SUGGEST RECOMMENDATION FOR LIBRARY USERS USING GRAPHS

Gheorghe-Cătălin Crișan[1],

[1] *PhD. student, University „Lucian Blaga" of Sibiu, Faculty of Science, Romania*

Abstract

The aim of this paper is to prove the usefulness of graphs in solving an ever-present problem for library users: finding books they like and they are looking for. Graphs are known as an important tool in solving conditioned optimization problems. We propose a graph-based system of recommendation which can be easy used in a library for assisting and helping users in finding in real time the books they like. The main advantage of the proposed graph-based approach lies in the ease with which new data or even new entities from different sources are added to the graph without disturbing the entire system. The system uses the similarity scores in order to find the similarity between objects and to get the best recommendation for a user's request. In the end, we will compare the results from used formulas..

Keywords: graph, optimization, similarity, library

1 Introduction

Graphs are everywhere. They are known as an important tool in solving conditioned optimization problems For example, Google Maps use graphs to find the best routes for cars, buses, and walks. Thus, we need to find a different approach for each type of route based on some requirements. Car drivers have to respect some traffic rules like speed limit, one-way streets, crowd and so on. All of these influence the time of arrival to destination. For buses, we have to know the routes of these, when the buses come in a specific station, even the time spent by the user during the route. This is useful when the user has to change two buses to reach the destination and we want the user to spend as little time as possible. Also for the walks, we have to know about restricted walk area or even the weather, so the user can bypass a rainy area.

Search engines use graphs to rank web pages. This feature allows us to get only relevant pages based on searched keywords. So, if a user searches for pet stores we want to show only pages for pet stores near the user location for example.

Another good example is social media networks. We have a graph which looks like a network where the users are connected based on some rules. If a user is friend with other user the graph will create an edge between them to make a

correlation. This is useful when we want to recommend new friends or possibly known peoples.

Nowadays, a large part of the commerce takes place in the online environment, which is why we want to find the most effective ways to increase sales. A very important factor is providing personalized recommendations to every user at the right time or even in real time. This ensures that the suggested products have a high level of interest for the consumer, so he is willing to purchase the product, and the business can achieve an increase in the percentage of sales of the products.

Product recommendation is not just a sales strategy, it's also an action that helps improve the user experience in the online environment. Thus, a pleasant experience can increase the number of conversions and sales and increase the potential ROI of marketing efforts to minimize the effort the user is making. For example, 35% of Amazon's revenue is generated by its recommendation engine [1].

The use of graphs for recommendations presents a number of advantages such as the ability to suggest real-time recommendations based on the latest user actions, ease of setting parameters to be taken into account when suggesting a recommendation, the ability to add data in graph from different sources (relational or nonrelated database, csv files, etc.) without compromising the already existing graph, but also the ease of integration with the existing application systems.

After this, we will go through formulas used for our experiment and we'll make a comparison between them.

The rest of the article is organized as follows: the second section presents the problem definition and why such a problem exists. The third section presents the proposed solution and what are the main benefits. In the fourth section we take a look at theoretical aspects of graphs and how we use the graphs to map the data and user data sets. The fifth section contains information about the tool used to map data and view them using Cypher and details about the results obtained using different mathematical approaches. And in the last section the conclusions and further research directions.

2 Problem definition

The main problems with libraries are that there is not so much interaction and also the reader only reads books recommended by friends.

Many libraries do not interact enough with readers which purchase books to know their preferences. Thus, the library is just a place where you come if the reader needs a book which he already knows the title. Only a small part of the readers are willing to read from other genres because there is a big chance they will do not like it. They will not risk with these books because they don't want to spend time for something that they will not enjoy. So, the readers

need somebody to know what they like to read and what they would like to read.

The second fact is that most of the readers tend to read books that are recommended by friends or written by the same author whose books they have read before and liked. This is because they know their friends preferences, they talk together about the books they read. Based on that, they will like to receive suggestions from them.

3 Proposed solution

The solution proposed in this article is a graph-based system of recommendation which can be easily used in a library for assisting and helping users in finding in real time the books they like.

The main advantages of this solution are:

- increased user experience because they receive good recommendation
- easy way to add new data to graph system from different sources like dumped database, .csv file or any other file with data
- easy to implement as an external micro service for recommendation

4 Theoretical Aspects

4.1 Applying Graph Theory

The graph is a mathematical structure with countless applications in real life. Based on Jonathan L. Gross et al. (2004) we will describe the main ideas of graph theory. Using the graph we can create relationships between different objects using the following elements of the graph:

- nodes: representing the objects that make up the data set
- edges: representing the relationship between two objects that have a certain connection

The nodes or entities we are going to use can be of several types. Thus, for example, we will have user nodes and book nodes, all interconnected according to the established relationships. In this way, if an X person buys a Y card, then an edge is created from node X to node Y.

In our case, the edges are to be unidirectional so that if node A has a relation (edge) with node B it does not mean that node B has a relation with node A. If we want to do this we use two edges, one from node A at node B and one from node B to node A, as can be seen in Fig. 1.

Another essential aspect is that each of the nodes represents an object that has a set of properties specific to each object type. Thus, a book object can have properties such as title, author, genreand publishing house. These sets of properties also apply to edges between nodes. So, each edge represents a

specific type of relationship that has a set of properties. For example, a "BOUGHT" type edge may have properties such as the acquisition date or rating offered for that product.

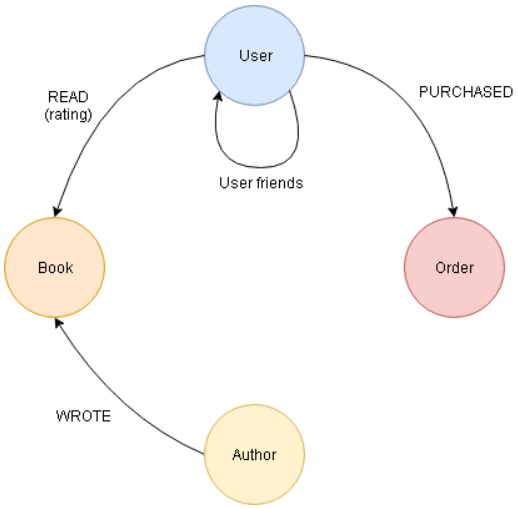

Fig. 1 – Example of graph with 3 types of nodes and 3 types of edges [2]

Thus, for the recommendation of books to the users of a library we have to have the following structure:

- the graph nodes:
  - book (with properties):
    - title
    - authorId
    - genreId
  - author
    - name
  - genre
    - name
  - user
    - name
    - years
    - gender
    - job

- the edges of the graph with various actions:
  - wrote
  - in_genre
  - bought
    - rating

4.2 Types of Recommendations and Formulas

Suggestion recommendations can be of two types: content-based or collaborative filtering [3]. Content-based filtering also has features comparing objects properties and making a score based on them, where each feature can have a weight more or less important than another feature. Using this approach, the object with the highest score will be recommended, considering the user's preferences remain constant. The problem with this approach is the fact that suggestions are suggested only from the categories that the user has bought without recommendations for products in other categories, so that the user may be interested in them but have not seen them before.

On the other hand, collaborative filtering is based on how other users responded to the same object as compared to the current user. This determines whether our client might like a particular product (Guy, N. N., 2017). This is done by filtering users who have interacted with the same objects and finding similar objects that have been purchased by other users with similar preferences by doing a filter after the best score of objects.

The following formulas are also described in Junmei Feng et al. (2018) article presenting different algorithms for finding similarity. Using the Jaccard index we can measure the level of similarity between two objects resulting in a score with the value in the range [0, 1]. This means that two identical objects have the score 1 and two different objects altogether have a score of 0. The Jaccard index counts the common properties of two objects, that is, the intersection of the two sets of properties, then divides them into the total number of unique object properties, that is, the meeting of the two sets of properties. This can be applied for content-based recommendations.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

Another function is the cosine distance with which we can compare the level of similarity between two objects resulting in a score that has a value in the range [-1, 1]. It transforms the values of the relations of the two objects into two vectors, and then calculates the distance representing the difference between the two objects. Thus for a 0° difference we have cos(0°)=1 meaning that the two objects are perfectly similar and cos(180°)= -1 meaning the two objects are totally opposite. It can be applied for collaborative filtering recommendations.

$$C(A,B) = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2}$$

Pearson correlation is another feature used for collaborative filtering that takes into account the fact that for each object the value of relationships can differ, for example, two people can give a different rating for a book because one person is more demanding. Therefore, Perason correlation takes into account

the average of values, so some objects tend to have higher values of relationships than other objects.

$$P(A,B) = \frac{\sum_{i=1}^{n}(A_i - \bar{A}) * (B_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \bar{A})^2 * (B_i - \bar{B})^2}} \quad (3)$$

5 Experimental setup

We have 3 main steps used in our project implementation to create the recommendation system. The first step is to create the dataset. Because we can't find an existing dataset for a library with users, borrowed books, ratings and so on we create a dataset with imaginary data. We put all these data in separated .csv file (one file for each entity – like books, author, reader, order).

The second step is to import created data into a graph database. There is plenty of graph database alternative: Neo4j, Titan, Cassandra, OrientDB, etc. before.

The third step is to choose a query-programming language. This depends on what graph database you choose: Cypher Query Language for Neo4j, Gremling for Titan, Cassandra Query Language for Cassandra. So, for our project, we will use Cypher and we have a syntax like the one presented below:

```
(user: User {name: "Alex"}) -> [b:BORROW] -> (book:Book)
```

Fig. 2 – Listing example for Cypher query matching all books borrow by user Alex

Cypher Query Language is a declarative graph query language that allows for expressive and efficient querying and updating of a property graph [4]. The Cypher type system is simple to use and use a specific syntax for declaring nodes, relationships, paths, maps, lists, integers, floating-point numbers, booleans, and strings:

- CREATE / DELETE: Used to create / delete nodes and relationships.
- SET / REMOVE: Used to set / remove values to properties on nodes and relationships.
- MERGE: Used to match existing or create new nodes.
- MATCH: Used to get data from the graph
- WHERE: Used to add for filter results
- WITH: Used to passing results or to give aliases to results
- RETURN: Used to get results

Data sets are to be loaded from .csv files as you can see in the example below in Fig.3.

```
LOAD CSV WITH HEADERS
FROM 'file:///books.csv'
AS line
MERGE (book:Book { id: line.id,
                   title: line.title,
                   authorId: line. authorId,
                   genreId: line. genreId
})
```

Fig. 3 – Loading book data set

6 Experiments and results

Based on data imported using the technique exemplified above we will create a graph like the next one. Here we can see authors and the books borrowed/bought by Maria Anders and Thomas Hardy represented as nodes:

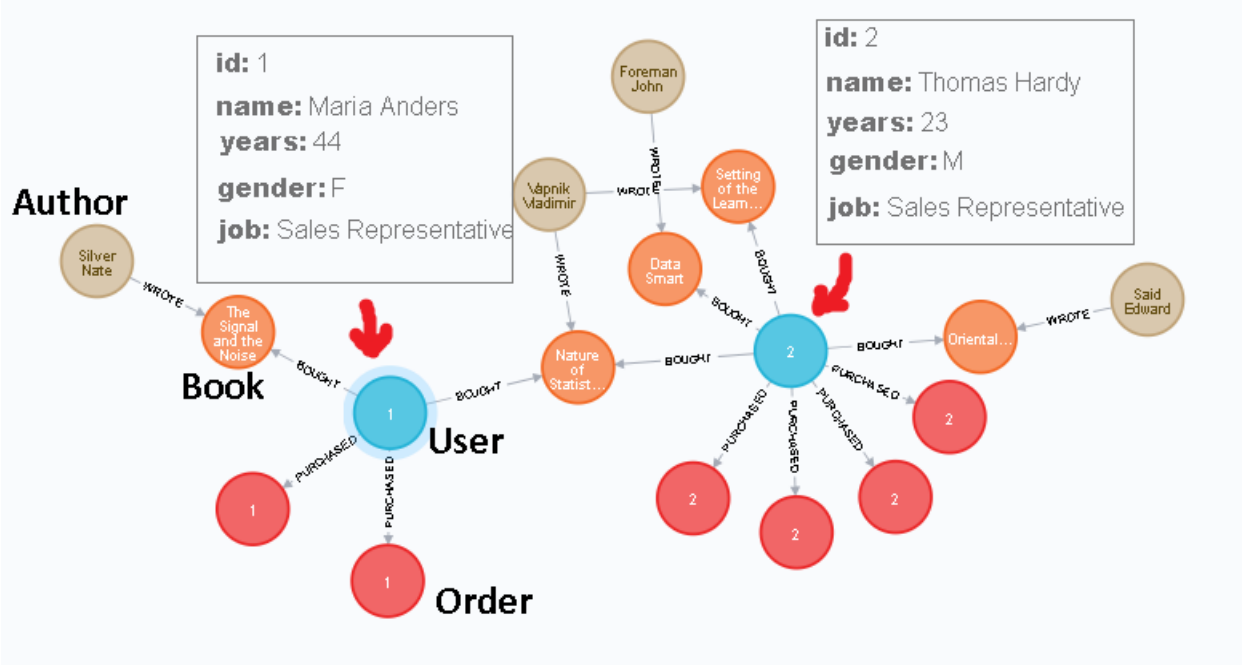

Fig. 4 – View graph relationships between two users

In the following section, we will describe formulas used for recommendations and what are the advantages and disadvantages of each one.

6.1 Jaccard Index

In the following, the recommended recommendations for the book "Nature of Statistical Learning Theory" using the Jaccard Index will be analyzed. The algorithm can be configured to take account of one or more features of objects. Thus, as can be seen in Fig. 5 the algorithm searches for similar

books that are the same or that are written by the same author. So, the higher the number of features in common, the better the score. However, this type of content-based recommendation is not very effective because it does not take into account which books other users are interested in, but just try to find cards with identical features.

The results obtained in Fig. 6 shows that the "Setting of the Learning Problem" book has the best score. This is due to the fact that the two books have the same author and the same genre, while the following recommendations have the same genre but different authors.

```
MATCH (b: Book {title: "Nature of Statistical Learning Theory"})-
[:IN_GENRE|:WROTE]-(g)<-[:IN_GENRE|:WROTE]-(other:Book)
WITH b, other, COUNT(g) AS intersection
MATCH (b)-[:IN_GENRE|:WROTE]-(bg)
WITH  other, intersection, COLLECT(bg.name) as set1
MATCH (other)-[:IN_GENRE|:WROTE]-(og)
WITH  other, intersection, set1, COLLECT(og.name) as set2
WITH other, intersection, set1, set2
WITH other, intersection,set1+filter(x IN set2 WHERE NOT x IN set1)
AS union
RETURN           other.title           AS           recommendation,
((1.0*intersection)/SIZE(union)) AS score ORDER BY score DESC LIMIT
5
```

Fig. 5 – Using of the Jaccard Index algorithm

recommendation	score
"Setting of the Learning Problem"	0.5
"Machine Learning for Hackers"	0.3333333333333333
"Analysis - Vol I"	0.3333333333333333
"Introduction to Algorithms"	0.3333333333333333
"Data Smart"	0.3333333333333333
"Fundamentals of Wavelets"	0.3333333333333333
"Python for Data Analysis"	0.3333333333333333

Fig. 6 – The recommendations obtained by applying the Jaccard Index algorithm

The advantage is that this approach is good when we want to match as many book properties as we can. So, we are looking for common properties like author, the genre of the book, etc.

The disadvantage is that there is no possibility to suggest books from other genres or even other authors. All these because the compared books will have a low score of similarity, so the recommendation will fail.

6.2 Cosine Distance

In Fig. 7 you can see the Cosine Distance algorithm listing on our data set. The algorithm suggests recommendations for the "Maria Anders" user looking

for other users who bought books that Maria bought. Thus, the algorithm performs collaborative filtering taking into account the preferences of other similar users. This is deduced from the user rating on shared cards. In this way, a classification of users who gave similar recharges for the books that the two users bought.

So, "Maria Anders" bought the book "Nature of Statistical Learning Theory" (rating 5) and the book "The Signal and the Noise" (rating 3). From the results obtained in Fig. 8 we can see that the algorithm suggests Mary as the first recommendation the book "Complete Sherlock Holmes - Vol I" bought by "Christina Berglund" (rating 5) who also bought the book "Nature of Statistical Learning Theory" (rating 3). This pattern shows that the two people have a book in common, and because the rating values given for the other books are similar, the algorithm finds this match.

```
 MATCH (u1:User {name: "Maria Anders"})-[x:BOUGHT]->(b:Book)<-
[y:BOUGHT]-(u2:User)
WITH SUM(x.rating * y.rating) AS ab_sum,
    SQRT(REDUCE(ai = 0.0, a IN COLLECT(x.rating) | ai + a^2)) AS
a_sqrt,
    SQRT(REDUCE(bi = 0.0, b IN COLLECT(y.rating) | bi + b^2)) AS
b_sqrt,
    u1, u2
WITH u1, u2, ab_sum / (a_sqrt * b_sqrt) AS cos_distance
ORDER BY cos_distance DESC LIMIT 10

MATCH (u2)-[r:BOUGHT]->(m:Book) WHERE NOT EXISTS ((u1)-[:BOUGHT]-
>(m))
RETURN m.title AS recommendation, SUM(cos_distance * r.rating) AS
score
ORDER BY score DESC LIMIT 5
```

Fig. 7 - Using of the Cosine Distance algorithm

recommendation	score
"Complete Sherlock Holmes - Vol I"	5.0
"Making Software"	5.0
"Machine Learning for Hackers"	5.0
"Data Smart"	4.0
"The Trial"	4.0
"Python for Data Analysis"	4.0

Fig. 8 – The recommendations obtained by applying the Cosine Distance algorithm

The advantage is that this approach is to match users with similar properties such as books borrowed by both having the same rating or users have the same age and so on. So, now we can find a book written by other authors and even from other genres because a similar user read that book and now we can recommend to our reader.

The disadvantage is that this is not so accurate when we have some readers who like a niche book genre and we do not have other readers with similar preferences. Thus, for our niche user, we can't find other users with the same

preferences and we have to use the Jaccard Index instead to find similar books.

6.3 Pearson Correlation

The following implementation shown in Fig. 9 of Pearson Correlation comes as an optimization of the Cosine Distance algorithm. This is because the algorithm takes into account an average of the user's ratings, so the algorithm takes into account the overall direction in which the rating values tend. So the algorithm solves the recommendation problem by finding similar books that two people buy, even if a person is more exigent on the rating.

This is shown in Fig. 10 where "Maria Anders" bought the book "Nature of Statistical Learning Theory" (rating 5) and the book "The Signal and the Noise" (rating 3), with an average rating of 4. And the "Francisco Chang" bought the book "Nature of Statistical Learning Theory" (rating 4), "Machine Learning for Hackers" (rating 5), "Superfreakonomics" (rating 3) and "Physics & Philosophy" (rating 3) with an average rating of 3.75. Thus, applying Pearson's formula results in a coefficient equal to 1 which means that there is an absolutely positive linear correlation. This leads to the recommendation of a book read by "Francisco Chang", which is chosen from the books that "Maria Anders" did not read and whose rating of "Francisco Chang" is the highest.

```
MATCH (u1:User {name:"Maria Anders"})-[r:BOUGHT]->(m:Book)
WITH u1, avg(r.rating) AS u1_mean
MATCH (u1)-[r1:BOUGHT]->(m:Book)<-[r2:BOUGHT]-(u2)
WITH u1, u1_mean, u2, COLLECT({r1: r1, r2: r2}) AS ratings MATCH
(u2)-[r:BOUGHT]->(m:Book)
WITH u1, u1_mean, u2, avg(r.rating) AS u2_mean, ratings UNWIND
ratings AS r
WITH sum( (r.r1.rating - u1_mean) * (r.r2.rating - u2_mean) ) AS
nom,sqrt( sum( (r.r1.rating - u1_mean)^2) * sum( (r.r2.rating -
u2_mean) ^2)) AS denom, u1, u2 WHERE denom <> 0
WITH u1, u2, nom/denom AS pearson
ORDER BY pearson DESC LIMIT 10
MATCH (u2)-[r:BOUGHT]->(m:Book) WHERE NOT EXISTS( (u1)-[:BOUGHT]-
>(m) )
RETURN m.title AS recommendation, SUM( pearson * r.rating) AS score
ORDER BY score DESC LIMIT 5
```

Fig. 9 - Using of the Pearson Correlation algorithm

recommendation	score
"Machine Learning for Hackers"	5.0
"Orientalism"	4.0
"Data Smart"	4.0
"Setting of the Learning Problem"	3.0
"Physics & Philosophy"	3.0
"Superfreakonomics"	3.0

Fig. 10 – The recommendations obtained by applying the Pearson Correlation algorithm

The advantage is that this approach is good to match readers with common properties such as books borrowed by both with the same rating. Another advantage is also that we take into account the mean of the ratings for example. In this way, some users tend to give higher ratings than others so we have to find the correct pattern.

The disadvantage is the same as for cosine distance.

Conclusions

This article aims to present how to use graphs to suggest book recommendations to library users. It proposes an approach where the library application uses a graph saving data structure to attract as many users as possible.

On my opinion, the proposed application could be an important factor in loyalty to users because they receive accurate, real-time recommendations without the need for additional resources. Thus, the article has been able to show the benefits of using different statistical algorithms to find the level of similarity between two objects and how to implement them in real life to increase the number of hits of the library application.

One useful extension consists of using a hybrid recommendation system that uses the algorithms shown above but which also takes into account the latest user-viewed products as well as the number of times the product was viewed.

References

[1] Sales force, *Product Recommendation Engines to Improve Customer Relationships*, https://www.salesforce.com/solutions/industries/retail/resources/product-recommendation-engines , (accessed 16 March 2019).

[2] Microsoft, *Create a graph database and run some pattern matching queries using T-SQL*, https://docs.microsoft.com/en-us/sql/relational-databases/graphs/sql-graph-sample?view=sql-server-2017 , (accessed 16 March 2019)..

[3] Valiance Solutions, *RECOMMENDER SYSTEMS 101*, https://valiancesolutions.com/recommender-systems-101 , (accessed 16 March 2019).

[4] Wikipedia, *Cypher Query Language*, https://en.wikipedia.org/wiki/Cypher_Query_Language , (accessed 16 March 2019).

[5] Jonathan L. Gross, Jay Yellen (2004). *Handbook of Graph Theory*, CRC Press

[6] Junmei Feng , Xiaoyi Fengs, Ning Zhang, Jinye Peng (2018). *An improved collaborative filtering method based on similarity*, https://doi.org/10.1371/journal.pone.0204003

[7] Guy, N. N. (2017). *A Recommender system for rental properties (Thesis)*. Strathmore University