Visualizing_Stranger_Danger-grouped

November 14, 2018

0.1 Analyzing "stranger","danger" query

Melissa was interested to hear how the phrase “stranger danger” has been used over time, and where it came from. Therefore we run a query against the BL books using the words “stranger” and “danger” within the same sentence, which generates an ouput file (stranger_danger.yml).

This output file has an entry for each book where sentences matching the criteria hashavebeen found, which follows this schema: - Tittle - Publisher - Year - ID_Book - SENTENCE MATCHING THE CRITERIA - Page - SENTENCE MATCHING THE CRITERIA - Page

For example: - Poems ... viz. The Hermaphrodite. The Remedie of Love. Elegies. Sonnets, with other poems - Richard Hodgkinson, for W. W. and Laurence Blaikelocke - 1640 - ’000241254’ - [stranger, ”, loves, delight, and, sweetest, blisse, is, got, with, greatest, danger] - ’000070’

Therefore the "sentences" are naturally grouped by the book where they have found. However, a book can be replicated in our initial dataset. So, we have to be carreful when we analyse the data. For that reason, we are going to read first the input file in a dataframe (called bdf), and later we are going to create a new one (called bdf_t) where we have grouped the books by code, and aggregate all the sentences into the same column.

0.1.1 Importing the python libraries

```
In [1]: import pandas as pd
        import yaml
        import matplotlib.pyplot as plt
        import numpy as np
        import collections
        import nltk
        #nltk.download('sentiwordnet')
        #nltk.download('wordnet')
        from nltk.corpus import sentiwordnet as swn
        from textblob import TextBlob
        import re
        from wordcloud import WordCloud
```

0.1.2 Reading the input file and importing into the bdf dataframe

```
In [130]: filename= "stranger_danger.yml"
          with open(filename, 'r') as f:
                  results = yaml.load(f)
```

```
bdf = pd.DataFrame(results['(stranger, danger)'])
bdf.rename(columns={0: 'title', 1: 'publisher', 2:'year', 3: 'code', 4: 'content'} ,
```

Exploring the bdf dataframe Each row represente a book entry, which sentences maching the criteria have been found.

```
In [131]: print "First row of the dataframe:"
          bdf.head(1)
```

```
First row of the dataframe:
```

```
Out[131]:                                                    title  \
          0  Poems ... viz. The Hermaphrodite. The Remedie ...

                                                   publisher  year       code  \
          0  Richard Hodgkinson, for W. W. and Laurence Bla...  1640  000241254

                                                     content
          0  [[[stranger, , loves, delight, and, sweetest, ...
```

```
In [132]: print "Number of books (including repetitions) - Number of columns: ", bdf.shape
```

```
Number of books (including repetitions) - Number of columns:  (948, 5)
```

```
In [133]: print "The total number of books (grouped by their codes) is ", bdf.groupby(['code']

          print "Same result:," , bdf['code'].nunique()
```

```
The total number of books (grouped by their codes) is  886
Same result:, 886
```

0.1.3 Creating a new dataframe (bdf_t)

We have merged in bdf_t all the rows of the same books into one, by aggregating all the sentences into "content" column. Furthermore, we have added a new column (called repetition), to store the number of times that a book appears in our dataset.

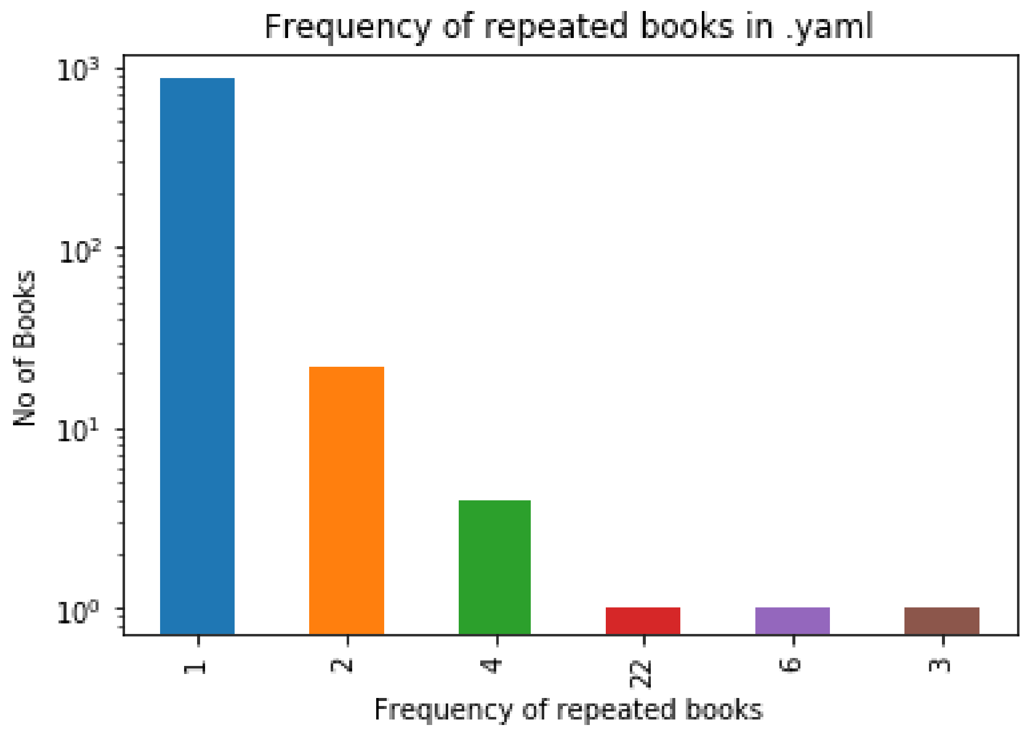
```
In [134]: bdf_t = pd.DataFrame()
          for c in list_codes:
              df = bdf[bdf['code'] == c]
              repetition=len(df)
              if repetition!=1:
                  publisher=df.iloc[0]["publisher"]
                  df=df.groupby(["code","title","year"], as_index=False)['content'].sum()
                  df['publisher']=publisher
              df=df.assign(repetition = repetition)
              bdf_t = bdf_t.append(df, ignore_index=True)
          bdf_t.set_index('code',inplace = True)
```

Exploring the bdf_t dataframe --> From now on, we are going to work with bdf_t dataframe

```
In [136]: print "My first line is:"
          bdf_t.head(1)

My first line is:


Out[136]:                                                           content  \
          code
          000241254  [[stranger, , loves, delight, and, sweetest, ...

                                                          publisher  repetition  \
          code
          000241254  Richard Hodgkinson, for W. W. and Laurence Bla...           1

                                                              title  year
          code
          000241254  Poems ... viz. The Hermaphrodite. The Remedie ...  1640

In [137]: print "Number of books (without repetitions) - Number of columns: ", bdf_t.shape

Number of books (without repetitions) - Number of columns:  (886, 5)
```

0.1.4 Exploring the frequency of books repetition in our dataset

```
In [138]: bdf_t['repetition'].value_counts().plot(kind='bar', title='Frequency of repeated bool
          plt.yscale('log', nonposy='clip')
          plt.xlabel('Frequency of repeated books', fontsize=10)
          plt.ylabel('No of Books', fontsize=10)
          print bdf_t['repetition'].value_counts()

1     857
2      22
4       4
22      1
6       1
3       1
Name: repetition, dtype: int64
```



0.1.5 Getting the most replicated book's title

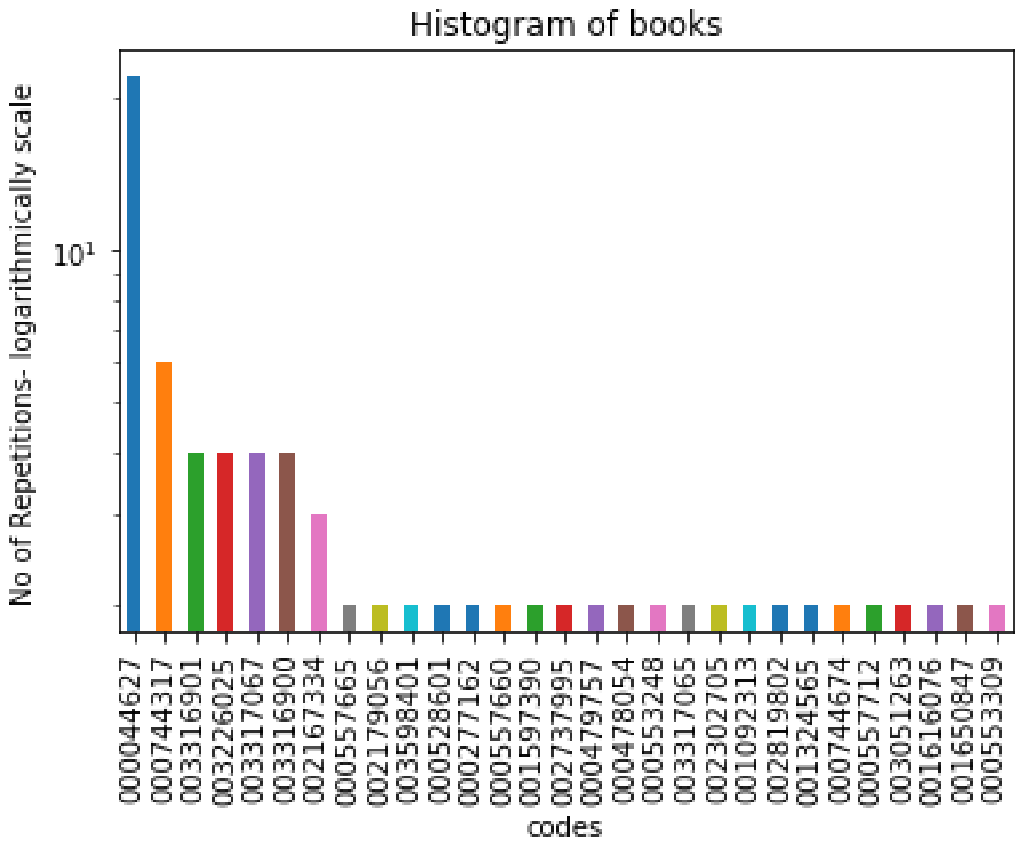
```
In [139]: print "The book:", bdf_t[bdf_t['repetition']== bdf_t['repetition'].max()]["title"], '
```

```
The book: code
000044627    Aldine O'er Land and Sea. Library
Name: title, dtype: object  --> has 22  replications
```

0.1.6 Exploring the books' repetitions that appear more than once in our dataset

```
In [140]: freq_code=bdf_t[bdf_t['repetition'] >1]['repetition']
          freq_code.sort_values(ascending=False).plot(kind='bar', title='Histogram of books')
          plt.yscale('log', nonposy='clip')
          plt.xlabel('codes', fontsize=10)
          plt.ylabel('No of Repetitions- logarithmically scale', fontsize=10)
```

```
Out[140]: Text(0,0.5,u'No of Repetitions- logarithmically scale')
```

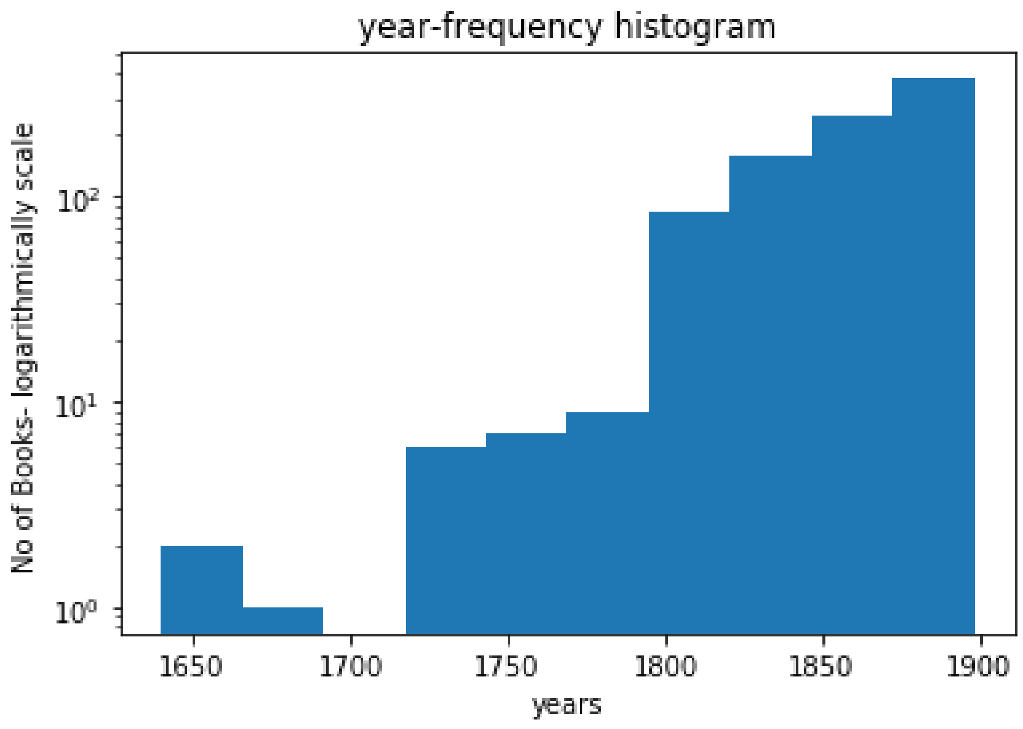


0.1.7 year-frequency histogram

```
In [141]: list_year = bdf_t["year"].values
          print "We have ", len(list_year)," books, which their year frequency/distribution is
          plt.hist(list_year)
          plt.yscale('log', nonposy='clip')
          plt.title("year-frequency histogram")
          plt.xlabel('years', fontsize=10)
          plt.ylabel('No of Books- logarithmically scale', fontsize=10)
```

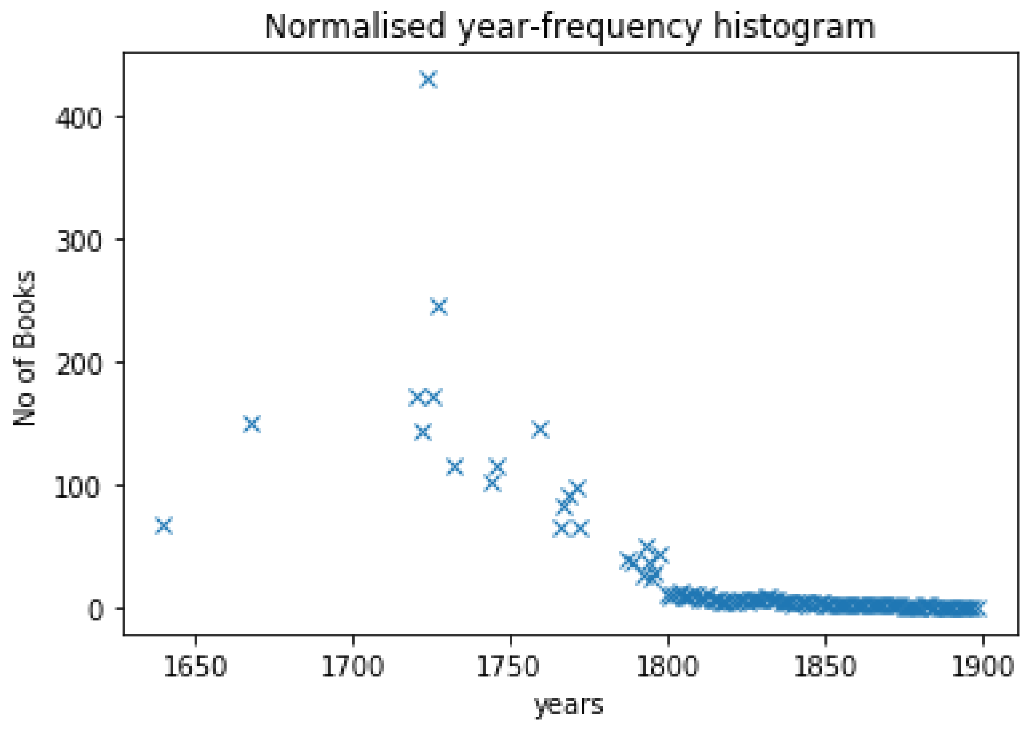
```
We have  886  books, which their year frequency/distribution is the following
```

```
Out[141]: Text(0,0.5,u'No of Books- logarithmically scale')
```



0.1.8 Normalisation

```
In [142]: normal_filename = 'normaliser.yml'
          with open('./' + normal_filename, 'r') as f:
              publication = yaml.load(f)

In [97]: normed_results = {}
         d_year=collections.Counter(list_year)
         for year in d_year:
             if year>0:
                 normed_results[year] = year/float(publication[year][0])
         plt.plot(normed_results.keys(), normed_results.values(), 'x')
         plt.title("Normalised year-frequency histogram")
         plt.xlabel('years', fontsize=10)
         plt.ylabel('No of Books', fontsize=10)

Out[97]: Text(0,0.5,u'No of Books')
```



These stages carry out normalisation: dividing the per year word occurence with the per year book occurence to get a words per book per year measure. - year: [count, count_pages, count_words]} - 1788: [102, 22588, 4055011]

0.1.9 Getting the total number of sentences found, and the book title which has the max number of sentences

```
In [155]: num_books = len(bdf_t)
          max_sentences =0
          total_sentences =0
          for index, row in bdf_t.iterrows():
              num_sentences= len(row['content'])
              total_sentences= total_sentences + num_sentences
              if num_sentences > max_sentences:
                  max_sentences= num_sentences
                  max_code = index
```

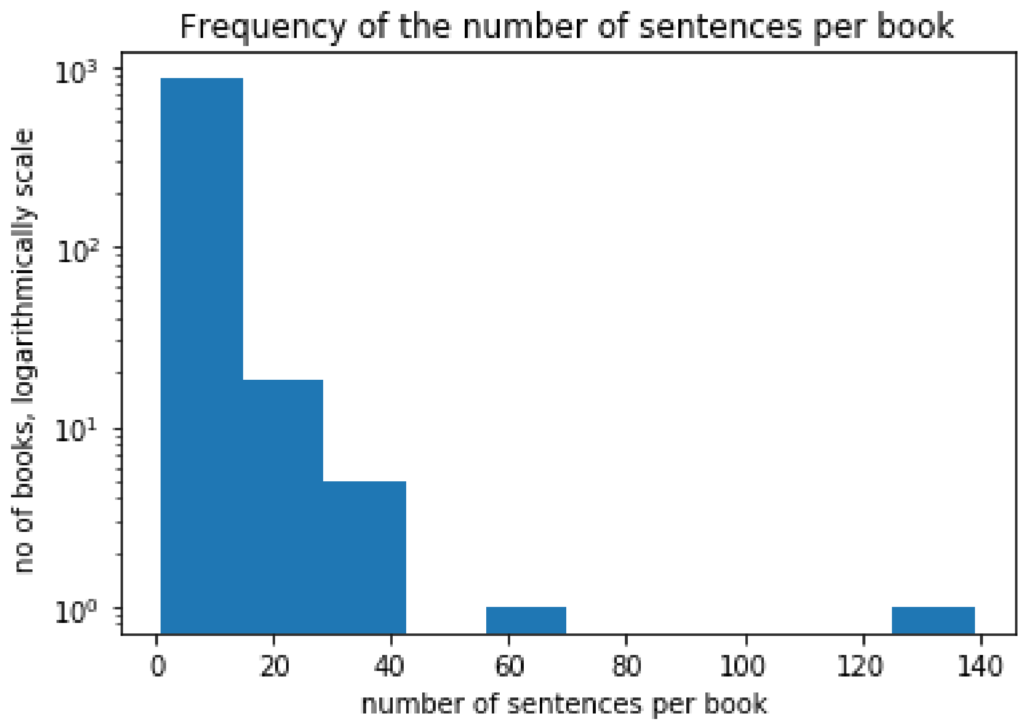
```
In [160]: print "Total sentences found is ", total_sentences, " \n"
          max_title=bdf_t.loc[max_code]["title"]
          max_year=bdf_t.loc[max_code]["year"]
          print"The book --", max_title, "--publish at ", max_year," ,(code  ",max_code, ") ha
```

```
Total sentences found is  5159

The book -- Aldine O'er Land and Sea. Library --publish at  1890  ,(code   000044627 ) has the
```

0.1.10 Frequency of the number of sentences per book

```
In [100]: num_books = len(bdf_t)
          max_sentences =0
          total_sentences =0
          num_senteces_freq=[]
          for index, row in bdf_t.iterrows():
              num_sentences= len(row['content'])
              num_senteces_freq.append(num_sentences)

          plt.hist(num_senteces_freq)
          plt.yscale('log', nonposy='clip')
          plt.title("Frequency of the number of sentences per book")
          plt.xlabel('number of sentences per book ', fontsize=10)
          plt.ylabel('no of books, logarithmically scale ', fontsize=10)
```

```
Out[100]: Text(0,0.5,u'no of books, logarithmically scale ')
```

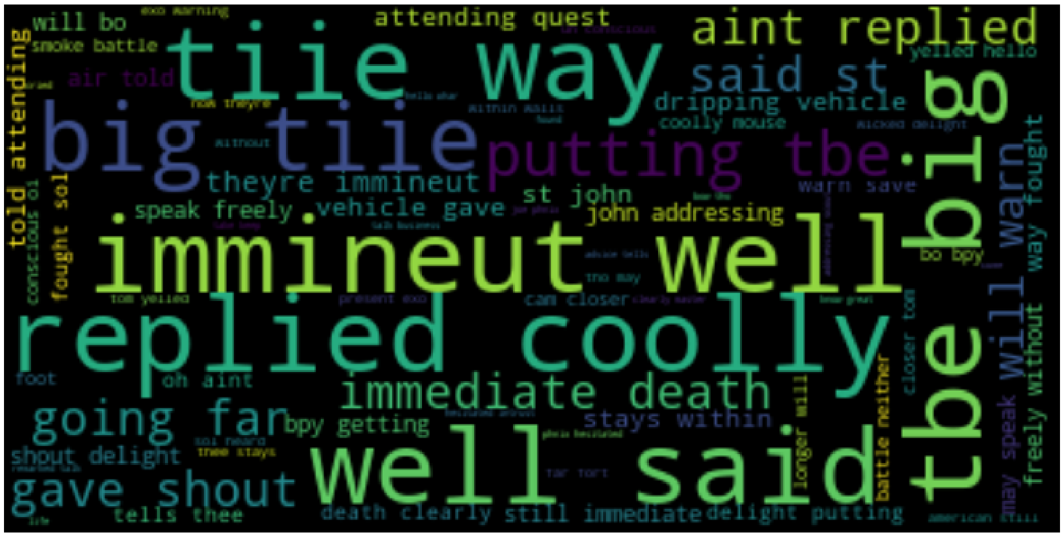


0.1.11 Exploring which words from the book "Aldine “O’er Land and Sea" appear more

Note: We have to remove "stranger" and "danger" words

```
In [150]: text_q1=''
          filter_book= bdf_t.loc[bdf_t["title"] == max_title]["content"]
```

```
for i in filter_book.sum():
    for w in i[0]:
       if w!="stranger" and w!="danger":
           text_q1= text_q1 +" " + w

wordcloud = WordCloud(max_font_size=40).generate(text_q1)
plt.figure(figsize=(12,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



0.1.12 Exploring the words of "Aldine “O’er Land and Sea" book

```
In [102]: def last_chars(x):
              '''
              Utility function to get the last 25 characters.
              '''
              return(x[-25:])

          def clean_text(text):
              '''
              Utility function to clean the text in a text by removing
              links and special characters using regex.
              '''
              return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", t

          def analize_sentiment(text):
              '''
```

```
        Utility function to classify the polarity of a tweet
        using textblob.
        '''

        analysis = TextBlob(clean_text(text))
        if analysis.sentiment.polarity > 0:
            return 1
        elif analysis.sentiment.polarity == 0:
            return 0
        else:
            return -1
```

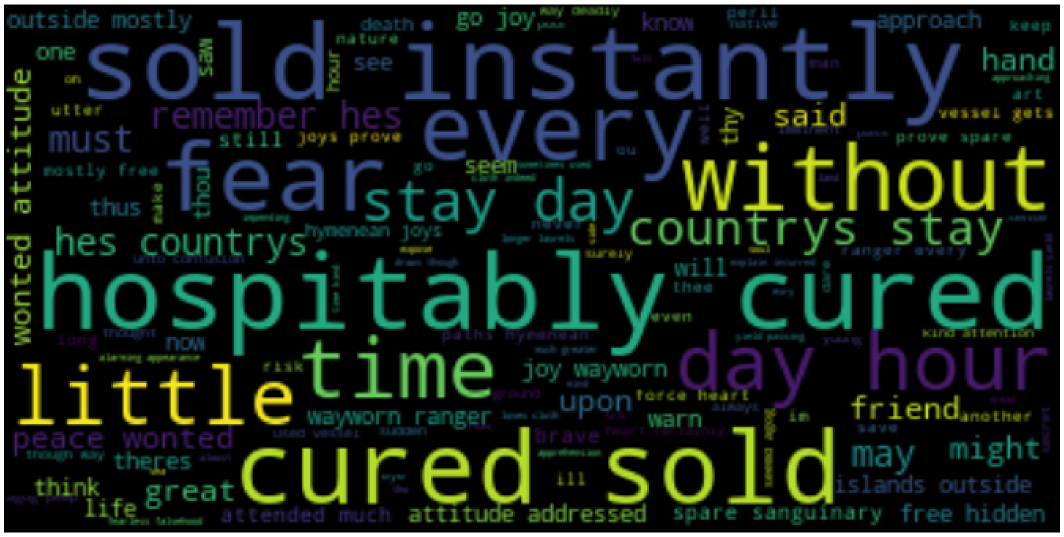
```
In [148]: filter_book= bdf_t.loc[bdf_t["title"] == max_title]["content"]
          text_q1=[]
          for i in filter_book.sum():
              text=''
              for w in i[0]:
                  text= text +" " + w
              text_q1.append(text)

          #print the first the sentiment analysis of the first 10 sentences
          for i in range(10):
              print "The sementiment of ", text_q1[i]," is : " ,analize_sentiment(text_q1[i])
```

```
The sementiment of   danger of his trying to escape while the tramp as a stranger  is :  0
The sementiment of   in love with danger as he cam closer tom yelled hello stranger  is :  1
The sementiment of   love with danger as he cam closer tom yelled hello stranger whar  is :  1
The sementiment of   with danger as he cam closer tom yelled hello stranger whar ye  is :  0
The sementiment of   danger as he cam closer tom yelled hello stranger whar ye gwine  is :  0
The sementiment of   no danger   the second man exolaimed and as the stranger  is :  0
The sementiment of   danger   the second man exolaimed and as the stranger entered  is :  0
The sementiment of   escape the danger  joe phnix hesitated to intrust to a stranger  is :  0
The sementiment of   the danger  joe phnix hesitated to intrust to a stranger the  is :  0
The sementiment of   danger  joe phnix hesitated to intrust to a stranger the secret  is :  -1
```

0.2 Exploring which words appears more in our senteces

Note--> we have to remove "stranger" and "danger" words

```
In [149]: text_q1=''
          for book in range(0,num_books):
                  for w in bdf_t["content"][book][0][0]:
                      if w!="stranger" and w!="danger":
                          text_q1= text_q1 +" " + w

          wordcloud = WordCloud(max_font_size=40).generate(text_q1)
          plt.figure(figsize=(12,10))
          plt.imshow(wordcloud, interpolation="bilinear")
          plt.axis("off")
          plt.show()
```


sold instantly
hospitably cured
cured sold
without
fear
every
little
time
day hour
stay day
countrys stay
hes countrys
remember hes
must
wonted attitude
peace wonted
friend
may
might
upon
hand
said
approach
outside mostly
islands outside
attendeded much
attitude addressed
spare sanguinary
free hidden
great
think
life
theres
wayworn ranger
joy wayworn
go joy
death
one
see
know
thus
now
will
warn