ADDIS ABABA UNIVERSITY

SCHOOL OF INFORMATION SCIENCE

INFORMATION STORAGE AND RETRIEVAL ASSIGNMENT II Section-2

Members of participant Name ID

- **Beyene Bekele…………… UGR/3182/15**
- **Enyew Yirga.......................UGR/4624/15**
- **Osama Ibrahim ……….... UGR/8819/15**
- **Samuel Mesfin …….....… UGR/5939/15**
- **Yitbarek Daniel……............UGR/2620/15**
- **Yonathan Tesfaye ……......UGR/2549/15**

INTRODUCTION

Text operations, also known as text processing, are techniques used to manipulate written or spoken language. In the field of Information Retrieval (IR), text operations are fundamental in the organization and retrieval of text-based information. Text operations can be categorized into three major types: syntactic, semantic, and statistical. Each type of text operation has its own distinct use in IR. Since more than 20 years ago, text processing in Amharic has frequently been done on computers. Nowadays, both government and non-governmental organizations, as well as individuals, process papers in Amharic using computers.

The main sources of information for researchers and the general public are documents. By offering irrelevant information, the growing availability of electronic documents causes a problem with information access effectiveness. We are drowning with much data at office, home either in printable or electronic form. Then finding the relevant information from this mass data is critical. At this end, information retrieval is a technology which creates the structured representation of unstructured texts by extracting relevant entities from them, thereby, making the data analysis realizable.

The number of Amharic documents available online and in other machine-readable formats keeps growing over time. This expansion makes it difficult to extract the relevant data for decision-making from the vast amounts of unstructured textual data that are already available. For years, a significant issue has been the lack of tools for extracting and utilizing useful information that are efficient enough to meet user needs. Semantic connection extraction is a difficult undertaking, particularly for languages like Amharic that have limited resources and complex linguistic structures.

The creation of significant features and the appropriate combination of these elements is also required by the complicated structure of languages. One of the poorly resourced and morphologically rich languages that faces the aforementioned difficulties is Amharic. This study was conducted on an Amharic corpus and it involves the fundamental steps of text processing: data cleaning, tokenization, normalization, and stopwords removal.Text preprocessing is a crucial step in natural language processing, as it helps to clean and prepare raw text data for further analysis. When it comes to Amharic, the process is no different. The first step in text preprocessing is data cleaning.

This involves removing any irrelevant or unnecessary data, such as HTML tags, punctuation marks, and numbers. The advantage of data cleaning is that it helps to reduce noise in the data and improve the accuracy of subsequent analyses. To develop Java script code for data cleaning, one can use regular expressions to identify and remove unwanted characters.

After cleaning the data, the next step is tokenization. Tokenization involves breaking down the text into individual words or tokens. This is important for subsequent analyses such as sentiment analysis and topic modeling. The advantage of tokenization is that it helps to simplify the text data and make it easier to analyze. This involves converting words to their base form or lemma, and reducing inflectional and derivational variations. The advantage of normalization is that it helps to reduce the size of the vocabulary and improve the accuracy of subsequent analyses. This involves removing common words that do not carry much meaning. The advantage of stopword removal is that it helps to reduce noise in the data and improve the accuracy of subsequent analyses. Objective of the study The objectives of this study is to experiment on text preprocessing and statistical analysis on texts are to extract meaningful insights and patterns from raw text data. Text preprocessing helps to clean and prepare the data for subsequent analyses, while statistical analysis helps to identify trends, relationships, and patterns in the data. Experimenting on tokenization, normalization, and stopword removal is important because it helps to determine the effects of these techniques on the accuracy and efficiency of subsequent analyses. For example, tokenization can affect the accuracy of sentiment analysis by changing the way words are counted and analyzed. Normalization can affect topic modeling by reducing the size of the vocabulary and improving the accuracy of topic assignments. Stopword removal can affect text classification by reducing noise in the data and improving the accuracy of predictions. By experimenting on these techniques, we can identify the most effective methods for specific use case and improve the accuracy and efficiency of their analyses. We believe can lead to better decisionmaking, improved customer experiences, and more efficient business processes. Because these operations are crucial in information retrieval. They enable search engines to understand which words are important and to retrieve documents that match the query.

Stemming involves reducing words to their root form, which improves recall by allowing the search engine to retrieve documents that contain variations of the query term. Stemming helps to improve the accuracy of the information retrieval system by reducing the number of variations of a word that need to be considered. This is particularly important in Amharic text, where there are many variations of words due to inflection.

Text operations and preprocessing are essential steps in the development of an information retrieval system for the Amharic text corpus we prepared. These steps help to improve the accuracy and efficiency of the system by reducing noise and irrelevant data, enabling the removal of stop words, stemming, and identifying named entities. By implementing these techniques, we can create more effective and efficient information retrieval systems for the document collection. Scope The focus of this study is the development of automatic algorithms to perform text operations on an Amharic corpora and apply statistical analysis as well as text processing on the text to extract meaningful insight on the raw text data. Methodology Collecting and preprocessing Amharic corpora is a crucial step in developing effective natural language processing systems. In this article, we will discuss the methodologies we followed to collect an Amharic corpus and apply text preprocessing and operations using Java script code.

Tokenization

Collecting Amharic corpora was conducted on the assignment I by manually collecting texts from various sources. Once the data has been collected, we cleaned it to remove any irrelevant or noisy information. This can be done by removing HTML tags, punctuation marks, and non-Amharic characters. Additionally, any duplicate or irrelevant texts should be removed from the corpus. Literature Review A literature review was conducted to identify existing research on Amharic text preprocessing and operations. This was useful in selecting appropriate techniques and tools for the task at hand. Tokenization Tokenization is the process of breaking text into smaller units such as words or phrases. This was challenging due to the complex morphology of the amharic language.

Tokenizing of a given text depends on the characteristics of language of the text which it is written. The Amharic language has its own punctuation marks which demarcate words in a stream of characters. The tokenization of text on this component uses Amharic punctuation marks and white spaces. Each token consists of a word I.e in this work,words are taken as tokens. The input text is tokenized with regard to the Amharic language characteristics. We developed a Java script code that tokenizes the corpus and analysed the results which are discussed under discussion session.

Normalization

Normalization involves converting text to a standard format that can be easily analyzed by computers. It is the process of transforming text into a single canonical form that it might not have had before. Normalizing text before storing or processing it allows for separation of concerns, since input is guaranteed to be consistent before operations are performed on it. Text normalization requires being aware of what type of text is to be normalized and how it is to be processed afterwards; there is no all-purpose normalization procedure. In Amharic writing system there are characters with the same pronunciation but different symbols which are called homophones. The letters such as አ, ኣ, ዐ and ዓ; ሠ and ሰ; ሀ, ኀ, ሃ, ኻ, ሓ, ኃ and ሐ, ጸ and ፀ are examples of characters with the same meaning and pronunciation but different symbol.

For example if we take the word ሁብታሙ it could have different forms like ሑብታሙ, ሃብታሙ, ሓብታሙ ኁብታሙ and ኃብታሙ. Therefore, all the above different forms must be normalized into ሁብታሙ by changing the first character of a word. Therefore, these characters should be normalized. We developed a Java script code and regex to normalise such things.

We also tried to normaize shortened words to non shortened words using a possible collection of dictionaries to map each shortened word. For example ዶ/ር to ዶክተር.

Stopword Removal

Stopwords are commonly used words in a language that don't carry significant meaning. Removing stopwords reduced the size of the corpus and improve the efficiency of the information retrieval system. we prepared a text file of amharic stopwords and removed them from the corpus by implementing a Java script code.

Stemming

Stemming involves reducing words to their root form. In Amharic, stemming is challenging due to the complex morphology of the language. We developed a Java script code that stems the corpus after completion of the above steps of text operations. We will discuss it in discussion part

Experiment Results and Discussion

Tokenize the text collected On this stage, we performed the tokenization of the Amharic corpus of 300 documents. We have done this automatically in a Javascript code. The code is attached in the zipped file and the output of the tokenization. Before the tokenization was performed the unprocessed text had total words of 150,493. After applying the Java script code on the corpus, it automatically cleaned the text by removing HTML tags, URLs, extra spaces and special characters from the text. Then the tokenization was performed on the cleaned text which produced tokens. The total number of words after the tokenization was reduced to 140,802.this shows that the corpus initially contained lots of noise which had to be removed before converting the text into tokens. Tokenization is an essential aspect of the text processing due to its ability to break down the text into smaller units that can be used to identify meaningful information and to perform further text operations.

Normalize the Text

The first reason why we need to normalize the Amharic text is to ensure consistency in spelling and grammar. Amharic has a complex writing system with over 200 characters, including consonants, vowels, and diacritics. This complexity makes it difficult to maintain consistency in spelling and grammar, which can affect the readability and understandability of the text. Normalization standardize the spelling and grammar of the corpus.

The second reason why we need to normalize the text is to facilitate information retrieval. Search engines and other information retrieval systems rely on normalized text to match search queries with relevant documents. Normalization can help convert Amharic text into a standard format that can be easily indexed and searched, improving the accuracy and relevance of search results. To do the normalization of the corpus we wrote a Java script code that uses regular expressions to substitute the shortened words with their respective correct form using mapping that was initially collected by us and process are attached in the zipped file. The normalization we have done automatically has 2 functions; normalizeChar normalizeAbbr That does Character Level Normalization such as "ጸጸጸ" and "ጸጸጸ" Labialized Character Normalzation such as "ጸጸጸጸጸ" to "ጸጸጸጸ" Short Form Expansion such as "ጸ.ጸ" to "ጸጸጸ ጸጸጸ" Punctuation Normalization such as :: to ጸ respectively. After the normalization we created a text file to store the normalized text and applied a Java script that we wrote to plot the rank vs frequency graph to facilitate statistical analysis on the text. After normalization on text file to Zipf's law and Luhn's idea, the frequency of occurrence of words in the text file is distributed according to Zipf's law. This means that the most frequent word occurs approximately twice as often as the second most frequent word, three times as often as the third most frequent word, and so on.

This distribution follows a power law, where the frequency of a word is inversely proportional to its rank. By normalizing the text file to these two ideas, we can create a more accurate representation of the text and identify important words. This can be useful in various applications such as information retrieval, text summarization, and natural language processing.

The output of the zipfian distribution for the normalized text is presented below:

The above Zipfian distribution graph represents the frequency of occurrence of words in the normalized text file. The x-axis represents the rank of the word in terms of frequency, and the y-axis represents the frequency of occurrence. The graph follows a power-law distribution, where a few words occur very frequently, and most words occur rarely.

Before tokenization and normalization, the text file contained punctuation marks, and other nonalphanumeric characters that can affect the frequency of occurrence of words. Therefore, the Zipfian distribution graph may not follow a perfect power-law distribution. After tokenization and normalization, the text file is divided into individual words. This process removes punctuation marks and other non-alphanumeric characters and standardizes the text. As a result, compared to the one we did for assignment I, the Zipfian distribution graph after normalization follows a more accurate power-law distribution, where the most frequent words are at the top, and the least frequent words are at the bottom. We decided the upper and lower cutoff points for the index based on the first and the third quartile for the rank distribution. The upper and lower cutoffs of frequency for the index are Q1: 9207.5 and Q3: 27620.5 Remove Stopwords In information retrieval systems, the analysis of word distribution in the corpus is important.

Accordingly the frequency distribution of words is considered. The usage of words in each sentence has to be analyzed. Sentences are formed by using a sequence of words or terms. The frequency distribution of these terms in the corpus has different values. In other words, there are high and low frequency terms. In information retrieval, content words are a more important factor in the performance of IR systems than function or stop words, which although occurring in high frequencies, have less importance. For this reason, stop words were removed automatically by the code we developed.

Stemming the text

The main logic of our stemmer is that,

- First we convert each word to their base form like ቤት to ብኤት
- Then we have collected possible suffixes and prefixes and stored them. They are also in there base form
- So, for ever word, we check if there's any suffix or prefix found, if so, we remove that prefix or suffix

In conclusion, our study delved into the intricate world of text processing and statistical analysis, focusing specifically on the complexities presented by the Amharic language. Through rigorous experimentation and implementation of various techniques, we aimed to enhance the efficiency and accuracy of information retrieval systems for Amharic corpora.

Our findings underscore the importance of text preprocessing steps such as tokenization, normalization, stopword removal, and stemming. These processes not only facilitate the organization and structuring of unstructured textual data but also play a pivotal role in extracting meaningful insights and patterns.

The results of our experiments demonstrated the effectiveness of these techniques in reducing noise, standardizing text formats, and improving the overall quality of data for subsequent analyses. Tokenization helped break down the text into manageable units, while normalization ensured consistency in spelling and grammar, thereby facilitating more accurate information retrieval.

Moreover, the removal of stopwords proved instrumental in filtering out irrelevant terms, focusing our analyses on content words crucial for understanding the underlying semantics of the text.

Additionally, our stemming algorithm showcased its ability to reduce inflectional variations, further refining the data for enhanced analysis.

As we move forward, it is essential to acknowledge the limitations of our study, particularly in the context of the Amharic language's rich morphological structure. Future research endeavors could explore more advanced techniques tailored to the intricacies of Amharic text processing, potentially incorporating machine learning approaches for even greater accuracy and efficiency.

In essence, our study contributes to the broader landscape of natural language processing by providing valuable insights into text processing methodologies tailored to the unique characteristics of the Amharic language. By continuing to refine and innovate in this domain, we can pave the way for more effective information retrieval systems and pave the way for broader applications across various domains.