自然言語処理：感情分析

・概要

今回は Word2Vec を用いて、与えた文章に対する簡単な感情分析を実装しました。

・簡単な流れ

1. まず、ダンプ化された Wikipedia の日本語データをダウンロードして、コーパスを用意しました。
2. 1.で用意したコーパスに対して、MeCab を用いて分かち書きを行い、Word2Vec のモデル作成を行いました。
3. 与えられた文章に対して、MeCab を用いて形態素解析をし、形態素ごとに感情分析のための処理をしました。

・コマンド

1. Wikipedia のダンプデータダウンロード(XML ファイル形式)

```
r.koike@koiketaitonoMBP wikiextractor % curl https://dumps.wikimedia.org/jawiki/latest/jawiki
-latest-pages-articles.xml.bz2 -o jawiki-latest-pages-articles.xml.bz2
```

2. XML ファイルから記事本文のみを取り出す(wiki.txt に保存)

   この際、WikiExtractor という Github で公開されているフリーソフトウェアを利用しました。

```
r.koike@koiketaitonoMBP wikiextractor % python WikiExtractor.py jawiki-latest-pages-articles.
xml.bz2
```

```
r.koike@koiketaitonoMBP wikiextractor % find text/ | grep wiki | awk '{system("cat "$0" >> wi
ki.txt")}'
```

3. wiki.txt 内のデータを、MeCab を用いて分かち書き(wiki_wakati.txt に保存)

```
r.koike@koiketaitonoMBP wikiextractor % mecab -Owakati wiki.txt -o wiki_wakati.txt
```

・ソースコード

1. Word2Vec モデル作成(gensim という python のフリーソフトウェアを利用)
   “wiki.model”が作成したモデル

```
from gensim.models import word2vec
import logging

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
sentences = word2vec.Text8Corpus('./wiki_wakati.txt')

model = word2vec.Word2Vec(sentences, size=200, min_count=20, window=15)
model.save("./wiki.model")
```

2. 与えられた文章に対して感情分析

```
import MeCab
from gensim.models import word2vec
import numpy as np

mecab = MeCab.Tagger("")
mecab.parse("")
model = word2vec.Word2Vec.load("./wiki.model")

def text_senti(text):
    senti = []
    node = mecab.parseToNode(text)
    while node:
        hinshi = node.feature.split(",")[0]
        if hinshi == '名詞' or hinshi == '動詞' or hinshi == '形容詞':
            res = list(map(model.wv.similarity, [node.surface]*6, ["嬉しい", "怒り", "悲しみ", "興奮", "恐怖", "安心"]))
            senti.append(res)
        node = node.next
    return senti

def main():
    print("文章を入れてください")
    sentence = input()
    senti_map = text_senti(sentence)
    senti_map = np.array(senti_map)
    mean_senti = np.mean(senti_map, axis=0)
    print(f"喜び:{mean_senti[0]}")
    print(f"怒り:{mean_senti[1]}")
    print(f"悲しみ:{mean_senti[2]}")
    print(f"興奮:{mean_senti[3]}")
    print(f"恐怖:{mean_senti[4]}")
    print(f"安心:{mean_senti[5]}")

if __name__ == "__main__":
    main()
```

具体的には、その形態素の品詞が”名詞”、”動詞”、”形容詞”のいずれかの場合、その形態素と感情要素「”嬉しい”、”怒り”、”悲しみ”、”興奮”、”恐怖”、”安心”」との cos 類似度をとり、その値を配列に格納しました。

出力される値は cos 類似度なので-1 から 1 までの値を取ります。-1 の時、単語間の関係性が最も弱く、1 の時、単語間に最も強く関係性があることを示します。

最後に感情要素ごとに cos 類似度の平均値をとり、その文章がどの感情要素と強く結びついているかを値として出力しました。

・結果

“嬉しい”や”楽しい”といった、直接的に感情を表すような単語が含まれない文章入力を行いました。
感情ごとに出力された値は-1 に近いほど、文章との関係性が小さく、1 に近いほど文章との関係性が大きい。

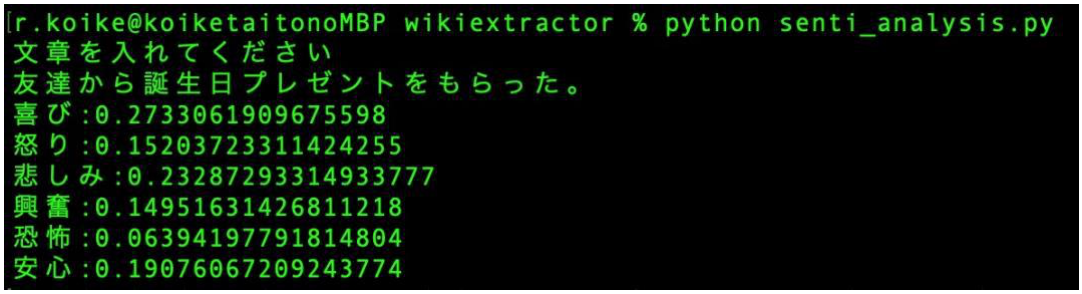
1. 入力文章(ネガティブ感情)：「アメリカでは拳銃による殺人事件が多く発生する。」

   ”喜び”、”安心”といったポジティブな感情の値は比較的小さくなり、一方で”怒り”、”恐怖”といったネガティブな感情の値は比較的大きくなった。“悲しみ”という感情の値は、その他のネガティブな感情ほど値が大きくならなかった。

   殺人事件自体の、”恐怖”といった「直接的な」感情などは読み取れているが、殺人事件後に発生する、人の死に対する”悲しみ”といった「間接的な」感情は読み取れていないことがわかる。

2. 入力文章(ポジティブ感情)：「友達から誕生日プレゼントをもらった。」

   “恐怖”に対する値は比較的小さくなり、”喜び”,”興奮”,”安心”に対する値は比較的大きくなった。
   しかし、”怒り”,”悲しみ”に対する値も比較的大きくなってしまった。



3. 入力文章(ニュートラル感情)：「今日は犬と散歩に行った。
いずれの感情においても同じような値となった。

```
r.koike@koiketaitonoMBP wikiextractor % python senti_analysis.py
文章を入れてください
今日は犬と散歩に行った。
喜び:0.13553451001644135
怒り:0.12080545723438263
悲しみ:0.14948973059654236
興奮:0.13985280692577362
恐怖:0.10596563667058945
安心:0.1354132741689682
```

・考察

今回、感情要素ごとの cos 類似度の値を形態素ごとに平均化したものを出力しましたが、そうすると前述の結果のように感情分析の精度はあまり良くないものとなりました。
この図を見ると、「合格」という単語は正しく感情を表せているのにもかかわらず、「第一志望の大学に合格した。」という文章になると「合格」以外の単語の値との平均をとるので、顕著に感情を表せていないことがわかる。(“喜び”と”怒り”の値が同程度になってしまっている。)

```
[r.koike@koiketaitonoMBP wikiextractor % python senti_analysis.py
文章を入れてください
第一志望の大学に合格した。
喜び:0.04844580590724945
怒り:0.0552915520966053
悲しみ:0.0223575476557016 37
興奮:0.05397895723581314
恐怖:0.018647029995918274
安心:0.08678823709487915
[r.koike@koiketaitonoMBP wikiextractor % python senti_analysis.py
文章を入れてください
合格
喜び:0.10161330550909042
怒り:-0.05333073064684868
悲しみ:-0.038136642426252365
興奮:0.01609606295824051
恐怖:-0.125708341598510 74
安心:0.09359527379274368
```

したがって、どの形態素における cos 類似度がどれほど全体の出力時に影響を与えるかの重み付けを教師あり学習で学習させれば、平均化を用いずに済み、より精度の高い感情分析が行えるのではないかと思いました。

・参考文献

1. 「Word2Vec の使い方」https://qiita.com/kenta1984/items/93b64768494f971edf86
2. 「Wikipedia ダンプデータ」https://dumps.wikimedia.org/jawiki/latest/
3. 「A tool for extracting plain text from Wikipedia dumps」 https://github.com/attardi/wikiextractor