Project Progress Report-1

Empirical Analysis of Predictive Algorithms for Collaborative Filtering

Dhayalini Nagaraj

A20359686

Abstract:

Collaborative filtering is a method of making automatic predictions about the interests of a user by collecting preferences or taste information from many users. It is also known as recommender systems. In collaborative filtering recommender system, the recommendations predicted for each user is based on the preferences of other similar users. The assumption is that if a person "A" has a opinion in an issue "C" and a person "B" has a similar opinion in the same issue "C", then more likely both the person "A" and "B" will have a similar opinion in a different issue. In this project, we use different algorithms for this task based on correlation coefficients, similarity calculations to predict the users preferences. We analyse and compare different algorithms based on accuracy in a set of problem domains. We will use books data and apply collaborative filtering techniques to recommend books to users and compare the predictive accuracy of each algorithms and decide the efficient algorithm. Books Dataset associated with Amazon Product data is used for the experiments which has reviews and ratings of the user for each data items.

1. Data

1.1 Data Source

Books datasets is used for this project. The dataset is obtained from Amazon Product data and it can be accessed from the site http://jmcauley.ucsd.edu/data/amazon/links.html. We have a two types of datasets for books. They are 'K-core' and 'Ratings only' dataset. These datasets contains reviews spanning May 1996 - July 2014. The 'K-core' data set contains 8,898,041 reviews and the 'Ratings only' dataset contains 22,507,155 ratings. The 'K-core' is dense and is approximately 3GB. The 'Ratings only' dataset includes no reviews but only ratings and the size is approximately 1 GB. The 'Ratings only' data can be processed on disk and we planning to implement different algorithms. After working with ratings, we are also planning to use the 'K-core' data with reviews and ratings and to perform the same algorithms and to compare the results with the previous implementations. Since the 'K-Core' dataset is more, it cannot be processed on disk. So we will be using an instance of AWS to process the data.

1.2 Data Format

Ratings dataset includes no metadata or reviews. It includes only User, item, rating and timestamp tuples.

- reviewerID - ID of the reviewer

- asin - ID of the product
- overall - rating of the product
- unixReviewTime - time of the review (unix time)

Sample:

{"reviewerID": “AH2L9G3DQHHAJ”,

"asin": "0000013714",

"overall": 4.0,

"unixReviewTime": 1019865600}

K-core:

This dataset is a reduced subset, in which each user and items have k reviews each. It has reviewerID, asin, reviewerName, reviewText, overall, summary, unixReviewTime, reviewTime

where,

- reviewerID - ID of the reviewer
- asin - ID of the product
- reviewerName - name of the reviewer
- helpful - helpfulness rating of the review
- reviewText - text of the review
- overall - rating of the product
- summary - summary of the review
- unixReviewTime - time of the review (unix time)
- reviewTime - time of the review (raw)

Sample:

{"reviewerID": "A1C0O09LOLVI39",

"asin": "B00005ML71",

"reviewerName": "Michael",

"helpful": [0, 0],

"reviewText": "I love it, I used this for my Yamaha ypt-230 and it works great, I would recommend it to anyone",

"overall": 5.0,

"summary": "awesome",

"unixReviewTime": 1371340800,

"reviewTime": "06 16, 2013"}

1.3 Programming Language, packages

Python language is used to implement and analyse the algorithms. The packages that we will use for this project are Scipy, Pandas, nltk, scikit-learn, Numpy, countVectorizer.

2. Preliminary Experiments:

2.1 Pre-process:

a) Load Dataset: The pandas framework is used to load the dataset. The books rating only file is downloaded from the above mentioned site and is loaded and read.

```
import pandas as pd
data = pd.read_csv('ratings_Books.csv')
```

b) Matrix data is formed using the following code

```
data_matrix = pd.DataFrame(index=data.user,columns=data.item)
```

Then cosine similarities are found and the recommendation system is further built based on that.

c) K-Core Data: Since the dataset has reviews and ratings, extract reviews from the dataset and preprocess it by converting the text to lower case and removing stop words, numbers, punctuations and by stemming. The redundant and noisy information is removed which will be useful for efficiency of the system. The LDA topic modeling algorithm will be implemented and sentiments and similarities are found later.

The stopwords can be retrieved using stopwords method of nltk package, stopwords and punctuations are removed using the following code:

```
import re
stopwords = nltk.corpus.stopwords.word('english')
copy_reviewText = reviewText.copy()
for word in reviewText:
      if word in stopwords:
            del copy_reviewText[word]
for word in copy_reviewText:
     word = re.sub(r'[^\w\s]',' ',word)
```

The document term matrix is formed using countVectorizer package from scikit-learn. Rows represent the review documents and column the count of each words.

```
import pandas as pd
import sklearn.feature_extraction.text import CountVectorizer
vectorizerObject = CountVectorizer()
dtm = vectorizerObject.fit_transform(reviewTextDocument)
dtmFrame =
pd.DataFrame(dtm.toarray().transpose(),index=vectorizer.get_feature_names())
return dtmFrame
```

Using this document term matrix the LDA topic modeling is then performed to obtain the topics and then sentiment and similarities are found

2.2 Experiments:

I will be doing experiments based on cluster model and collaborative filtering based on the clusters. The idea is that there are certain groups or types of users capturing a common set of preferences and tastes. Given the class, the preferences regarding the various items (expressed as votes) are independent. The probability model relating joint probability of class and votes to a tractable set of conditional and marginal distributions is the standard Naïve Bayes formulation.

References:

https://www.microsoft.com/en-us/research/publication/empirical-analysis-of-predictive-algorithms-for-collaborative-filtering/
http://jmcauley.ucsd.edu/data/amazon/links.html
http://www.haas.berkeley.edu/Courses/Spring2000/BA269D/BreeseHeckermanKadie99.pdf
https://en.wikipedia.org/wiki/Collaborative_filtering
http://ieeexplore.ieee.org/document/6468211/?reload=true
http://cseweb.ucsd.edu/~jmcauley/pdfs/kdd15.pdf
http://www.salemmarafi.com/code/collaborative-filtering-with-python/