HiLCoE
School of Computer Science and Technology.

Information Retrieval

Group 4 Members

1. Abubeker Nasir
2. Afomiya Belay
3. Asrat Yehombawork
4. Mahlet Zelalem
5. Sara Tilahun
6. Welid Abdulhafiz

Introduction

In computers and information science, information retrieval refers to the process of selecting information system resources from a collection of those resources that are pertinent to a particular information demand. Searches may use full-text indexing or another type of content-based indexing. The Information retrieval model also provides a means to get information that already exists in electronic format. It consists of three essential processes which are **query**, **documents** and **search results**. A user looking for information needs to formulate a query usually consisting of a small set of keywords summarizing the information needed. documents are represented by index terms or any form of representation. The goal of information retrieval is to find documents that satisfy the user's information needs.

Information Retrieval considers mainly the term weight, identification of word forms, elimination of stop words, stemming and selection of index terms. Term weighting is basically assigning importance level to index terms. it takes place during the text indexing process in order to assess the value of each term to the document. It assigns numerical values to terms that represent their importance in a document in order to improve retrieval effectiveness. To perform weighting We used the suggested methods on our slides which are The term frequency (tf) weights and cosine similarity to give us a score for different documents that share the same representation.

Stemming, as presented on phase one of our assignment, manages to lower the inflection in words to their root forms, hence aiding in the preprocessing of text, words, and documents for text normalization. We used words from our local language (Amharic) to extract the base form of the words by removing affixes from them.

For this project, python language was used with Jupyter IDE to build the back end of our project and for the front end we used Anvil which is an online Python IDE for building full stack web apps.

Text Preprocessing

Data preprocessing is an essential step in building a Machine Learning model and depending on how well the data has been preprocessed; the results are seen. In NLP, text preprocessing is the first step in the process of building a model.

from the various text processing steps we used Stemming.

Stemming is a natural language processing technique that lowers inflection in words to their root forms, hence aiding in the preprocessing of text, words, and documents for text normalization. We have tried to adopt a stemming algorithm for one of the local languages(Amharic). The programming language that has been used for the purpose of the research is Python using Jupyter notebook IDE.

Steps

1. First we got some suffixes and prefixes of Amharic language by browsing the internet and by finding similar published papers.

2. We then created a function that converts the input Amharic string into its phonetic form (cv form).

3. We then created a function that changes it back to a normal non phonetic form.

4. We took the input from the keyboard, stored it in a string and used them. ends with and. starts with function to check if it contains any of the suffixes and prefixes it removes.

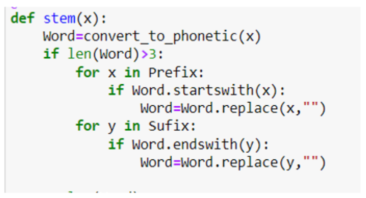
5. Lastly we convert it back to the non-phonetic form using the reverse function.

Test Words Used

የመዝርያ	ኩባያዎቹን
ማዳበሪያዎች	ባህላቸው
ከመፈልፈልዎ	ጨርቆቹን
በመዘረር	መንገዶቹ
ድመቶቹ	የበቆሎ
ወተቱ	መሬትዎ
ዘፋኞቹ	መሬትዎን
አለብዎት	የመጨረሻውን

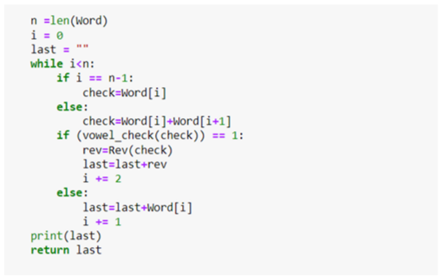
Most of the test words in the folder (Corpus.txt) work correctly when we apply the algorithm to them, however some of them did not. For instance, since the "አለብዎት" is regarded as a prefix, it removes it, the "አ" did not work well. Other prefixes also experience similar issues; for instance, since "በጎቹን" is regarded as a prefix, is removed. We refer to this issue as over-stemming. In comparison to over-stemming, we have encountered less under-stemming.

Some important functions of the stemmer



```python
def stem(x):
    Word=convert_to_phonetic(x)
    if len(Word)>3:
        for x in Prefix:
            if Word.startswith(x):
                Word=Word.replace(x,"")
        for y in Sufix:
            if Word.endswith(y):
                Word=Word.replace(y,"")
```

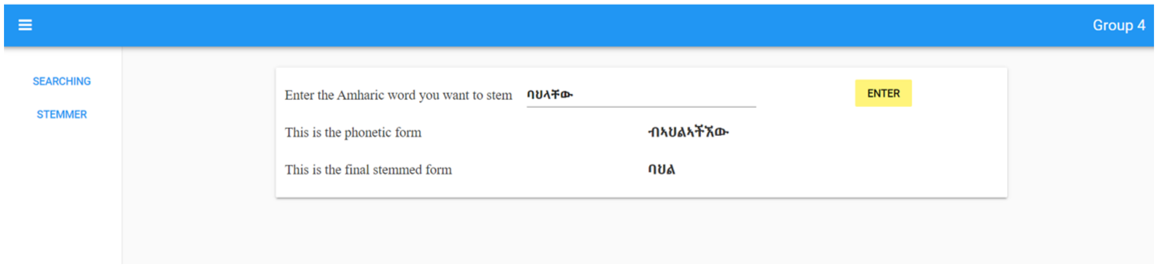
This is used to remove the prefixes and suffixes accordingly



```python
n =len(Word)
i = 0
last = ""
while i<n:
    if i == n-1:
        check=Word[i]
    else:
        check=Word[i]+Word[i+1]
    if (vowel_check(check)) == 1:
        rev=Rev(check)
        last=last+rev
        i += 2
    else:
        last=last+Word[i]
        i += 1
print(last)
return last
```

This is used to reverse to non-phonetic and print the stemmed form

Interface of the stemmer



Enter the Amharic word you want to stem ባህላቸው

This is the phonetic form ብኣህልኣችኧው

This is the final stemmed form ባህል

Stop Word Removal

Stop words are commonly used words in documents. These words do not really signify any importance as they do not help in distinguishing two documents.

we tried implementing stop word removal in our documents. the system tried to remove some stop words but it was not effective as it was hard to distinguish some Amharic words.

System Description

Our system was built in two phases

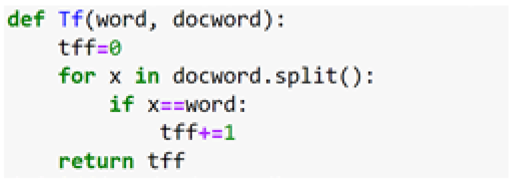
1st Phase – full functioning Amharic word stemmer

2nd Phase – Searching and Indexing

Searching and Indexing

Our system takes input queries from users then it indexes and retrieves ranked documents based on their weight. The corpus we used contains 100 documents from our local language. We used term frequency (tf) to evaluate the frequency of occurrence of a term to indicate its relevancy in describing a document.

The code below shows how we implemented tf in our system.

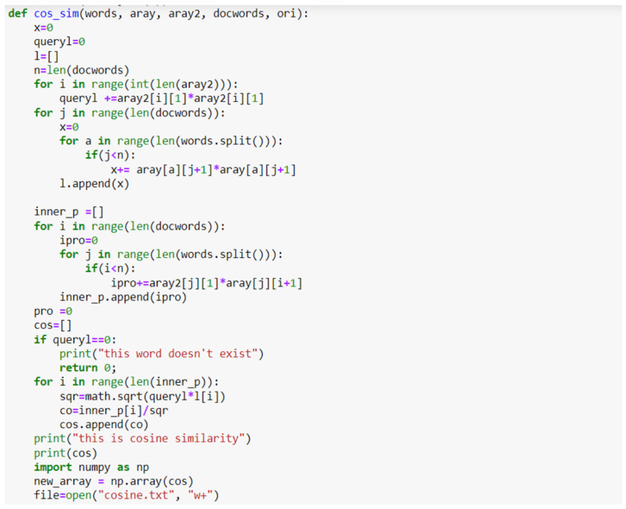


```
def Tf(word, docword):
    tff=0
    for x in docword.split():
        if x==word:
            tff+=1
    return tff
```

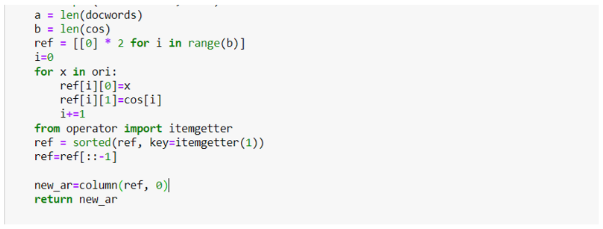
We've tried to work on inverse document frequency (IDF) or collection frequency) weights but it couldn't be implemented due to unforeseen reasons.

Cosine similarity has also been implemented to measure the similarity between vectors.

The code below shows implementation of cosine similarity in our system

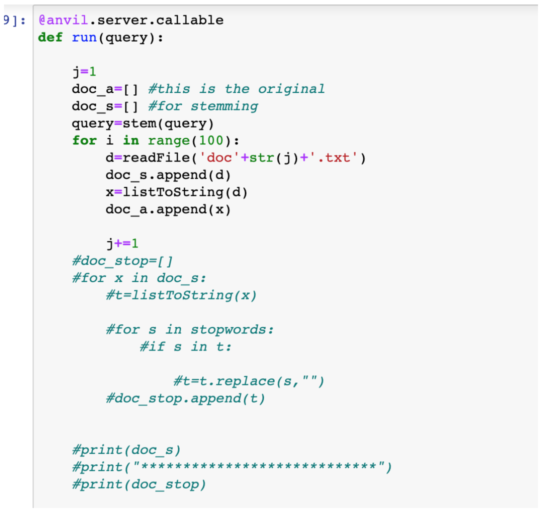


```python
def cos_sim(words, aray, aray2, docwords, ori):
    x=0
    queryl=0
    l=[]
    n=len(docwords)
    for i in range(int(len(aray2))):
        queryl +=aray2[i][1]*aray2[i][1]
    for j in range(len(docwords)):
        x=0
        for a in range(len(words.split())):
            if(j<n):
                x+= aray[a][j+1]*aray[a][j+1]
        l.append(x)

    inner_p =[]
    for i in range(len(docwords)):
        ipro=0
        for j in range(len(words.split())):
            if(i<n):
                ipro+=aray2[j][1]*aray[j][i+1]
        inner_p.append(ipro)
    pro =0
    cos=[]
    if queryl==0:
        print("this word doesn't exist")
        return 0;
    for i in range(len(inner_p)):
        sqr=math.sqrt(queryl*l[i])
        co=inner_p[i]/sqr
        cos.append(co)
    print("this is cosine similarity")
    print(cos)
    import numpy as np
    new_array = np.array(cos)
    file=open("cosine.txt", "w+")
```

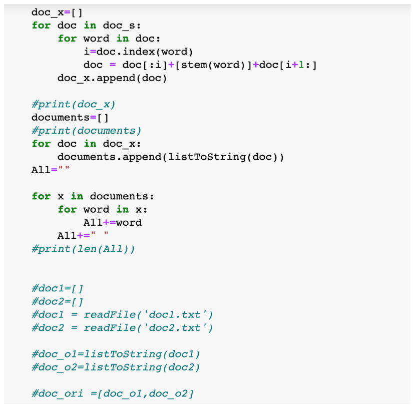


```python
    a = len(docwords)
    b = len(cos)
    ref = [[0] * 2 for i in range(b)]
    i=0
    for x in ori:
        ref[i][0]=x
        ref[i][1]=cos[i]
        i+=1
    from operator import itemgetter
    ref = sorted(ref, key=itemgetter(1))
    ref=ref[::-1]

    new_ar=column(ref, 0)
    return new_ar
```

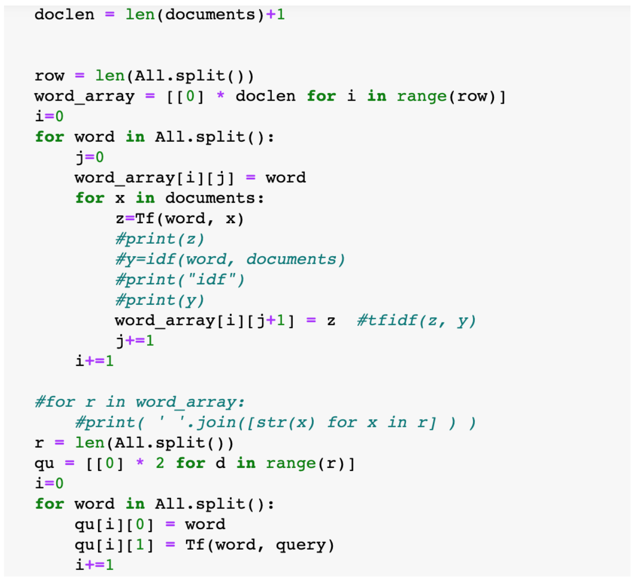
The code below shows implementation of a function that gets the query as a parameter and does the searching and returns the information in our system



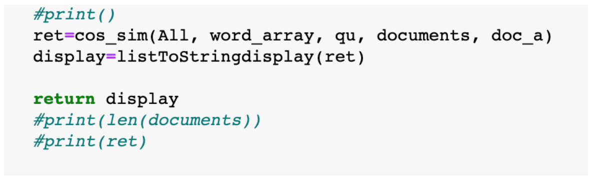
```
9]: @anvil.server.callable
    def run(query):

        j=1
        doc_a=[] #this is the original
        doc_s=[] #for stemming
        query=stem(query)
        for i in range(100):
            d=readFile('doc'+str(j)+'.txt')
            doc_s.append(d)
            x=listToString(d)
            doc_a.append(x)

            j+=1
        #doc_stop=[]
        #for x in doc_s:
            #t=listToString(x)

            #for s in stopwords:
                #if s in t:

                    #t=t.replace(s,"")
            #doc_stop.append(t)


        #print(doc_s)
        #print("****************************")
        #print(doc_stop)
```



```
    doc_x=[]
    for doc in doc_s:
        for word in doc:
            i=doc.index(word)
            doc = doc[:i]+[stem(word)]+doc[i+1:]
        doc_x.append(doc)

    #print(doc_x)
    documents=[]
    #print(documents)
    for doc in doc_x:
        documents.append(listToString(doc))
    All=""

    for x in documents:
        for word in x:
            All+=word
        All+=" "
    #print(len(All))


    #doc1=[]
    #doc2=[]
    #doc1 = readFile('doc1.txt')
    #doc2 = readFile('doc2.txt')

    #doc_o1=listToString(doc1)
    #doc_o2=listToString(doc2)

    #doc_ori =[doc_o1,doc_o2]
```

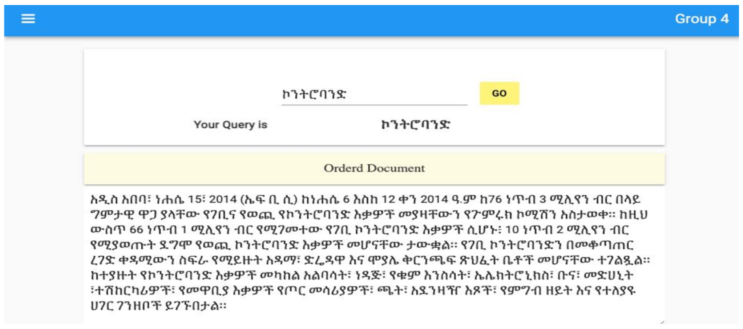


```
    doclen = len(documents)+1


    row = len(All.split())
    word_array = [[0] * doclen for i in range(row)]
    i=0
    for word in All.split():
        j=0
        word_array[i][j] = word
        for x in documents:
            z=Tf(word, x)
            #print(z)
            #y=idf(word, documents)
            #print("idf")
            #print(y)
            word_array[i][j+1] = z  #tfidf(z, y)
            j+=1
        i+=1

    #for r in word_array:
        #print( ' '.join([str(x) for x in r] ) )
    r = len(All.split())
    qu = [[0] * 2 for d in range(r)]
    i=0
    for word in All.split():
        qu[i][0] = word
        qu[i][1] = Tf(word, query)
        i+=1
```



```
    #print()
    ret=cos_sim(All, word_array, qu, documents, doc_a)
    display=listToStringdisplay(ret)

    return display
    #print(len(documents))
    #print(ret)
```

Interface of the System



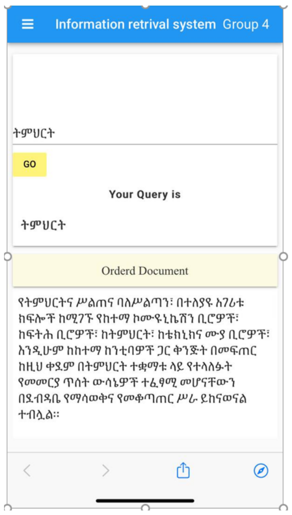
ኮንትሮባንድ

Your Query is ኮንትሮባንድ

Orderd Document

አዲስ አበባ፣ ነሐሴ 15፣ 2014 (ኤፍ ቢ ሲ) ከነሐሴ 6 እስከ 12 ቀን 2014 ዓ.ም ከ76 ነጥብ 3 ሚሊየን ብር በላይ ግምታዊ ዋጋ ያላቸው የገቢና የወጪ የኮንትሮባንድ እቃዎች መያዛቸውን የጉምሩክ ኮሚሽን አስታወቀ። ከዚህ ውስጥ 66 ነጥብ 1 ሚሊየን ብር የሚገመተው የገቢ ኮንትሮባንድ እቃዎች ሲሆኑ፣ 10 ነጥብ 2 ሚሊየን ብር የሚያወጡት ደግሞ የወጪ ኮንትሮባንድ እቃዎች መሆናቸው ታውቋል። የገቢ ኮንትሮባንድን በመቆጣጠር ረገድ ቀዳሚውን ስፍራ የሚይዙት አዳማ፣ ድሬዳዋ እና ሞያሌ ቅርንጫፍ ጽህፈት ቤቶች መሆናቸው ተገልጿል። ከተያዙት የኮንትሮባንድ እቃዎች መካከል አልባሳት፣ ነዳጅ፣ የቁም እንስሳት፣ ኤሌክትሮኒክስ፣ ቡና፣ መድሀኒት ፣ተሽከርካሪዎች፣ የመዋቢያ እቃዎች የጦር መሳሪያዎች፣ ጫት፣ አደንዛዥ እጾች፣ የምግብ ዘይት እና የተለያዩ ሀገር ገንዘቦች ይገኙበታል።

The system can also be accessed from our smartphones



ትምህርት

Your Query is

ትምህርት

Orderd Document

የትምህርትና ሥልጠና ባለሥልጣን፣ በተለያዩ አገሪቱ ክፍሎች ከሚገኙ የከተማ ኮሙዩኒኬሽን ቢሮዎች፣ ከፍትሕ ቢሮዎች፣ ከትምህርት፣ ከቴክኒክና ሙያ ቢሮዎች፣ እንዲሁም ከከተማ ከንቲባዎች ጋር ቅንጅት በመፍጠር ከዚህ ቀደም በትምህርት ተቋማቱ ላይ የተላለፉት የመመርያ ጥሰት ውሳኔዎች ተፈፃሚ መሆናቸውን በደብዳቤ የማሳወቅና የመቆጣጠር ሥራ ይከናወናል ተብሏል።

You can check it on “https://czmmqicthcg22rcm.anvil.app/PJ4ECCCFSZXKZZ3TEKJJQ3MN “

Note

Since our personal computer is being used as a server for the front end, you might encounter a server not found error or method run not found. In that case, run(compile) every code in

Conclusion

Finding documents that meet the user's information needs is the purpose of information retrieval, to sum up. Our strategy was to construct the back end of our project in Python using the Jupyter IDE, and the front end in Anvil, an online Python IDE for building full stack web apps.

When we apply the algorithm to the test words in the folder (Corpus.txt) for stemming, most of them operate as intended, but some of them do not. We have seen less under-stemming in this method compared to over-stemming.
The algorithm attempted to eliminate several stop words, but it was unsuccessful since it was difficult to tell some Amharic words apart. 100 documents in the corpus we used are in our local language. In spite of our best efforts, we were unable to put the inverse document frequency (IDF) or collection frequency weights into practice.

Reference

● https://openreview.net/pdf?id=r1edp2VYwH
●https://www.researchgate.net/publication/250900075_Stemming_of_Amharic_Wor ds_for_Information_Retrieval
●https://www.codegrepper.com/code-examples/python/read+amharic+pdf+in+python