Email recommendations using cosine similarity on Apache-Drill mailing lists

Execution time:

Total execution time is less than 2 mins.

1. Generating TF-IDF matrix
   ```
   CPU times: user 1min 6s, sys: 168 ms, total: 1min 7s
   Wall time: 1min 7s
   ```
2. Calculating cosine similarities
   ```
   CPU times: user 22.9 s, sys: 10.2 s, total: 33.1 s
   Wall time: 35.9 s
   ```

Memory usage:

Total memory required approx 4GB

1. TF-IDF Sparse Matrix:
   ```
   1MB
   ```
2. Pairwise Cosine-similarity matrix:
   ```
   4025MB ~ 3.93GB
   ```

Result quality:

Results look promising, top 10 recommendations for each email are stored in a csv file. In the testing portion of the script you can specify any email and get the recommendations. Just eyeball it to see how close they are.

Issues:

The size of the cosine_similarity matrix is the problem. It will be too large as the size of the data grows. Next steps will be to use Locality Sensitive Hashing and get 15 approximate nearest neighbours and then re-compute cosine similarities on the set of 15 neighbours and then store the top 10 similar emails

In [3]:
```
import re
import nltk
import pandas as pd
import numpy as np
from sys import getsizeof
from os import listdir
from os.path import isfile, join
from nltk.tokenize import RegexpTokenizer
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

In [5]:
```
tokenizer = RegexpTokenizer(r'\w+')
stopwords = set(nltk.corpus.stopwords.words('english'))
stemmer = SnowballStemmer("english")
```

In [6]:
```
# tokenizer.tokenize('Eighty-seven miles to go, yet.  Onward!')
def tokenize_stop_stem(text):
    tokens = tokenizer.tokenize(text)
    # filter out any tokens not containing letters (e.g., numeric tokens, ra
    filtered_tokens = set()
    for token in tokens:
        token = token.lower()
        if token not in stopwords:
            if not re.search('[0-9]', token):
                try:
                    token = stemmer.stem(token)
                    filtered_tokens.add(token)
                except UnicodeDecodeError:
                    print 'illeagal token ignored:',token
                    pass
    return filtered_tokens
```

In [7]:
```
files = [f for f in listdir("/Users/sanket/Desktop/nlp_emailrecs/sample_data
```

In [8]:
```
getsizeof(files)
```

Out[8]: 200328

In [9]:
```
all_emails = []
for file in files:
    f = open(join("/Users/sanket/Desktop/nlp_emailrecs/sample_data", file))
    text = f.read()
    f.close()
    all_emails.append(text)
```

In [28]:
```
print sum([getsizeof(k) for k in all_emails])/10**6,'MB'
```

34 MB

In [22]:
```
# alltokens = set()
# for file in files:
#     f = open(join("/Users/sanket/Desktop/nlp_emailrecs/sample_data", file),
#     text = f.read()
#     f.close()
#     alltokens = alltokens.union(tokenize_stop_stem(text))
# print len(alltokens)
```

Generate TF-IDF matrix on the emails we have

In [11]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [12]:
```
#define vectorizer parameters
tfidf_vectorizer = TfidfVectorizer(max_features=2000000, stop_words='englisl
```

In [13]:
```
%time tfidf_matrix = tfidf_vectorizer.fit_transform(all_emails)
# terms = tfidf_vectorizer.get_feature_names()
```

```
CPU times: user 1min 6s, sys: 168 ms, total: 1min 7s
Wall time: 1min 7s
```

In [27]:
```
print sum([getsizeof(k) for k in tfidf_matrix])/10**6,'MB'
print 'shape:',tfidf_matrix.shape
```

```
1 MB
shape: (22431, 31245)
```

Generate pairwise cosine similartiy

Distance = 1 - similarity

In [17]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

In [18]:
```
%time cosine_sim = cosine_similarity(tfidf_matrix)
```

```
CPU times: user 22.9 s, sys: 10.2 s, total: 33.1 s
Wall time: 35.9 s
```

In [20]:
```
getsizeof(cosine_sim)/10**6
```

Out[20]:
```
4025
```

In [14]:
```
cosine_sim.mean()
```

Out[14]:
```
0.04429858241268634
```

In [30]:
```
x=cosine_sim[0]
x.argsort()[::-1][:10]
```

Out[30]:
```
array([    0, 19805, 18548, 14945, 17521,   907,  3202, 16175, 20476, 110
95])
```

Top 10 similar emails based on Cosine similarity saved to similarity_results.csv

In [51]:
```
ls=[]
for i in range(cosine_sim.shape[0]):
    #print i
    temp = []
    temp.append(files[i])
    x = cosine_sim[i].argsort()[::-1][1:11]
    for j in x:
        temp.append(files[int(j)])
    ls.append(temp)

dataFrame = pd.DataFrame(ls)
dataFrame.head()
dataFrame.to_csv('/Users/sanket/Desktop/nlp_emailrecs/similarity_results.csv
```

...

Test recommendations

In [1]:
```
import pandas as pd
```

In [2]:
```
recommendations = pd.read_csv('/Users/sanket/Desktop/nlp_emailrecs/similarit
recommendations.head()
```

In [12]:
```
recommendations.shape
```

Out[12]:
```
(22431, 11)
```

In [31]:
```
index = np.random.randint(0,22430)
index
```

Out[31]:
```
9760
```

In [32]:
```
mainEmail = recommendations.iloc[index,:][0]
mainEmail = open('/Users/sanket/Desktop/nlp_emailrecs/sample_data/'+mainEmai
print mainEmail
```

```
[jira] [Created] (DRILL-3849) IOB Exception : ORDER BY ROW_KEY over date_
epoch_be encoded data
Khurram Faraaz created DRILL-3849:
-------------------------------------

             Summary: IOB Exception : ORDER BY ROW_KEY over date_epoch_be
encoded data
                 Key: DRILL-3849
                 URL: https://issues.apache.org/jira/browse/DRILL-3849 (h
ttps://issues.apache.org/jira/browse/DRILL-3849)
             Project: Apache Drill
          Issue Type: Bug
          Components: Execution - Flow
    Affects Versions: 1.2.0
         Environment: 4 node cluster CentOS
            Reporter: Khurram Faraaz
            Assignee: Smidth Panchamia
```

In [34]:
```
recEmail1 = recommendations.iloc[index,:][5]
recEmail1 = open('/Users/sanket/Desktop/nlp_emailrecs/sample_data/'+recEmail
print recEmail1
```

```
xt(AbstractSingleRecordBatch.java:78) ~[drill-java-exec-1.9.0.jar:1.9.0]
        at org.apache.drill.exec.physical.impl.project.ProjectRecordBatc
h.innerNext(ProjectRecordBatch.java:135) ~[drill-java-exec-1.9.0.jar:1.9.
0]
        at org.apache.drill.exec.record.AbstractRecordBatch.next(Abstract
RecordBatch.java:162) ~[drill-java-exec-1.9.0.jar:1.9.0]
        at org.apache.drill.exec.record.AbstractRecordBatch.next(Abstract
RecordBatch.java:119) ~[drill-java-exec-1.9.0.jar:1.9.0]

        at org.apache.drill.exec.record.AbstractRecordBatch.next(Abstract
RecordBatch.java:109) ~[drill-java-exec-1.9.0.jar:1.9.0]
        at org.apache.drill.exec.record.AbstractSingleRecordBatch.innerNe
xt(AbstractSingleRecordBatch.java:51) ~[drill-java-exec-1.9.0.jar:1.9.0]
        at org.apache.drill.exec.physical.impl.project.ProjectRecordBatc
h.innerNext(ProjectRecordBatch.java:135) ~[drill-java-exec-1.9.0.jar:1.9.
0]
        at org.apache.drill.exec.record.AbstractRecordBatch.next(Abstract
RecordBatch.java:162) ~[drill-java-exec-1.9.0.jar:1.9.0]
        at org.apache.drill.exec.record.AbstractRecordBatch.next(Abstract
RecordBatch.java:119) ~[drill-java-exec-1.9.0.jar:1.9.0]
```

Kmeans clustering of TF-IDF vectors

In [18]:
```
from sklearn.cluster import KMeans
```

In [19]:
```
def cluster_gridsearch(num_clusters):
    km = KMeans(n_clusters=num_clusters,n_jobs=-1)
    %time km.fit(tfidf_matrix)
    print km.inertia_
    return km.inertia_
```

In [33]:
```
num_clusters = 5
km = KMeans(n_clusters=num_clusters,n_jobs=-1)
```

In [34]:
```
%time km.fit(tfidf_matrix)
```

```
CPU times: user 76.4 ms, sys: 46.4 ms, total: 123 ms
Wall time: 1.43 s
```

Out[34]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=5, n_init=10, n_jobs=-1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [52]:
```
label_df = pd.DataFrame(km.labels_)
label_df[0].value_counts()
```

Out[52]:
```
2    315
0    296
1    179
3    159
4     51
Name: 0, dtype: int64
```

In [59]:
```
km.inertia_
```

Out[59]:
```
836.2358911719807
```

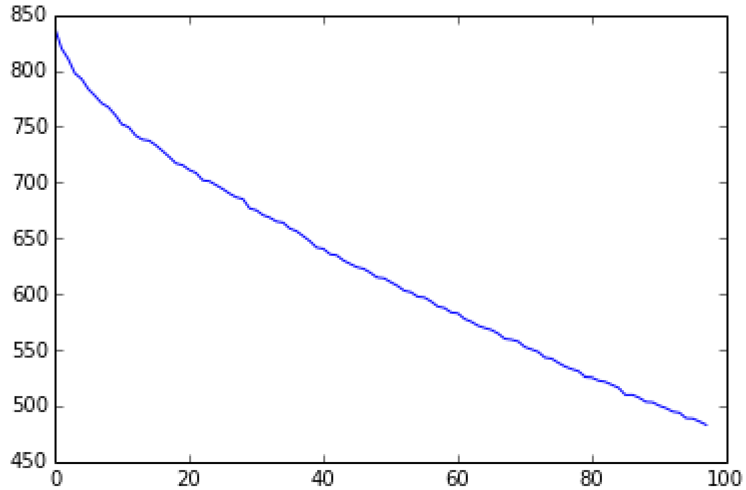
In [20]:
```
error_list = []
```

In [21]:
```
for i in range(5,15,5):
    print i,
    error_list.append(cluster_gridsearch(i))
```

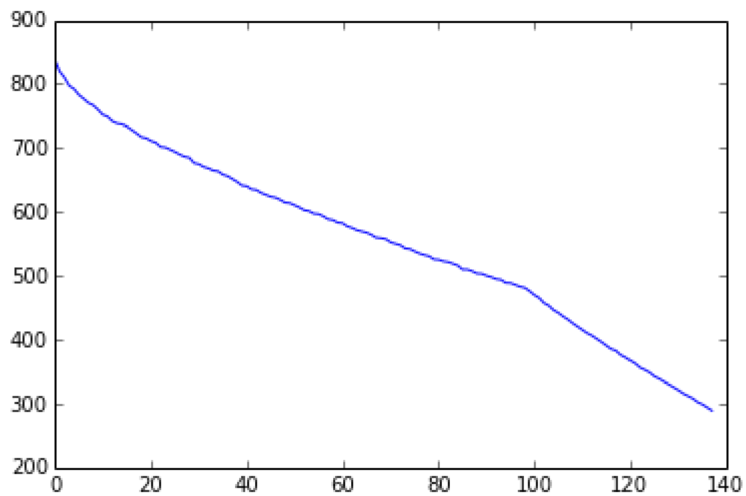
```
5CPU times: user 629 ms, sys: 336 ms, total: 965 ms
Wall time: 2min 5s
 19890.7005316
10CPU times: user 696 ms, sys: 142 ms, total: 839 ms
Wall time: 4min 15s
 19299.059155
```

In [22]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [23]:
```
plt.plot(error_list)
plt.show()
```



In [25]:
```
plt.plot(error_list)
plt.show()
```



Heirarchical Clustering on the data

In [39]:
```
# Warnning it will take too long to run. Remove comments to execute
from scipy.cluster.hierarchy import ward, dendrogram
dist = 1 - cosine_sim
# linkage_matrix = ward(dist)
# fig, ax = plt.subplots(figsize=(15, 20))
```