REPORT 2

1. REPORT 2-1

(ア) 推薦アルゴリズム(3):協調フィルタリング(モデルベース)(続き)の回に実施した「演習 2scikit - surprise による SUSHIPreferenceDataSets の推薦：推薦アルゴリズム評価」の実施結果について報告する。

① 使用したデータ

1. http://www.kamishima.net/sushi/

② 手順

1. 使用したデータの sushi3-2016.zip を解凍
2. 以下のプログラムを実行し、scikit-surprise に入れることができるデータ形式に変換

```
def convert(input_file_name):

    output_file_name = input_file_name + '_converted' #  変換後のファイル名
    output = ''

    with open(input_file_name, mode='r') as f:
        lines = f.readlines()

        user_id = 0
        for line in lines:
            words = line.strip().split(' ')
            for item_id, word in enumerate(words):
                score = int(word)
                if score != -1:
                    if score != 0:
                        output += '{0:04d} {1:02d} {2:01d}¥n'.format(user_id, item_id, score)

            user_id += 1

    with open(output_file_name, mode='w') as f:
        f.write(output)

    return output_file_name
```

```
file_name = convert('./sushi3-2016/sushi3b.5000.10.score')
```

3. 以下の SVD, SVDPP, NMF のプログラムを実行した
（ア）SVD

```
from surprise import SVD
from surprise import Dataset, Reader
from surprise.model_selection import cross_validate


# Load the movielens-100k dataset (download it if needed),
reader = Reader(line_format='user item rating', sep=' ')
dataset = Dataset.load_from_file('./sushi3-2016/sushi3b.5000.10.score_converted', reader = reader)
trainset = dataset.build_full_trainset()

# We'll use the famous NMF algorithm.
algo = SVD()
algo.fit(trainset)

# Run 5-fold cross-validation and print results
cross_validate(algo, dataset, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

（イ）SVDPP

```
from surprise import SVDpp
from surprise import Dataset, Reader
from surprise.model_selection import cross_validate


# Load the movielens-100k dataset (download it if needed),
reader = Reader(line_format='user item rating', sep=' ')
dataset = Dataset.load_from_file('./sushi3-2016/sushi3b.5000.10.score_converted', reader = reader)
trainset = dataset.build_full_trainset()

# We'll use the famous NMF algorithm.
algo = SVDpp()
algo.fit(trainset)

# Run 5-fold cross-validation and print results
cross_validate(algo, dataset, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

（ウ）NMF

```
from surprise import NMF
from surprise import Dataset, Reader
from surprise.model_selection import cross_validate


# Load the movielens-100k dataset (download it if needed),
reader = Reader(line_format='user item rating', sep=' ')
dataset = Dataset.load_from_file('./sushi3-2016/sushi3b.5000.10.score_converted', reader = reader)
trainset = dataset.build_full_trainset()

# We'll use the famous NMF algorithm.
algo = NMF()
algo.fit(trainset)

# Run 5-fold cross-validation and print results
cross_validate(algo, dataset, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

③　実行結果

```
Evaluating RMSE, MAE of algorithm NMF on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.0639  1.0512  1.0438  1.0661  1.0408  1.0531  0.0103
MAE (testset)     0.8542  0.8460  0.8397  0.8560  0.8352  0.8462  0.0081
Fit time          2.39    2.54    2.35    2.29    2.31    2.38    0.09
Test time         0.05    0.06    0.05    0.05    0.05    0.05    0.00
> /Users/10kaoru12/.local/share/virtualenvs/4y_university_information_rec
0kaoru12/Documents/WorkSpace/4y_university_information_recommender_system
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9518  0.9493  0.9497  0.9509  0.9594  0.9522  0.0037
MAE (testset)     0.7911  0.7890  0.7862  0.7860  0.7962  0.7897  0.0038
Fit time          2.09    2.03    2.02    2.02    1.96    2.02    0.04
Test time         0.07    0.06    0.06    0.07    0.06    0.07    0.01
> /Users/10kaoru12/.local/share/virtualenvs/4y_university_information_rec
0kaoru12/Documents/WorkSpace/4y_university_information_recommender_system
Evaluating RMSE, MAE of algorithm SVDpp on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9494  0.9493  0.9514  0.9505  0.9438  0.9489  0.0026
MAE (testset)     0.7799  0.7821  0.7843  0.7837  0.7764  0.7813  0.0029
Fit time          6.15    5.97    6.07    6.06    7.04    6.26    0.39
Test time         0.17    0.17    0.17    0.17    0.22    0.18    0.02
```

2. REPORT 2-2

(ア) 演習 3 で実施したプログラムを改造し、複数のテキスト文書を指定してその類似度を比較して結果を表示するプログラムを作成する。

① 使用したデータ

なし

② 作成したプログラム

```
import sys
import MeCab
from gensim.models.word2vec import Word2Vec
import numpy as np

mt = MeCab.Tagger('')
model_path = 'gensim-model/word2vec.gensim.model'
wv = Word2Vec.load(model_path)

# テキストのベクトルを計算
def get_vector(text):
    try:
        sum_vec = np.zeros(50)
        word_count = 0
        node = mt.parseToNode(text)
        while node:
            fields = node.feature.split(",")
            # 名詞、動詞、形容詞に限定
            if fields[0] == '名詞' or fields[0] == '動詞' or fields[0] == '形容詞':
                sum_vec += wv[node.surface]
                word_count += 1
            node = node.next
    except KeyError as error:
        print(error)

    return sum_vec / word_count


# cos 類似度を計算
```

```
def cos_sim(v1, v2):
    return np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))


if __name__ == "__main__":
    args = sys.argv
    v1 = get_vector(open("./text/"+sys.argv[1],"r",encoding="utf_8").read())
    v2 = get_vector(open("./text/"+sys.argv[2],"r",encoding="utf_8").read())

    print(cos_sim(v1, v2))
```

（イ）作成したプログラムを用いて、10 冊の書籍間の類似度を算出し、実施結果について考察する。

① 使用したデータ

1. 三体
2. 三体Ⅱ　黒暗森林（上）
3. 三体Ⅱ　黒暗森林（下）
4. 帝国の娘　上
5. 帝国の娘　下
6. 高校事変
7. 高校事変 II
8. 高校事変 III
9. ソロモンの偽証: 第Ⅰ部 事件 上巻
10. ソロモンの偽証: 第Ⅰ部 事件 下巻

② 使用したプログラム

```
import sys
import MeCab
from gensim.models.word2vec import Word2Vec
import numpy as np

mt = MeCab.Tagger('')
model_path = 'gensim-model/word2vec.gensim.model'
wv = Word2Vec.load(model_path)
output_file = open("./text/output.txt","w")

# テキストのベクトルを計算
def get_vector(text):
    try:
        sum_vec = np.zeros(50)
        word_count = 0
        node = mt.parseToNode(text)
        while node:
            fields = node.feature.split(",")
            # 名詞、動詞、形容詞に限定
            if fields[0] == '名詞' or fields[0] == '動詞' or fields[0] == '形容詞':
                sum_vec += wv[node.surface]
```

```
                    word_count += 1
                node = node.next
        except KeyError as error:
            print(error)

        return sum_vec / word_count


# cos 類似度を計算
def cos_sim(v1, v2):
        return np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))


if __name__ == "__main__":
        print("文書番号¥t",end='',file=output_file)
        for n in range(1,11):
            print("book", n, "¥t", end='',file=output_file)
        print("¥n",file=output_file)
        for i in range(1, 11):
            comparator = get_vector(open("./text/book"+str(i)+".txt","r",encoding="utf_8").read())
            print("book", i, "¥t", end='',file=output_file)
            for j in range(1, 11):
                target = get_vector(open("./text/book"+str(j)+".txt","r",encoding="utf_8").read())
                print("%8.3f"%cos_sim(comparator, target),end="",file=output_file)
            print("¥n",file=output_file)
```

③　実行結果

文書番号	book 1	book 2	book 3	book 4	book 5	book 6	book 7	book 8	book 9	book 10
book 1	1.000	0.765	0.777	0.482	0.318	0.107	0.532	0.462	0.742	0.713
book 2	0.765	1.000	0.626	0.530	0.388	0.273	0.355	0.322	0.558	0.524
book 3	0.777	0.626	1.000	0.544	0.255	0.172	0.447	0.433	0.587	0.588
book 4	0.482	0.530	0.544	1.000	0.478	0.319	0.310	0.384	0.536	0.585
book 5	0.318	0.388	0.255	0.478	1.000	0.362	0.481	0.301	0.448	0.514
book 6	0.107	0.273	0.172	0.319	0.362	1.000	0.514	0.384	0.118	0.279
book 7	0.532	0.355	0.447	0.310	0.481	0.514	1.000	0.493	0.619	0.714
book 8	0.462	0.322	0.433	0.384	0.301	0.384	0.493	1.000	0.485	0.509
book 9	0.742	0.558	0.587	0.536	0.448	0.118	0.619	0.485	1.000	0.938
book 10	0.713	0.524	0.588	0.585	0.514	0.279	0.714	0.509	0.938	1.000

④　考察

似たような結果になるべき同じ作品グループとしては以下である。

・book1, book2, book3

・book4, book5

・book6, book7, book8

・book9, book10

実行結果を見てみると、粗方似たような結果となるべきグループが関係性が強く出ていることがわかる。

同じ作品グループではないものの、 book7 と book10 は登場する年齢層が同じであり、関係が高く出ている。

同じ作品グループでも book4 と book5 は登場する固有名詞が違うため、関係が低く出ており、SF などの固有名詞が大幅に異なるような説明をすると、関係が低く出てしまうと推測される。

説明文単体だと文章自体が少ないため特定する要素が少ない場合があるため、説明文単体で関係を出すのは少々難しい面も垣間見えた。