SN Computer Science (2021) 2:74
https://doi.org/10.1007/s42979-021-00460-8

ORIGINAL RESEARCH

Recommendation of Influenced Products Using Association Rule Mining: Neo4j as a Case Study

Sudipta Sen[1] · Akash Mehta[2] · Runa Ganguli[2] · Soumya Sen[1]

Received: 8 September 2020 / Accepted: 8 January 2021 / Published online: 4 February 2021

Abstract

Recommendation systems are now inherent for many business applications to take important business decisions. These systems are built based on the historical data that may be the sales data or customer feedback etc. Customer feedback is very important for any organization as it reflects the view, sentiment of the customers. Online systems allow customers to purchase products at a glance from any e-commerce website. Generally, the potential buyers check the review of the products to take informed decision of purchase. In this work, we attempt to build a recommendation model to find out the influence of a product on another product so that if a user purchases the influential product then the recommender system can recommend the influenced products to the users. In this paper, the recommendation system has been built based on association rule mining. We proposed a new association rule mining technique for quick decision-making and it gives better performance over Apriori algorithm which is one of the most popular approaches for association rule mining. The entire framework has been developed in Neo4j graph data model for doing the data modelling from raw text file and also to perform the analysis. We used real-life customer feedback data of amazon for experimental purpose.

Keywords Recommendation system · Apriori algorithm · Association rule mining · Neo4j · Influential product

Introduction

The trend of e-commerce has increased rapidly in recent years with the development of the internet and due to the easy accessibility of internet usage. Easy access to the internet has driven consumers to shop online books, airline tickets/reservations, clothing/shoes videos/games and other electronic products which are the most popular items purchased on the internet. E-commerce enables the user to purchase goods and services without going to a physical market and thus saving time and energy of the user. Consumers have a wider choice not from their town or country but also round the globe. Consumers can customize or personalize products and services. There is an absolute flexibility of time and place. Consumers can check from a wide range of similar products and compare their prices and, therefore, can make a better choice. E-commerce enables suppliers to introduce and promote new markets and new products to meet the needs of individual buyers.

Recommender systems are changing from novelties used by a few e-commerce sites, to serious business tools that are reshaping the world of E-commerce. Users love it when companies can second-guess their thoughts. Recommender systems are used by e-commerce sites to suggest products to their customers. The products can be recommended based on the top overall sellers on a site, based on the demographics of the customer, or based on an analysis of the past buying behaviour of the customer as a prediction for future buying behaviour. In recent years, it has been seen

This article is part of the topical collection "Applications of Software Engineering and Tool Support" guest edited by Nabendu Chaki, Agostino Cortesi and Anirban Sarkar.

✉ Soumya Sen
iamsoumyasen@gmail.com

Sudipta Sen
sensudipta779@gmail.com

Akash Mehta
akash93mehta@gmail.com

Runa Ganguli
runa.ganguli@gmail.com

[1] A.K. Choudhury School of I.T, University of Calcutta, JD-2, Sector-3 Saltlake, Kolkata 700106, West Bengal, India

[2] Department of Computer Science, The Bhawanipur Education Society College, 5, L.L.R Sarani, Kolkata 700020, West Bengal, India

that recommender systems have been influential in boosting sales as well as user satisfaction [1]. Without a recommendation system, consumers must spend time searching for the product they like and in most of the cases it has a negative impact on the consumers. Recommender systems that are nicely designed and effectively implemented comes as a handy solution for both producer, consumer and also for business perspective. So regardless of the perspective—business or consumer, recommendation systems have been immensely beneficial [1].

Big data [3] has become an imminent part of all industries and business sectors today. Big data and recommendation engines have been providing an extremely useful combination for big corporations [2]. Big data is the driving force behind recommendation systems. A typical recommendation system cannot do its job without sufficient data and big data supplies plenty of user data such as past purchases, browsing history, and feedback for the recommendation systems to provide relevant and effective recommendations. As time passes on user behaviour data(historical data) such as log on-site activity (clicks, searches, page, and item views), Off-site activities (tracking clicks in emails, in mobile applications, and in their push notifications), customer past purchase history, goods and services customer have rated and liked, details of each item, user connections in various social media site have dramatically increases. The dataset increases so drastically that traditional relational databases cannot work efficiently. To handle these types of large and rapidly increasing datasets, we need big data. Even the most advanced recommenders cannot be effective without big data. Big data [3] analytics tool process as shown in Fig. 1 involves the data flow from collection of data from the larger dataset to provide valuable information for decision-making.

This research work has been done in Neo4j application which is an open source graph database with strong support for applications involving connected or linked data. Neo4j uses Cypher (query language) which is a declarative graph query language that allows for expressive and efficient data querying and visualization in a property graph model. Neo4j follows a property graph data model. In this model, domain data are expressed in a "node space"—a network of nodes, relationships and properties (key value pairs), compared to the relational model's tables, rows and columns. Relationships are first class objects and may also be annotated with properties, revealing the context in which nodes interact.

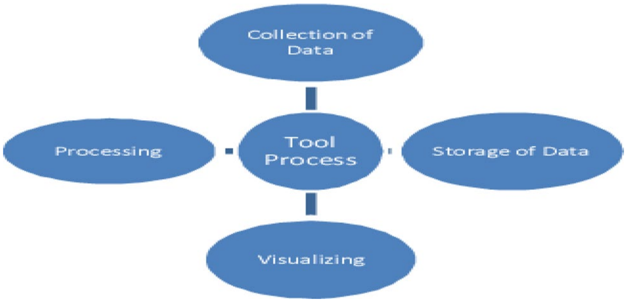


Fig. 1 Big data analytics tool process

The main focus of this research work is to introduce a novel association rule mining technique to identify the influenced product from the influential product of the given database. Apriori algorithm is one of the most popular association rule mining technique; however, it has high time complexity and henceforth incurs huge cost. Here we propose an alternative association rule mining method to reduce the time complexity to identify influenced products. The "Comparative Analysis" shows the computational benefit over apriori algorithm. This is going to be very much helpful for the business bodies who sell many products from their systems. As more transactions take place, the system can upgrade itself to reflect the association between the different products. The most interesting part of this recommendation system is that it can identify the relationship among the diversified products which are not possible without this type of computation.

The research work is organized in the following sections. In "Related Work", we present a survey on existing work. The proposed methodology is discussed in "Proposed Methodology" and this section also defines the objective and contribution of this research work. "Results and Discussion" is on the analysis and discussion of the result set. "Comparative Analysis" shows comparative analysis with Apriori algorithm and finally the paper is concluded in "Conclusion" along with the future scope of the work.

Related Work

The different models of recommendation system have been proposed over the last few years to analyze and forecast various business domains from different perspectives. Ioannis et al. in their paper [4] present a novel a product recommendation system that is based on the concept of Implicit Feedback. They named it MuSIF. MuSIF incorporates Collaborative Filtering with Matrix Factorization and Association Rule Mining. It is based on a hybrid recommendation algorithm that uses different methods to increase the overall accuracy of the system. This model can increase the accuracy of the matrix factorization algorithms via initialization of factor vectors. However, it does not perform well with fake reviews. The paper [5] uses only genuine reviews, takes the trustworthiness of the user into account and generates the results in a more significant manner. The proposed system collects reviews from different online websites and performs opinion mining and sentiment analysis. Factors like star ratings, buyer's profile, previous purchases, and whether the review has been given after purchasing or not, are included. These factors along with the user's trustworthiness make the system quite robust. Wang in his paper [6] proposed a product recommendation algorithm based on the DeepFM network. First the user's purchased products were embedded, and then the sparse features were transformed into the low-dimensional dense feature. DeepFM considers both low-level and high-level features at the same time to further improve the generalization ability of the model. The user's interests were learned from the user's purchases. This was later used to recommend products to the user. Sumaia et al. in their paper [7], propose a multi-criteria recommender systems (MCRSs) have to improve the accuracy of the recommender system. Text mining or sentiment analysis is used to extract valuable review elements. The review elements help improve the accuracy of the recommender system and mitigate most of its problems. Experimental results have shown that MCRSs outperform state of the art recommender systems. Aciar et al. [8] in their paper, design a recommender system that considers the degree of knowledge of the user, and the user's availability into account before making recommendations. This paper calculates the reputation of the users within their community, and uses this reputation to help design the recommendation systems. Suggestions given by reputed users are given more priority than others. This method genuinely improves the performance of the model. Recommendation systems are also used in different business domains such as E-learning platform [9, 10], friend recommendation [11], transportation system [12], energy management [13] etc.

To build the recommendation system association rule mining can be used effectively. Association rule mining helps to find out the association between different attributes of the system. One major issue with the original Association Rule Mining method is the time it takes to provide an acceptable result. The running time grows with increase in dimensionality. Czibula et al. in their paper [14] introduced a new approach named concurrent relational association rule mining (CRAR) which uses concurrency for the relational association rules discovery process. This helps to significantly reduce the mining time. Vougas et al. in their paper [15] provide an overview of the various supervised and unsupervised learning algorithms that are used specifically in drug response prediction applications. A novel in silico screening process, based on association rule mining, is presented in this paper. Experimental results show that this pipeline works well on large sample-spaces, while also being able to detect low frequency events. Multiple researches have shown that particle swarm optimization (PSO) algorithm is well suited for performing quantitative association rule mining (ARM). However, the method becomes inefficient when applied on huge datasets. Zhao et al. in their paper [16] have proposed a parallel PSO for quantitative association rule mining (PPQAR). This parallel algorithm strategies two methods—particle-oriented and data-oriented parallelization, to suit different application scenarios. Experimental results show that particle-oriented parallelization has a higher speedup, whereas the data-oriented method is more generic when applied on large datasets. Association rule mining is used in different business domains such as share market forecasting [17], materialized view formation [18], gene expression [19], drug reaction [20] etc. The most popular approach of applying association rule mining is Apriori algorithm [21]. Apriori algorithm is used in different business applications but it might be slower if the attributed are highly co-related. It is to be noted that the Apriori algorithm is also applied in big data applications. In both [22], the underlying data model is based on big data, however, applied on different business domains.

Among the different big data model, Neo4j is one of the most effective one for mining the data. In recommendation system, Neo4j was used for graph traversal to identify frequent pattern from the given customer reviews [23]. In this paper, Neo4j and its cypher query language was used to create directed weighted graph and access the desired paths to extract the frequent mining rules. Neo4j has been used in different social networking application for mining purpose. In [24], a time-varying social network modeling using Neo4j graph database was done for the recording of human activities and interactions from mobile devices and wearable sensors. The performance of real-world queries in terms of efficiency and scalability is demonstrated in this paper. In another application of Neo4j on social networking demonstrated the computation and evaluation of Twitter influence metrics [25]. Due to its capability of mining in big data applications, Neo4j has been used in different types of business applications such as decentralized ledger systems based on blockchain [26], book recommendation [27], healthcare applications [28] etc.

The use of Neo4j has increased rapidly in different applications with the spread of social networking, online systems etc. It is proven very much useful for these types of system for the data analytics purpose. In this research work, we will use Neo4j to build a recommendation system based on association rule mining. Here Neo4j is used both for data modelling and analytics purpose.

Proposed Methodology

The objective of this research work is to build a recommendation model in big data environment and use this model to recommend products to users. Our specific objectives are given below:

1. Extract the data from text files and to pre-process and load these data into a graph database (Neo4j) and model it for analytical purposes.
2. Analyzing the relationship between the products to find out what are the influential products.

The contribution of this research work is given below:

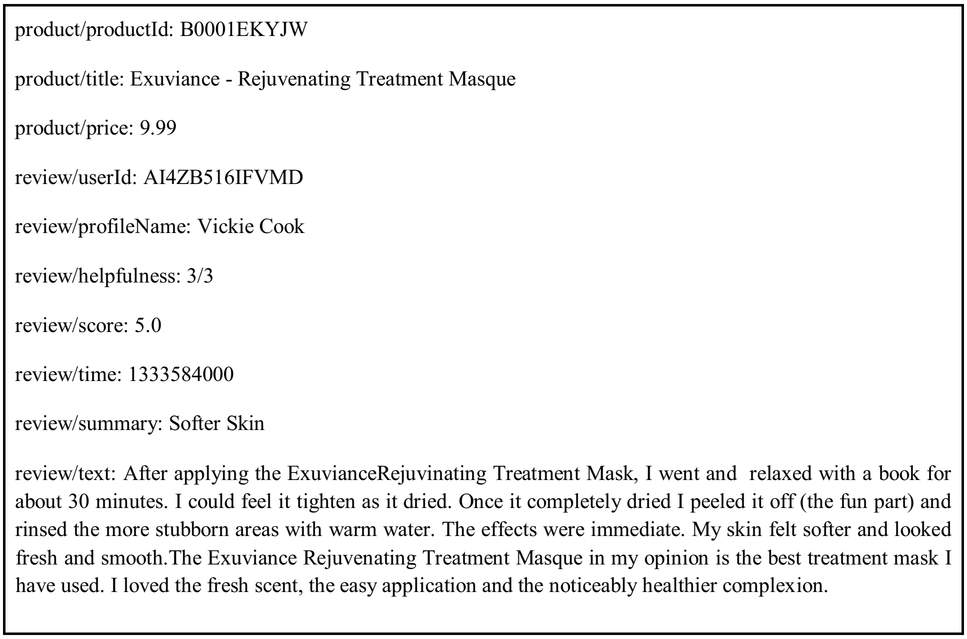
1. The dataset from text format, is converted into csv format using a python script.
2. Modeling the data in the graph database (Neo4j) for the purpose of analysis. Based on the given dataset it generates three types of nodes (labeled as user, product, and review) and two types of relationships or edges (labeled as Review_Of and Wrote). The properties of the nodes are also defined. After this, the data are imported in Neo4j.
3. Proposing a mining method to identify the set of products that influences the sale of other products.

This section is divided into three subsections:

1. Data acquisition;
2. data modeling and importing;
3. data analysis.

Data Acquisition

A real-world beauty dataset from Amazon [29] is used in this work. In this dataset, there are a total of 167,709 users, 29,004 products and 252,073 reviews. The format of the data is as given below.



product/productId: B0001EKYJW

product/title: Exuviance - Rejuvenating Treatment Masque

product/price: 9.99

review/userId: AI4ZB516IFVMD

review/profileName: Vickie Cook

review/helpfulness: 3/3

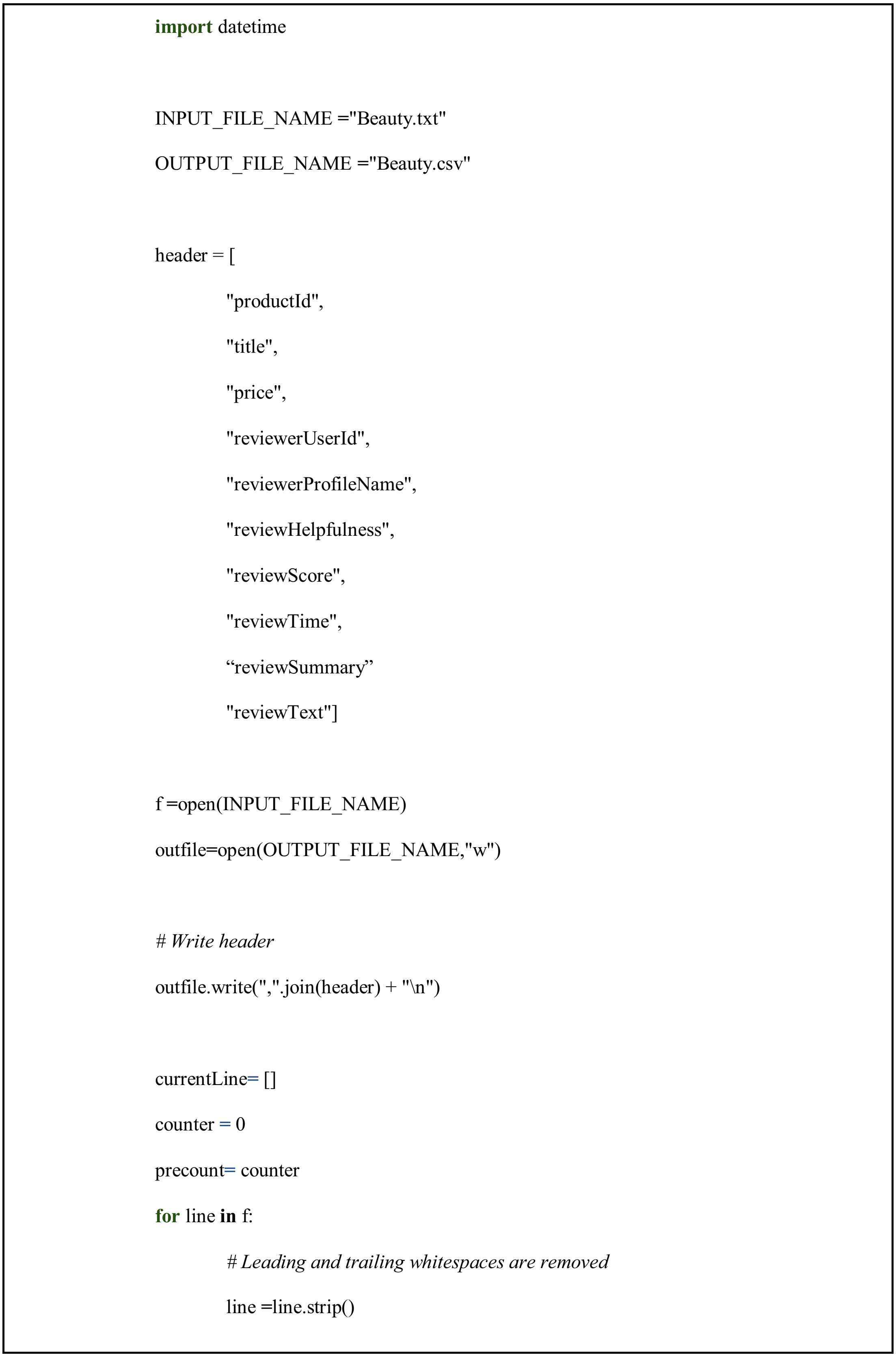
review/score: 5.0

review/time: 1333584000

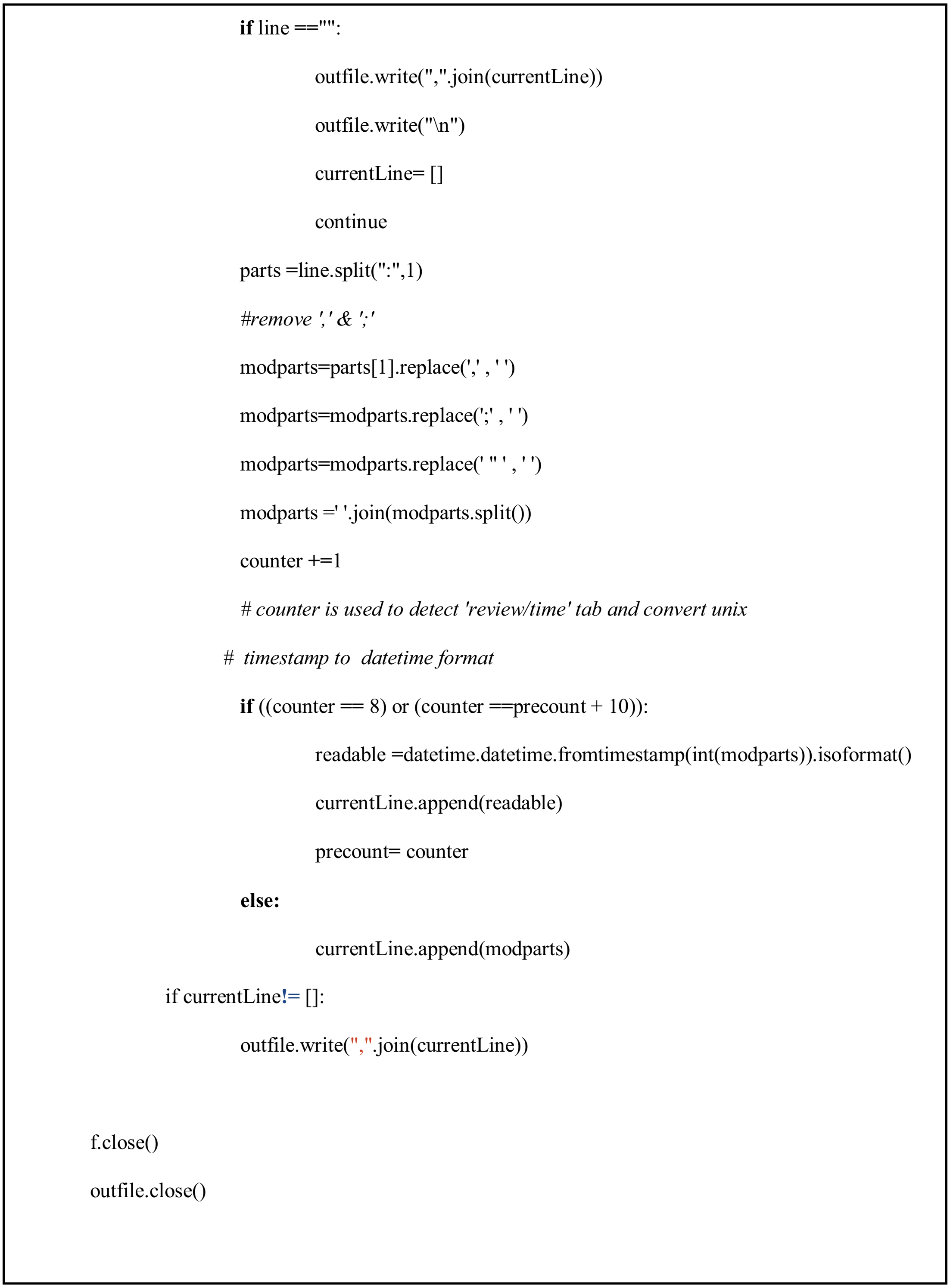
review/summary: Softer Skin

review/text: After applying the ExuvianceRejuvinating Treatment Mask, I went and relaxed with a book for about 30 minutes. I could feel it tighten as it dried. Once it completely dried I peeled it off (the fun part) and rinsed the more stubborn areas with warm water. The effects were immediate. My skin felt softer and looked fresh and smooth.The Exuviance Rejuvenating Treatment Masque in my opinion is the best treatment mask I have used. I loved the fresh scent, the easy application and the noticeably healthier complexion.

Here the time format is in UNIX time format and all other fields are self-explanatory. This dataset initially is in text format and then is converted into csv (comma separated value) format using the below python script for easily importing the data into Neo4j application.



```
import datetime


INPUT_FILE_NAME ="Beauty.txt"

OUTPUT_FILE_NAME ="Beauty.csv"


header = [

        "productId",

        "title",

        "price",

        "reviewerUserId",

        "reviewerProfileName",

        "reviewHelpfulness",

        "reviewScore",

        "reviewTime",

        “reviewSummary”

        "reviewText"]


f =open(INPUT_FILE_NAME)

outfile=open(OUTPUT_FILE_NAME,"w")


# Write header

outfile.write(",".join(header) + "\n")


currentLine= []

counter = 0

precount= counter

for line in f:

        # Leading and trailing whitespaces are removed

        line =line.strip()
```



```
                if line =="":
                    outfile.write(",".join(currentLine))
                    outfile.write("\n")
                    currentLine= []
                    continue
                parts =line.split(":",1)
                #remove ',' & ';'
                modparts=parts[1].replace(',' , ' ')
                modparts=modparts.replace(';' , ' ')
                modparts=modparts.replace(' " ' , ' ')
                modparts =' '.join(modparts.split())
                counter +=1
                # counter is used to detect 'review/time' tab and convert unix
               #  timestamp to  datetime format
                if ((counter == 8) or (counter ==precount + 10)):
                    readable =datetime.datetime.fromtimestamp(int(modparts)).isoformat()
                    currentLine.append(readable)
                    precount= counter
                else:
                    currentLine.append(modparts)
            if currentLine!= []:
                outfile.write(",".join(currentLine))


        f.close()
        outfile.close()
```

Data Modelling and Importing

This section is divided into two parts. In the subsection "Modelling Amazon Dataset into Neo4j", we demonstrate Neo4j modelling of the raw text data and in the subsection "Importing Data into Neo4j" we import the data from the text file to Neo4j.

Modelling Amazon Dataset into Neo4j

Data modelling is the process of creating a data model for the data to be stored in Database. This data model is a conceptual representation of Data objects, the associations between different data objects and the rules. Data modelling

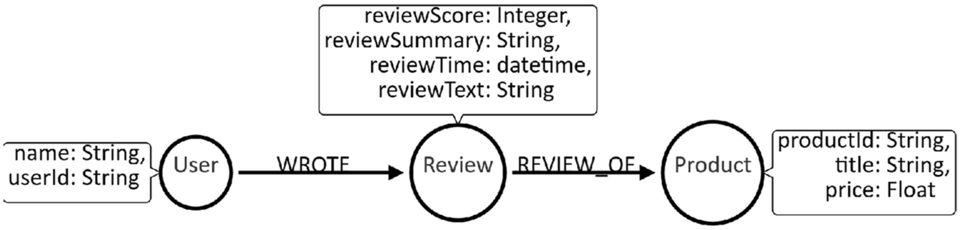
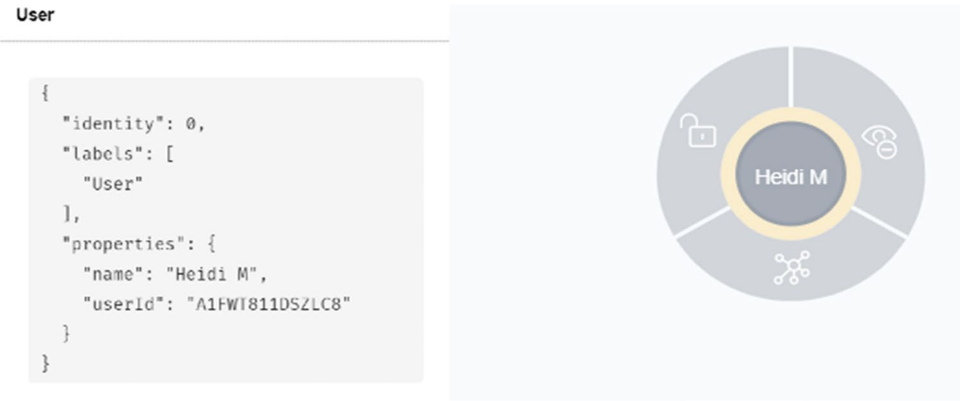
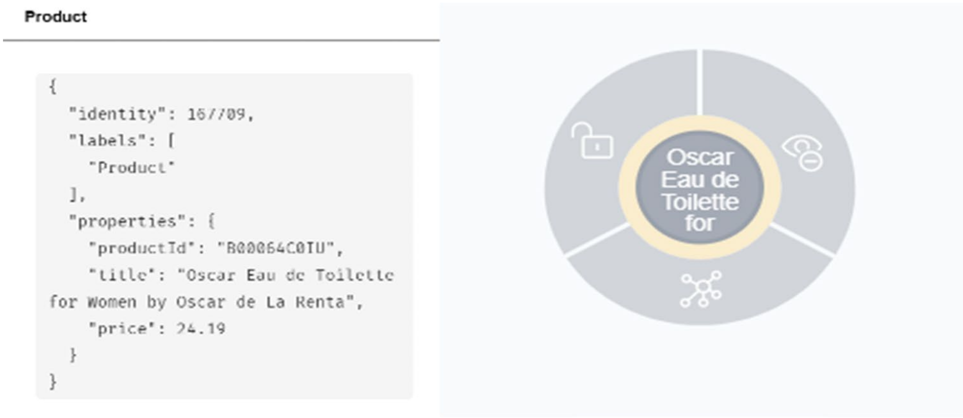


Fig. 2 Modelling amazon text dataset into Neo4j



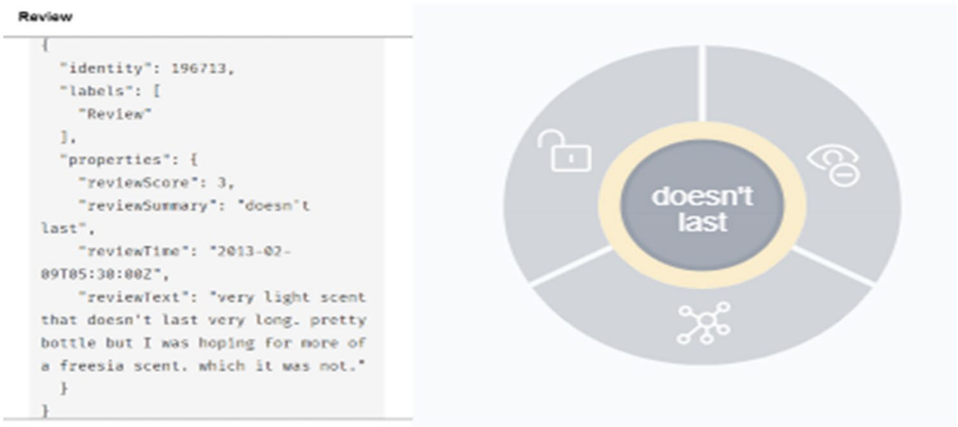
```
{
  "identity": 0,
  "labels": [
    "User"
  ],
  "properties": {
    "name": "Heidi M",
    "userId": "A1FWT811DSZLC8"
  }
}
```

Fig. 3 Sample "User" node



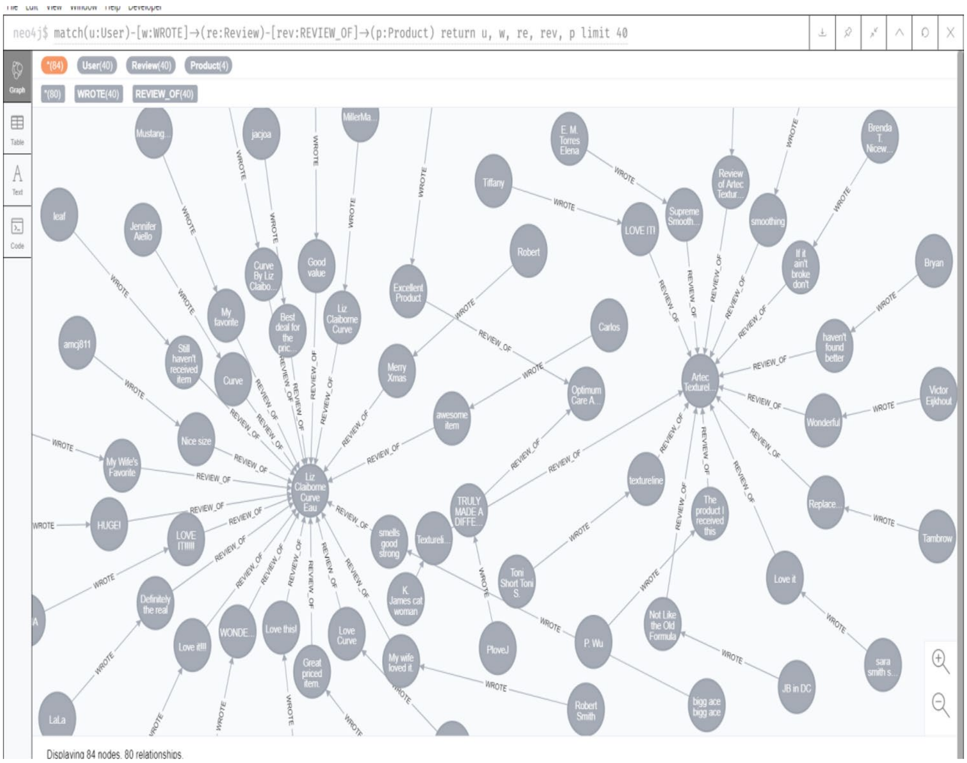
```
{
  "identity": 167709,
  "labels": [
    "Product"
  ],
  "properties": {
    "productId": "B00064C0TU",
    "title": "Oscar Eau de Toilette
for Women by Oscar de La Renta",
    "price": 24.19
  }
}
```

Fig. 4 Sample "Product" node



```
{
  "identity": 196713,
  "labels": [
    "Review"
  ],
  "properties": {
    "reviewScore": 3,
    "reviewSummary": "doesn't
last",
    "reviewTime": "2013-02-
09T05:30:00Z",
    "reviewText": "very light scent
that doesn't last very long. pretty
bottle but I was hoping for more of
a freesia scent. which it was not."
  }
}
```

Fig. 5 Sample "Review" node



```
neo4j$ match(u:User)-[w:WROTE]→(re:Review)-[rev:REVIEW_OF]→(p:Product) return u, w, re, rev, p limit 40
```

Displaying 84 nodes, 80 relationships.

Fig. 6 Overview of dataset in Neo4j

is an essential step in the process of creating any complex software. It helps developers understand the domain and organize their work accordingly.

The data formed three kinds of nodes which are labeled as “User”, “Product” and “Review”. We also have two kinds of relationship (edges) between these three nodes “Wrote” and “Review_Of”. Here “Wrote” is a relationship between the nodes labeled as “User” and “Review” and “Review_Of” is a relationship between the nodes labeled as “Product” and “Review”.

We design the data model in Fig. 2 using arrows tool. The data model shows the nodes and edges between them and property and label of every node and type of every relationship (edge).

Importing Data into Neo4j

After converting the dataset from text file to a csv file each row contains data of all the nodes “User”, “Review” and “Product” along with the relationships “Wrote” and “Product_Of”. We import each type of node from the dataset separately.

Import Node Labeled as User

Here each “User” node contains a unique userId. So we design a cypher while importing the dataset which internally checks all the existing userId. if a match is found then no new “User” node will be created and if no match was found then a new “User” node with that userId is created and then sets its name property (Fig. 3).

For faster execution, we create an index on userId which helps in searching for existing userId.

Cypher Query to Create Index

```
CREATE CONSTRAINT ON (u:User)ASSERT u.userId IS UNIQUE;
```

Cypher Query to Create User Node

```
load csv with headers from
    "file:///Beauty.csv" as csv
    MERGE(u:User {userId:csv.reviewerUserId})
    ON CREATE SET u.name = csv.reviewerProfileName
```

Import Node labeled as Product

Here each “Product” node contains a unique productId. So we design a cypher which while importing the dataset internally checks all the existing productId. If a match is found then no new “Product” node will be created and if no match is found then a new “Product” node with that productId is created and sets its title and price property (Fig. 4).

For faster execution, we create an index on productId which helps in searching for existing productId.

Cypher Query to Create Index

```
CREATE CONSTRAINT ON (p:Product) ASSERT
p.productId IS UNIQUE;
```

Cypher Query to Create User Node

```
load csv with headers from
    "file:///Beauty.csv" as csv
    MERGE(p:Product {productId:csv.productId})
    ON CREATE SET p.title = csv.title,
    p.price = toFloat(csv.price)
```

Import Node labeled as Review and Relationships "Wrote" and "Review_Of"

Here a Review node does not contain any unique id like that of "User" and "Product" node. Each row in the csv file is a review of a product given by a user. Since the review node does not contain any unique id, after creating all the review nodes it is difficult to find out the user and product associated with that specific "Review" node. So, we need to join the "User" and "Product" node after creating each "Review" node. Here joining simply means creating a relationship(edge) between "User" and "Review" node labeled as "Wrote" and relationship(edge) between "Review" and "Product" node labeled as "Review_Of" (Figs. 5 and 6).

Required Cypher Query

```
load csv with headers from "file:///Beauty.csv" as csv
    MATCH (p:Product{productId:csv.productId}),
    (u:User{userId:csv.reviewerUserId})
    CREATE(r:Review{
    reviewScore:toFloat(csv.reviewScore),
    reviewTime:datetime(csv.reviewTime),
    reviewSummary:csv.reviewSummary,
    reviewText:csv.reviewText})
    CREATE (r)-[:REVIEW_OF]->(p)
    CREATE (u)-[:WROTE]->(r)
```

Data Analysis to Identify Influential Product

In this section, we focus on finding the influential products in pairs and building a 2D array in the system which depicts the pairwise influence of products. We select those products which have a review count more than a threshold value (T1). A 2D array Influential_Product[][] of size n*n is used to uniquely identify all the selected products in the range of unique_identifier = 1 to n. Then we form pairs of all 2 item sets for these n products which has been reviewed by the same user. Given a product pair (i,j) our job is to identify whether the purchase of i influences j, or vice versa. For those selected pairs, we then find out the timestamps of purchase for the users who bought both the products in the pair for the first time. Now for each selected product pair(i,j), we take two integers count$_i$and count$_j$and initialize them with 0. Here count$_i$means number of times a user purchases product i before j and count$_j$ means number of times a user purchases j before i. The value of count$_i$is incremented if product i was purchased before product j, whereas count$_j$ is incremented if product j was purchased before product i. The following formula (1) is used to initialize Influential_Product[][]:

$$
\mathit{Influential\_Product}[i][j] = \begin{cases} 1, & \mathit{count}[i] > \mathit{count}[j] \\ 2, & \mathit{count}[j] > \mathit{count}[i] \\ 0, & (\mathit{count}[i] = \mathit{count}[j])\mathit{and}(\mathit{count}[i] \neq 0) \\ -1, & (\mathit{count}[i] = \mathit{count}[j])\mathit{and}(\mathit{count}[i] = 0) \end{cases} . \tag{1}
$$

Entry 1 for (i,j)th cell position in the matrix indicates that product i is influencing product j. Whereas entry 2 for (i,j)th cell position in the matrix indicates that product i is influenced by product j.

If both of the counts are equal and non-zero, it can't be inferred which product is more influential.

Whereas if both the counts are equal and zero, then it indicates that the two products have never been purchased together by the same user for the first time. Thus, to exclude these cases from our finding of influential products, we mark such entries to be − 1.

Thus to summarize, each cell (i,j) of 2D array Influential_Product[][] can obtain any of the four values:

1. 0 if a conclusion cannot be drawn. These products do not influence each other.
2. 1 if 'i' influences 'j'.
3. 2 if 'j' influences 'i'.
4. − 1 if 'i' and 'j' are never purchased together by same user.

When all the pairs are scanned, the 2D array Influential_Product[][] gets completely populated. Now whenever users search or buy a product say A, we can recommend other products that are influenced by A. The products influenced by A can be retrieved by only scanning the entire row for product A and whenever we get a value 1 the corresponding product in the column is influenced by A.

Algorithm 1: Influence_Pair

Input: m number of products.

Output: A 2D array Influential_Product[n][n] which depicts the pairwise influence of n products from the given m products.

```
1) Select n products out of total m products having review count >$T_1$
                              [n products shortlisted after applying given threshold T1]
2) Take a 2D array Influential_Product[][] of size n * n.
3) All the selected products are uniquely identified in the range of unique_identifier = 1 to n
                                                                   [n is the integer value]
4) For each product i = 1 to n
        a) For each product j = (i+1) to n
            i)   Initialize: $count_1$ = 0, $count_2$ = 0
            ii)  For each common user u = 1 to k, who purchased both products i and j,
                    (1) Find out the timestamp in which user-u bought product i and j for
                        the first time, let the timestamp be $t_i$, $t_j$
                    (2) if ($t_i$<$t_j$)
                                $count_1$ = $count_1$ + 1
                                                            [product i purchased before j]
                        else if ($t_i$>$t_j$)
                                $count_2$= $count_2$+ 1
                                                            [product j purchased before i]
                        End if
                    End For

            iii) if ($count_1$> $count_2$)
                        Influential_Product[i][j] = 1
                        Influential_Product[j][i] = 2
                                                    [product-i is influencing product-j]
                 else if ($count_1$< $count_2$)
                        Influential_Product[i][j] = 2
                        Influential_Product[j][i] = 1
                                                     [product-j is influencing product-i]

                 else if ($count_1$ == $count_2$ &&$count_1$ !=0 && $count_2$ !=0)

                        Influential_Product[i][j] = 0
                        Influential_Product[j][i]=0
                                                    [products don't influence each other]
                 else if ($count_1$ ==0 && $count_2$ ==0)

                        Influential_Product[i][j] = -1
                        Influential_Product[j][i] = -1
              [ product-i and j are never purchased together by same user]
                 End if

     End for

End for
```

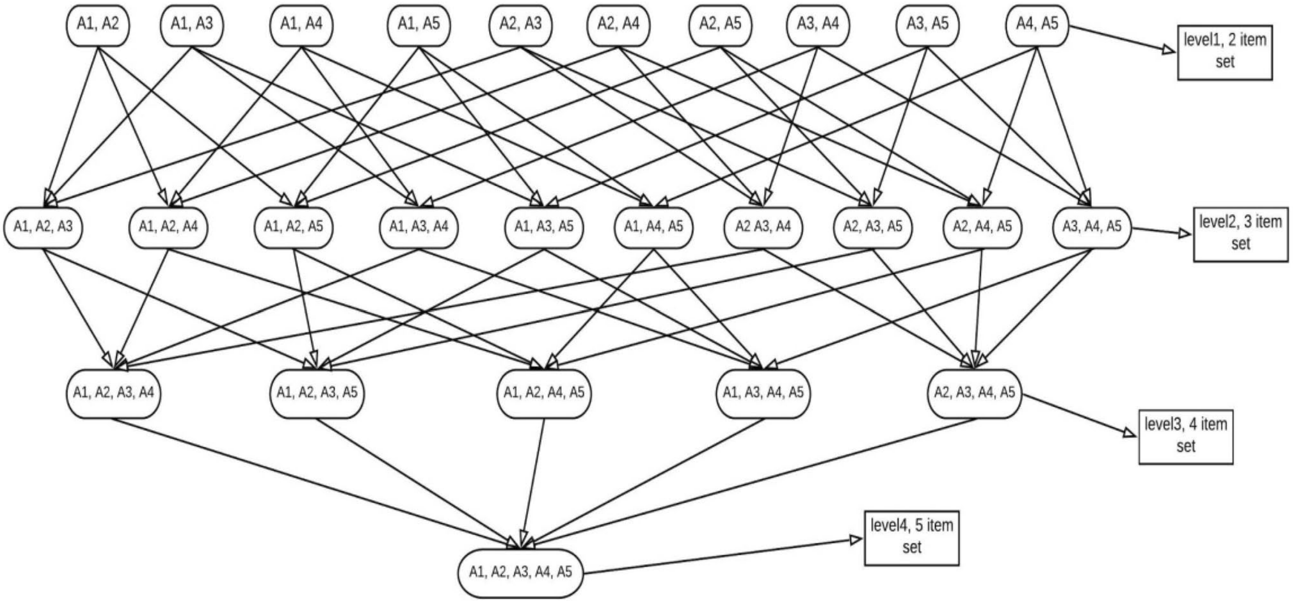


Fig. 7 Generation of itemset in apriori algorithm

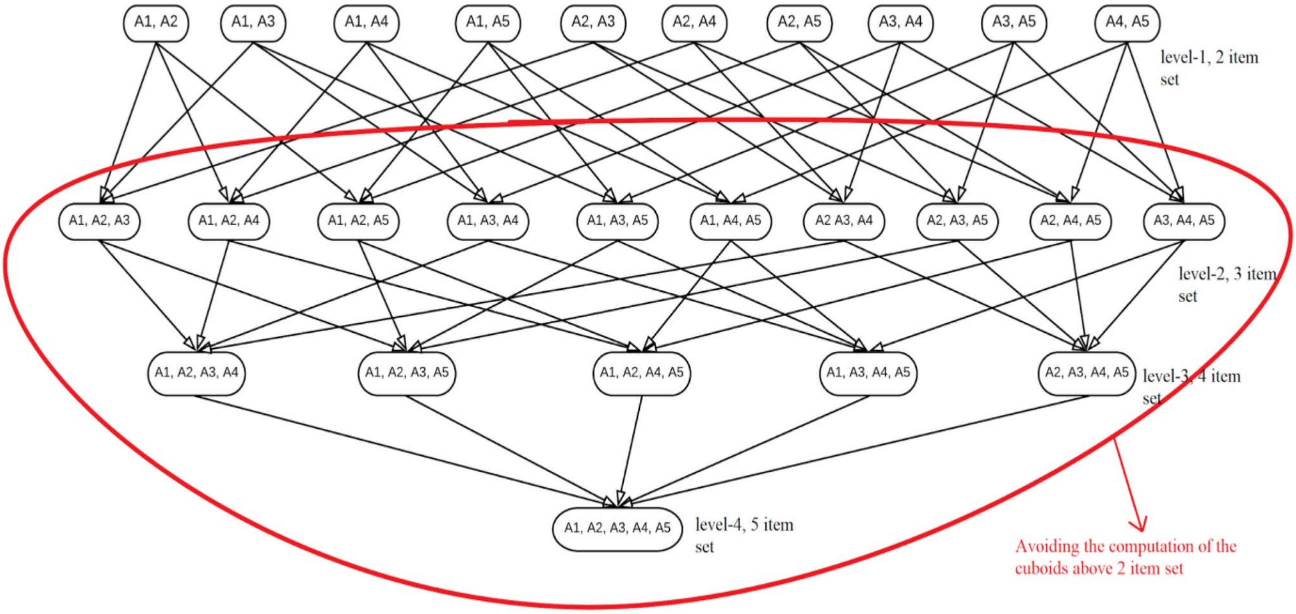
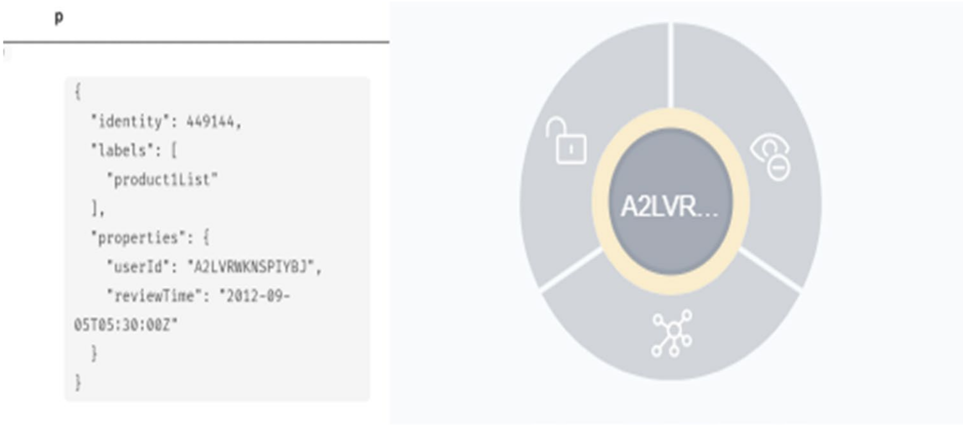


Fig. 8 Discarding the generation of itemset in proposed methodology



```
{
  "identity": 449144,
  "labels": [
    "product1List"
  ],
  "properties": {
    "userId": "A2LVRWKNSPIYBJ",
    "reviewTime": "2012-09-
05T05:30:00Z"
  }
}
```

Fig. 9 Sample Product1List node

This method can be extended to deal with itemsets having more than 2 items. For itemset with n items the Total numbers of itemset for computation is given by the following formula (2):

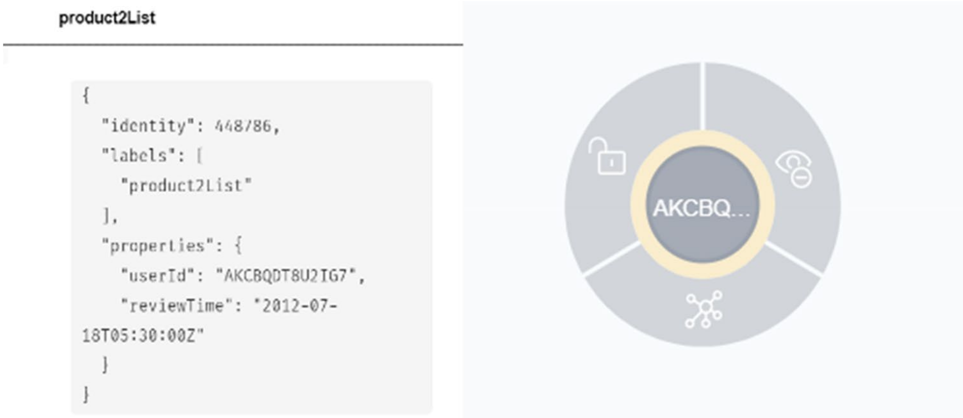
$$2^n - {}_1{}^nC - {}_0{}^nC = 2^n - n - 1 \tag{2}$$

The one disadvantage of this method is the extremely large number of itemsets that need to be computed for large values of n. In an online retail system, numbers of products are high in number hence computing (2^n-n-1) itemset is practically impossible. We depict this in Fig. 7 for 5 itemset.

To avoid this high computation, we propose an alternative approach to do the analysis. We will compute only 2 itemsets for all the given products. For n products the number of pairs will be ${}^nC_2$. Now using these 2 itemset pairs, we will compute any other itemset. Let us consider five products A1, A2, A3, A4, A5. For these items we will generate ${}^5C_2$ pairs. These are < A1,A2 > , < A1,A3 > , < A1,A4 > , < A1, A5 > , < A2,A3 > , < A2,A4 > , < A2,A5 > , < A3,A4 > , < A 3,A5 > , < A4,A5 > . We will use these 2 items sets for the computation of 3 item sets to 5 item sets. In this way we can avoid the huge computation as shown in Fig. 8.

Computation for More Than 2 Item Set

Any higher item set can be represented in terms of 2 item sets. For example < A,B,C > can be represented as < A,B > , < A,C > , < B,C > . For example < A,B,C,D > can be represented as < A,B > , < A,C > , < A,D > , < B,C > , < B,D > , < C,D > . In the following algorithm for any item set we will generate all the possible 2 item sets and check the matrix Influential_Product[n][n] to perform the analysis.



```
{
  "identity": 448786,
  "labels": [
    "product2List"
  ],
  "properties": {
    "userId": "AKCBQDT8U2IG7",
    "reviewTime": "2012-07-
18T05:30:00Z"
  }
}
```

Fig. 10 Sample Product2List node

Table 1 Snapshot of the users who purchased Product1 before Product2

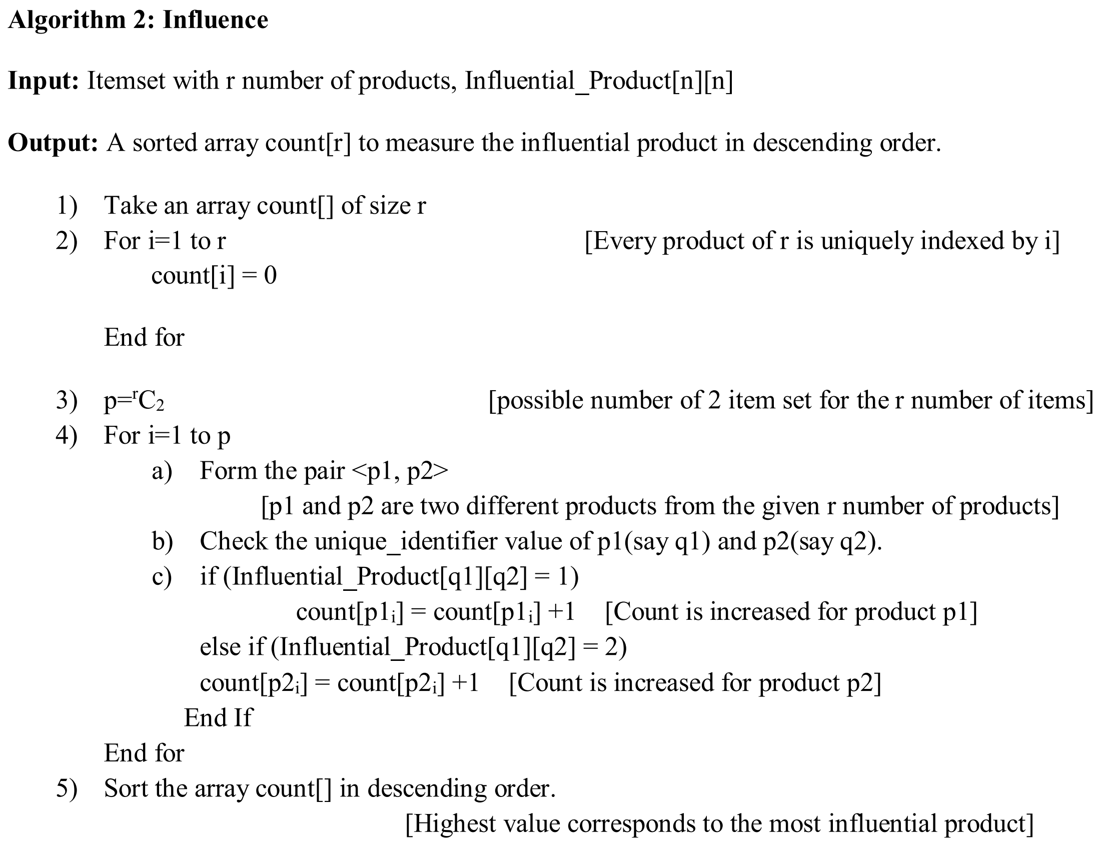
	userId	product1_ReviewTime	product2_ReviewTime
1	AKCBQDT8U2IG7	"2012-07-10T05:30:00Z"	"2012-07-18T05:30:00Z"
2	AK6PL8DNAZ03P	"2012-08-01T05:30:00Z"	"2012-09-19T14:39:00Z"
3	AGMK6ROKAGNMP	"2012-06-02T05:30:00Z"	"2012-08-01T05:30:00Z"
4	A23U3DMY3W9PCM	"2013-01-15T05:30:00Z"	"2013-01-21T05:30:00Z"
5	A1SWKVKTCWILIX	"2012-09-11T05:30:00Z"	"2012-09-18T23:51:00Z"
6	A2XSA2ZXDQK2Y	"2010-09-26T05:30:00Z"	"2010-09-30T09:30:00Z"
7	ARIDDYTQ1ND1N	"2012-08-31T05:30:00Z"	"2012-09-13T05:41:00Z"
8	A7BXPJ4ZB0FJ8	"2013-02-02T05:30:00Z"	"2013-02-14T02:30:00Z"
9	A2ZPVN2IJ9VA70	"2012-08-01T05:30:00Z"	"2012-08-10T05:30:00Z"
10	A3W499L5IJ2U4	"2012-08-14T05:30:00Z"	"2012-08-23T19:00:00Z"
11	A1MP8B0M2BR9Z5	"2009-01-16T05:30:00Z"	"2009-01-17T05:10:00Z"
12	AWFMPKT4EE9Z9	"2008-03-29T05:30:00Z"	"2008-04-04T08:12:00Z"
13	A38IL2GXD8866Z	"2011-08-06T05:30:00Z"	"2011-08-27T13:37:00Z"
14	A1TLT5UPB0QPXY	"2012-09-01T05:30:00Z"	"2012-09-01T06:30:00Z"
15	A281FJJ7JM6W0Z	"2012-12-07T05:30:00Z"	"2013-02-01T14:26:00Z"
16	A1IOGEZUZPII8C	"2012-06-24T05:30:00Z"	"2012-07-09T09:31:00Z"
17	A3H61JF6HG6KCW	"2011-07-15T05:30:00Z"	"2011-08-23T18:02:00Z"
18	A2MEJOMP8M4DPG	"2008-04-07T05:30:00Z"	"2008-04-11T16:08:00Z"
19	A2Q5ZCO7P5NZTH	"2012-08-02T05:30:00Z"	"2012-08-12T05:15:00Z"
20	A2336BHPQW6UQH	"2011-02-08T05:30:00Z"	"2012-09-12T05:30:00Z"

Table 2 Snapshot of the users who purchased Product2 before Product1

	userId	product1_ReviewTime	product2_ReviewTime
1	A1GHSW62QHK6KR	"2009-05-05T12:08:00Z"	"2008-11-27T17:46:00Z"
2	A22SFPYPRS4V6F	"2013-05-16T13:11:00Z"	"2013-01-18T05:30:00Z"
3	A2G9Q433DWW6ZS	"2012–08-10T06:38:00Z"	"2012–08-01T05:30:00Z"
4	A4POIU8EL0OY8	"2013–02-28T04:19:00Z"	"2013–01-21T07:41:00Z"
5	A395C2BVSD7GM9	"2008–07-19T01:32:00Z"	"2008–01-21T19:23:00Z"
6	A3BQ1WV4P043QH	"2012–02-09T08:02:00Z"	"2011-04-01T08:51:00Z"
7	A1URKU7WVVW4KL	"2013-03-12T19:47:00Z"	"2013-01-23T02:34:00Z"
8	A1WK2GKU50XEU	"2009-09-21T11:30:00Z"	"2009-05-27T05:36:00Z"
9	A1ZFFYXP9DDK6A	"2010-04-09T023:16:00Z"	"2010-05-28T16:03:00Z"
10	A19YSUOANEA8W7	"2012-09-04T12:41:00Z"	"2012-07-18T22:53:00Z"
11	A14TT2GL0MQMU1	"2011-11-27T21:39:00Z"	"2011-10-12T14:49:00Z"
12	A8VGX0V9OM4T5	"2012-10-03T17:22:00Z"	"2012-06-27T09:28:00Z"
13	A1XC0CJB0UJYW7	"2012-11-18T15:24:00Z"	"2012-06-26T00:18:00Z"
14	AOOVVRAYAKRJO	"2012-05-01T03:19:00Z"	"2012-04-19T13:56:00Z"
15	AQVJJ8F47K9M6	"2013-01-10T10:16:00Z"	"2013-01-06T08:12:00Z"

Table 3 Snapshot of the users who purchased Product2 and Product1 at the same time

	userId	product1_ReviewTime	product2_ReviewTime
1	A3BQ1WV4P043QH	"2011-04-01T05:30:00Z"	"2011-04-01T05:30:00Z"
2	A1URKU7WVVW4KL	"2013-01-23T19:53:00Z"	"2013-01-23T15:53:00Z"
3	A1WK2GKU50XEU	"2012-05-21T12:18:00Z"	"2012-05-21T12:18:00Z"
4	A1ZFFYXP9DDK6A	"2011-04-28T12:39:00Z"	"2011-04-28T12:39:00Z"
5	A19YSUOANEA8W7	"2010-07-18T23:11:00Z"	"2010-07-18T23:11:00Z"
6	A14TT2GL0MQMU1	"2009-10-12T00:47:00Z"	"2009-10-12T00:47:00Z"
7	A8VGX0V9OM4T5	"2012-06-27T08:31:00Z"	"2012-06-27T08:31:00Z"
8	A1XC0CJB0UJYW7	"2010-06-26T19:02:00Z"	"2010-06-26T19:02:00Z"
9	AOOVVRAYAKRJO	"2009-04-19T16:59:00Z"	"2009-04-19T16:59:00Z"
10	AQVJJ8F47K9M6	"2013-02-06T05:06:00Z"	"2013-02-06T05:06:00Z"



Algorithm 2: Influence

Input: Itemset with r number of products, Influential_Product[n][n]

Output: A sorted array count[r] to measure the influential product in descending order.

```
1) Take an array count[] of size r
2) For i=1 to r                                  [Every product of r is uniquely indexed by i]
       count[i] = 0

   End for

3) $p={}^{r}C_{2}$                          [possible number of 2 item set for the r number of items]
4) For i=1 to p
       a) Form the pair <p1, p2>
                 [p1 and p2 are two different products from the given r number of products]
       b) Check the unique_identifier value of p1(say q1) and p2(say q2).
       c) if (Influential_Product[q1][q2] = 1)
                    count[$p1_i$] = count[$p1_i$] +1    [Count is increased for product p1]
          else if (Influential_Product[q1][q2] = 2)
          count[$p2_i$] = count[$p2_i$] +1    [Count is increased for product p2]
         End If
   End for
5) Sort the array count[] in descending order.
                              [Highest value corresponds to the most influential product]
```

Results and Discussion

Now by our developed algorithm, we will find influence between specific two products using neo4j cypher query. Product1 has productId ′B00011JKRW′ and Product2 has ′B00011JKSG′. We first find all the users who bought the product with productId ′B00011JKRW′ and find the timestamp when the users bought the product for the first time. For this, we made nodes labelled as product1List with two properties: userId and timestamp. userId denotes the Id of the user who bought that product and timestamp denotes the time when the user bought that product for the first time. So every node labelled as product1List had a unique userId. So, for faster execution of query we build index on userId for nodes labelled as product1List.

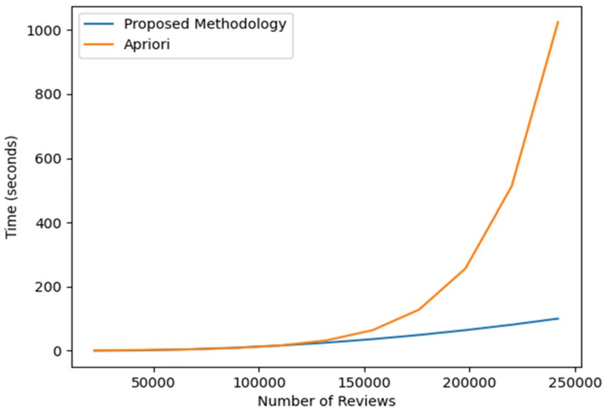


Fig. 11 Comparison of run time execution of apriori and proposed methodology

Query to build index

CREATE CONSTRAINT ON (u:product1List) ASSERT u.userId IS UNIQUE;

Query to build nodes labeled as product1List

MATCH (u:User)-[:WROTE]- > (r:Review)-[:REVIEW_OF]- > (p:Product {productId:'B00011JKRW'})
merge(t:product1List{userId:u.userId})
ON CREATE SET t.reviewTime = datetime(r.reviewTime)
ON MATCH set t.reviewTime = (case when t.reviewTime > r.reviewTime then datetime(r.reviewTime) else datetime(t.reviewTime) end)

Now we will take every 'User' and 'Review' node associated with Product2 i.e. the 'Product' node with productId 'B00011JKRW' and for every 'User' node we are checking a 'product1List' node with the same userId. If a 'product1List' node with the same userId already exists then we update the timestamp of 'product1List' node based on the condition that the timestamp of 'product1List' node is greater than the timestamp of the associated 'Review' node and if a 'product1List' node with the same userId does not exists then we create a 'product1List' node with the same productId and timestamp as of the corresponding 'Product' and 'Review' node (Fig. 9).

Next, we perform the same thing for productId B00011JKSG but here we create a different set of nodes labeled as product2List.

Query to build index

CREATE CONSTRAINT ON (u:product2List) ASSERT u.userId IS UNIQUE;

Query to build nodes labeled as product2List

MATCH (u:User)-[:WROTE]- > (r:Review)-[:REVIEW_OF]- > (p:Product {productId:'B00011JKSG'})
merge(t:product2List{userId:u.userId})
ON CREATE SET t.reviewTime = datetime(r.reviewTime)
ON MATCH set t.reviewTime = (case when t.reviewTime > r.reviewTime then datetime(r.reviewTime) else datetime(t.reviewTime) end)

Now we have to find how many times product1 is bought before the product2 and vice versa (Fig. 10).

Query to find how many times product1 is bought before product2 by any user

match(u1:product1List),(u2:product2List)
where u1.userId = u2.userId AND
u1.reviewTime < u2.reviewTime
return u1.userId AS userId, u1.reviewTime AS product1_ReviewTime, u2.reviewTime AS product2_ReviewTime

Result

Table1 is a snapshot of 20 rows of the result which contains a total of 202 results.

Query to find how many times product1 is bought after product2 by any user

match(u1:product1List),(u2:product2List)
where u1.userId = u2.userId AND
u1.reviewTime > u2.reviewTime
return u1.userId AS userId, u1.reviewTime AS product1_ReviewTime, u2.reviewTime AS product2_ReviewTime

Result

Table2 is a snapshot of 15 rows of the result which contains a total of 89 results.

Query to find how many times product1 and product2 are brought together by any user

match(u1:product1List),(u2:product2List)
where u1.userId = u2.userId AND
u1.reviewTime = u2.reviewTime
return u1.userId AS userId, u1.reviewTime AS product1_ReviewTime, u2.reviewTime AS product2_ReviewTime

Result

Table3 is a snapshot of 10 rows of the result which contains a total of 67 results.

So, there are a total (202 + 89 + 67) = 358 who bought both product1 with productId: 'B00011JKRW' and product2 with productId: 'B00011JKSG'. Out of these, 202 users bought product1 before product2, 89 users bought product2 before product1 and 67 users bought both the products together. So, we can draw a conclusion that the product1 i.e. the product with productId 'B00011JKRW' influences the product2 i.e. the product with productId 'B00011JKSG'.

Comparative Analysis

The product recommendation system in this proposed work is based on association rule mining. Apriori algorithm is one of the popular methods to execute association rule mining. Our proposed methodology works efficiently over Apriori algorithm from many aspects.

- In Apriori algorithm based on the given minimum threshold, the number of items can be generated upto 2^n where n is the number of products, whereas in the proposed system ${}^nC_2$ numbers of items are only generated.
- The proposed methodology is built as a customized system. After the generation of ${}^nC_2$ numbers of items, the generation of remaining items are done based on the user queries. Whereas, in apriori algorithm all the possible itemsets (2^n) may be generated. Henceforth the proposed approach has better time complexity and saves space.
- The runtime complexity of Apriori algorithm is very high. It is in the order of exponential.

Let us consider M is the number of transactions and n is the number of items. Then the runtime complexity of the Apriori algorithm is: $O(M*2^n)$.

Whereas the complexity of the proposed method is given as: $O(M) + O(n^2) \approx O(n^2)$

We applied the proposed method and apriori algorithm on the dataset as described in "Data Acquisition". In this database, the number of review is 252073. We selected different numbers of reviews to show the proposed methodology is much faster than the Apriori algorithm. Figure 11 shows the comparison of execution time.

Moreover in Apriori algorithm the computation result of 2^n items are stored in worst case where as in the proposed methodology the computation result of ${}^nC_2$ items are only stored. This implies significant improvements of space complexity too.

Conclusion

This research work proposes a recommendation model framework in big data environment. Many recommendation models have been proposed over the time using association rule mining in traditional database systems. Here we carry out this work in Neo4j for data modelling and analysis to make the system more flexible in terms of the underlying data source. The modelling of the data in Neo4j for the analysis purpose from the raw text file is one of the major contributions of this work. Other systems can adopt this framework for analysis purpose in graph database model. After the modelling of data, the analysis is done. A new association rule mining technique is proposed that gives significantly better time and space complexity over Apriori algorithm. The proposed framework is generic in nature and can be customized for any other online system to recommend items. The research work could be extended for different types of analysis. This analysis is done on review rating of the products. The work can be extended further by analyzing reviewer's comments based on natural language processing (NLP). Another approach could be to rate the reviewers so that the potential customers can identify the quality of reviews written. Moreover, the reviewers could be clustered into different groups based on the reviews written. We can incorporate other parameters such as product price, location of the customers etc. to infer more interesting patterns.

Funding This study was not funded by any agency or organization.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. S Shaikh, S Rathi, P Janrao. Recommendation System in E-Commerce Websites: A Graph Based Approached. 2017 IEEE 7th International Advance Computing Conference (IACC), Hyderabad, 2017; pp. 931-934, doi: https://doi.org/10.1109/IACC.2017.0189
2. A. Juneja, N. N. Das. Big Data Quality Framework: Pre-Processing Data in Weather Monitoring Application. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019; pp. 559–563, doi: https://doi.org/10.1109/COMITCon.2019.8862267.
3. J. Patel. An Effective and Scalable Data Modeling for Enterprise Big Data Platform. 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019; pp. 2691–2697, doi: https://doi.org/10.1109/BigData47090.2019.9005614.
4. IoannisSchoinas, Christos Tjortjis. MuSIF: A Product Recommendation System Based on Multi-source Implicit Feedback. 15th IFIP International Conference on Artificial Intelligence

Applications and Innovations (AIAI), Hersonissos, Greece. 2019; 660–672 doi: https://doi.org/10.1007/978-3-030-19823-7_55

5. Soor GK, Morje A, Dalal R, Vora D. Product recommendation system based on user trustworthiness and sentiment analysis. ITM Web of Conf EDP Sci. 2020;32:03030. https://doi.org/10.1051/itmconf/20203203030.
6. Wang S. Research of shopping recommendation system based on improved wide-depth network. Mater Sci Eng Conf Ser. 2020;768(7):72072. https://doi.org/10.1088/1757-899X/768/7/072072.
7. Al-Ghuribi SM, Mohd Noah SA. Multi-criteria review-based recommender system-the state of the art. IEEE Access. 2019;7:169446–68. https://doi.org/10.1109/ACCESS.2019.2954861.
8. Aciar SV, Aciar GI, Collazos CA, González CS. User recommender system based on knowledge, availability, and reputation from interactions in forums. IEEE RevistaIberoamericana de Tecnologias del Aprendizaje. 2016;11(1):18–22. https://doi.org/10.1109/RITA.2016.2518441.
9. Wu D, Lu J, Zhang G. A fuzzy tree matching-based personalized e-learning recommender system. IEEE Trans Fuzzy Syst. 2015;23(6):2412–26. https://doi.org/10.1109/TFUZZ.2015.2426201.
10. Ghosh S, Roy S, Sen S. An efficient recommendation system on e-learning platform by query lattice optimization. In: Sharma N, Chakrabarti A, Balas V, Martinovic J, editors. Data management analytics and innovation advances in intelligent systems and computing. Singapore: Springer; 2021. https://doi.org/10.1007/978-981-15-5616-6_6.
11. R Ganguli, A Mehta, NC Debnath, S Aljahdali, S Sen. An Integrated Framework for Friend Recommender System Using Graph Theoretic Approach. Proc. of the 35th International Conference on Computers and Their Applications (CATA 2020) EPiC Series in Computing 2020; 69:242–255. DOI: https://doi.org/10.29007/4bwn
12. Yuan NJ, Zheng Y, Zhang L, Xie X. T-finder: a recommender system for finding passengers and vacant taxis. IEEE Trans Knowl Data Eng. 2013;25(10):2390–403. https://doi.org/10.1109/TKDE.2012.153.
13. Pinto T, Faia R, Navarro-Caceres M, Santos G, Corchado JM, Vale Z. Multi-Agent-Based CBR Recommender System for Intelligent Energy Management in Buildings. IEEE Syst J. 2019;13(1):1084–95. https://doi.org/10.1109/JSYST.2018.2876933.
14. Czibula G, Czibula I, Miholca D, Crivei L. A novel concurrent relational association rule mining approach. Expert Syst Appl. 2019. https://doi.org/10.1016/j.eswa.2019.01.082.
15. Vougas K, Sakellaropoulos T, Kotsinas A, Foukas GRP, Ntargaras A, Koinis F, Georgoulias V. Machine learning and data mining frameworks for predicting drug response in cancer: an overview and a novel in silico screening process based on association rule mining. Pharmacol Ther. 2019;203:107395. https://doi.org/10.1016/j.pharmthera.2019.107395.
16. Yan D, Zhao X, Lin R, et al. PPQAR: Parallel PSO for quantitative association rule mining. Peer-to-Peer Netw Appl. 2019;12:1433–44. https://doi.org/10.1007/s12083-018-0698-1.
17. Maji G, Sen S, Sarkar A. Share Market Sectoral Indices Movement Forecast with Lagged Correlation and Association Rule Mining. In: Saeed K, Homenda W, Chaki R, editors. Computer Information Systems and Industrial Management CISIM 2017 Lecture Notes in Computer Science. Cham: Springer; 2017. https://doi.org/10.1007/978-3-319-59105-6_28.
18. S Roy, B Shit, S Sen. Association Based Multi-Attribute Analysis to Construct Materialized View. Springer 3rd International Doctoral Symposium on Applied Computations and Security Systems (ACSS 2016) Aug 12-14, Kolkata, India. Advances in Intelligent Systems and Computing book series (AISC, volume 567) pp 115-131; ISBN: 978-981-10-3409-1; https://doi.org/https://doi.org/10.1007/978-981-10-3409-1_8
19. Mallik S, Mukhopadhyay A, Maulik U. RANWAR: rank-based weighted association rule mining from gene expression and methylation data. IEEE Trans NanoBiosci. 2015;14(1):59–66. https://doi.org/10.1109/TNB.2014.2359494.
20. Jin H, Chen J, He H, Williams GJ, Kelman C, O'Keefe CM. Mining unexpected temporal associations: applications in detecting adverse drug reactions. IEEE Trans Inf Technol Biomed. 2008;12(4):488–500. https://doi.org/10.1109/TITB.2007.900808.
21. Luna JM, Padillo F, Pechenizkiy M, Ventura S. Apriori versions based on mapreduce for mining frequent patterns on big data. IEEE Trans Cybern. 2018;48(10):2851–65. https://doi.org/10.1109/TCYB.2017.2751081.
22. Sheng G, Hou H, Jiang X, Chen Y. A novel association rule mining method of big data for power transformers state parameters based on probabilistic graph model. IEEE Trans Smart Grid. 2018;9(2):695–702. https://doi.org/10.1109/TSG.2016.2562123.
23. JamshidiNejad S, Ahmadi-Abkenari F, Bayat P. A combination of frequent pattern mining and graph traversal approaches for aspect elicitation in customer reviews. IEEE Access. 2020;8:151908–25. https://doi.org/10.1109/ACCESS.2020.3017486.
24. C Cattuto, M Quaggiotto, A Panisson, A Averbuch. Time-varying social networks in a graph database: a Neo4j use case. In First International Workshop on Graph Data Management Experiences and Systems (GRADES '13). Association for Computing Machinery, New York, NY, USA, Article 11, 2013; 1–6. DOI: https://doi.org/10.1145/2484425.2484442
25. Georgios D, Andreas K, Phivos M, Spyros S. Defining and evaluating twitter influence metrics: a higher order approach in Neo4j. Soc Netw Anal Min (SNAM). 2017. https://doi.org/10.1007/s13278-017-0467-9.
26. Tsoulias K, Palaiokrassas G, Fragkos G, Litke A, Varvarigou TA. A graph model based blockchain implementation for increasing performance and security in decentralized ledger systems. IEEE Access. 2020;8:130952–65. https://doi.org/10.1109/ACCESS.2020.3006383.
27. INPW Dharmawan, R Sarno. Book recommendation using Neo4j graph database in BibTeX book metadata. 2017 3rd International Conference on Science in Information Technology (ICSITech), Bandung, 2017; 47–52 doi: https://doi.org/10.1109/ICSITech.2017.8257084.
28. Stothers JAM, Nguyen A. Can Neo4j replace PostgreSQL in Healthcare? AMIA Jt Summits Transl Sci Proc. 2020;2020:646–53.
29. Ni J, Li J, McAuley J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) 2019;188–197). DOI: https://doi.org/10.18653/v1/D19-1018