問68
英語の文書のcos類似度の比較 uberについて書かれた記事2本とlyftについて書かれた記事1本の類似度を比較してみる。

- uber1.txt (https://medium.com/free-code-camp/dark-genius-how-programmers-at-uber-volkswagen-and-zenefits-helped-their-employers-break-the-law-b7a7939c6591)
- uber2.txt (https://medium.com/sandpapersuit/side-hustle-as-a-sign-of-the-apocalypse-e7027a889fc2)
- lyft1.txt (https://medium.com/@johnzimmer/all-lyft-rides-are-now-carbon-neutral-55693af04f36)

文章の量を一致させるため一部削っている。

In [167]:

```
pip install nltk
```

```
Requirement already satisfied: nltk in /anaconda3/lib/python3.7/site-packages (3.4.
1)
Requirement already satisfied: six in /anaconda3/lib/python3.7/site-packages (from n
ltk) (1.12.0)
Note: you may need to restart the kernel to use updated packages.
```

In [188]:

```
import numpy as np
import nltk
from functools import reduce
import collections
nltk.download('punkt')

def word_list(file_name):
    word_list1 = []
    with open(file_name) as f:
        for s_line in f:
            word_list1.append(nltk.word_tokenize(s_line))
    word_list1 = reduce(lambda a, b: a + b, word_list1)
    return np.array(word_list1)

def count_f(word_list, all_word_list):
    count_list = []
    for i in range(all_word_list.size):
        word = all_word_list[i]
        count = np.count_nonzero(word_list == word)
#         t1 = (word, count)
        t1 = count
        count_list.append(t1)
    return np.array(count_list)

def cos(f, g):
    return np.dot(f, g)/(np.linalg.norm(f)*np.linalg.norm(g))
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/taishieguchi/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [189]:

```
uber1_list = word_list('uber1.txt')
uber2_list = word_list('uber2.txt')
lyft1_list = word_list('lyft1.txt')

all_word_list = np.unique(np.hstack((uber1_list, uber2_list, lyft1_list)))

uber1_vec = np.array(count_f(uber1_list, all_word_list))
uber2_vec = np.array(count_f(uber2_list, all_word_list))
lyft1_vec = np.array(count_f(lyft1_list, all_word_list))
```

In [190]:

```
cos(uber1_vec, uber1_vec)
```

Out[190]:

1.0

In [191]:

```
cos(uber1_vec, uber2_vec)
```

Out[191]:

0.6395783804293877

In [192]:

```
cos(uber2_vec, lyft1_vec)
```

Out[192]:

0.6849740392966016

In [193]:

```
cos(uber1_vec, lyft1_vec)
```

Out[193]:

0.7263543672138268

uber1とuber2の値が近くなることを期待したが精度はイマイチであった（むしろlyftとuberの方が近づいてしまっている）。

別の文書(newstimesの記事)で比較してみる。

- realestate1.txt (https://medium.com/free-code-camp/dark-genius-how-programmers-at-uber-volkswagen-and-zenefits-helped-their-employers-break-the-law-b7a7939c6591)
- realestate2.txt (https://medium.com/sandpapersuit/side-hustle-as-a-sign-of-the-apocalypse-e7027a889fc2)
- sports1.txt (https://medium.com/@johnzimmer/all-lyft-rides-are-now-carbon-neutral-55693af04f36)

In [194]:

```
realestate1_list = word_list('realestate1.txt')
realestate2_list = word_list('realestate2.txt')
sports1_list = word_list('sports1.txt')

all_word_list = np.unique(np.hstack((realestate1_list, realestate2_list, sports1_list)))

realestate1_vec = count_f(realestate1_list, all_word_list)
realestate2_vec = count_f(realestate2_list, all_word_list)
sports1_vec = count_f(sports1_list, all_word_list)
```

In [195]:

```
cos(realestate1_vec, realestate1_vec)
```

Out[195]:

1.0

In [196]:

```
cos(realestate1_vec, realestate2_vec)
```

Out[196]:

0.8383622904249884

In [197]:

```
cos(realestate1_vec, sports1_vec)
```

Out[197]:

0.7632852370301972

In [198]:

```
cos(realestate2_vec, sports1_vec)
```

Out[198]:

0.7255375589022625

不動産の記事とスポーツの記事だが、きちんとその類似度を分類できた。

残りの課題については締め切りに間に合わなかったため以下のリンクに貼った。
https://github.com/shierote/NLP_practice (https://github.com/shierote/NLP_practice)

ここでcommon wordを取り除いて考えてみる。
common wordのリストは以下のサイトに載っているものを利用した。
https://www.ef.com/wwen/english-resources/english-vocabulary/top-1000-words/
(https://www.ef.com/wwen/english-resources/english-vocabulary/top-1000-words/)

またカンマや括弧などの記号、文書の類似度に関係のなさそうな数字も目立ったためこれらも取り除いてみる。

In [25]:

```
import numpy as np
import nltk
from functools import reduce
import collections
nltk.download('punkt')

def word_list(file_name):
    word_list1 = []
    with open(file_name) as f:
        for s_line in f:
            word_list1.append(nltk.word_tokenize(s_line))
    word_list1 = reduce(lambda a, b: a + b, word_list1)
    return np.array(word_list1)

def count_f(word_list, all_word_list):
    count_list = []
    for i in range(all_word_list.size):
        word = all_word_list[i]
        count = np.count_nonzero(word_list == word)
#         t1 = (word, count)
        t1 = count
        count_list.append(t1)
    return np.array(count_list)

def cos(f, g):
    return np.dot(f, g)/(np.linalg.norm(f)*np.linalg.norm(g))

def abstract_not_common_word(target_list):
    common_words_list = word_list('common_words.txt')
    res_list = []
    for i in target_list:
        res_list.append(not(i in common_words_list) and i.isalpha())

    return target_list[res_list]
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/taishieguchi/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [23]:

```
realestate1_list = word_list('realestate1.txt')
abstract_not_common_word(realestate1_list)
```

```
KeyboardInterrupt
```

In [35]:

```
realestate1_list = word_list('realestate1.txt')
realestate2_list = word_list('realestate2.txt')
sports1_list = word_list('sports1.txt')

all_word_list = np.unique(np.hstack((realestate1_list, realestate2_list, sports1_list)))

realestate1_vec = count_f(realestate1_list, all_word_list)
realestate2_vec = count_f(realestate2_list, all_word_list)
sports1_vec = count_f(sports1_list, all_word_list)
```

In [36]:

```
print('取り除かなかった場合')
print("cos(realestate1_vec, realestate1_vec)", end=": ")
print(cos(realestate1_vec, realestate1_vec))

print("cos(realestate1_vec, realestate2_vec)", end=": ")
print(cos(realestate1_vec, realestate2_vec))

print("cos(realestate2_vec, sports1_vec)", end=": ")
print(cos(realestate2_vec, sports1_vec))

print("cos(realestate1_vec, sports1_vec)", end=": ")
print(cos(realestate1_vec, sports1_vec))
```

```
取り除かなかった場合
cos(realestate1_vec, realestate1_vec): 1.0
cos(realestate1_vec, realestate2_vec): 0.8383622904249884
cos(realestate2_vec, sports1_vec): 0.7255375589022625
cos(realestate1_vec, sports1_vec): 0.7632852370301972
```

In [64]:

```
realestate1_list = abstract_not_common_word(word_list('realestate1.txt'))
realestate2_list = abstract_not_common_word(word_list('realestate2.txt'))
sports1_list = abstract_not_common_word(word_list('sports1.txt'))

all_word_list = np.unique(np.hstack((realestate1_list, realestate2_list, sports1_list)))

realestate1_vec = count_f(realestate1_list, all_word_list)
realestate2_vec = count_f(realestate2_list, all_word_list)
sports1_vec = count_f(sports1_list, all_word_list)
print('取り除いた場合')
print("cos(realestate1_vec, realestate1_vec)", end=": ")
print(cos(realestate1_vec, realestate1_vec))

print("cos(realestate1_vec, realestate2_vec)", end=": ")
print(cos(realestate1_vec, realestate2_vec))

print("cos(realestate2_vec, sports1_vec)", end=": ")
print(cos(realestate2_vec, sports1_vec))

print("cos(realestate1_vec, sports1_vec)", end=": ")
print(cos(realestate1_vec, sports1_vec))
```

```
取り除いた場合
cos(realestate1_vec, realestate1_vec): 1.0
cos(realestate1_vec, realestate2_vec): 0.4483906358282693
cos(realestate2_vec, sports1_vec): 0.11821621346237848
cos(realestate1_vec, sports1_vec): 0.1559997520720196
```

common wordを取り除くとその差がより明確になった。
uberとlyftでも検証してみる。

In [40]:

```
uber1_list = word_list('uber1.txt')
uber2_list = word_list('uber2.txt')
lyft1_list = word_list('lyft1.txt')

all_word_list = np.unique(np.hstack((uber1_list, uber2_list, lyft1_list)))

uber1_vec = np.array(count_f(uber1_list, all_word_list))
uber2_vec = np.array(count_f(uber2_list, all_word_list))
lyft1_vec = np.array(count_f(lyft1_list, all_word_list))
print('取り除かなかった場合')
print("cos(uber1_vec, uber1_vec)", end=": ")
print(cos(uber1_vec, uber1_vec))

print("cos(uber1_vec, uber2_vec)", end=": ")
print(cos(uber1_vec, uber2_vec))

print("cos(uber2_vec, lyft1_vec)", end=": ")
print(cos(uber2_vec, lyft1_vec))

print("cos(uber1_vec, lyft1_vec)", end=": ")
print(cos(uber1_vec, lyft1_vec))
```

取り除かなかった場合
cos(uber1_vec, uber1_vec): 1.0
cos(uber1_vec, uber2_vec): 0.6395783804293877
cos(uber2_vec, lyft1_vec): 0.6849740392966016
cos(uber1_vec, lyft1_vec): 0.7263543672138268

In [70]:

```
uber1_list = abstract_not_common_word(word_list('uber1.txt'))
uber2_list = abstract_not_common_word(word_list('uber2.txt'))
lyft1_list = abstract_not_common_word(word_list('lyft1.txt'))

all_word_list = np.unique(np.hstack((uber1_list, uber2_list, lyft1_list)))

uber1_vec = np.array(count_f(uber1_list, all_word_list))
uber2_vec = np.array(count_f(uber2_list, all_word_list))
lyft1_vec = np.array(count_f(lyft1_list, all_word_list))
print('取り除いた場合')
print("cos(uber1_vec, uber1_vec)", end=": ")
print(cos(uber1_vec, uber1_vec))

print("cos(uber1_vec, uber2_vec)", end=": ")
print(cos(uber1_vec, uber2_vec))

print("cos(uber2_vec, lyft1_vec)", end=": ")
print(cos(uber2_vec, lyft1_vec))

print("cos(uber1_vec, lyft1_vec)", end=": ")
print(cos(uber1_vec, lyft1_vec))
```

取り除いた場合
cos(uber1_vec, uber1_vec): 0.9999999999999999
cos(uber1_vec, uber2_vec): 0.29304356514021723
cos(uber2_vec, lyft1_vec): 0.32191127046385176
cos(uber1_vec, lyft1_vec): 0.24494958586944585

idfを使ってみる

In [54]:

```
import numpy as np
import nltk
from functools import reduce
import collections
nltk.download('punkt')

def word_list(file_name):
    word_list1 = []
    with open(file_name) as f:
        for s_line in f:
            word_list1.append(nltk.word_tokenize(s_line))
    word_list1 = reduce(lambda a, b: a + b, word_list1)
    return np.array(word_list1)

def count_f(word_list, all_word_list):
    count_list = []
    for i in range(all_word_list.size):
        word = all_word_list[i]
        count = np.count_nonzero(word_list == word)
#         t1 = (word, count)
        t1 = count
        count_list.append(t1)
    return np.array(count_list)

def cos(f, g):
    return np.dot(f, g)/(np.linalg.norm(f)*np.linalg.norm(g))

def abstract_not_common_word(target_list):
    common_words_list = word_list('common_words.txt')
    res_list = []
    for i in target_list:
        res_list.append(not(i in common_words_list) and i.isalpha())

    return target_list[res_list]

def idf(list1, list2, list3):
    for i in range(list1.size):
        c = 0.0
        if list1[i] > 0:
            c += 1.0
        if list2[i] > 0:
            c += 1.0
        if list3[i] > 0:
            c += 1.0
        list1[i] = list1[i]*np.log((c+1)/(3+1))
    return list1
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/taishieguchi/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

uber2とlyft1が近づいてしまった。

In [ ]:

```
```

In [74]:

```
uber1_list = abstract_not_common_word(word_list('uber1.txt'))
uber2_list = abstract_not_common_word(word_list('uber2.txt'))
lyft1_list = abstract_not_common_word(word_list('lyft1.txt'))

all_word_list = np.unique(np.hstack((uber1_list, uber2_list, lyft1_list)))

uber1_vec = np.array(count_f(uber1_list, all_word_list))
uber2_vec = np.array(count_f(uber2_list, all_word_list))
lyft1_vec = np.array(count_f(lyft1_list, all_word_list))
print('idfをつけなかった場合')
print("cos(uber1_vec, uber1_vec)", end=": ")
print(cos(uber1_vec, uber1_vec))

print("cos(uber1_vec, uber2_vec)", end=": ")
print(cos(uber1_vec, uber2_vec))

print("cos(uber2_vec, lyft1_vec)", end=": ")
print(cos(uber2_vec, lyft1_vec))

print("cos(uber1_vec, lyft1_vec)", end=": ")
print(cos(uber1_vec, lyft1_vec))

uber1_list = abstract_not_common_word(word_list('uber1.txt'))
uber2_list = abstract_not_common_word(word_list('uber2.txt'))
lyft1_list = abstract_not_common_word(word_list('lyft1.txt'))

all_word_list = np.unique(np.hstack((uber1_list, uber2_list, lyft1_list)))

uber1_vec = np.array(count_f(uber1_list, all_word_list))
uber2_vec = np.array(count_f(uber2_list, all_word_list))
lyft1_vec = np.array(count_f(lyft1_list, all_word_list))

uber1_idf_vec = idf(uber1_vec, uber2_vec, lyft1_vec)
uber2_idf_vec = idf(uber2_vec, uber1_vec, lyft1_vec)
lyft1_idf_vec = idf(lyft1_vec, uber1_vec, uber2_vec)
print('')
print('idfをつけた場合')
print("cos(uber1_vec, uber1_vec)", end=": ")
print(cos(uber1_idf_vec, uber1_idf_vec))

print("cos(uber1_vec, uber2_vec)", end=": ")
print(cos(uber1_idf_vec, uber2_idf_vec))

print("cos(uber2_vec, lyft1_vec)", end=": ")
print(cos(uber2_idf_vec, lyft1_idf_vec))

print("cos(uber1_vec, lyft1_vec)", end=": ")
print(cos(uber1_idf_vec, lyft1_idf_vec))
```

idfをつけなかった場合
cos(uber1_vec, uber1_vec): 0.9999999999999999
cos(uber1_vec, uber2_vec): 0.29304356514021723
cos(uber2_vec, lyft1_vec): 0.32191127046385176
cos(uber1_vec, lyft1_vec): 0.24494958586944585

idfをつけた場合
cos(uber1_vec, uber1_vec): 1.0
cos(uber1_vec, uber2_vec): 0.05913123959890826
cos(uber2_vec, lyft1_vec): 0.11809437324333252
cos(uber1_vec, lyft1_vec): 0.008534859274986486

In [75]:

```
realestate1_list = abstract_not_common_word(word_list('realestate1.txt'))
realestate2_list = abstract_not_common_word(word_list('realestate2.txt'))
sports1_list = abstract_not_common_word(word_list('sports1.txt'))

all_word_list = np.unique(np.hstack((realestate1_list, realestate2_list, sports1_list)))

realestate1_vec = count_f(realestate1_list, all_word_list)
realestate2_vec = count_f(realestate2_list, all_word_list)
sports1_vec = count_f(sports1_list, all_word_list)
print('idfをつけなかった場合')
print("cos(realestate1_vec, realestate1_vec)", end=": ")
print(cos(realestate1_vec, realestate1_vec))

print("cos(realestate1_vec, realestate2_vec)", end=": ")
print(cos(realestate1_vec, realestate2_vec))

print("cos(realestate2_vec, sports1_vec)", end=": ")
print(cos(realestate2_vec, sports1_vec))

print("cos(realestate1_vec, sports1_vec)", end=": ")
print(cos(realestate1_vec, sports1_vec))
realestate1_list = abstract_not_common_word(word_list('realestate1.txt'))
realestate2_list = abstract_not_common_word(word_list('realestate2.txt'))
sports1_list = abstract_not_common_word(word_list('sports1.txt'))

all_word_list = np.unique(np.hstack((realestate1_list, realestate2_list, sports1_list)))

realestate1_vec_tmp = count_f(realestate1_list, all_word_list)
realestate2_vec_tmp = count_f(realestate2_list, all_word_list)
sports1_vec_tmp = count_f(sports1_list, all_word_list)

realestate1_vec = idf(realestate1_vec_tmp, realestate2_vec_tmp, sports1_vec_tmp)
realestate2_vec = idf(realestate2_vec_tmp, realestate1_vec_tmp, sports1_vec_tmp)
sports1_vec = idf(sports1_vec_tmp, realestate1_vec_tmp, realestate2_vec_tmp)
print('')
print('idfをつけた場合')
print("cos(realestate1_vec, realestate1_vec)", end=": ")
print(cos(realestate1_vec, realestate1_vec))

print("cos(realestate1_vec, realestate2_vec)", end=": ")
print(cos(realestate1_vec, realestate2_vec))

print("cos(realestate2_vec, sports1_vec)", end=": ")
print(cos(realestate2_vec, sports1_vec))

print("cos(realestate1_vec, sports1_vec)", end=": ")
print(cos(realestate1_vec, sports1_vec))
```

```
idfをつけなかった場合
cos(realestate1_vec, realestate1_vec): 1.0
cos(realestate1_vec, realestate2_vec): 0.4483906358282693
cos(realestate2_vec, sports1_vec): 0.11821621346237848
cos(realestate1_vec, sports1_vec): 0.1559997520720196

idfをつけた場合
cos(realestate1_vec, realestate1_vec): 1.0
cos(realestate1_vec, realestate2_vec): 0.2152572818734065
cos(realestate2_vec, sports1_vec): 0.036037498507822355
cos(realestate1_vec, sports1_vec): 0.04986857554914075
```

よりrealestated同士が近づいた。

問69
LSIを行ってみる。

In [78]:

```
import numpy as np
import nltk
import re
import collections
nltk.download('punkt')

def sentece_list(file_name):
    sentence_list = []
    with open(file_name) as f:
        for s_line in f:
            sentence_list.append(s_line)
    return sentence_list

def make_x(file_name):
    sentence_list = sentece_list(file_name)
    all_sentence_list = []
    all_word_list = []
    for sentence in sentence_list:
        not_symbol_list = abstract_not_symbol(sentence_divide(sentence))
        if not_symbol_list.size != 0:
            all_word_list.append(not_symbol_list)
            all_sentence_list.append(not_symbol_list)
    all_sentence_list = np.hstack(all_sentence_list)
    x_list = []
    for sentence in  all_word_list:
        x_list.append(count_f(sentence, all_sentence_list))
    return np.array(x_list).T

def abstract_not_symbol(target_list):
    res_list = []
    for i in target_list:
        res_list.append(i.isalpha())
    return target_list[res_list]

def sentence_divide(sentence):
    return np.array(nltk.word_tokenize(sentence))

def count_f(word_list, all_word_list):
    count_list = []
    for i in range(all_word_list.size):
        word = all_word_list[i]
        count = np.count_nonzero(word_list == word)
        count_list.append(count)
    return np.array(count_list)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/taishieguchi/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

次のセンテンスをSVD分解してみる。

In [84]:

```
d1 = 'I like computer.'
d2 = 'I like machine.'
d3 = 'I like apple.'
sentence_divide(d1)
sentence_divide(d2)
sentence_divide(d3)

all_word_list = np.unique(np.hstack((sentence_divide(d1) , sentence_divide(d2), sentence_divide(d
3))))
l1 = count_f(sentence_divide(d1), all_word_list)
l2 = count_f(sentence_divide(d2), all_word_list)
l3 = count_f(sentence_divide(d3), all_word_list)
all_list = [l1, l2, l3]
x = np.array(all_list).T

print(x)
U, s, V = np.linalg.svd(x, full_matrices=False)
print('U', end=': ')
print(U)
print('s', end=': ')
print(s)
print('V', end=': ')
print(V)
print(np.dot(np.dot(U, np.diag(s)), V))
print(np.dot(U, U.T))
print(np.dot(V, V.T))
```

```
[[1 1 1]
 [1 1 1]
 [0 0 1]
 [1 0 0]
 [1 1 1]
 [0 1 0]]
U: [[-5.47722558e-01 -9.28133183e-18 -1.17457637e-17]
 [-5.47722558e-01 -9.28133183e-18 -1.17457637e-17]
 [-1.82574186e-01 -7.07106781e-01  4.08248290e-01]
 [-1.82574186e-01  5.22860278e-17 -8.16496581e-01]
 [-5.47722558e-01 -2.84412419e-17  7.08520558e-17]
 [-1.82574186e-01  7.07106781e-01  4.08248290e-01]]
s: [3.16227766 1.         1.        ]
V: [[-0.57735027 -0.57735027 -0.57735027]
 [-0.         0.70710678 -0.70710678]
 [-0.81649658  0.40824829  0.40824829]]
[[ 1.00000000e+00  1.00000000e+00  1.00000000e+00]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00]
 [-3.46795453e-17  7.28509239e-17  1.00000000e+00]
 [ 1.00000000e+00  1.01294450e-16 -9.72785220e-18]
 [ 1.00000000e+00  1.00000000e+00  1.00000000e+00]
 [-5.50525174e-17  1.00000000e+00  2.75262587e-17]]
[[ 0.3  0.3  0.1  0.1  0.3  0.1]
 [ 0.3  0.3  0.1  0.1  0.3  0.1]
 [ 0.1  0.1  0.7 -0.3  0.1 -0.3]
 [ 0.1  0.1 -0.3  0.7  0.1 -0.3]
 [ 0.3  0.3  0.1  0.1  0.3  0.1]
 [ 0.1  0.1 -0.3 -0.3  0.1  0.7]]
[[ 1.00000000e+00  5.89014774e-17  6.62768797e-17]
 [ 5.89014774e-17  1.00000000e+00 -3.70692371e-17]
 [ 6.62768797e-17 -3.70692371e-17  1.00000000e+00]]
```

以下の論文をSVDしてみる。 https://queue.acm.org/detail.cfm?id=3321612 (https://queue.acm.org/detail.cfm?id=3321612)

In [81]:

```
x = make_x('cs_article.txt')
print(x)
U, s, V = np.linalg.svd(x, full_matrices=False)
print(s)
```

```
[[1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 [1 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 1]
 [0 1 0 ... 0 1 1]
 [0 1 0 ... 0 1 1]]
[6.47939674e+02 2.07666070e+02 1.30352921e+02 1.06857142e+02
 1.02528556e+02 7.83366701e+01 7.55658842e+01 6.72394660e+01
 5.86674306e+01 5.66937976e+01 5.22863044e+01 5.00232963e+01
 4.71757941e+01 4.51682048e+01 3.98331196e+01 3.89736709e+01
 3.64840960e+01 3.51670497e+01 3.29756341e+01 3.25784043e+01
 3.03715948e+01 2.98550849e+01 2.90008952e+01 2.79982640e+01
 2.71358128e+01 2.64232907e+01 2.58010643e+01 2.49118691e+01
 2.45103435e+01 2.29111878e+01 2.23604557e+01 2.16685174e+01
 2.11064110e+01 2.05447637e+01 1.91634958e+01 1.87057488e+01
 1.82491282e+01 1.77822217e+01 1.70118288e+01 1.67673398e+01
 1.64170213e+01 1.56815648e+01 1.53665556e+01 1.43682886e+01
 1.43224346e+01 1.39919834e+01 1.37865953e+01 1.32528006e+01
 1.30822670e+01 1.22250656e+01 1.20407769e+01 1.19585102e+01
 1.14596965e+01 1.14409308e+01 1.06636034e+01 1.02592325e+01
 9.91478456e+00 9.85118211e+00 9.70690834e+00 9.30555175e+00
 9.19287107e+00 8.65940716e+00 8.43741992e+00 8.19365995e+00
 7.69205443e+00 7.58361434e+00 7.18100204e+00 7.05900241e+00
 6.92594887e+00 6.34661423e+00 6.30111804e+00 6.19890266e+00
 6.06190867e+00 5.62712677e+00 5.48806509e+00 4.83580054e+00
 4.48147509e+00 4.02088519e+00 3.59980347e+00 3.45225037e+00
 3.09181564e+00 2.94082927e+00 2.80410257e+00 2.44702306e+00
 2.16108453e+00 2.11120942e+00 1.93631056e+00 1.53733283e+00
 1.41421356e+00 1.00000000e+00 5.19077655e-14 9.78715931e-15]
```