自然言語処理　課題

61716621 星川　潤
メールアドレス：jun527@keio.jp

１．目的

実際の英語圏の子供達が英語を勉強する際、どんな英語から学び始め、やがてどんな言葉を使うようになるだろうか。大人向けのクラシック小説と子供向けの童話のそれぞれに使われている単語のデータを整理し、年齢と言葉の相関を調べることによってこれについて考察していく。

そしてこれらのクラシック小説と童話のデータから、大人より子供がよく使われる頻出単語を集めた「子供向け単語帳」と子供より大人がよく使う言葉を集めた「大人向け単語帳」の作成を試みたいと思う。

２．手法

(1) サンプル単語の抽出

上記のような目的に達するために、まずは童話に登場する単語とクラシック小説に登場する単語を区切る必要がある。

今回これらのデータを収集にあたり、ある 1 冊のみの本から単語を抽出しそれらをサンプルとしてしまうと、その本の話題に関係する単語のみが頻出してしまい、本来の目的を果たせなくなる可能性が高い。よって、今回のサンプルとなる単語集はオンラインに公開されているアメリカの童話とクラシック小説それぞれ任意の 30 冊から合計およそ 5500 字抽出することにした。つまり合計で 60 冊、11000 文字の単語をこの課題では扱うことになる。

童話の単語集を‘ForChild.txt’というファイルに、クラシック小説の単語集を’ForAdult.txt’というファイルに入力し、それを Python3 の以下のようなコードでクオーテーションやピリオドといった記号を除き単語ごとに分割する。

```
In [59]: import numpy as np
         import pandas as pd
         file=open('ForChild.txt','r',encoding='utf-8')
         text=file.read()
         text=text.replace('" ',' ')
         text=text.replace(',',' ')
         text=text.replace('.',' ')
         text=text.replace(';',' ')
         text=text.replace('"','" ')
         text=text.replace('"',' ')
         text=text.replace('?',' ')
         text=text.replace('!',' ')
         words=[]
         words=text.split()
         print(len(words))
         print(words)

         5624
         ['The', 'Little', 'Gingerbread', 'Man', 'Once', 'upon', 'a', 'time', 'there', 'was', 'an', 'old', 'woman', 'who', 'loved', 'baking', 'ging
         erbread', 'She', 'would', 'bake', 'gingerbread', 'cookies', 'cakes', 'houses', 'and', 'gingerbread', 'people', 'all', 'decorated', 'with',
         'chocolate', 'and', 'peppermint', 'caramel', 'candies', 'and', 'colored', 'frosting', 'She', 'lived', 'with', 'her', 'husband', 'on', 'a',
         'farm', 'at', 'the', 'edge', 'of', 'town', 'The', 'sweet', 'spicy', 'smell', 'of', 'gingerbread', 'brought', 'children', 'skipping', 'an
         d', 'running', 'to', 'see', 'what', 'would', 'be', 'offered', 'that', 'day', 'Unfortunately', 'the', 'children', 'gobbled', 'up', 'the',
```

図１　童話に登場する単語の抽出と分割

(2) 各単語の登場回数を計算

童話において全ての単語を小文字表記し、各単語ごとにそれが文章の中でどれくらいの頻度で登場しているかを数え、それを単語ごとに振り当てｘという（単語,登場頻度）を要素に持つ二次元配列にまとめる。
以下がそのコードである。
仕組みは至ってシンプルであり、各単語ごとにそれが何回登場しているかを数え count という配列にまとめ、それをもとに頻度(=登場回数×1000/単語数)を導出した後それを単語と結合し wordlist という配列にまとめる。
それが終了すれば set という関数を使い同じ単語は圧縮すれば以下のように重なる単語が存在しないリストｘが出来上がる。
ｘの長さは 1552 であり、元々の単語数 5624 のおよそ 3.63 倍に圧縮されている。これを童話の圧縮率と呼ぶとする。

```
In [61]: count=np.zeros(len(words))
wordslist=[]
for i in range(len(words)):
    words[i]=words[i].lower()
    count[i]=float((words.count(words[i]))/len(words)*1000)
    wordslist+=[(words[i],count[i])]
wordslist=set(wordslist)
x=[]
for i in wordslist:
    x+=[i]
print(len(x))
print(x)
```

```
1552
[('but', 7.823613086770981), ('if', 2.6671408250355615), ('walk', 0.17780938833570414), ('outside', 0.3556187766714083), ('one-piece', 0.17780938833570414), ('counting', 0.17780938833570414), ('strange', 0.3556187766714083), ('reward', 0.5334281650071124), ('teachers', 0.3556187766714083), ('absolutely', 0.17780938833570414), ('near', 0.17780938833570414), ('hard', 0.8890469416785206), ('bites', 0.17780938833570414), ('build', 0.17780938833570414), ('plan', 0.17780938833570414), ('joy', 0.3556187766714083), ('backwards', 0.17780938833570414), ('katharine' s', 0.3556187766714083), ('debated', 0.17780938833570414), ('pecks', 0.17780938833570414), ('thinks:', 0.17780938833570414), ('unnaturally', 0.17780938833570414), ('take', 0.17780938833570414), ('grabbed', 0.17780938833570414), ('three', 0.8890469416785206), ('his', 7.467994310099573), ('counted', 0.17780938833570414), ('wheels', 0.17780938833570414), ('go', 1.0668563300142249), ('friendship', 0.17780938833570414), ('color', 0.3556187766714083), ('very', 2.6671408250355615), ('pioneer', 0.17780938833570414), ('lovely', 0.3556187766714083), ('monsieur', 0.7112375533428166), ('gnarble', 0.8890469416785206), ('ago', 0.17780938833570414), ('which', 0.17780938833570414), ('heavens', 0.3556187766714083), ('trouble', 0.17780938833570414), ('foothold', 0.17780938833570414), ('singing', 0.17780938833570414), ('little', 2.6671408250355615), ('rules', 0.17780938833570414), ('for', 7.467994310099573), ('secretly', 0.17780938833570414), ('sliver', 1.0668563300142249), ('feet', 0.7112375533428166), ('zanele', 1.2446657183499288), ('surprises', 0.17780938833570414), ('thing', 0.3556187766714083), ('wish', 0.5334281650071124), ('both', 0.17780938833570414), ('described', 0.3556187766714083), ('as', 3.9118065433854907), ('president', 0.5334281650071124), ('explorer', 0.17780938833570414), ('of', 15.647226173541963), ('roly-poly', 0.17780938833570414), ('rested', 0.17780938833570414), ('frosting', 0.17780938833570414), ('book', 0.17780938833570414), ('life', 0.17780938833570414), ('cheated', 0.17780938833570414), ('done', 0.3556187766714083), ('santa', 0.3556187766714083), ('tests', 0.17780938833570414), ('jacks', 0.17780938833570414), ('thou', 0.17780938833570414), ('station', 0.17780938833570414), ('pure', 0.17780938833570414), ('ducks', 0.17780938833570414), ('suffering', 0.17780938833570414), ('buses', 0.17780938833570414), ('pieces', 0.3556187766714083), ('reach', 0.17780938833570414), ('lying', 0.17780938833570414),
```

図２　童話における各単語の出現頻度

今までは童話のみについて処理を行っていたが、クラシック小説の方でも工程 1~2 と同様に処理し、単語とその出現頻度をまとめたリストｙを作成する。
ちなみに、クラシック小説については、元々のサンプル単語数が 5408、圧縮後のｙの単語数が 1887 であり、圧縮率は 5408/1887=2.87 である。これは童話の圧縮率に比べてかなり小さいことが分かる。

(3) 各単語の出現頻度の図式化

ここで、いままでバラバラのデータであったxとyを結合し、横軸を童話での登場頻度、縦軸をクラシック小説における登場頻度として、各単語を散布図にプロットして観察したいと思う。

そのためには、まず要素(‘単語’，童話の登場頻度 ，クラシック小説の登場頻度)を各単語ごとにまとめたリスト“wordlist”が必要になる。

以下の図3がそのためのPython言語のコードである。

仕組みとしては、まず童話、クラシック小説ともに各単語ごとにflagという要素を加える。そして童話に登場する各単語ごとにそれがクラシック小説のほうで登場しているかを判定し、もしその単語が童話にもクラシック小説にも登場しているのならばflagを立て、童話における登場頻度とクラシック小説における登場頻度をwordlistに代入する。

そしてflagが立っていないような単語は、童話もしくはクラシック小説の登場していない方の登場頻度を0として代入させる。

これでwordlistは出来上がる。

```
In [13]: flugx=np.zeros(len(x))
         flugy=np.zeros(len(y))
         wordlist=[]

         for i in range(len(x)):
             for j in range(len(y)):
                 if x[i][0]==y[j][0]:
                     wordlist+=[(x[i][0],x[i][1],y[j][1])]
                     flugx[i]+=1
                     flugy[j]+=1

         for i in range(len(x)):
             if flugx[i]==0:
                 wordlist+=[(x[i][0],x[i][1],0)]
         for j in range(len(y)):
             if flugy[j]==0:
                 wordlist+=[(y[j][0],0,y[j][1])]
         print(wordlist)
```

```
[('but', 7.823613086770981, 3.5133136094674557), ('if', 2.6671408250355615, 1.6642011834319528), ('walk', 0.17780938833570414, 0.1849112426035503), ('outside', 0.3556187766714083, 0.1849112426035503), ('near', 0.17780938833570414, 0.3698224852071006), ('hard', 0.8890469416785206, 0.5547337278106509), ('joy', 0.3556187766714083, 0.1849112426035503), ('take', 0.17780938833570414, 0.1849112426035503), ('three', 0.8890469416785206, 0.3698224852071006), ('his', 7.467994310099573, 7.581360946745562), ('go', 1.0668563300142249, 0.5547337278106509), ('very', 2.6671408250355615, 2.7736686390532544), ('lovely', 0.3556187766714083, 0.1849112426035503), ('ago', 0.17780938833570414, 0.3698224852071006), ('which', 0.17780938833570414, 6.841715976331361), ('little', 2.6671408250355615, 2.2189349112426036), ('for', 7.467994310099573, 5.362426035502958), ('feet', 0.7112375533428166, 1.2943786982248522), ('thing', 0.3556187766714083, 0.3698224852071006), ('both', 0.17780938833570414, 0.3698224852071006), ('as', 3.9118065433854907, 5.177514792899409), ('of', 15.647226173541963, 39.755917159763314), ('life', 0.17780938833570414, 1.6642011834319528), ('done', 0.3556187766714083, 0.3698224852071006), ('pure', 0.17780938833570414, 0.1849112426035503), ('lying', 0.17780938833570414, 0.1849112426035503), ('happen', 0.3556187766714083, 0.1849112426035503), ('wings', 0.3556187766714083, 0.1849112426035503), ('children', 0.7112375533428166, 1.6642011834319528), ('awake', 0.17780938833570414, 0.1849112426035503), ('livelihood', 0.17780938833570414, 0.1849112426035503), ('smart', 0.17780938833570414, 0.1849112426035503), ('number', 0.17780938833570414, 0.1849112426035503), ('like', 3.200568990042674, 1.8491124260355031), ('hours', 0.5334281650071124, 0.1849112426035503), ('rushes', 0.1778093883357
```

図3　配列wordlistの作成

Wordlistは第一要素に単語、第二要素に童話における登場頻度、第三要素にクラシック小説における登場頻度を記録している。これを見やすくするといかの表になり、今回はこのxを横軸yを縦軸としてプロットしてみる。

In [66]:
```
length=len(wordlist)
w=[]
x_child=np.zeros(length)
y_adult=np.zeros(length)
for i in range(length):
    w+=[(wordlist[i][0])]
    x_child[i]+=wordlist[i][1]
    y_adult[i]+=wordlist[i][2]

df=pd.DataFrame({
    'word' : w,
    'x(child)':x_child,
    'y(adult)':y_adult
})
df.head(10)
```

Out[66]:

	word	x(child)	y(adult)
0	but	7.823613	3.513314
1	if	2.667141	1.664201
2	walk	0.177809	0.184911
3	outside	0.355619	0.184911
4	near	0.177809	0.369822
5	hard	0.889047	0.554734
6	joy	0.355619	0.184911
7	take	0.177809	0.184911
8	three	0.889047	0.369822
9	his	7.467994	7.581361

図 4　wordlist のデータの表

実際にプロットした結果を図 5 に表す。

In [15]:
```
import matplotlib.pyplot as plt
% matplotlib inline
df_x=df.loc[:,['x(child)']]
df_y=df.loc[:,['y(adult)']]
```

In [16]:
```
plt.scatter(df_x,df_y)
plt.title('The Relation Of Frequency')
plt.xlim(-5,40)
plt.ylim(-5,40)
plt.xlabel('The Number Of Words In Kids Books')
plt.ylabel('The Number Of Words In Adult Books')
```

Out[16]: Text(0,0.5,'The Number Of Words In Adult Books')

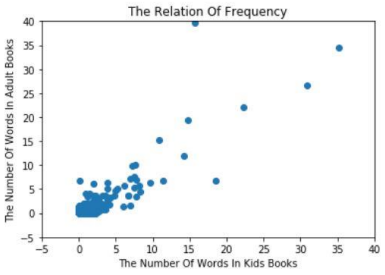

図 5　頻度をプロットした散布図

これから分かる通り、多くの単語は頻度 0~10 の枠に収まっており、意外にも童話に登場する単語とクラシック小説に登場する単語は線形の相関関係にある。

(4) 頻度差の計算

x と y の差分（=童話における登場頻度とクラシック小説における登場頻度の差分）を計算すれば、各単語ごとの頻度の差が求められる。しかし、多く登場する単語ほどその差が大きくなってしまうので、それでは本当の意味での「偏り度」は測れない、よって、ここでの頻度差は以下のような式で出すとする。

$$頻度差=(x-y)^2/(x+y)$$

このようにすることで、登場頻度がそこまで多くない単語も大きな頻度差を持つことができるようになると考えている。

頻度差順に並べた表を図 6 に、頻度差が 0.5 以上の単語を赤色にプロットした散布図を図 7 に表す。

```
In [79]: df_fd=pd.DataFrame({
             'FrequencyDifference': frequent_difference
         })
         dif_df=pd.concat([df,df_fd],axis=1)
         dif_df.sort_values(by=['FrequencyDifference'],ascending=False)
```

Out[79]:

	word	x(child)	y(adult)	FrequencyDifference
21	of	15.647226	39.755917	10.490903
14	which	0.177809	6.841716	6.326304
519	he	18.492176	6.656805	5.569849
210	you	6.934566	1.479290	3.537027
150	the	55.298720	76.553254	3.426230
369	t	6.045519	1.294379	3.075429
981	magic	2.311522	0.000000	2.311522
1385	cake	2.311522	0.000000	2.311522
118	by	1.955903	6.102071	2.133378
322	its	0.889047	4.068047	2.038703
1219	things	1.955903	0.000000	1.955903
403	just	3.556188	0.739645	1.846653
818	scooter	1.778094	0.000000	1.778094
1094	ella	1.778094	0.000000	1.778094
655	didn	1.778094	0.000000	1.778094
146	big	2.667141	0.369822	1.737812

図 6　頻度差も含んだ表

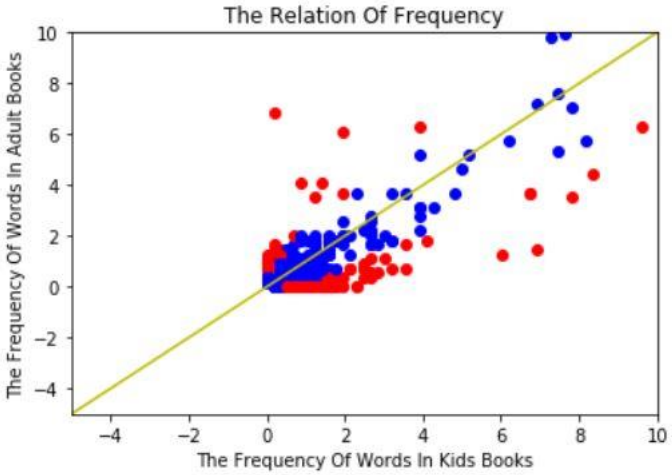

図 7　頻度差が 0.5 以上の単語に赤色をプロットした散布図

ここから、クラシック小説の方が頻繁に使う単語として of, which, the, by, its などが挙げられ、童話に方が頻繁に使う単語として he, you, t, magic, cake, things, just, big, scooter などが挙げられる。やはり which といった関係代名詞は文法的な難易度が高いためクラシック小説の方に頻繁に登場していることが分かる。
そして散布図を見てみると、真ん中の黄色線からある程度離れている点のみが赤色になっている。これにより、頻度の大小にかかわらず頻度差をある程度正当に評価されているということが分かる。

(5) 単語帳の作成
これらのデータを用いて、「子供向け単語帳」と「大人向け単語帳」を実際に作成してみようと思う。
まず、子供向け単語帳に載せる単語の条件として、

・なるべく多めの出現頻度（多すぎてもいけない）
・ある一定以上の頻度差を持つ
・童話での出現頻度の方がクラシック小説での出現頻度よりも多い

の 3 つである考えた。同時に似たようなもので、大人向け単語帳の条件は以下の 3 つとなる。

・なるべく多めの出現頻度（多すぎてもいけない）
・ある一定以上の頻度差を持つ
・クラシック小説での出現頻度の方が童話での出現頻度よりも多い

各単語のうち、頻度さが 0.5 以上のモノでかつ童話のほうにより多く出現している単語を「童話に比較的出やすい単語」と捉えると、以下の図 8 の上の方がそのリストであり、逆に「クラシック小説に比較的出やすい単語」は図 8 の下の方である。面白いことに、その単語数は子供の方がかなり多いことが分かる。

```
In [98]: wordsdata=np.array(dif_df.loc[:,'word'])
         words_for_child=[]
         for i in range(len(reald)):
             if reald[i]>0.5:
                 if realx[i]>realy[i]:
                     words_for_child+=[(wordsdata[i],float(reald[i]))]
         print(len(words_for_child))
         print(words_for_child)
```

```
169
[('but', 1.6387758412698743), ('too', 0.6763601936063272), ('s', 1.200469827655292), ('did', 0.9233266061345556), ('big', 1.737812089040514
7), ('him', 0.8452745531681602), ('i', 0.6915233741961949), ('couldn', 0.6213830380516898), ('all', 0.8947521633210705), ('get', 0.948103158
4988747), ('beautiful', 1.1221634700545327), ('around', 0.6213830380516898), ('you', 3.537027275116812), ('m', 1.0816633275852112), ('your',
0.9528289937009002), ('small', 1.384989909201833), ('because', 0.7685048440468623), ('inside', 0.6213830380516898), ('emma', 0.948103158498
747), ('that', 1.2367455860462218), ('t', 3.075429168436857), ('then', 0.6763601936063272), ('make', 1.5428174034643156), ('just', 1.846653211
22691113), ('said', 1.5370096880937245), ('so', 0.6856985687281885), ('violet', 0.9233266061345556), ('able', 0.6213830380516898), ('they',
0.8947521633210705), ('thought', 1.5712054398462412), ('can', 0.8858871464434852), ('see', 0.7172238081273748), ('help', 1.2930332647787608),
('pink', 0.6213830380516898), ('tree', 0.6424670353580753), ('he', 5.569848794154825), ('reward', 0.5334281650071124), ('monsieur', 0.711237
5533428166), ('gnarble', 0.8890469416785206), ('sliver', 1.0668563300142249), ('zanele', 1.2446657183499288), ('wish', 0.5334281650071124),
('president', 0.5334281650071124), ('tumma', 0.5334281650071124), ('metal', 1.600284495021337), ('gingerbread', 1.600284495021337), ('brough
t', 0.5334281650071124), ('mcfeeglebee', 0.7112375533428166), ('pete', 1.0668563300142249), ('didn', 1.7780938833570412), ('worked', 0.71123
75533428166), ('gugu', 0.5334281650071124), ('makes', 0.5334281650071124), ('workshop', 0.5334281650071124), ('tails', 0.5334281650071124),
('lucky', 1.0668563300142249), ('ve', 0.5334281650071124), ('portrait', 0.5334281650071124), ('although', 0.5334281650071124), ('elf', 1.066
8563300142249), ('oh', 0.5334281650071124), ('absulum', 0.8890469416785206), ('pool', 0.7112375533428166), ('lot', 0.7112375533428166), ('pa
trick', 1.0668563300142249), ('d', 1.2446657183499288), ('playing', 0.5334281650071124), ('teddy', 0.7112375533428166), ('sky', 0.889046941
```

```
In [99]: words_for_adult=[]
         for i in range(len(reald)):
             if reald[i]>0.5:
                 if realx[i]<realy[i]:
                     words_for_adult+=[(wordsdata[i],float(reald[i]))]
         print(len(words_for_adult))
         print(words_for_adult)
```

```
91
[('which', 6.3263039459988), ('of', 10.490902611291158), ('life', 1.199428821089376), ('was', 0.7163213775435987), ('from', 0.553149961762289
9), ('are', 0.5368970109985857), ('may', 0.5058492568056149), ('been', 1.0817119830222053), ('seemed', 0.6881197266741225), ('by', 2.133378
227340606), ('the', 3.4262304114236666), ('right', 0.6373758142976693), ('window', 0.5341039630199651), ('behind', 0.5058492568056149), ('it
s', 2.03870310841646), ('an', 1.2747516285953386), ('between', 1.0221786942434716), ('first', 0.5341039630199651), ('in', 0.634763471313053
6), ('lucian', 0.5547337278106509), ('vampyre', 0.5547337278106509), ('quite', 0.7396449704142012), ('doing', 0.5547337278106509), ('view',
0.5547337278106509), ('taylor', 0.7396449704142012), ('times', 0.7396449704142012), ('lord', 0.9245562130177515), ('society', 0.554733727810
6509), ('various', 0.5547337278106509), ('appeared', 0.5547337278106509), ('fort', 0.5547337278106509), ('helped', 0.7396449704142012), ('st
ood', 1.1094674556213018), ('perhaps', 0.7396449704142012), ('subject', 0.5547337278106509), ('tunstall', 0.5547337278106509), ('sofa', 0.73
96449704142012), ('last', 0.9245562130177515), ('sounds', 0.5547337278106509), ('london', 0.5547337278106509), (')', 0.9245562130177515),
('ship', 0.5547337278106509), ('north', 0.5547337278106509), ('endless', 0.5547337278106509), ('speaker' s', 0.9245562130177515), ('began',
0.7396449704142012), ('youth', 0.5547337278106509), ('country', 0.5547337278106509), ('liberty', 0.5547337278106509), ('study', 0.5547337278
106509), ('pleasant', 0.5547337278106509), ('possess', 0.5547337278106509), ('certain', 0.5547337278106509), ('today', 0.5547337278106509),
('doarty', 1.1094674556213018), ('proper', 1.1094674556213018), ('several', 0.7396449704142012), ('bring', 0.5547337278106509), ('women', 0.
5547337278106509), ('margaret', 0.9245562130177515), ('miss', 0.5547337278106509), ('smoke', 0.7396449704142012), ('yet', 0.554733727810650
9), ('turgenev', 0.5547337278106509), ('street', 0.7396449704142012), ('nora', 0.5547337278106509), ('among', 1.1094674556213018), ('influen
ce', 0.5547337278106509), ('wife', 0.5547337278106509), ('vampire', 0.7396449704142012), ('unity', 0.9245562130177515), ('farris', 0.7396449
```

図 8　童話により頻繁に登場する単語（上）と
クラシック小説により頻繁に登場する単語（下）

単語帳を作るための条件として、これらの単語のうちさらにある程度一定の登場頻度を満たすという条件を付け加える必要がある。

よって今回、頻度差が 0.2～2（2 以上になると、図 6 から分かる通り of や he といった基本的な単語のみしか入ってこなくなるため）かつ総合出現頻度が上位 2863 位に入っている単語を例にとってみると、以下のような単語帳を作ることができた。

【子供向け単語帳】

334words

['if', 'three', 'do', 'blue', 'forest', 'after', 'felt', 'too', 'don', 'did', 'big', 'even', 'much', 'couldn', 'wind', 'get', 'beautiful', 'animals', 'high', 'around', 'm', 'let', 'covered', 'opened', 'horse', 'sometimes', 'never', 'your', 'sister', 'give', 'time', 'grew', 'bed', 'small', 'because', 'inside', 'girl', 'put', 'made', 'something', 'emma', 'my', 'large', 'found', 'love', 'then', 'make', 'already', 'here', 'what', 'just', 'said', 'violet', 'called', 'able', 'named', 'am', 'told', 'heaven', 'thought', 'can', 'see', 'barn', 'help', 'pink', 'tall', 'tree', 'while', 'strange', 'reward', 'teachers', 'katharine’s', 'color', 'monsieur', 'gnarble', 'heavens', 'sliver', 'zanele', 'wish', 'described', 'president', 'santa', 'pieces', 'tumma', 'laughed', 'gogo', 'each', 'metal', 'gingerbread', 'treats', 'brought', 'break', 'digging', 'alligators', 'quilts', 'mcfeeglebee', 'pete', 'didn', '‘what', 'worked', 'cookies', 'invisible', 'wrong', 'gugu', 'makes', 'workshop', 'tails', 'happened', 'wrapping', 'lucky', 'ride', 'heat', 'legs', 've', 'sitting', 'yards', 'portrait', 'although', 'elf', 'oh', 'catch', 'absulum', 'stay', 'wouldn', 'pool', 'lot', 'patrick', 'd', 'playing', '‘i', 'beadwork', 'tells', 'chocolate', 'teddy', 'sky', 'bought', 'well', 'idea', 'tributary', 'wise', 'cool', 'scooter', 'll', 'bank', 'shot', 'picador', 'instead', 'abbaye', 'played', 'gets', 'sad', 'swam', 'nose', 'run', 'wizard', 'asks', 'simply', '--', 'bear', 'fish', 'off', 'doll', 'article', 'eee', 'rocks', 'decorated', 'wolstencroft', 'loads', 'bake', 'insist', 'wrapped', 'fur', 'creatures', 'gnarbles', 'baking', 'uncle', 'tiniest', 'ice', 'attic', 'wasn', 'race', 'stickers', 'senior', 'size', 'gonna', 'nobody', 'reindeer', 'safe', 'liquid', 'map', 'shooflies', 'beavers', 'themselves', 'colored', '45', 'eat', 'colony', 'round', 'jungle', 'crab', 'waves', 'sparkles', 'toy', 'sees', 'paint', 'hole', 'numbers', 'fast', 'katharine', 'she’s', 'counts', 'buckskin', 'toys', 'ella', 'tie', 'maybe', 'tightly', 'pirate', 'bartoli', 'judges', 'sat', 'rachel', 'furnace', 'rock', 'onto', 'loomploy', 'homework', 'tibley', 'coyote', 'night', 'points', 'callum', 'unable', 'crazy', 'turns', 'wonderful', 'pond',

'words', 'neck', 'allow', 'share', 'spoke', 'things', 'edge', 'peace', 'god', 'wendy', 'families', 'cream', 'food', 'wins', 'monkey', 'next', '“one', 'craft', 'loud', 'package', 'woke', 'wumpalump', 'name', 'dog', 'says', 'ordinary', 'feels', 'songs', 'upside', 'castle', 'banks', 'nita', 'having', 'dam', 'tears', 'gold', 'problem', 'shelf', 'l', 'moved', 'ears', 'anyone', 'store', 'order', 'find', 'woodpecker', 'bearing', 'bounces', 'pointy', 'birds', 'driver', 'puts', 'royal', 'coats', 'ocean', 'cars', 'peck', 'memory', 'fins', 'navi', 'sti', 'arrows', 'blyfish', 'shoes', 'terrible', 'favorite', 'mirror', 'fairy', 'making', 'delane', 'handles', 'shouldn', 'closed', 'trail', 'followed', 'bugs', 'he’s', 'gargoyle', 'suzie', 'spot', 'can’t', 'lights', 'killed', 'everyone', 'goes', 'bone', 'care', 'matter', 'fun', 'warned', 'wobble', 'bad', 'mom', 'tries', 'contest', 'swimming', 'cat']

【大人向け単語帳】

278words

['life', 'children', 'may', 'been', 'most', 'seemed', 'being', 'alone', 'river', 'has', 'right', 'these', 'window', 'always', 'hair', 'only', 'behind', 'through', 'side', 'those', 'nothing', 'since', 'leave', 'young', 'room', 'many', 'upon', 'between', 'first', 'road', 'might', 'without', 'byron', 'ribbons', 'lucian', 'our', 'pulled', 'vampyre', 'yesterday', 'quite', 'village', 'doing', 'russian', 'son', 'particularly', 'view', 'zeus', 'south', 'philosophical', 'ruthven', 'custom', 'plantation', 'taylor', 'mildness', 'heavy', 'saying', 'times', 'further', 'fancy', 'dressed', 'mary', 'baden', 'often', 'george', 'silence', 'period', 'russians', 'lord', 'society', 'various', 'appeared', 'fort', 'silent', '(the', 'fathers', 'talking', 'helped', 'sensible', 'rich', 'stood', 'summons', 'published', 'weather', 'cloak', 'whom', 'polidori', 'sunlight', 'rural', 'branches', 'show', 'perhaps', 'subject', 'tunstall', 'boats', 'drawing-room', 'american', 'laws', 'sofa', 'soon', 'matters', 'last', 'creative', 'apparently', 'yes', 'easily', 'sounds', 'field', 'foliage', 'ralph', 'london', 'moment', 'left', ')', 'ship', 'chill', 'steering', 'somewhat', 'childhood', 'mystery',

'table', 'malice', 'anxiety', 'novel', 'faces', 'dry', 'north', 'softened', 'endless', 'speaker’s', 'german', 'represented', 'against', 'poor', 'steam', 'began', 'youth', 'including', 'week', 'country', 'afternoon', 'liberty', 'lost', 'dark', 'study', 'twilight', 'faint', 'believed', 'pleasant', 'principle', 'character', 'possess', 'certain', 'today', 'crossed', 'doarty', 'proper', 'caerleon', 'rush', 'beauty', 'several', 'bring', 'blood', 'difficulty', 'women', 'later', 'margaret', 'miss', 'power', 'claude', 'bare', 'rare', 'smoke', 'windows', 'stained', 'standing', 'group', 'almighty', 'yet', 'dying', 'virgin', 'accomplishment', 'led', 'portions', 'less', 'machen', 'former', 'turgenev', 'moat', 'ghost', 'street', 'nora', 'among', 'account', 'gave', 'force', 'southampton', 'influence', 'wife', 'converse', 'moon', 'passengers', 'seductive', 'form', 'gone', 'vampire', 'woodhouse', 'unity', 'hamlet', 'farris', 'daggers', 'nearly', 'hurstwood', 'battle', 'cut', 'daughters', 'knowledge', 'doubt', 'industrialist', 'lavender', 'governess', 'greasy', 'thérèse', 'emphasis', 'wonder', 'june', 'observations', 'general', 'wedding', 'works', 'fought', 'worlds', 'muslin', 'line', 'position', 'barred', 'narrow', 'social', 'history', 'though', 'forward', 'romantic', 'want', 'church', 'men', 'authority', 'question', 'experience', 'edith', 'simple', 'hoarse', 'service', 'priest', 'somber', 'hardly', 'possessed', 'passed', 'pictures', 'morris', 'asleep', 'hills', 'point', 'towards', 'coming', 'different', 'casual', 'expression', 'valley', 'facts', 'struggle', 'park', 'fallen', 'heliotrope', 'preceding', 'minutes', 'stiff', 'athenian:', 'personal', 'holding']

３．考察

・童話とクラシック小説の圧縮率の違い

手法の(2)から、童話の単語の圧縮率は 3.63 であったのに対し、クラシック小説の圧縮率は 2.87 と大きな違いがあった。この事からどんな事象が予想できるであろうか。
圧縮率とはいわば単語がどれくらい繰り返し使われているのかを指す指標であり、この値が大きいほど同じ単語が繰り返し被って登場することを示している。よって、ここから「大

人が読む書物は子供が読む童話よりも同じ単語が繰り返し使われやすい」という仮設を立てることができる。
もちろんこの仮説が本当に正しいかは、さらなる量と正確さに富んだデータ分析を行ってみないとわからないが、もしこの仮説が正しいということになればこの結果は少し面白い。それというのも、本来私はこの課題に取り組む前に、童話こそ単語をあまり多く知らない子供のために同じ様な単語が繰り返えし使われているのではないかという仮説を立てていたからである。しかし、今回の結果はそれと反することとなった。私が考えつく理由としては以下のようなものが挙げられる。

１．クラシック小説は話の展開が遅い。
通常、クラシック小説というのはやはりある程度成熟した成人向けであり、細かい心情描写、状況の説明などが婉曲的で芸術的に丁寧に表現される傾向がある。その結果、童話と同じような規模でサンプル単語を取ってきても、童話よりも話の展開は遅いためある程度関連のある単語や心情描写に使いやすい単語が繰り返し使われる可能性が高い。

２．童話の方がジャンルに富んでいる。
子供というのは、通常沢山の物事を広く浅く学ぶ時期である。そして大人になっていくにつれ、その知識は狭く広く、つまり専門的になっていく。それが今回の課題でも表れたのではないかと思う。子供は様々なことを学ぶ必要があるからこそ、様々なジャンルに触れさせる必要がある。よってその結果、童話という文化はバリエーションに重きを置いたのではないだろうか。

３．抽象的概念に対する理解の差
通常人は成長するにつれ、具体的で分かりやすい表現より、汎用性に富んだ抽象的な表現に対する理解が進み、徐々に後者を利用機会が多くなる。その観点から考えれば、まだ具体的で特定の概念しか理解していない子供に対しては様々な特定の言葉を巧みに使う必要があり、既にある程度抽象的で包括的な表現に慣れ親しんだ大人に対しては一般的で汎用的な表現を組み合わせて使うことが多くなる。

・大人向け単語帳と子供向け単語帳から分かること

大人向けの単語帳と子供向けの単語帳を比べてみる。
子供向け単語帳の傾向としては、emma, Katharine, nita, gingerbread, santa, elf といった、人の名前やキャラクターの名前を指す言葉が多い。この事から、童話ではキャラクターの名前を頻繁に登場させる傾向があることが示唆される。

それに加えて、gogo や gugu といった擬音語があるのも興味深い。これは明らかに大人向けの単語帳にはない子供向け単語帳独自のモノであるといえる。
対して大人向けの単語帳では、those や these といった複数系の指示語や boats や summons など複数系の名詞が増え、童話よりも話が複雑化していることが予想される。
そして London や Russian, American といった特定の地域やそこに住む人々に言及する言葉も多くなり、Zeus や Hamlet といった世間一般に知られている人物も登場させていることにより、本を読むのにある程度の常識や知識が必要になってくることが分かる。
そして今回、子供向け単語帳で自分が知らない単語は 25 個存在し、大人の方では 41 個存在していた。このことから、やはり大人向けの単語帳に記載されている単語の方が比較的難易度が高いことがわかる。

・反省点

今回、このような言語処理を行うにあたり、反省し再検討すべき点はいくつか発見された。
まず、今回の作品である大人向けの単語帳と子供向け単語帳についてだが、自分はこの結果に対しあまりうまくいったとは考えていない。
その理由としてはあらゆる数値の選び方があまりにも恣意的すぎるからである。
例えば、今回の単語帳は頻度差が 0.2〜2 かつ総合出現頻度が上位 2863 位のものを選んできたが、別に頻度差が 0.5~2 かつ総合出現頻度が上位 3000 位のものでも良かったのである。「大人向け単語帳」や「子供向け単語帳」などというサンプルは今回オンラインでは発見できなかったのでもちろんこれは教師なし学習となり、数値の選び方は難しいものであるが、それでも恣意的ではなく何か根拠を持って単語帳の選び方を決めておくべきであったと思っている。
さらに反省点としては、サンプルとなる文章の量と質が挙げられる。
はじめ私は、子供向けの単語が多いのはきっと童話であるだろうという考えから童話をサンプルとして使用し、それに対し大人をターゲットにした書類として New York Times のような新聞を取り上げようとした。しかしそれでは、童話(フィクション)と新聞(ノンフィクション)という大きなジャンルの相違が生まれてしまうため、ある程度フィクションとして共通し、かつ大人にターゲットを当てたクラシック小説を題材として取り上げたのである。
だが今思えば、クラシック小説と童話も時代の相違というものがあり、理想的な対比とはなっていない。よって私は、もし時間が許すならば、さらに色んな種類のサンプルを集めこの実験をさらに深めていきたい。例えば、若者向けのニュースである CNN Students(現　CNN 10)と New York Times の報道における単語頻度差を比べたり、小学校の国語のテキストと大学で読ませるような本を比べてみたりしたい。
文章の量も当然結果を左右する。今回は互いに 30 冊のみのサンプルとなってしまい、emma,

や gingerbread といった特定の本に登場する特定の名詞が単語帳に記載されてしまったが、サンプルとなる本が何百冊、何千冊ともなればこのような単語の重みは軽くありもっと価値のある単語帳が作成できるはずである。

４．参考文献

https://freekidsbooks.org/　　2019/05/31
本のタイトル
The Little Gingerbread Man
The Journey of the Noble Gnarble
Invisible Alligators
The Brave Monkey Pirate
Wolstencroft The Bear
the zoo animals
The halloween house
Mr. Coyote Meets Mr. Snail
Absulum the Reindeer Elf
Grow Your Own Gargoyle
The Wumpalump
The Loomploy
McFeeglebee's Pond
Who Did Patrick's Homework?
It Could Happen...
the master artist
a tale of friendship
second thoughts
Sliver Pete
wind song
shooflies

http://www.magickeys.com/books/　　2019/05/31
本のタイトル
A FRACTIONATED FAIRY TALE
Why is Nita Upside Down?
Emma's crazy day

Zanele Sees Numbers
Race Driver Callum Makes A Car
Explorer Ella’s Magic Forest
Where Does Metal Come From?
THE SNOW QUEEN
the great cake contest

https://americanliterature.com/books　　2019/05/31
本のタイトル：
one of ours
A country doctor
creative unity
Emma
hard times
laws
life's little ironies
pandora
smoke
the battle of life
The Girl from Farris's
the vampyre
the hill of dreams
the black arrow
a dool's house
the titan
mother
old chrismas
north and south
On liberty
Summer
A Room With View
A Story Of The Day to Come
Lord Jim
A Simple heart
pirates of vanus
sister carrie

at fault
youth
women in love