**HiLCoE**School of Computer Science and Technology  
**Book review Search Engine**

**DRB1902-B**

Natnael Mebratu, Nardos Dagnachew, Nazrawit Berhanu, Romhay Yemane, Saron Bisrat, and Solyana Moges

# 1. Introduction

Welcome to our information retrieval project, where our objective is to retrieve only relevant information that our user needs. The aim of this project is to develop an Amharic search engine that is capable of accurately retrieving relevant information from a large corpus of Amharic texts. To achieve this goal, several processes were implemented, including stemming, stop word removal, corpus collection, calculation of tf-idf, term weighting, and ranking by cosine similarity.

## 2. Problem Statement

## How to retrieve relevant data easily?

## 3. Objective

Given a corpus collection with different attributes, the six of us decided to answer the above question using information retrieval techniques we learned in our information retrieval course.

## 4. Methodology

Firstly, to develop a search engine, a **corpus** of Amharic texts is required. The corpus used in this project was collected using an automatic scraper for authors and book reviews. The scraper was designed to crawl websites and extract text data, which was then stored in a database for further processing.

Then, **Stemming** applied which is the process of reducing a word to its base or root form. In Amharic, this involves removing prefixes and suffixes to obtain the core meaning of the word. The Amharic stemmer used in this project is based on a rule-based approach, which involves applying a set of linguistic rules to the input word to obtain its stem.

After stemming, a **dictionary** was used to check if the stemmed words are not over stemmed or under stemmed. Over stemming occurs when multiple words are reduced to the same stem, while under stemming occurs when a stem is not reduced enough, resulting in multiple stems for a single word. The dictionary used in this project was developed based on expert knowledge of the Amharic language.

We used a **set of written stop words** in this project to remove the commonly occurring stop words in Amharic. Such as: "ስለሚሆን", "አና", "ስለዚህ", "በመሆኑም", "አንዳች", "አንድ", "እንደ", "እንጂ", "ያህል" etc. Stop words are words that are commonly used in a language but do not carry much meaning, such as "the," "a," "an," and "in." in English version. These words are removed from the text before analysis to reduce noise and improve the accuracy of the results. The function of removing stop word takes a text (corpus) as input and processes it to remove the stop words. Before removing the stop words from the input text, the sortedStopWords array sorts the stop words by length in descending order to avoid biased positives and negatives during the removal process. The function iterates through the sortedStopWords array and removes each stop word from the input text using a regular expression.

Calculation of **TF and IDF** (Term Frequency-Inverse Document Frequency ) was used to evaluate the importance of a term in a document. This task focus on processing text which can be used to search for documents that is similar to a given query. TF measures how frequently a term appears in a document, while IDF measures how important a term is in the entire corpus of documents. In this project, TF and IDF were calculated for each term in the corpus, but the values were not normalized. After tf-idf values are calculated, term weighting was done. **Term weighting** is the process of assigning weights to terms in a document based on their importance. The weights are used to rank the documents based on relevance to a search query.

In this project, we used **ranking by cosine similarity**. In the search method preprocesses the input query by performing lexical analysis( processing the input text and breaking it into tokens, such as words or phrases), stopword removal, and stemming. Then, it converts the query into a vector using the tf-idf weighting scheme. Finally, it calculates the cosine similarity between the query vector and each document vector or between two vectors in a high-dimensional space. It is used to rank the documents based on their similarity to the search query. The cosine similarity score was calculated by taking the dot product of the query vector and the document vector and dividing it by the product of their magnitudes.

At last, to make the project more attractive and user friendly, the front-end of the search engine was developed using React, a popular JavaScript library for building user interfaces. The front-end provides a simple and intuitive interface for users to enter their search queries and view the results.

# 5. Concluding remarks

The goal of the Amharic search engine developed in this project is to search and retrieve relevant information from a large corpus of Amharic texts. The processes implemented, including stemming, stop word removal, corpus collection, calculation of tf-idf, term weighting, and ranking by cosine similarity, ensure that the search engine provides accurate and relevant results to users. The front-end developed using React provides an easy-to-use interface for users to interact with the search engine.

# 6. System Description

# Recall = Retrieved / Total Relevant

# Precision = Relevant Retrieved / Total Retrieved

# Query1 – 100% Recall and 100% Precision

# Recall = Retrieved / Total Relevant

# Precision = Relevant Retrieved / Total Retrieved

# Query2 – 100% Recall and 25% Precision

# Recall = Retrieved / Total Relevant

# Precision = Relevant Retrieved / Total Retrieved

# Query3 – 50% Recall and 25% Precision

# Recall = Retrieved / Total Relevant

# Precision = Relevant Retrieved / Total Retrieved

# Query4 – 100% Recall and 100% Precision

# Recall = Retrieved / Total Relevant

# Precision = Relevant Retrieved / Total Retrieved

# Query5 – 33.3% Recall and 42% Precision

# Recall = Retrieved / Total Relevant

# Precision = Relevant Retrieved / Total Retrieved

# Query6 – 45.45% Recall and 53.3% Precision

# Recall = Retrieved / Total Relevant

# Precision = Relevant Retrieved / Total Retrieved

# Query7 – 23% Recall and 87% Precision

# Recall = Retrieved / Total Precision

# Precision = Relevant Retrieved / Total Retrieved

# Query8 – 36.7% Recall and 73.3% Relevant

# Recall = Retrieved / Total Relevant

# Precision = Relevant Retrieved / Total Retrieved

# Query9 – 89% Recall and 100% Relevant

# Query10 – 66.6% Recall and 100% Relevant

# 7. References

1. Genet Mezemir Fikremariam. ”Automatic Stemming for Amharic text: An experiment using successor variety approach”. AAU. Jan 2009.

2. https://github.com/liulalemx/felig-toolkit/tree