# 2023 年 10 月 24 日課題

## 1 課題 1: テキストファイルからのモデルの学習

### 1.1 ソースコード

テキストファイルを読み込み, 単語をリストに入れてモデルの学習を行なった. そのコードをソースコード 1 に示す.

ソースコード 1: 課題 1 のソースコード

```
1  from gensim.models import Word2Vec
2  import re
3
4  f = open('2023_jikkenII_05_textdata.txt')
5  moji = re.sub(r"[^a-zA-Z' '.?!]+", "", f.read())
6  sent = re.split('[.?!]', moji.lower())
7  li = []
8  for i in range(len(sent)):
9    li.append(sent[i].split(' '))
10 print(li)
11 model = word2vec.Word2Vec(sentences=li, vector_size=25, window= 5, min_count=1, epochs=10
12 f.close()
13
14 n = 5
15 print(model.wv.most_similar('cats', topn=n))
16 print()
17 print(model.wv.most_similar('well', topn=n))
18 print()
19 print(model.wv.most_similar('started', topn=n))
20
21 print(model.wv.similarity('alice', 'rabbit'))
```

### 1.2 実行結果および考察

cats に類似した上位 5 件は dinah, whether, made, austlalia, listen となっている. テキストファイルに”Dinah was the cat.”という文があることから, cats と dinah が類似しているのは正しいと考えられる. また well に類似した単語は dry, please, me, soon, practice となり, started に類似した単語は this, you, either, of, bats となった. これらの単語は類似しているとは考えられないため, 今回用いたテキストファイルでの出現頻度やテキストファイル自体の単語量などが問題となっていると考えられる.

alice と rabbit の類似度は 0.17486736 程度になった. ここでモデルのエポック数を 20 に増やしてみたところ, alice と rabbit の類似度は 0.8453393 になった. ここから適切なエポック数を与えることでモデルの学習が進むことがわかった.

# 2 課題 2: 日本語の Word2Vec の調査

## 2.1 ソースコード

調査に用いたソースコードを以下に示す.

ソースコード 2: 課題 2 のソースコード

```
1  import gensim
2
3  # データの（chiVePATHkv:）KeyedVectors
4  model_path = "/content/chive-1.2-mc90_gensim/chive-1.2-mc90.kv"
5
6  # モデルの読み込み
7  m_chiVe = KeyedVectors.load(model_path)
8
9  print('単語数')
10 print(len(m_chiVe))
11 print('分散表現の次元数')
12 print(m_chiVe.vector_size)
13
14 # 類似度上位件を取得10
15 match = m_chiVe.most_similar("ヘリコプター", topn=10)
16 # 見やすい形式で表示
17 print('ヘリコプターの分散表現')
18 print(m_chiVe['ヘリコプター'])
19 print('ヘリコプター類似度上位件:10')
20 for x in match:
21   print(x)
22
23 # 類似度上位件を取得10
24 match = m_chiVe.most_similar("嘯く", topn=10)
25 # 見やすい形式で表示
26 print('嘯くの分散表現')
27 print(m_chiVe['嘯く'])
28 print('嘯く類似度上位件:10')
29 for x in match:
30   print(x)
31
32 # 類似度上位件を取得10
33 match = m_chiVe.most_similar("美しい", topn=10)
```

```
34    # 見やすい形式で表示
35    print('美しいの分散表現')
36    print(m_chiVe['美しい'])
37    print('美しい類似度上位件:10')
38    for x in match:
39      print(x)
```

---

## 2.2 実行結果および考察

chiVe の単語数は 482238 個で, 分散表現の次元数は 300 であった. それぞれの単語についての分散表現は長くなったため省略する. "ヘリコプター"の類似度上位 10 件は, ヘリ, リコプター, コプター, 輸送機, セスナ機, ジェット機, 戦闘機, セスナ, 航空機, 離着陸となった. 最も類似度の高い "ヘリ"は類似度が 0.8820430636405945 となっている. 十分な単語を保有した chiVe ではより類似性の高いものが出力されることがわかった. 同様に"嘯く"の類似度上位 10 件は, 宣う, 言い放つ, 言い募る, せせら笑う, 居直る, ほざく, したり顔, 言い草, 平然, 臆面となった. この中で最も類似度の高い"宣う"の類似度は 0.6980857849121094 となっている. また, "美しい"の類似度上位 10 件は, 綺麗, 艶やか, 映える, 素晴らしい, 美しくも, 端正, 気品, 優美, 魅力的, うっとりとなった. 考えていた通りの結果が得られ, 課題 1 で行ったテキストファイルからの学習とは学習量が異なるため, これだけの精度の違いが生まれているのだと考えた.