DOI: 10.6977/IJoSI.202206_7(2).0004
Z. Nassima, L. Zineb / Int. J. Systematic Innovation, 7(2), 48-60 (2022)

Towards the development of a recommender system for product delivery using graph databases and related algorithms

ZAIKER Nassima[1], LAMGHARI Zineb[2]
1 ETIS Laboratory, cy-tech Engineering School, CY-CERGY University in Paris, France
2 LRIT associated unit to CNRST (URAC 29), Rabat IT Center, Faculty of Sciences,
Mohammed V University in Rabat, Morocco
zaikernass@cy-tech.fr, zineb_lamghari2@um5.ac.ma

(Received 6 January 2022; final version received 21 March 2022; accepted 21 March 2021)

Abstract

Recommendation systems are among the promising strands of machine learning that have revolutionized information retrieval operations. These systems are designed to make recommendations to users based on different factors.

The realization of a recommender system requires a study of the users' needs and the metrics that may influence each recommendation, as well as the attributes that can be entered into the application but that have no effect on the system's functioning.

In this context, SoftCentre[1] aims to develop a delivery recommender system using graph databases and related algorithms, in order to figure out the best path for each delivery to its destination. In this context, the deliverer will respect deadlines, specifications, and deduce the best itinerary to travel on.

Therefore, our project revolves around the design, modeling, and implementation of a recommendation system based on these main phases: 1)Data collection and preprocessing, 2)Graph database creation, and 3)Applying recommendation and optimization algorithms.

Keywords: Recommendation system, process model, Hybrid filtering, Graph database, Neo4j, Cypher, Python

[1] organization that creates national champions in the software industry to improve Morocco's international attractiveness

1. Introduction

There are many mobile (Jumia, Food on Demand, Glovo, etc.) or desktop applications (El Morocco Club, Best Restaurants, etc.) that present a wide choice of restaurants, menus, meals, and best places to visit, as well as suggest delivery services according to the customer's profile, his budget, his preferences, and his current geographical location. However, as a deliveryman, there is no system that considers orders' preferences and availability, especially for independent delivery people who have to do auto-orders. This random process makes the deliverer lose time in navigating and selecting deliveries, as well as allows him to make an unfavorable selection with incompatible orders and deliver a minimum number of orders across a very long route.

Thus, the current method has several limitations: deliverers suffer during the search from information overload due to published orders, publications on several sites, waste of time, lack of user experience personalization, and are shocked by the fact that the amount of information they are confronted with is greater than their capacity to process it. Moreover, they always wait for a user's request to recommend references. Indeed, there is a lack of informational data from deliverers and customers' requests. In this sense, general information provides a lack of adaptation to the user's interest in recommending and valuing contents such as deliverer metadata, customer metadata, descriptive information about orders, new deliveries, and funds that have historical value.

From a technical point of view, a delivery referral system is effective when it meets the needs of functional requirements. The system must take into consideration the delivery customer's evaluation and build a user model from the collected data, relying on real-time data (current geographic location). Also, it is required to take into consideration non-functional requirements.

The evaluation result aims to improve the performance and quality of the functional requirements of the recommended system. This is done by personalizing recommendations according to the deliverer profile and availability of customer point of view as recommendations for any deliverer connected to the application. In this sense, the deliverer has to respect deadlines, specifications, and deduce the right itineraries. Therefore, the deliverer has to respect the deadlines and specifications of each command and order and propose the optimal delivery route. In this context, we aim to create a recommender system that will work as background of a mobile application. Thus, the focus of this system is to provide real-time recommendations to each deliverer, about the best route path between two specific destinations, based on their geographical location, vehicle type, capacity, and availability, in order to reduce the total distance that will be travelled/forwarded (the total cost). In addition, it is rentable in terms of increasing the density of routes, the size of the delivery points by grouping deliveries according to location, proximity, delivery schedule, and order specifications.

To resolve this problem, the key technology for enabling real-time recommendations is the graph database (Kamphuis, 2020), a technology that is quickly leaving traditional relational databases behind. Graph databases easily outperform relational and other NoSQL databases in connecting masses of user and product data (connected data in general) to better understand customer needs and product trends. The biggest advantage of using a graph data model is that there is no need to connect entities within the data using special properties such as foreign keys. In a graph database, it becomes very easy to understand relationships between entities because the data structure is

well organized and very noticeable.

In this paper, we use Neo4j (Zhu et al., 2019), which is a graph database management system that also provides tools to visualize and extract important information from the graph database. This is done based on the Neo4j's graphical query language that allows users to store and retrieve data from the graphical database (Cypher).

Cypher (Francis et al., 2018) is not only the best way to interact with data and Neo4j. Cypher's syntax provides a visual and logical way to match patterns of nodes and relationships in the graph, which allows users to build expressive and efficient queries to handle the necessary creation, reading, updating, and deleting functionality. Indeed, SoftCentre[2] aims to develop a delivery recommender system using graph databases in order to optimize routes for each delivery to its destination.

The reminder of this paper is organized as follows: The second section presents a literature background on existing recommender systems, techniques, and algorithms. The third section presents our recommended system architecture with an illustrative example of its application. The fourth section summarizes the paper and introduces future work.

2. Related Work

We will devote this section to presenting how the literature defines some key concepts and notions related to our study. Thus, we will present how existing recommendation systems operate.

[2] organization that creates national champions in the software industry to improve Morocco's international attractiveness

2.1 Techniques for recommendation

Recommendation or recommender systems (Karimi et al., 2018) are algorithms that suggest relevant elements (movies to watch, texts to read, products to buy or any other element depending on the sector of activity) to users.

All recommendation algorithms are based on the following concepts:

These algorithms aim to find entities with similar properties and calculate their "similarity" measure (Prasetya et al., 2018). For example, customers who live in the same area, have the same age, or share common interests can be "grouped". This requires the analysis of customer choices. Indeed, it is possible to propose suitable recommendations by taking into consideration the executed activities of two similar clients. These two steps of the recommendation process depend on logical links between customers and between customers and their purchases. Therefore, the faster we can query and traverse these links, the stronger our ability to provide real-time recommendations is. In this context, there are many techniques used to make a recommendation system, which are illustrated in the diagram of Figure 1.

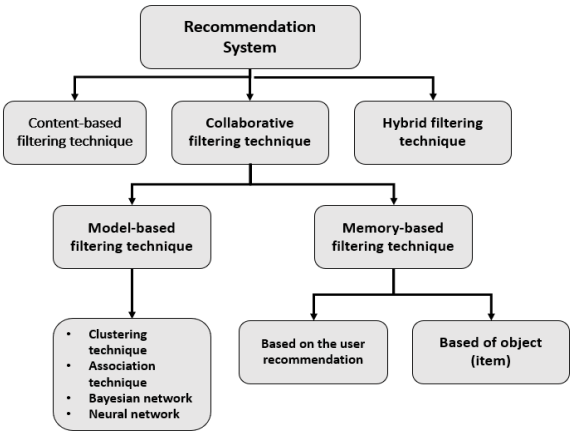

Fig. 1. A summary of existing approaches to recommendation systems

a. Content-based filtering technique

The content-based filtering technique (Bharti and Gupta, 2019) is used to analyze a set of items that have been evaluated previously by users in order to generate a prediction based on their evaluation.

To generate a complete sense of recommendation, different types of models will be used to find the similarity between documents, such as vector space models, probabilistic models, and neural networks. With this technique, we do not need a user profile, as it has been done using machine learning or statistical analysis.

The advantages of this technique are as follows: a) it provides privacy, b) there is no need to share the user's profile, c) recommendations are made in a short period of time, and it can recommend new items even if there is no rate on them.

On the other hand, it has disadvantages, namely: a) content-based filtering depends on the metadata of the item, and b) it is limited and requires a detailed description of the items.

b. Collaborative filtering technique

This technique (Nilashi et al., 2018) cannot be described easily using metadata because it is a domain-independent prediction technique. This technique is used to build a database between a user and an article as a performance table for articles written by users. Then it is required to calculate similarity by matching users with relevant articles.

The item recommended to the user depends on how similar users rate this article. In addition, this technique has two types: 1) model-based filtering techniques, which depends on clustering, association, Bayesian network, and neural network techniques; and 2) memory-based filtering technique like user-based and item-based techniques. It also has some advantages, such as the need for user evaluation to find similarity between them for establishing recommendations; it displays the recommendation items that an unknown user likes or evaluates; a new item can be suggested even if it has not been evaluated. On the other hand, this method has a disadvantage for new users, where recommendations will not be provided correctly, and items will not be recommended if there is no information to discriminate against.

c. Hybrid filtering technique

This technique (Parsian et al., 2017) is performed by combining several techniques to avoid the systems' limitations. Therefore, the result will be more accurate than a single algorithm. Each technique has weaknesses which can be overcome by combining them with another technique.

There are different ways to match this combination, such as implementing algorithms separately and then combining the results. Thus, we can use the content-based filtering technique (Bharti and Gupta, 2019) in the collaborative filtering approach and the collaborative filtering technique (Nilashi et al., 2018) in the content-based filtering approach.

2.2 Social media Recommendation systems

Currently, several recommendation systems are based on graph databases. 1) hybrid video recommendation system based on a graph-based algorithm (Öztürk, 2010); 2) book recommendation using Neo4j Graph Database in BibTeX Book Metadata (Dharmawan and Sarno, 2017); and 3) the Impact of the YouTube Recommendation System on Video Views (Zhou et al., 2010).

Firstly, the hybrid video recommendation system

(Öztürk, 2010) is based on a graph algorithm using a graphical algorithm called Adsorption. Adsorption (Raj et al., 2020) is a collaborative filtering algorithm in which logical links between users are used to make recommendations. Adsorption is used to generate the basic recommendation list. To overcome problems that have occurred in a collaborative system, content-based filtering is injected. Content-based filtering uses the idea of suggesting similar items that match the user's preferences, and the recommendation list is updated by removing weak recommendations. Then, similarities between items in the remaining list are calculated, and new items are inserted to form the final recommendation. Thus, recommendations are strengthened by considering the similarity algorithm between items. Therefore, the developed hybrid system combines both the collaborative approach and the content-based approach, to produce more effective suggestions.

Secondly, book recommendations Using Neo4j Graph Database in BibTeX Book Metadata (Dharmawan and Sarno, 2017) consists of processing the book metadata so information can be displayed to the user who needs a book recommendation. By combining BibTeX book metadata with Neo4j's graphical database, the data and metadata can be processed. Then, with an encrypted query, by entering the author or book type parameter, the user can get book recommendations based on their input criteria.

The result is exactly the same as with manual processing of metadata in the relational database. According to this article, it takes 180 milliseconds to run a cypher query with author criteria, and 184 milliseconds to run a query with book type criteria.

Last, the YouTube Recommendation System on Video Views (Zhou et al., 2010) presents a measurement study on datasets from YouTube. We find that related video recommendation, which recommend videos that are related to the video a user is currently watching, are one of the most important sources of video views. Despite the fact that YouTube video search is the top source of viewing in aggregation, related video recommendations is the top source of views for the majority of videos on YouTube (Lopezosa et al., 2020).

Furthermore, results reveal that there is a strong correlation between the number of views a video has and the average number of views of its most recommended videos. This implies that a video is more likely to become popular when it is placed on popular video recommendation lists. We also find that the click-through rate from a video to its related videos is high, and that the position of a video in a list of related videos plays a key role in the click-through rate. Indeed, the evaluation of the impact of the related video recommendation system on the diversity of video views indicates that the current recommendation system increases the diversity of video views in the aggregation.

3. Our Proposed Delivery Recommender system

In this section, we will detail our recommended system architecture, as well as the steps to follow in realizing it.

Our recommender system uses both collaborative filtering techniques and content-based recommendation (a hybrid recommender system). The collaborative part is where a set of algorithms will be used. The content-based technique is used to make recommendations in a short period of time, and it can recommend new items even if there is no rate on them. Therefore, our recommender system will be used to recommend the optimal path for a deliverer to achieve its order destination.

3.1 Framework

a. Overview

The system involves several actors, either directly or indirectly. These different actors are:

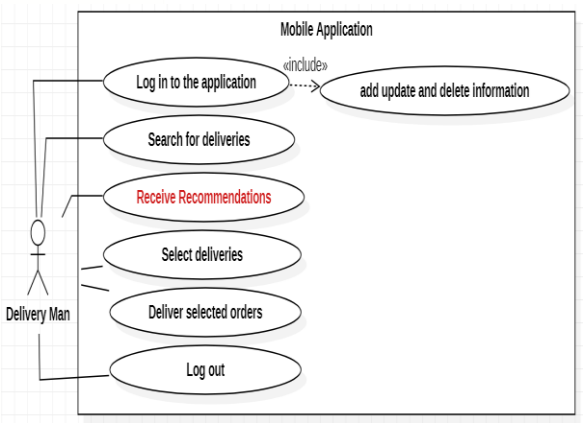

Fig. 2. Use Case

The administrator intervenes in an indirect way, by managing the system and by having the right to act through numerous actions, defined as follows: Log in and log out; Adding a user or an order (command), updating the system information, or deleting specific information. The user (the deliverer or delivery man): this is the main actor in the system. He interacts through the following actions: logging in, accessing his account, or logging out; He can update his profile information, search and view search results, as well as receive delivery recommendations based on his profile.

The diagram in Figure 2 models the use case of the mobile application combined with our recommended system for visualizing results.

b. Exemplar scenario

Figure 3 presents a scenario that is defined by a departure point and an arrival point. Throughout this scenario, participants react to the system through several actions. First, the driver must connect to the application according to his detected geographical position and the keywords of the search he makes. Afterwards, the system can propose deliveries that are in turn defined by numerous metrics, namely the size, the weight, the geographical points of pickup and destination, as well as the type of logical links between the deliverer and the customer. This is based on the rating that the deliverer gives to the customer. These static and dynamic parameters (in real time) are the core of our system, which affects the results of the recommendation (Figure 3 shows this recommendation process).

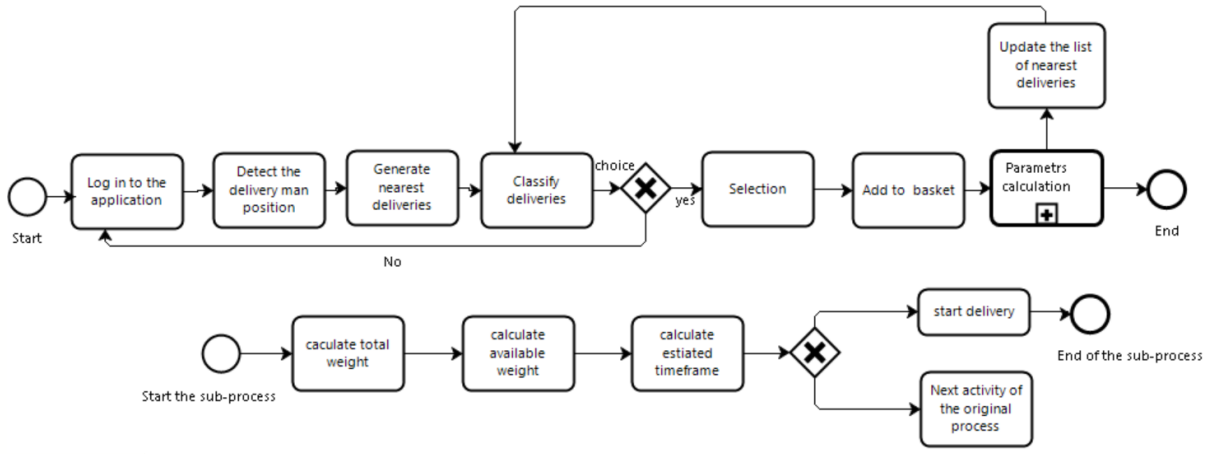

Fig. 3. Full Recommendation scenario with the parameters' calculation sub-process

3.2 Data collection

Our system is part of a project that develops a mobile application dedicated to independent delivery drivers. The delivery requests are published on the application and can be defined by data and metadata, which belong to the description of each request. This data is generated by the mobile application, and it will be considered the input data (output) of our system, on which the recommendation algorithms of our solution are based.

```
Commandes.head()
```

	IdCommande	Contenu	Catégorie	Nature	Taille	Date_de_livraison	Heure de Livraison	Org_lat	Org_lon	Des_lat	Dest_lon	ClientID
0	MVCV00000270	sandwitch, soda	Produits alimentaires	Fragile	Petite	2020-08-17	14:59:00	11.877200	7.063020e+01	12.7124	77.8125	ALLEXCHE450
1	VCV000142711	chat	Animeaux domestiques	Lourd	Moyenne	2020-08-27	16:21:00	12.786517	7.697522e+01	12.8319	79.9514	DMREXCHEUX1
2	VCV000143822	four	Electromenager	Léger	Petite	2020-08-27	17:57:00	30.000500	7.673221e+07	11.8711	79.7339	LUTGCCHE062
3	VCV000147433	vetements	Autres	Fragile	Grande	2020-08-28	00:42:00	13.087428	1.319509e+01	12.8239	79.9524	DMREXCHEUX3
4	VCV000147444	vélo	Autres	Lourd	Moyenne	2020-08-28	01:13:00	11.711540	7.781325e+01	11.8372	79.6322	LUTGCCHE064

```
Livreurs.head()
```

	IdLivreur	Curr_lat	Curr_lon	VéhiculeId	Disponibility	Mobility
0	VIJEXHOSR7	12.663500	78.649870	73fc7af87114b39712e6da79b0a377eb	1	I
1	VJLEXSHE09	12.836757	79.954428	a548910a1c6147796b98fdf73dbeba33	1	O
2	GSTEXLAK1Q	13.073956	80.225780	f9e4b658b201a9f2ecdecbb34bed034b	0	O
3	ARVEXNAM09	12.846686	79.950560	658677c97b385a9be170737859d3511b	1	I
4	SRTEXKOR96	12.429501	79.831556	8e6bfb81e283fa7e4f11123a3fb894f1	0	O

```
Clients.head()
```

	ClientID	Client_lat	Client_lon	Genre	Age	Note
0	ALLEXCHE450	12.7124	77.8125	Female	42	3.75
1	DMREXCHEUX1	12.8319	79.9514	Male	20	1.00
2	LUTGCCHE062	11.8711	79.7339	Male	40	2.25
3	DMREXCHEUX3	12.8239	79.9524	Male	70	2.75
4	LUTGCCHE064	11.8372	79.6322	Male	56	3.75

```
Vehicules.head()
```

	VéhiculeId	VehiculeType	VehiculeCapacity	IdLivreur
0	73fc7af87114b39712e6da79b0a377eb	Moto	20	VIJEXHOSR7
1	a548910a1c6147796b98fdf73dbeba33	Triporteur	125	VJLEXSHE09
2	f9e4b658b201a9f2ecdecbb34bed034b	Camion	27000	GSTEXLAK1Q
3	658677c97b385a9be170737859d3511b	Camionnette	1200	ARVEXNAM09
4	8e6bfb81e283fa7e4f11123a3fb894f1	Voiture	300	SRTEXKOR96

Fig. 4. Generated database (tables: orders, deliverers, customers, and vehicles)

In our case, the project is under development, which means that there is no data to be generated. Therefore, the production of false (true) data is required for the realization of the first step of our solution. To that end, there are many existing databases on the web that contain data that is compatible with our solution. Kaggle[3] is one of the platforms with rich and varied content in the field of data science, and on which there are different types of delivery request da-

[3] https://www.kaggle.com

tabases to generate useful data from this platform.

With the help of the Python packages, we can easily resolve several problems with simple code. In this step, we relied on the following two packages, Beautifulsoup and Faker, to generate the rest of the data, because we were not able to pull this data according to the first scenario.

There are numerous platforms to generate data, such as Mockaroo[2]. Mockaroo is a website that we have used to generate realistic but entirely fake data. It allows generating a wide variety of fake data from many domains and with high volumes. Also, it can be used to simulate the beta version of our API until the real APIs are completed. All these methods are explored for the data generation of a database that is made up of 4 tables in CSV format (see Figure 4). The first table contains information about orders (French: commandes), the second table depicts information about deliverers (French: livreurs), the third table contains information about customers (French: clients), and the fourth table contains information about vehicles (French: véhicules).

3.3 Data preparation

After database creation with specific metrics, we will process these metrics to yield logical results by removing noise.

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is prone to many errors. Data preprocessing is a proven method to solve these problems. To this end, we have based ourselves on four steps: data cleaning, data integration, data reduction, and data transformation (see Figure 6). For instance, the database contains several missing values (see Figure 5). This data can negatively influence the learning results. To solve this problem, we have to re-encode some of those missing values. To do this, we have used imputation, which means replacing the missing value with any number. In most cases, the imputed value will not be exactly accurate, but it usually gives more accurate models than dropping the column completely.

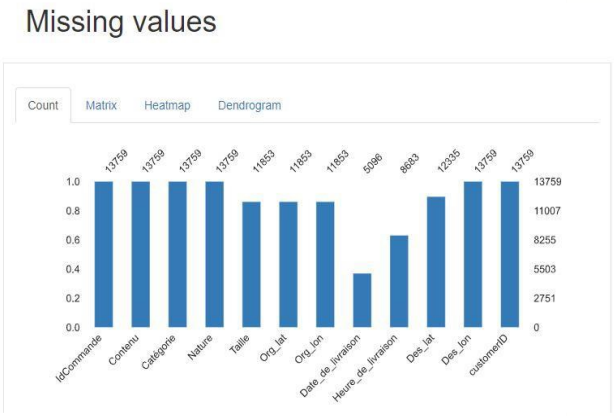

Fig. 5. Exploring missing values

3.4 Graph database

After processing the database and exporting it into new CSV files, we proceed to create the graph database, on which we will apply all the algorithms of the recommendation process. Then, we derive a graph model from a relational model, and we have to apply the following guidelines: a row is a node; a table name is a label name; and a join or foreign key is a logical link.

a. Nodes' creation

To create the nodes of our graph database, we begin by importing CSV files that contain the tables of the relational database. The output of each script gives the number of nodes created from the CSV file, the number of properties, and the execution time. e.g., 13759 order nodes, 6521 customer nodes, and 6880 deliverer nodes (see Figure 6).

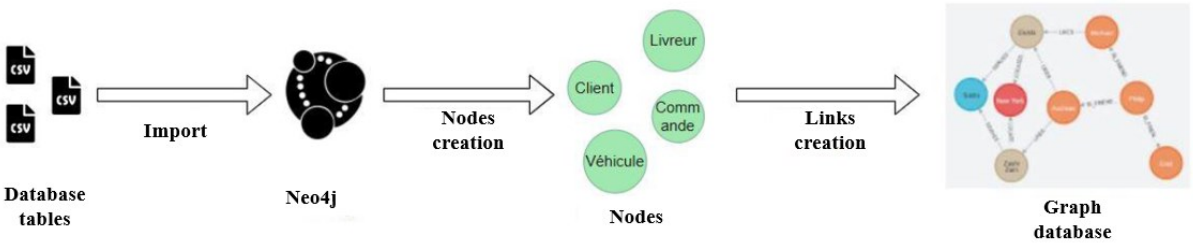

Fig. 6. The process of creating a graph database

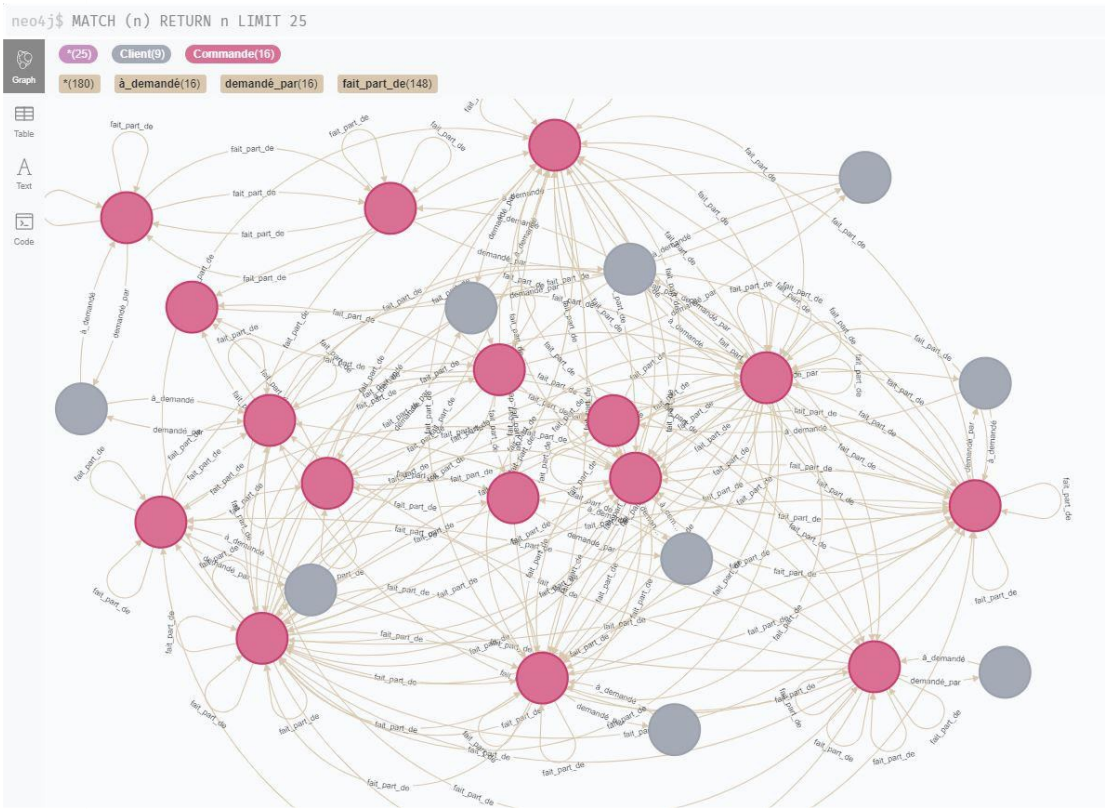

Fig. 7. Our graph database

b. Logical links and constraints creation

At this stage, we will create logical links between different nodes to obtain the graph of our database. To do this, we need to start by creating the indexes and the constraints on the properties that will be the links between the nodes. To successfully create the relationships, a constraint must be placed on the order identifier, which must be unique.

The goal of our study is to recommend a set of deliveries to a deliveryman, so the first relations are, on the one hand, those that link the orders with the customers, and on the other hand, those between the deliverers and the vehicles.

The graphical representation of the database (see Figure 7), based on the connections between the nodes, is faster when dealing with large amounts of data, so it is more efficient in the sense of searching for recommendations based on the relationships that link the different entities in the database.

3.5 Implementation

After creating the graph database, it is time to manipulate its data using the algorithms of the Neo4j Graph Data Science library (Needham and Hodler, 2019) as shown in Figure 8.

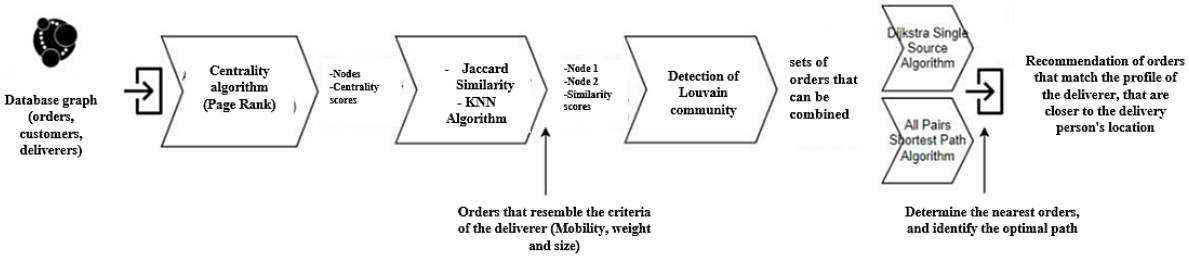

Fig. 8. The process of applying recommendation algorithms to the graph database

a. PageRank and Similarity Algorithm

The first step is to calculate the centrality of the order nodes using the PageRank algorithm (Xing and Ali, 2005). This algorithm calculates a score for each node and then displays them in order. This allows us to have the most important nodes in the graph that influence the rest of the algorithms. Then, we will calculate the similarity between a node of a given supplier and the properties of the orders using the node similarity algorithm based on Jaccard Similarity (Niwattanakul et al., 2013).

The configuration of this similarity considers the mobility and availability properties of the deliverer along with the mobility of the order. These algorithms allow the filtering of orders that do not match the profile of the deliverer, which will not be counted in the selection of recommendations.

b. Route Path optimization

To recommend an action based on route optimization, we need to select the orders closest to the location of the deliverer. The path taken by the delivery will be as shown in Figure 9. In order to optimize this path, we need to calculate the distance between the current position of the origin of the order as well as the distance between the starting point of the order and the order's point of departure and the point of arrival. To do this, we will use the Neo4j spatial library (Ashokkumar Arunkumar, et al. 2018).

First, we need to convert the geographic (see Figure 9) coordinates and create a point defined by the latitude and longitude. These parameters are used to calculate the distance between two geographical points, which are the origin of the order and its destination. The same calculation is applied to the distance between the delivery and the origin of the order.

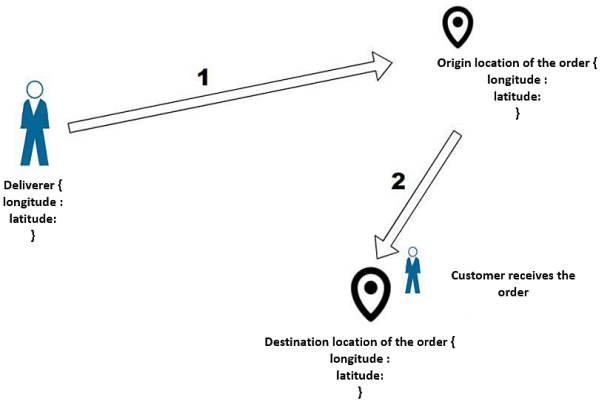

Fig. 9. Deliverer route

After calculating the distances, we can apply the shortest path algorithm (Gómez et al., 2019) to optimize the route path (ALL Pairs Shortest Path Algorithm). The result is illustrated as a ranking of the shortest paths between a pair of nodes using the total distance values between these aforementioned nodes. Figure 10 shows the result obtained by applying the shortest path algorithm to our database.

To define the closest orders to a given deliverer, we used the Dijkstra Single algorithm (Permana et al., 2018). This algorithm calculates the shortest paths between a source node, which is the delivery node in this case, and all the nodes reachable from this node, which are the orders. The application of this algorithm requires the use of a fixed starting node, so that the result can be represented as the distance between this node and all other nodes.

After selecting the closest orders to the delivery driver, we create a new group based on the nature and content of each order. At this point, the delivery driver can make multiple deliveries. To do that, we have used the community detection algorithm, the LOUVAIN algorithm (Zhang et al., 2021). This algorithm is used to nominate groups that can classify general orders and orders that belong to a specific group. The main idea consists of using these groups to provide the set of recommendations that will be displayed to the deliverer (see Figure 11).

①

②

```
CALL gds.nodeSimilarity.stream('pfeGraph')
YIELD node1, node2, similarity
RETURN gds.util.asNode(node1).name AS Commande, gds.util.asNode(node2).name AS Livreur, similarity
ORDER BY similarity DESCENDING, Commande, Livreur
```

③

```
LOAD CSV WITH HEADERS FROM "file:///Commande.csv" AS row
CREATE ( n : OrgLoc) SET n=row,
n.longitude =toFloat(row.Org_lon),
n.latitude =toFloat(row.Org_lat)

CALL spatial.geocodeOnce('geom_noeuds',{OrgLoc , DesLoc},0.1) YIELD location
WITH location, distance(OrgLoc,DesLoc) AS distance
RETURN OrgLoc.name, DesLoc.name, distance
ORDER BY distance
```

Fig. 10. Recommendation algorithms' application (Page rank ①, Similarity ②, Distance ③)

Fig. 11. Optimization algorithms' application (Shortest Paths ①, Dijkstra ② and Louvain③)

4. Conclusion

In this paper, a hybrid recommender system is presented. The system uses both collaborative filtering and content-based recommendation techniques. In this context, we use a set of recommendation and optimization algorithms. Therefore, our project revolves around the design, modeling, and implementation of a recommendation system based on the active filtering of orders published in the discussed mobile application.

To that end, we collected a large volume of data to create our relational database. Then, we converted the resultant database to a graph database using Neo4j libraries. Also, we applied a set of algorithms dedicated to data manipulation, which allowed us to generate recommendations for a given deliverer using Cypher and Python libraries. Furthermore, this helps the system personalize orders, contents, and deliveries by exploiting connections between data -all in real time- and matching deliverers with orders based on their profiles, preferences, and past online activities. In this context, we opt for a hybrid filtering approach that combines several algorithms and techniques. Therefore, the result is more accurate than a single algorithm.

In further work, it is important to treat the ability of process mining techniques to model uncertain behaviors of self-defined processes related to information retrieval systems. This is in line with the objective of achieving business process maturity and measuring how effectively and efficiently the self-defined BP is working.

References

Ashokkumar, P., Arunkumar, N., & Don, S. (2018). Intelligent optimal route recommendation among heterogeneous objects with keywords. Computers & Electrical Engineering 68, 526-535.

Bharti, R. & Gupta, D. (2019). Recommending top N movies using content-based filtering and collaborative filtering with hadoop and hive framework. In : Recent Developments in Machine Learning and Data Analytics. Springer, Singapore, 109-118.

Dharmawan, I. N. P. W., & Sarno, R. (2017). Book recommendation using Neo4j graph database in BibTeX book metadata. In 2017 3rd International Conference on Science in Information Technology (ICSITech), 47-52. IEEE.

Francis, N., Green, A., Guagliardo, P., et al. (2018). Cypher: An evolving query language for property graphs, in the Proceedings of the 2018 International Conference on Management of Data, 1433-1445.

Gómez, L. I., Kuijpers, B., & Vaisman, A.A. (2019). Analytical queries on semantic trajectories using graph databases. Transactions in GIS, 2, 5, 1078-1101.

Kamphuis, C. (2020). Graph databases for information retrieval. Advances in Information Retrieval, 12036, 608.

Karimi, M., Jannach, D., & Jugovac, M. (2018). News recommender systems–Survey and roads ahead. Information Processing & Management, 54, 6, 1203-1227.

Lopezosa, C., Enrique O., & Mario P. (2020). Making video news visible: Identifying the optimization strategies of the cybermedia on YouTube using web metrics. Journalism practice, 14, 4, 465-482

Needham, M. & Hodler, A.E. (2019). Graph Algorithms: Practical Examples in Apache Spark and Neo4j. O'Reilly Media.

Nilashi, M., Ibrahim, O., & Bagherifard, K. (2018). A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques. Expert Systems with Applications, 92, 507-520.

Niwattanakul, S., et al. (2013). Using of Jaccard coefficient for keywords similarity. Proceedings of the international multiconference of engineers and computer scientists, 1, 6.

Öztürk, G. (2010). A hybrid video recommendation system based on a graph-based algorithm. MS thesis.

Parsian, A., Ramezani, M., & Ghadimi, N. (2017). A hybrid neural network-gray wolf optimization algorithm for melanoma detection. Biomedical Research, 28, 8.

Permana, S.H., et al. (2018). Comparative analysis of pathfinding algorithms a*, dijkstra, and bfs on maze runner game. IJISTECH (International J. Inf. Syst. Technol, 1, 2, 1.

Prasetya, D.D., Wibawa, A., & Hirashima, T. (2018). The performance of text similarity algorithms. International Journal of Advances in Intelligent Informatics, 4, 1, 63-69.

Raj, J., Amirul, H., & Ashim, S. (2020). Various Methodologies for Micro-Video Recommendation System: A Survey.

Xing, W., & Ali, G. (2005). Weighted pagerank algorithm. Proceedings. Second Annual Conference on Communication Networks and Services Research, IEEE.

Zhang, J., et al. (2021). An Improved Louvain Algorithm for Community Detection. Mathematical Problems in Engineering 2021.

Zhou, R., Khemmarat, S. & Gao, L. (2010). The impact of YouTube recommendation system on video views. Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC. 404-410. 10.1145/1879141.1879193.

Zhu, Z., Zhou, X., & Shao, K. (2019). A novel approach based on Neo4j for multi-constrained flexible job shop scheduling problem. Computers & Industrial Engineering, 130, 671-686.