60.単語ベクトルの読み込みと表示

解答：

```
from gensim.models import KeyedVectors

model = KeyedVectors.load_word2vec_format('./GoogleNews-vectors-negative300.bin.gz', binary=True)

model.word_vec('United_States')
```

実行結果：

```
array([-3.61328125e-02, -4.83398438e-02,  2.35351562e-01,  1.74804688e-01,
       -1.46484375e-01, -7.42187500e-02, -1.01562500e-01, -7.71484375e-02,
        1.09375000e-01, -5.71289062e-02, -1.48437500e-01, -6.00585938e-02,
        1.74804688e-01, -7.71484375e-02,  2.58789062e-02, -7.66601562e-02,
       -3.80859375e-02,  1.35742188e-01,  3.75976562e-02, -4.19921875e-02,
       -3.56445312e-02,  5.34667969e-02,  3.68118286e-04, -1.66992188e-01,
       -1.17187500e-01,  1.41601562e-01, -1.69921875e-01, -6.49414062e-02,
       -1.66992188e-01,  1.00585938e-01,  1.15722656e-01, -2.18750000e-01,
       -9.86328125e-02, -2.56347656e-02,  1.23046875e-01, -3.54003906e-02,
       -1.58203125e-01, -1.60156250e-01,  2.94189453e-02,  8.15429688e-02,
        6.88476562e-02,  1.87500000e-01,  6.49414062e-02,  1.15234375e-01,
       -2.27050781e-02,  3.32031250e-01, -3.27148438e-02,  1.77734375e-01,
       -2.08007812e-01,  4.54101562e-02, -1.23901367e-02,  1.19628906e-01,
        7.44628906e-03, -9.03320312e-03,  1.14257812e-01,  1.69921875e-01,
       -2.38281250e-01, -2.79541016e-02, -1.21093750e-01,  2.47802734e-02,
        7.71484375e-02, -2.81982422e-02, -4.71191406e-02,  1.78222656e-02,
       -1.23046875e-01, -5.32226562e-02,  2.68554688e-02, -3.11279297e-02,
       -5.59082031e-02, -5.00488281e-02, -3.73535156e-02,  1.25976562e-01,
```

（結果の一部）

まとめ：

上に示した方法はモデル中のすべてのデータを読み込むことであり、モデルに含まれる単語の数は非常に膨大であるため、この方法は非常に時間とメモリ空間を費やし、パラメータ limit を設定することで読み込む単語ベクトルを制限することができ、これにより所定数の単語ベクトルを順番に最初から読み込むことができ、一般的には、この単語モデルでは、頻繁に使用される単語ほど前になる。

例えば：`model = KeyedVectors.load_word2vec_format('./GoogleNews-vectors-negative300.bin.gz', binary=Ture, limit=100)`

61.単語の類似度

解答：

```
model.similarity('United_States', 'U.S.')
```

実行結果：

```
0.73107743
```

6 2.類似度の高い単語 10 件

解答：

```
model.most_similar('United_States', topn=10)
```

実行結果：

```
[('Unites_States', 0.7877248525619507),
 ('Untied_States', 0.7541370391845703),
 ('United_Sates', 0.74007248878479),
 ('U.S.', 0.7310774326324463),
 ('theUnited_States', 0.6404393911361694),
 ('America', 0.6178410053253174),
 ('UnitedStates', 0.6167312264442444),
 ('Europe', 0.6132988929748535),
 ('countries', 0.6044804453849792),
 ('Canada', 0.6019070148468018)]
```

まとめ：

以上は類似度の高い 10 単語を探して、topn の値を変更することで任意の数を探すことができます

6 3.加法構成性によるアナロジー

解答：

```
vec = model['Spain'] - model['madrid'] + model['Athens']
model.most_similar(positive=['Spain', 'Athens'], negative=['Madrid'], topn=10)
```

実行結果：

```
[('Greece', 0.6898481249809265),
 ('Aristeidis_Grigoriadis', 0.5606848001480103),
 ('Ioannis_Drymonakos', 0.5552908778190613),
 ('Greeks', 0.545068621635437),
 ('Ioannis_Christou', 0.5400862693786621),
 ('Hrysopiyi_Devetzi', 0.5248444676399231),
 ('Heraklio', 0.5207759737968445),
 ('Athens_Greece', 0.516880989074707),
 ('Lithuania', 0.5166866183280945),
 ('Iraklion', 0.5146791934967041)]
```

6 4.アナロジーデータでの実験

解答：

```
!wget http://download.tensorflow.org/data/questions-words.txt
with open('./questions-words.txt', 'r') as f1, open('./questions-words-
add.txt', 'w') as f2:
  for line in f1:  # f1 から 1 行ずつ読込み、求めた単語と類似度を追加して f2 に書込む
    line = line.split()
    if line[0] == ':':
      category = line[1]
    else:
```

```
      word, cos = model.most_similar(positive=[line[1], line[2]], negative=[line[0]
], topn=1)[0]
      f2.write(' '.join([category] + line + [word, str(cos) + '\n']))
```

実行結果：

```
capital-common-countries Athens Greece Baghdad Iraq Iraqi 0.6351870894432068
capital-common-countries Athens Greece Bangkok Thailand Thailand 0.7137669324874878
capital-common-countries Athens Greece Beijing China China 0.7235777974128723
capital-common-countries Athens Greece Berlin Germany Germany 0.6734622120857239
capital-common-countries Athens Greece Bern Switzerland Switzerland 0.4919748306274414
capital-common-countries Athens Greece Cairo Egypt Egypt 0.7527809739112854
capital-common-countries Athens Greece Canberra Australia Australia 0.583732545375824
capital-common-countries Athens Greece Hanoi Vietnam Viet_Nam 0.6276341676712036
capital-common-countries Athens Greece Havana Cuba Cuba 0.6460992097854614
capital-common-countries Athens Greece Helsinki Finland Finland 0.6899983882904053
```

（結果の一部）

まとめ：

ファイルには 2 つのケースがあります。1 つ目は「：capital-common-countries」タイプの文で、この文はカテゴリをマークしています，「capital-common-countries」を取り出して category に保存する必要があります。2 つ目は「Athens Greece Baghdad Iraq」タイプの文です，vec[1]+vec[2]-vec[0]を計算し、計算結果を word、cos に保存する必要があります。word は類似度の高い単語である，cos は類似度です。

6 5.アナロジータスクで正解率

解答：

```
with open('./questions-words-add.txt', 'r') as f:
  sem_cnt = 0
  sem_cor = 0
  syn_cnt = 0
  syn_cor = 0
  for line in f:
    line = line.split()
    if not line[0].startswith('gram'):#如果不是以 gram 开头则为句意分析
      sem_cnt += 1
      if line[4] == line[5]:
        sem_cor += 1
    else:
      syn_cnt += 1
      if line[4] == line[5]:
        syn_cor += 1

print(f'意味的アナロジー正解率: {sem_cor/sem_cnt:.3f}')
print(f'文法的アナロジー正解率: {syn_cor/syn_cnt:.3f}')
```

実行結果：

```
意味的アナロジー正解率: 0.731
文法的アナロジー正解率: 0.740
```

まとめ：

まず gram で始まるかどうかを判断し、gram でなければ文法解析の結果です。

capital-common-countries Athens Greece Baghdad Iraq Iraqi 0.6351870894432068

line[4]と line[5]が同じかどうかを比較する。

6 6.WordSimilarity-353 での評価

解答：

```
ws353 = []
with open('./combined.csv', 'r') as f:
  next(f)
  for line in f:  # 1 行ずつ読込み、単語ベクトルと類似度を計算
    line = [s.strip() for s in line.split(',')]
    line.append(model.similarity(line[0], line[1]))
    ws353.append(line)

# 確認
for i in range(5):
  print(ws353[i])

import numpy as np
from scipy.stats import spearmanr

# スピアマン相関係数の計算
human = np.array(ws353).T[2]
w2v = np.array(ws353).T[3]
correlation, pvalue = spearmanr(human, w2v)

print(f'スピアマン相関係数: {correlation:.3f}')
```

実行結果：

```
['love', 'sex', '6.77', 0.2639377]
['tiger', 'cat', '7.35', 0.5172962]
['tiger', 'tiger', '10.00', 0.99999994]
['book', 'paper', '7.46', 0.3634626]
['computer', 'keyboard', '7.62', 0.39639163]
スピアマン相関係数: 0.685
```

まとめ：

スピルマン（Spearman）相関は、2 つの変数の依存性を測定する非パラメータ指標である。単調方程式を用いて 2 つの統計変数の相関を評価した。データに重複値がなく、2 つの変数が完全に単調に相関している場合、スピルマン相関係数は＋1 または－1 になります。

まず 1 行ずつ読込み、単語ベクトルと類似度を計算、3 列目と 4 列目の数値を取り出して転置する、転置後の 2 つの配列の計算スピアマン相関係数。

6 7.k-means クラスリング

解答：

```
# 国名の取得
countries = set()
```

```
with open('./questions-words-add.txt') as f:
  for line in f:
    line = line.split()
    if line[0] in ['capital-common-countries', 'capital-world']:
      countries.add(line[2])
    elif line[0] in ['currency', 'gram6-nationality-adjective']:
      countries.add(line[1])
countries = list(countries)

# 単語ベクトルの取得
countries_vec = [model[country] for country in countries]

from sklearn.cluster import KMeans

# k-means クラスタリング
kmeans = KMeans(n_clusters=5)
kmeans.fit(countries_vec)
for i in range(5):
    cluster = np.where(kmeans.labels_ == i)[0]
    print('cluster', i)
    print(', '.join([countries[k] for k in cluster]))
```

実行結果：

```
cluster 0
Slovakia, Greenland, Jordan, Georgia, Ireland, Denmark, Slovenia, Italy, Hungary,
Bulgaria, Macedonia, Latvia, Poland, Lithuania, Norway, Spain, Greece, Israel,
Iceland, Turkey, Serbia, Finland, Liechtenstein, Sweden, Croatia, Austria,
Montenegro, Switzerland, Cyprus, Europe, Netherlands, Albania, England, Germany,
Romania, Belgium, Portugal, Malta, Estonia, France, USA
cluster 1
Bangladesh, Philippines, Oman, Fiji, Vietnam, Laos, Bhutan, Australia, Japan,
Qatar, Cambodia, Indonesia, Thailand, India, Bahrain, Malaysia, Nepal, Tuvalu,
Korea, Taiwan, China
cluster 2
Suriname, Samoa, Chile, Honduras, Canada, Brazil, Dominica, Uruguay, Peru,
Argentina, Nicaragua, Colombia, Venezuela, Belize, Bahamas, Mexico, Cuba, Ecuador,
Guyana, Jamaica
cluster 3
Niger, Somalia, Gambia, Morocco, Namibia, Mali, Liberia, Kenya, Zambia, Uganda,
Malawi, Iraq, Botswana, Sudan, Eritrea, Lebanon, Rwanda, Algeria, Mauritania,
Angola, Ghana, Mozambique, Guinea, Madagascar, Egypt, Gabon, Libya, Burundi,
Nigeria, Tunisia, Zimbabwe, Senegal
cluster 4
Ukraine, Syria, Afghanistan, Kyrgyzstan, Pakistan, Kazakhstan, Armenia, Azerbaijan,
Moldova, Iran, Belarus, Turkmenistan, Tajikistan, Uzbekistan, Russia
```

まとめ：

K-means アルゴリズムは典型的な距離に基づくクラスタリングアルゴリズムであり、距離を類似性の評価指標として採用し、すなわち 2 つの対象の距離が近いほどその類似度が大きいと考えられる。このアルゴリズ

ムでは、クラスタは距離の近いオブジェクトで構成されていると考えられているため、コンパクトで独立したクラスタを最終目標としています。

まず 1 行ずつ読込み、国名の取得と単語ベクトルの取得、次に k-means クラスタリングを計算します
KMeans(n_clusters=5)：これはいくつのクラスに集まっているかを表すもので、クラスターの数です。
kmeans.fit(countries_vec)k 平均クラスタリングの実行。
最終的に各クラスタを取り出す

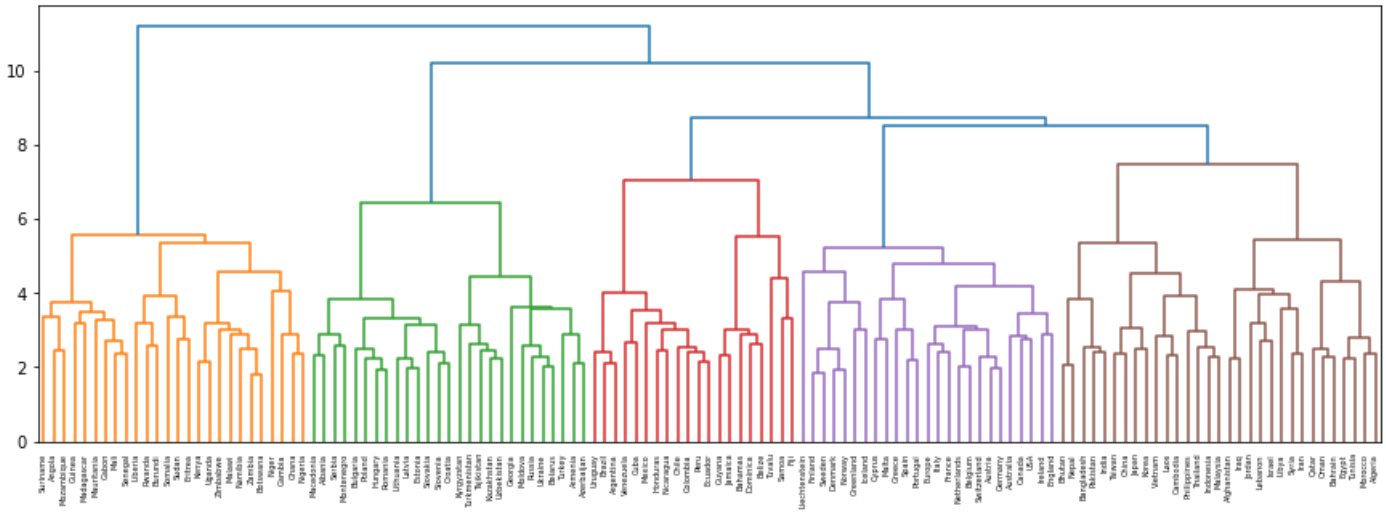
6 8. Ward 法によるクラスリング

解答：

```
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage

plt.figure(figsize=(15, 5))
Z = linkage(countries_vec, method='ward')
dendrogram(Z, labels=countries)
plt.show()
```

実行結果：



まとめ：

```
  Z = linkage(countries_vec, method='ward')
 Parameter method=
'Single': one norm distance
'Complete': infinite norm distance
'Average': average distance
'Centroid': two norm distance
'Ward': Sum of squares of deviations
```

dendrogram 関数階層クラスタリング図の描画

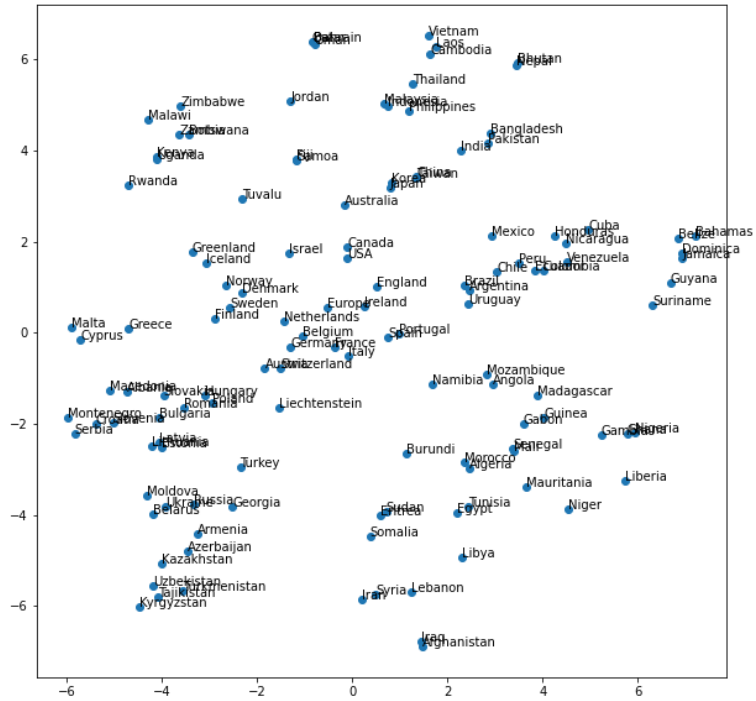
6 9. t-SNE による可視化

解答：

```
import bhtsne

embedded = bhtsne.tsne(np.array(countries_vec).astype(np.float64), dimensions=2, rand_seed=123)
plt.figure(figsize=(10, 10))
plt.scatter(np.array(embedded).T[0], np.array(embedded).T[1])
for (x, y), name in zip(embedded, countries):
    plt.annotate(name, (x, y))
plt.show()
```

実行結果：



まとめ：

　可視化を主な目的とした次元削減の問題は，「高次元空間上の類似度をよく表現する低次元空間の類似度を推定する」問題だと考えられるわけですが，

　t-SNE はこれを確率分布に基づくアプローチで解くものです．

　具体的には，(1)高次元空間上の類似度 と (2)低次元空間上の類似度 をそれぞれ確率分布 pijpij と qijqij で表現して，pijpij と qijqij を最小化するように確率分布のパラメータを最適化します．