MIDDLE EAST TECHNICAL UNIVERSITY

Department Of Computer Engineering

CENG 300

Summer Practice Report

METU Data Mining Research Group
Start Date: End Date:
Total Working Dates:

September 24, 2020

Student:
Onat ÖZDEMİR

Instructors:
Prof.Dr.Pınar KARAGÖZ

Prof.Dr.İsmail Hakkı TOROSLU

Student's Signature

Organization Approval

Contents

1 Introduction

I have done my summer internship at METU Data Mining Research Group under the supervision of Prof.Dr.Pınar KARAGÖZ and Prof.Dr.İsmail Hakkı TOROSLU. The task I have worked on was implementing a Trust Based Recommender using the collaborative filtering method and testing it on provided dataset. The dataset contains information about customers and the products they have bought. In addition to dataset, I was able to use eigentrust calculation and community detection modules provided by the TACOREC.

2 Project

During the internship, I implemented two different trust based recommenders

1. Eigentrust Weighted Recommender
2. Inverse Distance Weighted Recommender

The details of these two recommenders can be found in section 2.4 and 2.5, respectively.

2.1 Analysis Phase

There were two problems I need to solve:
1 - Dataset was very sparse
2 - There was no explicit trust information
In the both implementations I have made, first case handled by filtering the customers and products which purchased and were bought more than filtering threshold times. To gain better understanding on the second problem, I studied implicit trust calculation methods and looked into lots of research papers.

2.2 Design Phase

2.3 Implementation Phase

Since there are two different implementations, I have divided the details of the recommenders into two subsections: section 2.4 and 2.5. Under this subsection, libraries and technologies used in implementations are explained.

2.3.1 Neo4j

Driver Installation :

```
pip install neo4j
```

Configuration :

```
import neo4j

...
uri = self._config["database"]["neo4j"]["uri"]
user = self._config["database"]["neo4j"]["user"]
password = self._config["database"]["neo4j"]["password"]

self._driver = neo4j.Driver(uri, auth=(user, password))
```

Sample Usage :

```
import neo4j
...
    def get_customer_trust(self, customer_id):

    query = (
      f"MATCH (u:Customer)-[r:BELONGS_IN]->(:Community) "
      f"WHERE u.id = {repr(customer_id)} "
      f"RETURN r.eigentrust"
    )

    with self._driver.session() as session:
      return tuple(session.run(query).single())
```

Listing 1: Neo4j cypher example

2.3.2 Numpy

Installation :

```
pip install numpy
```

Sample Usage :

```
import numpy as np

class TrustBasedFilterer(object):
...

  def _create_customers_versus_products_table(self):

    self._customers_versus_products_table = np.zeros(
        (self._unique_customers.shape[0],
        self._unique_products.shape[0]),
        dtype=np.bool,
      )

    self._customers_versus_products_table[
        self._sales[:, 0],
        self._sales[:, 1],
      ] = True
```

Listing 2: Numpy example

2.3.3 Scipy

Installation :

```
pip install scipy
```

Sample Usage :

```
from scipy.sparse import csr_matrix
from scipy.sparse.csgraph import dijkstra

class Graph(object):
...

  def _create_distance_matrix(self):

    self._create_adjacency_matrix()

    self._adjacency_matrix = \
      csr_matrix(self._adjacency_matrix)

    self._distance_matrix = dijkstra(
      csgraph=self._adjacency_matrix,
      directed=False,
      return_predecessors=False,
```

```
18        unweighted=True,
19        limit=self._max_distance)
20
21    self._distance_matrix\
22      [~np.isfinite(self._distance_matrix)] = 0
```

Listing 3: Scipy example

2.4 Eigentrust Weighted Trust Based Recommender

2.5 Inverse Distance Weighted Trust Based Recommender

Inverse Distance Weighted Trust Based Recommender consists of three different modules:

1. **graph.py**: Responsible for creating adjacency matrix from given customer versus product matrix, calculating the distances between customers applying djkstra algorithm to the adjacency matrix using scipy library, and calculating the trust by taking the recreciprocal of the distance matrix.

2. **trust_based_filterer.py**: The module initially takes list of unsorted and unfiltered transaction list and filters it to create more denser customer versus product matrix. Then creates a graph object by giving the customer versus product matrix as parameter and use the trust matrix created by the graph object to calculate recommendation coefficient for each product. Finally, for each user, the module sorts all products with respect to their recommendation coefficients in descending order and recommends top k of them.

3. **trust_based_recommender.py**: This module sends the list of transactions to the trust_based_filterer and takes the list of recommendations for each customer in the database as a response. The main task of this module is writing the recommendations to the database using neo4j driver.

2.6 Testing Phase

2.6.1 Surprise

Installation :

```
pip install scikit-surprise
```

Sample Usage :

```
from surprise import AlgoBase, PredictionImpossible, Dataset
from surprise.model_selection import cross_validate

class Inverse_distance_weighted_tbr(AlgoBase):
...

reader = reader = Reader(line_format='user item rating
    timestamp', sep=';', rating_scale=(1, 5))

data = Dataset.load_from_file('./dataset.csv', reader=reader)
algo = Inverse_distance_weighted_tbr()

cross_validate(algo, data, cv=5, verbose=True)
```

Listing 4: Surprise example

2.6.2 Matplotlib

Installation :

```
pip install matplotlib
```

Sample Usage :

```
import matplotlib.pyplot as plt
...
plt.hist(eigentrust_list,
     color = 'blue',
     edgecolor = 'black',
     bins = 1000)
plt.title('Distribution of the Eigentrust Values')
plt.xlabel('Range')
plt.ylabel('Number of Eigentrust Values in the range')
plt.show()
```

Listing 5: Matplotlib example

2.6.3 Leave-one-out Cross Validation

3 Organization

3.1 METU Data Mining Research Group

4 Conclusion

References