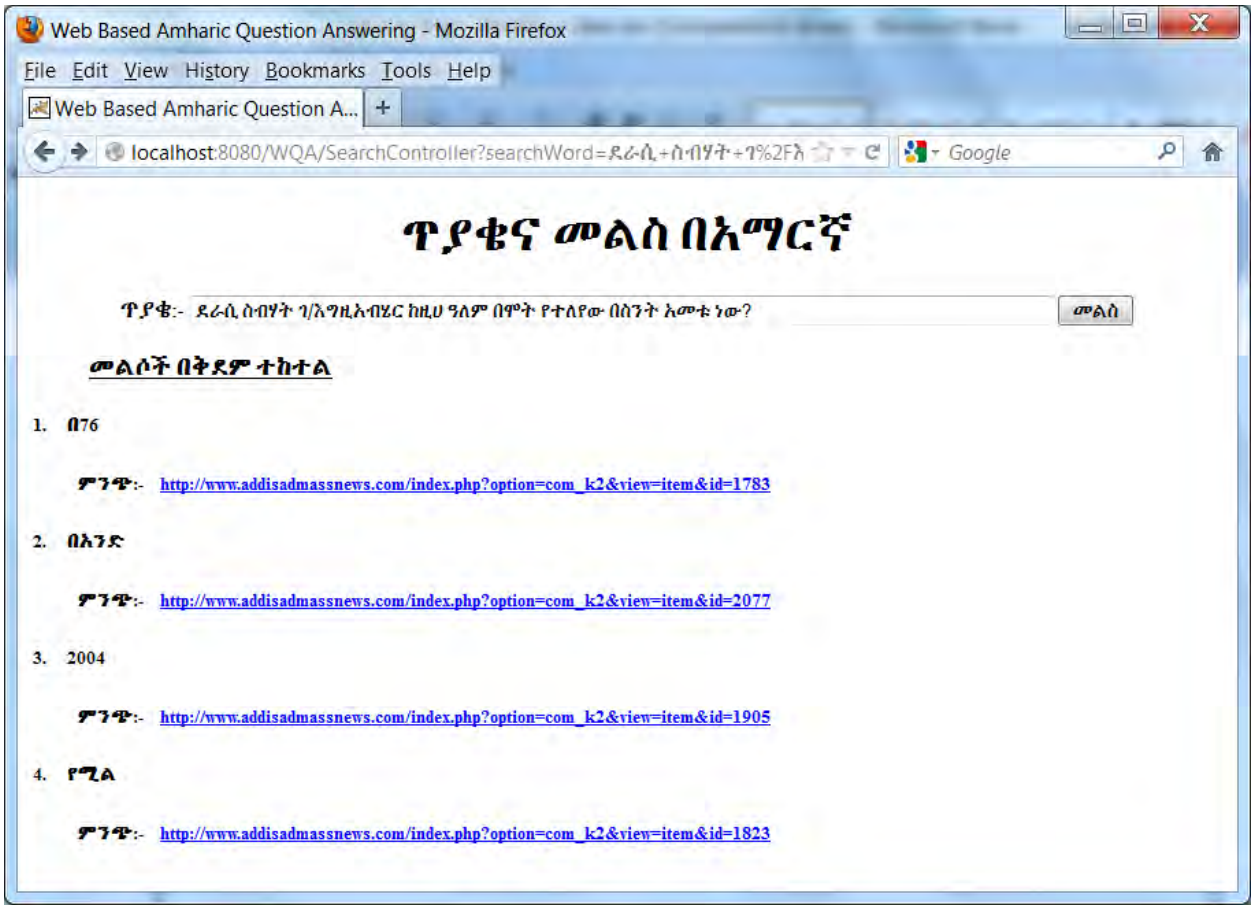

Figure 6.4: Screen shot of the system returning a correct answer in the 1st rank

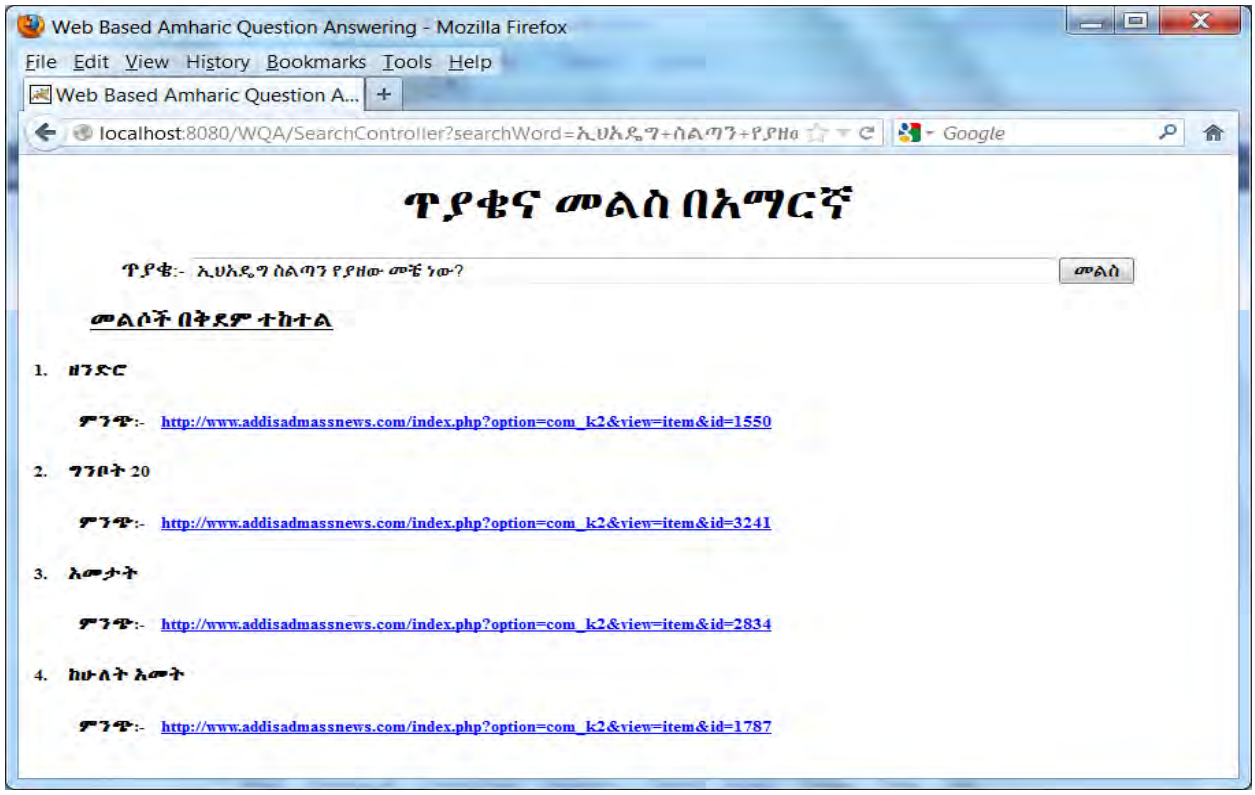

Figure 6.5: Screen shot of the system returning a correct answer in the 2nd rank

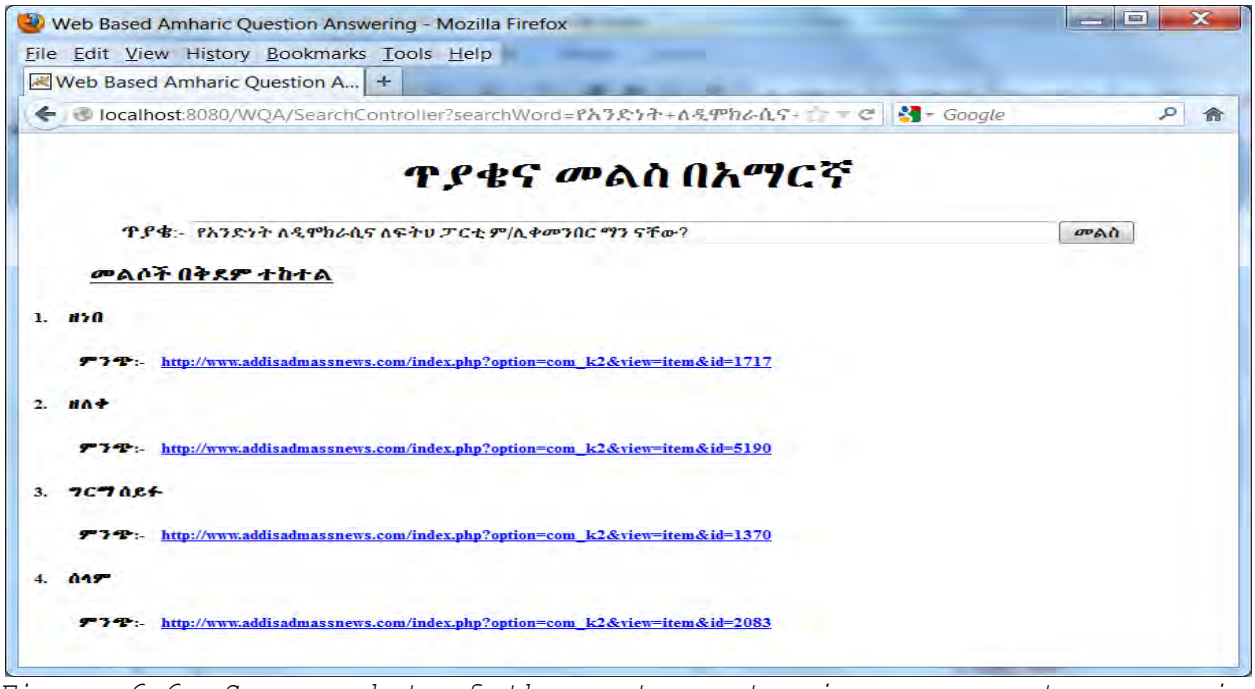

Figure 6.6: Screen shot of the system returning a correct answer in the 3rd rank

Chapter Seven - Conclusion and Future Work

When users need for a certain fact and try requesting search engines for it, they get back a bunch of addresses and snippets which are „related" to their need and it is up to the user to decide which address to choose expecting that the requested fact could be found there. Opening the address could present the user with lots of pages of information and it is the user"s duty to go through the information and extract the actual fact. This is how ordinary search engines help users in need of facts. What they do is accept the users" query, search documents in their repository which contains any of the words in the query, rank the retrieved documents and present to the user with title of the page, a snippet, and address (URL) of the page included in the response.

With the help of natural language processing, the information extraction is performed automatically and the user will be presented with the list of answers believed to fulfill the users" request. Most importantly, the users will be more pleased if the computer serves their fact inquiry needs like human beings i.e., if they could be able to communicate with the machine as if they are talking to someone else using their own natural language. This is the task of Question answering systems like ours.

Question answering systems could use preformatted corpus or crawled web documents as a search space or they may incorporate third party search engine tools to send their request to and work on the search results to extract the necessary fact.

7.1 Conclusion

The web based Amharic factoid question answering system has two major parts, the search engine part and the question answering part.

As the corpus used is gathered from Amharic web sites, we used a crawler to download pages from these sites. Then, the downloaded pages will go through an identification phase to filter out non-Amharic content pages from our search space. This is the language (script) identification phase. The Amharic documents are then indexed in a way to facilitate information retrieval later in the system. When the search space is ready, then the question answering part will start playing its role serving the users.

The question answering part has three main components, namely, question analysis, document retrieval and answer extraction.

The question analysis component takes in user"s request, identifies the type of fact the user needs as person, time, place, or quantity. This is called question classification and it is done in a

machine learning (statistical) approach. Furthermore, question analysis component regenerates user‟s question to make it suitable for getting better result out of the document retrieval phase.

Document retrieval is the component which receives generated queries from the preceding question analysis component and looks for documents containing information related to what the user has asked. This searching is made in the index directory containing indexed documents. When relevant documents are found, they will be ranked and returned to the answer extraction component.

The answer extraction is a component responsible to dig out the fact which is believed to be an answer to the user‟s question and to return these facts in a certain order back to the user. While trying to extract facts from relevant documents, the type of the question the user posed is of great importance as extracting name of a person from the documents wouldn‟t be of any use to the user looking for name of a city.

The first thing that we use is to prepare the search space containing Amharic web pages downloaded from known Amharic news sites. This is done by customizing the open source web crawler, JSpider. We then identified pages containing majority of their content written in Amharic language and put into our final repository. This is possible using another open source tool called Language Identification Module (LIM). Documents passing the identification process are indexed and made ready for a document retrieval process using the known open source tool called Lucene. Hence, the document retrieval is an index searching process.

For natural language processing applications such as question answering, the importance of other fundamental language processing modules like part of speech tagger (POS), named entity recognizer (NER), and stemmer is unquestionable. With the absence of these such modules in Amharic language, Seid Muhie [4] has designed the first Amharic factoid question answering system using a rule based approach employed in the question analysis component resulting in 89% of correct question type identification; as is reported in [4]. The system had also a performance of 97% in retrieving related documents. Using a gazetteer in place of the named entity recognizer and a regular expression, the researcher has reported a 72% overall system performance in correctly answering user questions.

This work of ours aims to apply a machine learning approach to the question analysis component hoping to improve precision of the question classification task in question analysis, whose accuracy impacts the accuracy of the entire system.

With application of a known machine learning based classification algorithm, support vector machine (SVM), we attained an accuracy of 94.2% in question classification.

Besides, we tried to incorporate a web crawler in our system to help automatically gather web documents and prepare a search space. With the integration of the JSpider based web crawler, we are able to download Amharic content pages from known Amharic news sites and keeping the

crawling process active while periodically running the language (script) identification and indexing processes would enrich our search space resulting in the likelihood of answering Amharic factoid questions.

The top 5 relevant documents are used in our system. Therefore, among the top hits returned by the document retrieval component, the top 5 are believed to contain much of the query terms and have potential to contain the correct answer.

Extracting correct answers, given relevant documents with the potential to contain correct answers, requires extracting candidate answer terms from documents and picking up one best answer among the candidates which is closer to the query terms in the document than other candidate answer terms. Our system‟s precision in extracting candidate answer terms from relevant documents is 62.3% on average for the four question types. This is mainly due to the absence of part of speech tagger and named entity recognizer which could have helped a lot for identifying the part of speech where a term is used in a sentence and named entities found in the document. But, the selection of best answer out of the candidate answer terms is done by computing the character distance between candidate answers and query terms in the document. In this regard, our system is able to pick out 77% of answers correctly.

7.2 Contribution of the work

Besides the modifications done in improving the efficiency of algorithms used in the previous Amharic factoid question answering system [4], we have following contributions:

- The study has implemented question answering techniques used in other languages.
- The study has integrated a search engine component with that of a question answering system for enriching the search space.
- The system has customized different open source tools to make a functional prototype.
- The system has implemented a machine learning based automatic question classification which simplifies the task of working on other question types other than the four factoid question types studied here.
- The study is designed to be a web based application which gives end users with a web based interface for interaction with the system.
- The study has identified important natural language processing tasks which could improve the performance.

7.3 Future work

There are fundamental natural language processing tasks which not only improves the Amharic question answering system but also serve as a pillar in getting the best out of Amharic based natural language processing applications.

Next is the list of tasks we recommend to be done in future for improving the performance of Amharic question answering system.

- Experimenting how the automatic question classification could perform using machine learning algorithms other than Support Vector Machine (SVM). For other languages, SVM yields a best classification performance than other machine learning algorithms like Naïve Bayes, Neural networks, and decision trees [11, 24, 37]. Though we believe the same could happen in Amharic question classification, we didn‟t make experiments. So, we believe it is worth experimenting and use other algorithms for the question classification task if it turns out to outsmart the SVM one.
- Working on other question types. Our research aims in answering factoid questions of type “Person”, “Place”, “Time”, and “Quantity”. But there are other question types like “Definition”, “List”, “How”, “Acronym”, and so on. So, we believe they are potential concepts for research in having a full-fledged question answering system.
- Implementing the question answering system in areas like organizations‟ user support, in online education systems, and so on. We believe the question answering system could help in automating such application areas and serve with a good performance towards answering the four factoid questions types.
- Improving the Amharic stemmer. The stemmer has an important role in natural language processing applications like ours. But the Amharic stemmer we have used is not of the state of the art one. Hence, improving the stemmer will hopefully bring a plus in the performance of such systems.
- Developing Amharic part of speech tagger and named entity recognizer. Though, there had been researches undertaken to develop such natural language applications for the Amharic language, to our knowledge, there are no such systems available to integrate to our system. So, to improve the performance of a question answering system and to support other Amharic based natural language processing applications, their role is unquestionable.
- Developing Amharic spelling checker. In our system, if a user commits a spelling error while posing a question, the system will blindly start processing the user request without checking the correctness of the words in the query. So, it may probably end up responding with a wrong answer. But, if there is such a system to check the input query for its correctness according to the language‟s rules, it will definitely improve our system.

- Developing Amharic WordNet. Had there been an Amharic WordNet developed, it would be easy for systems like ours to better understand the users‟ intentions and return to them with as much possible answers; taking into consideration the synonyms of the terms in the query.

References

[1] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman, Natural language processing: an introduction, J Am Med Inform Assoc, 2011.

[2] http://www.wisegeek.com/what-is-question-answering.htm, Last accessed on Sept 29, 2011.

[3] http://start.csail.mit.edu/ - START (SynTactic Analysis using Reversible Transformations) – The world‟s first web-based Question Answering system, Last accessed on Sept 29, 2011.

[4] Seid Muhie, 2009, Automatic Amharic Question Answering for Factoid Question Types, Master‟s Thesis, Addis Ababa University, Unpublished.

[5] Hwee Ton Ng, Leong Hwee Teo, and Jennifer Lai Pheng Kwan, A Machine Learning Approach to Answering Questions for Reading Comprehension Tests, DSO National Laboratories , 20 Science Park Drive, Singapore.

[6] Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin, Question Answering in Webclopedia, Information Sciences Institute, University of Southern California.

[7] Tessema Mindaye, 2007, Design and Implementation of Amharic Search Engine, Master‟s Thesis, Addis Ababa University, Unpublished.

[8] Nick Kewney, Using the web as a knowledge base for question answering, http://www.kewney.com/posts/technology/, accessed on January 21, 2012.

[9] Marcin Skowron and Kenji Araki, Effectiveness of combined features for Machine Learning based question classification, Journal of natural language processing, vol. 12, no. 6, Nov. 2005.

[10] Thamar Solorio, Manuel Pérez-Coutiño, Manuel Montes-y-Gómez, Luis Villaseñor-Pineda, and Aurelio López-López, A language independent method for question classification, Language Technologies Group, Computer Science Department, National Institute of Astrophysics, Optics and Electronics, Puebla, Mexico.

[11] Dell Zhang and Wee Sun Lee, Question classification using Support Vector Machines, Department of Computer Science, School of Computing, National University of Singapore, ACM , July 28-August 1, 2003, Toronto, Canada.

[12] Matthew W. Bilotti, Boris Katz, and Jimmy Lin, What works better for question answering: Stemming or morphological query expansion?, In Proceedings of the Information Retrieval for Question Answering (IR4QA) Workshop at SIGIR 2004, July 2004, Sheffield, England.

[13] Atelach Alemu Argaw and Lars Asker, An Amharic Stemmer : Reducing Words to their Citation Forms, Proceedings of the 5th Workshop on Important Unresolved Matters, pp 104–110, Prague, Czech Republic, June 2007, Association for Computational Linguistics.

[14] Dan Roth, Gio Kao Kao, Xin Li, Ramya Nagarajan, Vasin Punyakanok, Nick Rizzolo, and Wen-tau Yih, Cecilia Ovesdotter Alm, and LIam Gerard Moran, Learning components for a question-answering system, Department of Computer Science and Department of Linguistics, University of Illinois, Urbana Champaign.

[15] Li Xin, Research of passage retrieval for question answering system, PhD Dissertation, City University of Hong Kong, May 2010.

[16] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton, Quantitative evaluation of passage retrieval algorithms for question answering, In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 2003, Toronto, Canada.

[17] David Dominguez-Sal and Mihai Surdeanu, A Machine Learning Approach for Factoid Question Answering, Universitat Politécnica de Catalunya, Barcelona.

[18] Mengqiu Wang, A Survey of Answer Extraction Techniques in Factoid Question Answering, School of Computer Science, Carnegie Mellon University, Association for Computational Linguistics, Volume 1, Number 1, 2006.

[19] Diego Mollá, Menno Van Zaanen, and Daniel Smith, Named Entity Recognition for Question Answering, Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006), pp 51–58.

[20] Gautam Pant, Padmini Srinivasan, and Filippo Menczer, Crawling the web, the University of Iowa, USA and Indiana University, Bloomington.

[21] V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.

[22] Steve Gunn, Support Vector Machines for Classification and Regression, ISIS Technical Report, Image, Speech and Intelligent Systems Group, University of Southampton, 14 May, 1998.

[23] Chih-Wei Hsu and Chih-Jen Lin, A Comparison of Methods for Multi-Class Support Vector Machines, Department of Computer Science and Information Engineering, National Taiwan University.

[24] Thorsten Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features, Universität Dortmund Informatik LS8, Dortmund, Germany.

[25] Kevyn Collins Thompson, Jamie Callan, Egidio Terra, and Charles L.A. Clarke, The Effect of Document Retrieval Quality on Factoid Question Answering Performance, School of Computer Science, Carnegie Mellon University, and University of Waterloo, Canada, ACM, July 25–29, 2004.

[26] Christof Monz, From Document Retrieval to Question Answering, ILLC Dissertation Series DS-2003-4, Institute for Logic, Language and Computation, Universiteit van Amsterdam, 11 December, 2003.

[27] Lili Aunimo, Methods for Answer Extraction in Textual Question Answering, Series of Publications A Report A-2007-3, Department of Computer Science, University of Helsinki, Finland, 2007.

[28] L. Hirschman and R. Gaizauskas, Natural Language Question Answering: the view from here, Natural Language Engineering PP. 275-300, Cambridge University Press, 2001.

[29] Wei Li, Question Classification Using Language Modeling, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts.

[30] Nega Alemayehu and Willet P., Stemming of Amharic Words for Information Retrieval, In Literary and Linguistic Computing, Oxford, Oxford University press, Vol. 17, No.1, pp 1-17, 2002.

[31] JSpider User Manual, Version 0-5-0-dev, http://j-spider.sourceforge.net

[32] Amiya Patanaik and Sudeshna Sarkar, Open Domain Factoid Question Answering System, Department of Electrical Engineering, Indian Institute Of Technology, Kharagpur, India, MAY 2009, BTec Thesis, Unpublished.

[33] Lucene Tutorial By Steven J. Owens from http://darksleep.com/lucene/ accessed on October 01, 2012.

[34] Sandeep Sharma and Ravinder Kumar, Web-Crawling Approaches in Search Engines, Master‟s thesis, Department of Computer Science and Engineering, Thapar University, Patiala, June 2008.

[35] Language Observatory, How does Lim Work: Explanation of LIM, 2006.

[36] Shanjian Li, Katsuhiko Momoi, A Composite Approach to Language/Encoding Detection, In Proceedings of the 19th International Unicode Confrence, San Jose, USA, September 2001.

[37] István Pilászy, Text Categorization and Support Vector Machines, Department of Measurement and Information Systems, Budapest University of Technology and Economics.

[38] http://www.cs.ru.nl/~biniam/geez/crawl.php , last accessed on Feb 11, 2013.

Appendices

Appendix A

This appendix shows the 30 test questions in each of the question types and their resulting question types as automatically classified by the SVM.

Person Question Types

```
{
   1. ሁለተኛው የኢትዮጵያ ህዝብ ብሄራዊ መዝሙር ግጥም ደራሲ ማን ይባላሉ? =Person,
   2. ግብጻዊ የተባበሩት መንግስታት ድርጅት መሪ የነበሩ ሰው ማን ይባላሉ? =Person,
   3. የአምቦ ከተማ ከንቲባ ስማቸው ማን ነው? =Person,
   4. የጤና ጥበቃ ሚኒስቴር ሚኒስትር ማን ይባላሉ? =Person,
   5. የመጀመሪያው ፓኪስታናዊ የኖቤል ተሸላሚ ማን ይባላል? =Person,
   6. የደቡብ ክልል ርዕሰ መስተዳድር ማን ይባላል? =Person,
   7. በሮም የማራቶን ውድድር በባዶ እግሩ ሮጦ ያሸነፈው ኢትዮጵያዊ ማን ይባላል? =null,
   8. ኢትዮጵያን በዘመናዊነት እና በለውጥ ጎዳና እንድትሄድ ያደረጉ የመጀመሪያው ሰው ማን ናቸው? =Person,
   9. በምርኮ የተወሰደው የአጼ ቴዎድሮስ ልጅ ማን ነው? =Place,
   10. ህልም አለኝ የሚል ታሪካዊ ንግግር ያደረጉ አሜሪካዊ ማን ነበሩ? =Person,
   11. ማራቶንን ለመጀመሪያ ጊዜ ያሸነፉ ጥቁር አትሌት ማን ይባላል? =Person,
   12. ሽንብራ ኩሬ ላይ የተካሄደው ጦርነት በማንና በማን መካከል ነበር? =Person,
   13. የአይ ቢ ኤም ኩባንያ መስራች ማን ነው? =Person,
   14. ኢትዮጵያን ከ1889 እስከ 1913 ያስተዳደሩ ንጉስ ማን ይባላሉ? =Person,
   15. የሀረሪ ክልል ርዕሰ መስተዳድር ማን ይባላሉ? =Person,
   16. ከአድማስ ባሻገር ልቦለድ መጽሀፍ ደራሲ ማን ናቸው? =Person,
   17. የመጀመሪያውን ተንቀሳቃሽ ምስል ያሳየ ማን ይባላል? =Person,
   18. የጠቅላይ ሚኒስትር መለስ የመጀመሪያ ስም ማን ይባላል? =Person,
   19. የመጀመሪያው የዛጉዌ ስርወ መንግስት ንጉስ ማን ነበር? =Person,
   20. ሀረር የተመሰረተችው በማን ነው? =Person,
   21. ለመጀመሪያ ጊዜ አንታርክቲካን ያቋረጡ ሰዎች ማንና ማን ይባላሉ? =Person,
   22. እውነተኛው የመድሃኒት አባት በመባል የሚታወቀው ማን ይባላል? =Person,
   23. የዲላ ዩኒቨርሲቲ ፕሬዝዳንት ማን ይባላሉ? =Person,
   24. የደብረ ብርሃን ስላሴ ቤተክርስቲያን የተሰራው በማን ዘመነ መንግሰት ነው? =Person,
   25. በ1986 የኢትዮጵያ ፕሬዝዳንት የነበረ ማን ነው? =Person,
   26. በአሜሪካ የተገደሉት የቀድሞ የኢራቅ ፕሬዝዳንት ማን ነበሩ? =Person,
   27. ኢትዮጵያን በዘመናዊነት እና በለውጥ ጎዳና እንድትሄድ ያደረጉ የመጀመሪያው ሰው ማን ናቸው? =Person,
   28. በአለም ትልቁ አጥቢ እንስሳ ማን ይባላል? =Person,
   29. በደርግ ስርአት የነበሩ የመጨረሻው የኢትዮጵያ ጠቅላይ ሚኒስትር ማን ይባላሉ? =Person,
   30. የሀረር ቢራ ስራ አስኪያጅ ማን ይባላሉ? =Person
}
```

Time Question Types

```
{
  1. አጼ ዳዊት የነገሱት ከመቼ እስከ መቼ ነበር? ? =Time,
  2. ምኒሊክ 2ኛ የተወለዱት መቼ ነበር? =Time,
  3. በኢትዮጵያ የመጀመሪያው የስልክ አገልግሎት መቼ ተጀመረ ? =Time,
```

4. ደቡብ ሱዳን ነጻነቷን ያገኘችው መቼ ነው? =Time,
5. የማይጨው ጦርነት የተካሄደው መቼ ነበር? =Time,
6. አዲሱ የአፍሪካ ህብረት ዋና መስሪያ ቤት ህንጻ መቼ ተመረቀ? =Time,
7. የዓለም ዋንጫ መቼ ተጀመረ? =Time,
8. የአፍሪካ ህብረት መቼ ተመሰረተ? =Time,
9. ከአዲስ አበባ ጂቡቲ የተዘረጋው የባቡር መስመር ስራ የጀመረው መቼ ነበር? =Time,
10. ልእልት ዲያና የሞቱት መቼ ነበር? =Time,
11. ደቡብ ሱዳን ከሱዳን የተገነጠለችው መቼ ነው? =Time,
12. ፀሀይ ምድርን እንደምትዞር የታወቀው መቼ ነበር? =Time,
13. አክሊሉ ለማ መቼ ተወለዱ? =Time,
14. የለንደን ማራቶን መቼ ተካሄደ? =Time,
15. ኮለኔል መንግስቱ ሀይለ ማርያም መቼ ኮበለሉ? =Time,
16. የመቅደላ ጦርነት መቼ ተካሄደ? =Time,
17. የምድር ባቡር ድርጅት መቼ ተቋቋመ? =Time,
18. በኢትዮጵያ የህግ ሳምንት ከመቼ እስከ መቼ ይከበራል? =Time,
19. የኢትዮጵያ ብሄራዊ ባንክ መቼ ተቋቋመ? =Time,
20. የሰሜን ጦር ቃልኪዳን ድርጅት ወይም ኔቶ መቼ ተቋቋመ? =Time,
21. በኢትዮጵያ የመደራጀት መብት በህግ እውቅና የተሰጠው መቼ ነው? =Time,
22. የኢትዮጵያ ቴሌቪዥን ስርጭት የተጀመረው መቼ ነበር? =Time,
23. የኢትዮጵያ ኡጋዴንና የሶማሊያ ጦርነት የተካሄደው መቼ ነበር? =Time,
24. የሙስሊም መፍትሄ አፈላላጊ ኮሚቴዎች የፍርድ ዉሳኔ ለመቼ ተቀጠረ? =Time,
25. የአማራ ልማት ማህበር መቼ ተቋቋመ? =Time,
26. እጂግ የተከበሩ የአለም ሎሬት ሜትር አርቲስት አፈወርቅ ተክሌ መቼ ሞቱ? =Time,
27. ቫስኮ ደጋማ ወደ ህንድ የሄደው መቼ ነበር? =Time,
28. ባራክ አባማ ፕሬዝዳንት የሆኑት መቼ ነው? =Time,
29. 50ኛው አመት የአፍሪካ ህብረት የመሪዎች ጉባኤ ከመቼ እስከ መቼ ይካሄዳል? =Number,
30. የኢትዮጵያ ንግድ ባንክ መቼ ተመሰረተ? =Time
}

Place Question Types

{
1. አህመድ ግራኝ የየት ሀገር ገዥ ነበር? =Place,
2. ተሸከሞት ወደ ባህር በሚገባው ውሃ መጠን ቀዳሚው ወንዝ ማን ይባላል? =Place,
3. በ1998 የመሬት መንቀጥቀጥ የነበረው የት ሀገር ነበር? =Place,
4. በምስራቅ አፍሪካ ረጂሙ ወንዝ ማን ነው? =Place,
5. የኤቨረስት ተራራ የት ይገኛል? =Place,
6. በአለም ረጂሙ ወንዝ የት ይገኛል? =Place,
7. በአለም በቸኮሌት አምራችነት የምትታወቀው አገር ማን ነች? =Place,
8. በአፍሪካ በቆዳ ስፋት ትልቋ ሀገር ማን ነች? =Place,
9. የደቡብ ብሄራዊ ክልላዊ መንግስት መቀመጫ ከተማ ማን ይባላል? =Place,
10. ደብረ ዳሞ የት ይገኛል? =Place,
11. የአዋሽ ወንዝ መነሻ የት ነው? =Place,
12. የሬጌ ሙዚቃ መገኛ ሀገር ማን ነች? =Place,
13. በሜድትራንያን ባህር ከሚገኙት ደሴቶች ውስጥ ትልቁ ደሴት ማን ይባላል? =Place,
14. ኖቬምበር 14 2012 የከተማ ባቡር አገልግሎት መስጠት የጀመረች አፍሪካዊት ሀገር ማን ነች? =Place,
15. የሶማሌ ብሄራዊ ክልላዊ መንግስት መቀመጫ ከተማ ማን ይባላል? =Place,
16. የላሊበላ ውቅር አብያተ ክርስቲያናት የት ይገኛሉ? =Number,
17. የአልጀሪያ ዋና ከተማ ማን ይባላል? =Place,
18. የ2012 የአለም አትሌቲክስ ውድድር የተካሄደው የት ነበር? =Place,
19. ኢትዮጵያ የቡና ምርቷን የምትልከው የት የት ሀገራት ነው? =Place,
20. የጢያ ትክል ድንጋይ የት ይገኛል? =Place,

21. የካልሃሪ ደሴት በየት አህጉር ይገኛል? =Place,
22. የኒውዮርክ ትክክለኛ ስሟ ማን ይባላል? =null,
23. በኢትዮጵያ የመጀመሪያው ትምህርት ቤት የት ተገነባ? =Place,
24. በአለም ከፍተኛ ፏፏቴ የት ይገኛል? =Place,
25. የግብጽ ዋና ከተማ ማን ይባላል? =Place,
26. ከስምጥ ሸለቆ ሀይቆች ሁሉ ጥልቁ ማን ነው? =Place,
27. በአለም ትልቁ ቲያትር ቤት የት ይገኛል? =Place,
28. በአለም ብቸኛዋ ብዙ የኢንጂነሪንግ ኮሌጅ ያለባት ከተማ ማን ትባላለች? =Place,
29. የሉሲ ቅሬተ አካል የተገኘው በየትኛው ክልል ነው? =Place,
30. ጠቅላይ ሚኒስትር ሀይለ ማርያም ደሳለኝ የተወለዱት የት ነው? =Place
}

Quantity Question Types

{
1. የጤነኛ ሰው የልብ ምት በአማካይ በደቂቃ ምን ያህል ነው? =Number,
2. አንድ ጋሎን ስንት ሊትር ነው? =Number,
3. ኢትዮጵያ በትራፊክ አደጋ ብዛት ከአለም ስንተኛ ደረጃ ላይ ትገኛለች? =Number,
4. ከአዲስ አበባ ጂንካ ስንት ኪሎ ሜትር ነው? =Number,
5. እንግሊዝ ያዘጋጀችው ስንተኛውን ኦሎምፒክ ነው? =Number,
6. በ2012 የአፍሪካ ዋንጫ በአጠቃላይ ስንት ጎሎች ተቆጠሩ? =Number,
7. የራስ ዳሽን ተራራ ርዝመቱ ምን ያህል ነው? =Number,
8. የአዞ እንቁላል ለመፈልፈል ምን ያህል ጊዜ ይፈጅበታል? =Number,
9. ታላቁ የህዳሴ ግድብ እስከ ስንት አመት ይፈጃል? =Number,
10. በአዲስ አበባ ከተማ ስንት ሙዚየሞች ይገኛሉ? =Number,
11. ዛምቢያን ስንት ሀገሮች ያዋስኗታል? =Number,
12. ሰሞኑን በአዲስ አበባ የተካሄደው የሳይንስ የሂሳብና የምርምር ሽልማት የተካሄደው ለስንተኛ ጊዜ ነው? =null,
13. አፄ ዮሀንስ በስንት አመታቸው ነገሱ? =Number,
14. ስንተኛ ምን ያህል ጊዜ እንገላበጣለን በአማካይ? =Number,
15. በማርስ ከባቢ አየር ውስጥ የካርቦንዳይ አክሳይድ መጠን ምን ያህሉን ይሸፍናል? =Number,
16. በአለም ውስጥ ስንት ቋንቋዎች ይነገራሉ? =Number,
17. በፓርላማ ውስጥ ስንት የተቃዋሚ ፓርቲ አባላት አሉ? =Number,
18. የቼዝ መጫወቻ ቦርድ ምን ያህል ካሬዎች አሉት? =Number,
19. በአፍሪካ ዋንጫ ስንት አገራት ይሳተፋሉ? =Number,
20. ኮንሶ ከአዲስ አበባ ስንት ኪ.ሜ. ርቀት ላይ ትገኛለች? =Number,
21. ኤርትራ ከስንት ሀገሮች ጋር ትዋሰናለች? =Number,
22. ጨረቃ ከምድር በምን ያህል ርቀት ላይ ትገኛለች? =Number,
23. የኢትዮጵያ ሰራዊት በሶማሊያ ለምን ያህል ጊዜ ቆየ? =Number,
24. በዘዋይ ሀይቅ ላይ ስንት ደሴቶች ይገኛሉ? =Number,
25. የጢስ አባይ ፏፏቴ ርዝመቱ ምን ያህል ነው? =Number,
26. የኢትዮጵያ ጠ/ሚንስቴር በስንት አመታቸው አረፉ? =Number,
27. ቀዳማዊ ሀይለ ስላሴ ኢትዮጵያን ለስንት ዓመት አስተዳደሩ? =null,
28. የአንድ ሊትር ዘይት መሸጫ ዋጋ ስንት ብር ነው? =Number,
29. ከአዲስ አበባ አዋሳ ስንት ኪሎ ሜትር ነው? =Number,
30. የሱዳን ህዝብ ብዛት ስንት ነው? =Number
}

Appendix B

This appendix shows factoid questions from the four question types, the expected answers for these questions, the list of answers returned by our system, and the rank of the expected answers in the returned answers list.

No.	Questions	Expected Answer	Returned Answer	Rank
Person Question Types				
1	ድሬቲዮብ የተባለውን የቪዲዮ ድረገጽ የጀመረው ማን ነው?	ቢኒያም	ቢኒያም አንዱ መስከረም	1
2	የአንድነት ለዲሞክራሲና ለፍትህ ፓርቲ ም/ሊቀመንበር ማን ናቸው?	ግርማ ሰይፉ	ዘነበ ዘለቀ ግርማ ሰይፉ ሰላም	3
3	በኢትዮጵያ የግብፅ አምባሳደር ማን ይባላሉ?	መሀመድ ኢድሪስ	መሀመድ ኢድሪስ ብርሀነ ገንዘብ ሀይለማርያም ደሳለኝ አዲስ አበባ	1
4	በኢትዮጵያ ኮሪያ ዘማቾች ማህበር ፕሬዝደንት ማን ይባላሉ?	ኮሎኔል መለሰ ተሰማ	አዲስ አድማስ ጤና ከተማ አዲስ	-
5	ምስጢር የተሰኘው መፅሐፍ ደራሲ ማን ነው?	መሀመድ ሰልማን	መሀመድ ሰልማን አለማየሁ ደስታ ዋጋው	1
6	የአፍሪካ አንድነት ድርጅት ሲመሰረት የመጀመሪያው ዋና ጸሐፊ የሆኑት ማን ናቸው?	አቶ ክፍሌ ወዳጆ	መሰረት ነፃነት ጉባኤ በነበሩ	-
7	የኢትዮጵያ ፉትቦል ፌዴሬሽንን የመሠረቱት ማን ናቸው?	ይድነቃቸው ተሰማ	ይድነቃቸው አንዱ አዲስ አበባ ሀኪም	1
8	በጋና የኢትዮጵያ አምባሳደር ማን ይባላሉ?	ጊፍቲ አባሲያ	መቅደስ ብርሀነ ገንዘብ ሀይለማርያም ደሳለኝ	-
9	የሚሽን ፎር ኮሚዩኒቲ ዴቨሎፕመንት ፕሮግራም መሥራችና ኤክስኪዮቲቭ ዳይሬክተር ማን ናቸው?	ወይዘሮ ሙሉ ሀይለ	ሙሉ ሀይለ ህይወት ዳንኤል አምባቸው	1
10	የኢትዮጵያ እግር ኳስ ፌደሬሽን ተቀዳሚ ምክትል ፕሬዝዳንት ማን ናቸው?	ተካ አስፋው	በላይ ሚሊዮን ዮናስ ተሾመ መታሰቢያ	-
11	የ ሸገር ሬዲዮ መስራች ባለቤትና ሥራ አስኪያጅ ማን ይባላሉ?	መአዛ ብሩ	ዳዊት መላኩ መአዛ ብሩ መልካ ድረስ	2
12	የግርማዊ ቀዳማዊ አፄ ኃይለሥላሴ መታሰቢያ ማህበር የቦርድ ሊቀመንበር ማን ይባላሉ ?	ናሁሰናይ አርአያ	ናሁሰናይ አርአያ አንድነት አለም ልደት ሀይለ ስላሴ	1

13	የትምህርት ሚኒስቴር ሚኒስትር ዴኤታ ማን ይባላሉ?	አቶ ፉአድ ኢብራሂም	ፉአድ ኢብራሂም ብርሀነ መኮንን ሽመልስ ከማል	1
14	የመጀመርያውን ታላቁ ሩጫ በኢትዮጵያ ያሸነፈው ማን ነበር?	ሀይሌ ገብረስላሴ	ታሪኳ አለም አከሊል ልዩ መሀመድ አማን	-
15	የእንጀራ ማምረቻ ማሽን ዲዛይን ያደረገው ማን ይባላል?	ሙሉጌታ ቢጋሻው	ሙሉጌታ ቢጋሻው	1
16	በለንደኑ አሊምፒያድ ኢትዮጵያን በ800 ሜትር ሩጫ ወክሎ የተወዳደረዉ ማን ነው?	መሀመድ አማን	መሀመድ አማን ታሪኳ ፋንታየ ሲራክ አምሳለ ወልደ ገብርኤል ፋንቱ በላይ	1
17	የኢትዮጵያ ሴቶች በጎ አድራጉት ማኅበራት ጎብረት ሥራ አስኪያጅ ማን ናቸው?	አዜብ ቀለመወርቅ	አዜብ ቀለመወርቅ ንብረት በላይ ደረጀ	1
18	የግብፅ ፕሬዚዳንት ማን ይባላሉ?	መሀመድ ሞርሲ	መሀመድ ሞርሲ አለባቸው	1
19	ቀዳሚው የኢትዮጵያ እግር ኳስ ክለብ ማን ይባላል?	ቅዱስ ጊዮርጊስ	ቅዱስ ጊዮርጊስ ገበያው አሙቱ በላይ	1
20	የኢትዮጵያ ምርት ገበያ መስራችና ዋና ስራ አስፈፃሚ ማን ይባላሉ?	እሌኒ ገ/መድህን	እሌኒ ገ/መድህን መስፍን አቢ ዳዊት መድህኒት	1
21	የኢትዮጵያ ብሔራዊ ቡድን አሰልጣኝ ማን ይባላሉ?	ሰውነት ቢሻው	አዲስ አበባ ሰውነት ቢሻው ሙሉ እምነት ምትክ	2
22	የመጀመሪያዋ የሴት ባለቅኔ ማን ይባላሉ?	እማሆይ ገላነሽ	እማሆይ ገላነሽ ውበት	1
23	የኢትዮጵያ ሴት ደራስያን ማህበር ፕሬዚዳንትነት ማን ትባላለች?	የምወድሽ በቀለ	የምወድሽ በቀለ እንቁጣጣሽ ከተማ አዲስ አንጋፋ ዳግማዊ	1
24	የመጀመሪያዋ ሴት የፊልም ደራሲ አዘጋጅና ፕሮዲዩሰር ማን ትባላለች?	ቅድስት ባየልኝ	ቅድስት አላማ ሰይፉ ማሞ ሜሮን ጌትነት ማሞ ውድነህ	1
25	የመንግሥት ኮሙዩኒኬሽን ጉዳዮች ሚኒስትር ማን ናቸው?	በረከት ስምኦን	በረከት ስምኦን ረገድ መለስ ጠና ጉዳይ መንግስቱ	1
Time Question Types				
26	ጉማ የፊልም ሽልማት መቼ ይቀርባል?	በታህሳስ ወር	ታህሳስ በ64 ሰኞ 3 ሰአት	1
27	የሞጆ ደረቅ ወደብ መቼ ተመሠረተ?	በ2001	በ2001	1

28	ልጅ እንዳልካቸው መኮንን መቼ ተወለዱ?	ጳጉሜ 3 ቀን 1920	በቀን ጳጉሜ 3 ቀን 1920 ወር ወራት ጥር	2
29	ዲናሞ የተሰራዉ መቼ ነው?	በ1831	-	–
30	ኔልሰን ማንዴላ ታላቁን ዓለም አቀፍ የሠላም ሽልማት የተቀበሉት መቼ ነው?	በ1985	በ1985 ዛሬ ለአመታት ጥር	1
31	ኢኳቶሪያል ጊኒ ከስፔን ቅኝ አገዛዝ ነፃነቷን የተቀዳጀችው መቼ ነበር ?	በ1968	በ1968 በ2012 ዛሬ	1
32	የኢትዮጵያ ሴቶች በጎ አድራጐት ማኅበራት ጎብረት መቼ ተመሰረተ?	በ2010	በ2010 ትናንት ጥር ለ20 አመታት	1
33	እስራኤል ከግብጽ ጋር በካምፕ ዴቪድ የሰላም ስምምነት ያደረገችው መቼ ነው?	በ1979	በ1979 ከ1991 ሳምንት	1
34	ፀጋዬ ገብረ መድህን መቼ ተወለደ?	ነሀሴ 1928	ነሀሴ 1928 ዛሬ አስር አመት	1
35	የኒውስዊክ መጽሔት የመጀመሪያ እትም መቼ ገበያ ላይ ዋለ?	የካቲት 17 ቀን 1933	ከአንድ ሳምንት በኋላ ከ8 አመት በፊት ነሀሴ 2002	–
36	ዮፍታሄ ንጉሴ መቼ ተወለዱ?	በ1885	ትናንት ከሰአት በኋላ በ1885	2
37	አቡነ ጴጥሮስ መቼ ተወለዱ?	በ1885	በ1885 አመታት በኋላ	1
38	የቀዳማዊ ኃይለሥላሴ መታሰቢያ ማህበር መቼ ተቋቋመ?	ሀምሌ 16 ቀን 1987	ሀምሌ 16 ቀን 1987 ባለፈው ሳምንት 22 ቀን ከ1995 አመት	1
39	የጽሕፈት ማተሚያ ማሽን ወደ ኢትዮጵያ የገባው መቼ ነው?	በ1889	በ1889 አመት በ1843 ትናንት	1
40	በዓለማችን የተንቀሳቃሽ ስልክ ስርጭት መቶ በመቶ የሚሆነው መቼ ነው?	በ2016	በ2016 በቀን ቀን በ2003	1
41	ለአፍሪካ ህብረት ሊቀመንበርነት ያሸነፉት ዶ/ር ዙማ መቼ ቃለ መሐላ ፈጸሙ?	ሀምሌ 10 ቀን 2004	እሁድ	–
42	ዘመናዊ ኦሊምፒክ መቼ ተጀመረ?	1896	ከ1896 በ10	1
43	ኢትዮጵያ ለመጀመርያ ጊዜ በኦሊምፒክ መሳተፍ የጀመረችው መቼ ነው?	1956	ከ800 ቀን ዛሬ ከ1896	–
44	የኢትዮጵያ አትሌቲክስ ፌዴሬሽን ዓመታዊ ጠቅላላ ጉባኤዉን ያካሄደው መቼ ነው?	ህዳር 22 እና ህዳር 23	ህዳር 22 23 ቀን ረቡእ ጥር	1

45	የንጉሱ ጊዜ የኢትዮጵያ ሕዝብ መዝሙር መቼ ተጻፈ?	በ1919	በ1919 ረቡእ ቀን ጥር	1
46	በዳግማዊ ምኒልክ ሆስፒታል ቅጥር ውስጥ የሚገኘው ጤና ሳይንስ ኮሌጅ መቼ ተቋቋመ?	በ1941	በ1941	1
47	ኢህአዴግ ስልጣን የያዘው መቼ ነው?	ግንቦት 20	ዘንድሮ ግንቦት 20 አመታት ከሁለት አመት	2
48	የቀድሞዋ ሶቪየት ኅብረት አፍጋኒስታንን የወረረችዉ መቼ ነበር?	1979	ከ1979 በ1938	1
49	መሐመድ አማን መቼ ተወለደ?	ጥር 1 ቀን 1986	ዛሬ ሁለት አመት ወር በ2009	–
50	ሱዳን በአፍሪካ ዋንጫ ሻምፒዮን የሆነችው መቼ ነው?	በ1970	ዛሬ ለ28 ከሳምንት በፊት ሰኞ	
Place Question Types				
51	ፍራንሷ ኦላንድ የየት አገር ፕሬዚዳንት ናቸው?	ፈረንሳይ	ፈረንሳይ ገባ አምቦ	1
52	ፀጋዬ ገብረ መድህን የት ተወለደ?	አምቦ ከተማ	አምቦ ከተማ ኢትዮጵያ ልዩ ልዩ	1
53	ሳልቫ ኪር የየት አገር ፕሬዚዳንት ናቸው?	ደቡብ ሱዳን	ደቡብ ሱዳን ገባ	1
54	ዘመናዊ ኦሊምፒክ የት ተጀመረ?	በግሪክ አቴንስ	አቴንስ ልዩ	1
55	20ኛው የዓለም ዋንጫ የት ይካሄዳል?	ብራዚል	ከተማ ብራዚል አውሮፓ ደቡብ	2
56	ኤልክላሲኮ የየት አገር ክለቦች የደርቢ ፍልሚያ ነው?	ስፔን	ስፔን አዲስ ማድሪድ ኢትዮጵያ	1
57	የሞዛምቢክ ዋና ከተማ ማን ይባላል?	ማፑቶ	ማፑቶ ዳካ ለንደን	1
58	የሶሪያ ዋና ከተማ ማን ይባላል?	ደማስቆ	ደማስቆ ዳካ ፈረንሳይ	1
59	የኢትዮጵያ አትሌቲክስ ፌዴሬሽን ዓመታዊ ጠቅላላ ጉባኤ የት ተካሄደ?	ባህር ዳር	ባህር ዳር ከተማ መቀሌ ከተማ አዲስ አበባ መሀመድ አማን	1
60	6ኛው ቢግ ብራዘርስ አፍሪካ–አብሮ የመኖር ውድድር የት ተካሄደ?	ደቡብ አፍሪካ	አፍሪካ ኢትዮጵያ ለንደን	–
61	የፖላንድ ዋና ከተማ ማን ይባላል?	ዋርሶ	አዲስ ዳካ	–
62	ሦላሳ አንደኛው ኦሊምፒክ የት አገር ይካሄዳል?	ሪዮ ዲ ጃኔሮ ከተማ	ሪዮ ዲ ጃኔሮ ከተማ ጃፓን ሩዋንዳ ኪጋሊ	1

63	ዮፍታሄ ንጉሴ የት ተወለዱ?	ጎጃም	ስላሴ ጎጃም	2
64	አቡነ ጴጥሮስ የት ተወለዱ?	ፍቾ	ኢትዮጵያ ቴዎድሮስ	–
65	የኡጋንዳ ዋና ከተማ ማን ይባላል?	ካምፓላ	ካትማንዱ ካምፓላ ኢትዮጵያ	2
66	አንደኛው የምሥራቅ አፍሪካ የወጣቶች የሙያ ሻምፒዮን የት ይካሄዳል?	አዲስ አበባ ከተማ	አዲስ አበባ ከተማ ጊዮርጊስ	1
67	የአርቲስት አስናቀች ወርቁ ስርአተ ቀብር የት ተፈጸመ?	መንበረ ፀባኦት ቅድስት ስላሴ ደብር	መንበረ ፀባኦት ቅድስት ስላሴ ደብር አሜሪካ	1
68	የእሥራኤል የዓይን ሐኪሞች ነፃ ሕክምና የሰጡት የት ነው?	አዲስ አበባ	አዲስ አበባ ልዩ ልዩ ቻይና	1
69	ሱባ ደን የት ይገኛል?	ሆለታ	ሆለታ	1
70	የምሥራቅና መካከለኛው አፍሪካ ዋንጫ የት ተካሄደ?	ኡጋንዳ	ኳታር ኤርትራ አዲስ ካርቱም ሞሮኮ	–
71	የአማራ ብሔራዊ ክልላዊ መንግሥት ርዕሰ መዲና ማን ናት?	ባህር ዳር	አዲስ አበባ አንዱ	–
72	የቦትስዋና ዋና ከተማ ማን ይባላል?	ጋብሮኒ	አዲስ አበባ	–
73	ቤትሆቨን የት ተወለደ?	ጀርመን	ጀርመን	1
74	የመንግስታቱ ድርጅት አብዛኛውን ወጪ የምትሸፍነው አገር ማን ናት?	አሜሪካ	ሺታ አሜሪካ ጋራ አፍሪካ	2
75	የአቡነ ጳውሎስ የቀብር ስነስርዓት የት ተፈጸመ?	በመንበረ ፀባኦት ቅድስት ስላሴ ካቴድራል	ኢትዮጵያ አውራ ደበላ	–
Quantity Question Types				
76	ድሬቲዮብ የቪዲዮ ድረገጽ ምን ያህል ብር ወጥቶበታል?	ሁለት ሚሊዮን ብር	በሁለት ሚሊዮን ብር	1
77	የአንስቴቲስቶች ማህበር ስንት አባላት አሉት?	ከ300 በላይ	ከ300 በላይ ለሁለተኛ .. 20012 ሰባት	1
78	በአዲስ አበባ የሲጋራ ልምድ በምን ያህሉ ወንዶች ላይ ይታያል?	በስምንት በመቶ	መቶ በአንድ ሶስት 1	–
79	አድዋ ከተማ ከአዲስ አበባ በስንት ኪሎ ሜትር ርቀት ላይ ይገኛል?	1066 ኪሎ ሜትር	1066 ኪሎ ሜትር በ17 በአራት ኪሎ በ1500 ሜትር	1
80	ጠ/ሚ መለስ በረሃ የገቡት በስንት አመታቸው ነው?	በሀያ	በሀያ ከ7 በመቶ በ1989	1
81	እጅግ የተከበሩ የዓለም ሎሬት ሜትር አርቲስት አፈወርቅ ተክሌ በስንት አመታቸው ከዚህ አለም በሞት ተለዩ?	በ80	በ80	1

82	የተባበሩት መንግሥታት አባል አገራት ብዛት ስንት ነው?	192	በአንድ በ1979 በቢሊዮን 000 ሜትር በሁለተኛው	-
83	በመዲናችን ያሉት የኤፍ ኤም ሬዲዮ ጣቢያዎች ብዛት ስንት ነው?	ስድስት	ስድስት 98.1 በአንድ በ2002	1
84	የኢንተርፖል አባል አገሮች ብዛት ስንት ነው?	188	ከ188 አንድ በ2012	1
85	በደሌ ቢራና ሄኒከን የኢትዮጵያ ብሔራዊ ቡድንን በስንት ብር ስፖንሰር አደረጉ?	ከ24 ሚሊዮን ብር በላይ	አንድ 80 ሚሊዮን ብር 9	-
86	በሎካርቢ አውሮፕላን ፍንዳታ ምን ያህል ሰዎች አለቁ?	270	270 አንድ	1
87	በ 30ኛው የለንደን አለምምፒያድ የአፍሪካን አህጉር በመወከል ስንት አገራት ተሳተፉ?	53	ለ30ኛው በ30ኛው ከአራት አመት	
88	በአመት ምን ያህል በገጠር የሚኖሩ ኢትዮጵያውያን ሴቶች ለቤት ሰራተኝነት ወደ መሃል አገር ያቀናሉ?	10,000	በአንድ	
89	የሐረር ከተማ ከአዲስ አበባ በምን ያህል ኪሎ ሜትር ርቀት ላይ ትገኛለች?	በ526 ኪሎ ሜትር	በ526 ኪሎ ሜትር የ መኖ በአራት ኪሎ	1
90	በአለማችን ምን ያህል ሰዎች በየዓመቱ በስትሮክ ሳቢያ ለሕልፈተ ህይወት ይዳረጋሉ?	ስድስት ሚሊዮን	ስድስት ሚሊዮን የ 24 አንድ	1
91	የቅዱስ ያሬድ የዜማ ምልክቶች ብዛት ምን ያህል ነው?	10	10 ..... በሀለት	1
92	ምን ያህል የኤርትራ ተጫዋቾች ኮበለሉ?	17	17 አራት	1
93	የአሰላ ብቅል ፋብሪካ ምን ያህል ኩንታል የቢራ ገብስ ከውጭ ገዛ?	ከ132 ሽህ ኩንታል በላይ	ከ132 ሽህ ኩንታል ለ30 አንድ 118 ብር ለሀለተኛ	1
94	ከ ሁምቦ አርባ ምንጭ ያለው መንገድ ስንት ኪሎ ሜትር ርዝመት አለው?	106 ኪሎ ሜትር	106 ኪሎ ሜትር 6.4 ኪሎ ሜትር ሀለት	1
95	አዲሱ የአፍሪካ ህብረት ህንፃ ምን ያህል ዶላር ወጣበት?	200 ሚ	200 30 አንድ በአራተኛው	1
96	ሐበሻ ሲሚንቶ ለደቡብ አፍሪካ ኩባንያዎች የምን ያህል ዶላር አክሲዮኖች ሸጠ?	የ21 ሚሊዮን ዶላር	የ21 ሚሊዮን ዶላር ሀለት 52	1
97	ሊፋን ሞተርስ ምን ያህል አገር በቀል ግብረሰናይ ድርጅቶችን ለመርዳት ወሰነ?	ሀለት	ሀለት 60″	1
98	በጎንደር ከተማ አዲስ የተከፈተው ደስታ ሲኒማ ቤት ስንት መቀመጫዎች አሉት?	ሀለት መቶ ሀምሳ	ሀለት መቶ ሀምሳ 1890 አንድ በ3 ሰአት	1

99	ደራሲ ስብሃት ገ/እግዚአብሄር ከዚህ ዓለም በሞት የተለየው በስንት አመቱ ነው?	በ76	በ76 በአንድ 2004 አራተኛ	1
100	አርቲስት አስናቀች ወርቁ በስንት አመቷ ከዚህ አለም በሞት ተለየች ?	በ78	በ78 2004 የ አምስት አመት	1

Declaration

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

Declared by:

Name: _____

Signature: _____

Date: _____

Confirmed by advisor:

Name: _____

Signature: _____

Date: _____