ISSN 1816-6075 (Print), 1818-0523 (Online)
Journal of System and Management Sciences
Vol. 12 (2022) No. 5, pp. 105-120
DOI:10.33168/JSMS.2022.0507

Content-based Recommender System with Descriptive Analytics

Su-Cheng Haw[1], Lit-Jie Chew[1], Kyle Ong[1], Kok-Why Ng[1], Palanichamy Naveen[1], Elham Abdulwahab Anaam[2]

[1] Faculty of Computing and Informatics, Multimedia University,63100 Cyberjaya, Malaysia

[2] Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 Bangi, Malaysia

sucheng@mmu.edu.my (Corresponding author)

Abstract. A recommendation system (RS) is an information filtering system that provides users with information, which one may be interested in. Ontology modelling has been widely used to conceptualize items and their semantic relationship together. Hence, in this paper, we propose an intelligent CB RS that allows users to not only access the product recommendations, but also the dashboard systems, which contain descriptive analytics, modeled using ontology. The dashboard allows users to have insight into past data. It consists of five main features: (i) Highlight Dashboard, (ii) Customer Dashboard, (iii) Advanced Search, (iv) Pivot Table and Pivot Chart, and (v) Report. Experimental evaluations show that the CB RS can return the accurate recommended product in a real propriety dataset.

Keywords: content-based, recommender system, recommender technique, ontology, e-commerce.

1. Introduction

Recommender Systems (RS) aim to capture the user's behaviour by suggesting or recommending to users with relevant items or services that they find interesting in (Sridevi et al., 2016; Sharma et al., 2021). In this digital age, there are an extremely huge amount of data and information available on the internet. Bancu et al. (2012) stated that a filtering mechanism is compulsory as the information on the internet keeps increasing; most users find that it is difficult and exhausting to find exactly what they want on the internet due to the overload of information. In addition, with personalized recommendations, user interface friction can be reduced, and users' experience can be improved, where users can easily find what they look for or are interested in. Therefore, current online services in different fields such as e-commerce, tourism and music streaming apply a recommender system to improve user experience in finding information online.

Descriptive analytics is about describing the historical performance. By applying different data mining techniques upon past and current data, several insights are able to be obtained from different hierarchical levels.

2. Literature Review

Since the inception of RS, these can be grouped into Collaborative Filtering (CF), Content-based (CB), Knowledge-based (KB) and Hybrid-based (HB) as elaborated in these sub-sections.

2.1. Collaborative filtering-based recommendation techniques

CF operates mainly based on information about users' behaviour and activities (Li et al., 2021). In another words, CF is the process of filtering or evaluating items using the opinions of other people. It tries to predict what users might like based on the behaviour of other similar users or the behaviour of users with similar tastes. CF faces certain limitation such as cold start, scalability and sparsity (Khojamli et al., 2021). A cold start happens because of a new item or new user. It is difficult to recommend items to a new user as there is very less information about the user or item. Scalability happens when there are millions of users and items, recommender system will face challenges in handling the information. Sparsity refers to the situation when many users do not participate in rating most of the items, thus the rating matrix is sparse and difficult to find similar users. Some other recent works combine the context-based or side-information to enhance the CB approaches (Abhijit et al., 2022; Javed et al., 2022, Le et al., 2022).

2.2. Content-based recommendation techniques

CB approaches calculate the similarity between items according to the similarities of their content. These algorithms are very sensitive to the quantity of available information about items. More available information will produce better results as the

similarities would be calculated more precisely. The problem with content-based methods is overspecialization which often results in recommending only items that are very similar to those that have been rated or seen by a user (Isinkaye et al., 2015). According to Beel et al. (2015), from the year 1998 until 2016, about 55% of the research papers touches on CB recommendation techniques.

2.3. Knowledge-based recommendation techniques

This group of recommendation techniques recommend items based on the domain knowledge (Sharma et al., 2016) that are based on the knowledge about the user, items and relation between the user and item (Sharma et al., 2016). To achieve this, a popular approach which uses the ontology to represent the semantic meaning and representation has been adopted (Sulthana et al., 2019; Li et al., 2018). Ontologies are a good modelling, conceptualization and visualization tool to model the user profile, item data, as well as the relationship in a hierarchical structure. This allows users to analyses the data at various abstraction levels. In addition, the ontology-based approach overcomes the problems such as rating sparsity, cold-start, and overspecialization. The domain ontologies are used to calculate the semantic similarity between items and users.

2.4. Hybrid-based recommendation techniques

Hybrid recommender techniques combine two or more techniques in generating the recommendation. It is used to overcome the issue caused by the specific technique to another such as cold start and over-specification (Sharma et al., 2021). A good combination will have better accuracy of the recommendation whereas a bad combination will cause the performance of the recommendation drops. None of the techniques is perfect as we can notice that each technique has their own advantages and disadvantages. Table 1 shows the advantages and disadvantages of each technique. None of the techniques is perfect as we can notice that each technique has their own advantages and disadvantages. Thus, choosing the correct techniques is important.

3. Methodology

The proposed system is mostly written in Python programming language. Various packages and libraries have been utilized during the development phase which includes flask, py2neo, scikit-learn and pandas.

Table 1: Recommendation method: advantages and disadvantages.

Method	Advantages	Disadvantages
Collaborative filtering-based	-No domain knowledge is required. -Better result when having a huge amount of data.	-Cold start problems. -Sparsity problems.
Content-Based	-Do not rely on the user profile. -Quality of the system improves over time. -No item cold-start problem.	-Overspecialization. -User cold start problems.
Hybrid-Based	-Overcome some of the issues cause by general techniques. -Performance is better.	-Need to have knowledge of using correct techniques for a particular domain
Knowledge-based	-No cold-start problem. -User ratings are not required	-Need for knowledge acquisition. -Static recommendations.

3.1. Dataset

The dataset is in Comma-Separated-Value (CSV) format. The dataset consists of 47 attributes and approximately 80,000 entries collected from January 2018 to May 2019 (18 months). In addition, three synthetic datasets were also created to support the recommender system module for more effective recommendations. The synthetic datasets created are user profiles, user transactions and item datasets.

3.2. Data pre-processing

Data pre-processing is a technique that transforms raw data into a compatible format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviour or transaction, and is likely to contain many errors. The purpose of data pre-processing is to resolve such issues on the dataset. Data that has been cleaned through the data pre-processing process will be stored in the database and utilized by all the modules - recommendation system, descriptive analytics and predictive analytics. Fig. 1 shows the data preprocessing flowchart. From Fig. 1, the data preprocessing process has several subprocesses, i.e., data transformation, stripping all strings within the curly bracket, spelling correction, alphabet upper-lowercase correction and data derivation.

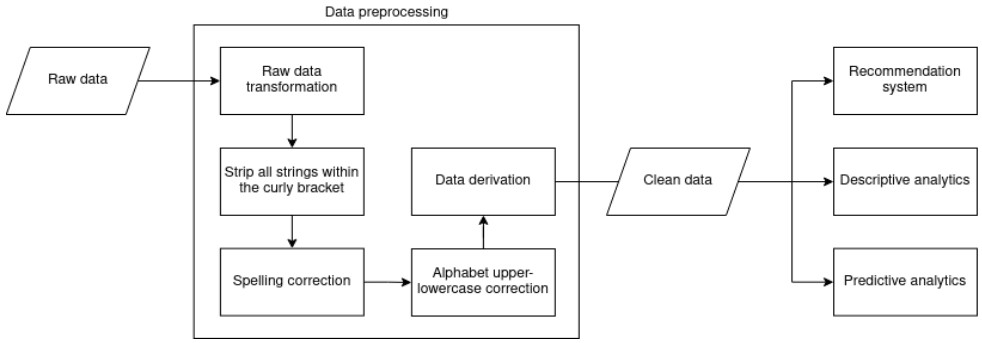


Fig. 1: Data preprocessing flowchart.

3.2.1. Data transformation

It was initially found that the insert_date and update_date attributes are in the format (month/day/year) to Universal DateTime acceptable format (year-month-day), for example, 1/13/2018 18:24:19 to 2018-1-13 18:24:19. Other data cleaning tasks involved are removing any columns that only contain one unique value or null value.

3.2.2. Strip all strings within brackets

Several attributes such as group and sub-group, contain values within the bracket. Several rows contain whitespace before or after some character, such as (C), (C), (C), and so on. Hence, it is important to strip — removing both the leading and the trailing whitespace. Fig. 2 shows the code snippets, while Table 2 shows the partial results.

```
# Strip within brackets. eg: Golf ( C ) -> Golf (C)
def strip_in_bracket(df_column):
    return df_column.apply(
        lambda x:
            re.sub("(\()\s+", "(", re.sub("\s+(\))", ")", x))
                if not pd.isna(x) else x
    )
```

Fig. 2: Code snippets for the stripping process.

Table 2: Partial result before and after stripping process.

Attribute	Before	After
group	Sports Goods (C), Swimming (C), Golf (C), Sport Apparel (C)	Sports Goods (C), Swimming (C), Golf (C), Sport Apparel (C)

3.2.3. Spelling correction

Fig. 3 depicts the code snippet for finding the potentially misspelt words. Spelling for each categorical value should be correct and consistent, as misspelt words can cause

additional unique value. For example, “availability“ and “availability“ in sub_category_2, should be identical but the machine may treat them differently. Table 3 depicts some samples of the before and after spelling correction.

```
spell = SpellChecker() # Python package
def find_misspells(df, col):
    misspells = list(
        filter(None,
        np.vectorize(lambda x:
        spell.unknown(re.findall(r"[\w']+|[.,!?;]", x)))
        (df[col].dropna().unique()))
    )
    return misspells
```

Fig. 3: Code snippets for finding potential misspelt words.

Table 3. Partial sample before and after spelling correction.

Attributes	Before	After
division	accesorries	accessories
sub_group	tradiotional	traditional
sub_category_2	availabilty, reuseable, validaity	availability, reusable, validity

3.2.4. Alphabet upper-lowercase correction

Besides, similar categorical values with different upper-lower cases may also lead to additional unique values. For example, “No Staff” and “No staff” in sub_category_3 should be identical but machines may treat them as unique. Our solution is to take the categorical value with the larger number of rows as the chosen value. For example, “No Staff” has 100 rows while “No staff” has 10 rows, since “No Staff” has a greater number of rows, “No staff” will be replaced with “No Staff”. Fig. 5 depicts the code snippets for upper-lowercase correction, while Table 4 depicts some examples of the attributes before and after the upper-lowercase correction.

```
def letter_case_correction(df, col):
    uniques = np.unique(np.vectorize(lambda x:
              str(x).lower())
              (df[col].dropna().unique()),
              return_counts = True)
    duplicates = list(uniques[0][uniques[1]>1])
    if duplicates:
        # Fix letter case
        for duplicate in duplicates:
            duplicate_value_counts = df[df[col].str.fullmatch(
                duplicate, case = False, na = False
            )][col].value_counts().to_dict()
            value = max(duplicate_value_counts)
            return df[col].apply(lambda x:value
                    if duplicate == str(x).lower() else x)
```

Fig. 5: Code snippets for upper-lowercase correction.

Table 4: Partial view of before and after spelling correction.

Attributes	Before	After
sub_category_3	No staff, No Staff	No staff
	Not available, Not Available	Not available
	Not given, Not Given	Not given
root_cause	High Usage by the customer, High usage by customer	High usage by customer
	SOP verification, SOP Verification	SOP verification

3.2.5. Data derivation

Two attributes have been derived from the dataset — resolution_time_second (resolution time in seconds) and extremity. The resolution_time_second can be derived by subtracting the insert_date from the update_date (see Table 5), while extremity can be derived from resolution_time_second based on certain conditions shown in Table 6. The resolution_time_second is used as the target variable for regression prediction, while extremity is used as the target variable for classification prediction.

Table 5: Examples of resolution_time_second.

insert_date	update_date	resolution_time_second
2019-05-01 10:22:15	2019-05-04 12:34:24	267129.0
2019-05-01 10:34:32	2019-06-03 18:22:59	2879307.0
2019-05-01 10:22:15	2019-05-04 12:34:24	267129.0
2018-12-29 16:31:25	2018-12-31 14:22:52	165087.0
2018-12-29 16:30:44	2018-12-29 16:44:24	820.0

Table 6: Extremity, conditions and number of rows.

extremity	resolution time in seconds	number of rows
0	0 to 172800 seconds (48 hours)	30164
1	172801 to 691200 seconds (8 days)	20612
2	6912001 seconds above (8 days and above)	5765

3.3. Ontology modelling

Ontology helps users to have a better understanding of the data structure as the hierarchical structure of the ontology allows users to analyze the data at different abstraction levels. The ontology is constructed by understanding the data and then making the relationship between each attribute. The relationship between each attribute can be visualized and it helps in the CB recommender system while calculating the similarity between each node. Fig. 6 shows the sample ontology which was extracted from the system. This ontology tells us that a user named Hairul, has made a request to create a service request with id 64769, and the request belongs to the facilities category.

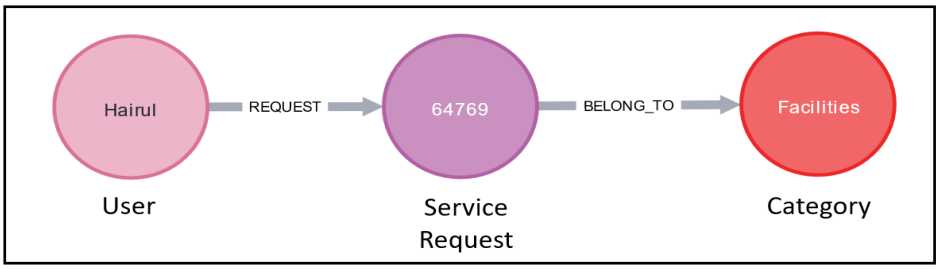


Fig. 6: Sample ontology extracted from the system.

Fig. 7 above is the ontology created in this system. As shown in the figure, each node represents an attribute, and each node is connected with the relationship between each other. This ontology is then stored in our graph database, in which the ontology structure is maintained in the database.

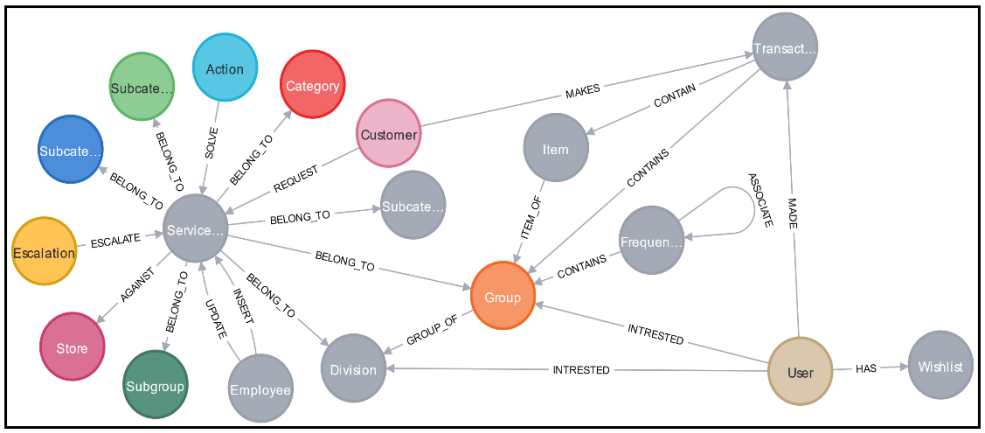


Fig. 7: Ontology constructed in this system.

3.4. CB method

Our CB method uses the user preferences as a reference, then calculate the similarity between user and items. By using the graph database, we can get similar item nodes by calculating the node similarity between user and item nodes as they are connected via edges, which is the relationship between each other. Besides user preferences, we link the user preferences with the service request they created. For example, if the user has requested an enquiry about Home Fashion, we will also include the Home Fashion item in the recommendation list. The similarity between users and items are calculated on-demand, thus the results are real-time results. Fig. 8 shows the CB method architecture.

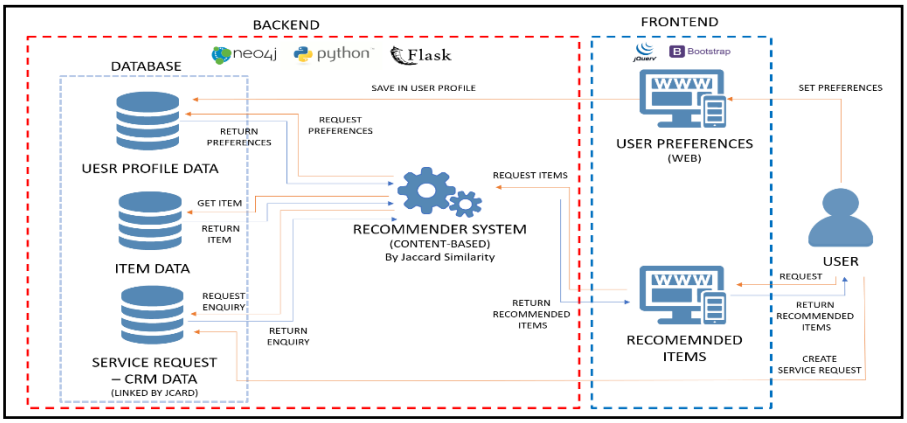


Fig. 8: CB recommendation method architecture.

3.5. Descriptive analytics

A descriptive analytics admin dashboard has been developed for users. It provides statistical analysis of the data and allows users to have a better understanding of the historical data and the changes within some specific period. All the data for the dashboard is pulled from the database thus it is a real-time dashboard. The backend of the system is using python and flask, which connect to the neo4j graph database. The frontend is developed using jQuery, Bootstrap and Bokeh. It consists of 5 features: (1) Highlight Dashboard; (2) Customer Dashboard; (3) Advanced Search; (4) Pivot Table; and (5) Report.

3.5.1. Highlight dashboard

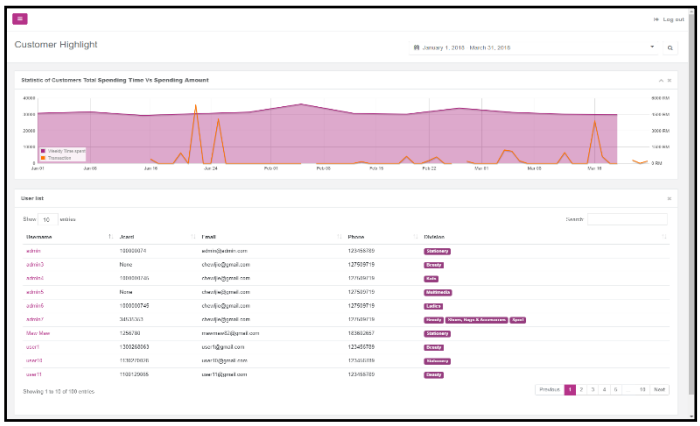
In this part, the user can view and select the desired period for the charts on the Highlight dashboard (see Fig. 9). Each chart is interacting with each other and acts as a filter for the dashboard. This enables the user to drill down the data by selecting multiple filters. There are multiple pages in this dashboard, as the original dataset needs to be grouped into different categories first then only can drill down to a specific level. Most of the graph is generated by the backend using the bokeh library. Bokeh is an interactive visualization library based on python, which enables users to create interactive dashboards easily. For some graphs such as the time series graph which the bokeh library has no applicable graph that suits our needs, we will use frontend javascript to generate the graph and the data is served by API.

Fig. 9: Highlight dashboard.

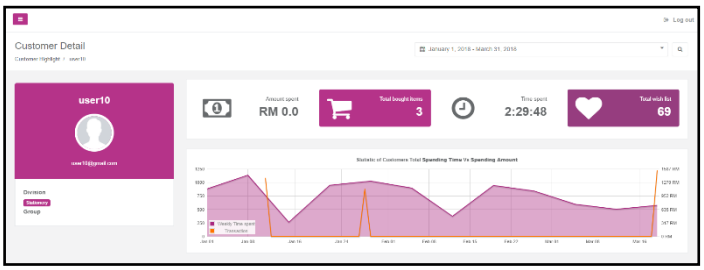
3.5.2. Customer dashboard

In the customer dashboard, users can get information on customers' total time spent on the e-commerce website and the total spending amount in the chart (see Fig. 10(a)).

Users can get insight that when a user will buy the products after browsing through the website. This dashboard is generated by using the synthetic data we created as we do not have the exact user profile data. Besides the highlight pages, we can also get the customer's individual information such as total amount and time spent on the website, total buying items and so on on the individual customer page (see Fig. 10(b)).



(a)



(b)

Fig. 10: (a) Customer highlights dashboard (b) Individual customer detail.

3.5.3. Advanced search

On the advance search page (see Fig. 11), the user can search through and apply multiple filters to the data and display the data as a table. Users are allowed to select the column that they want to show, and the column that they want to use as a filter. The generated table can be downloaded as either PDF or Excel. The data for this table is served by our API and pre-compressed. The pre-compress action makes the transferred data smaller and the download time for the end-user is reduced.

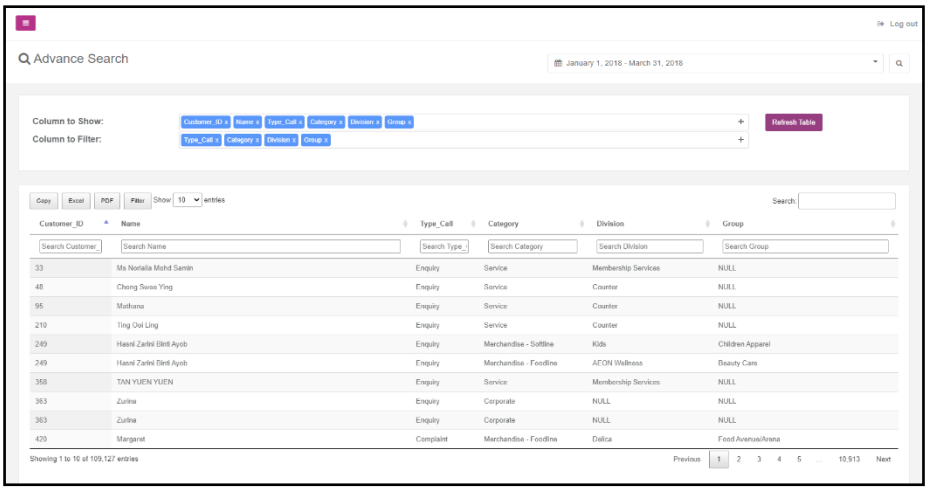
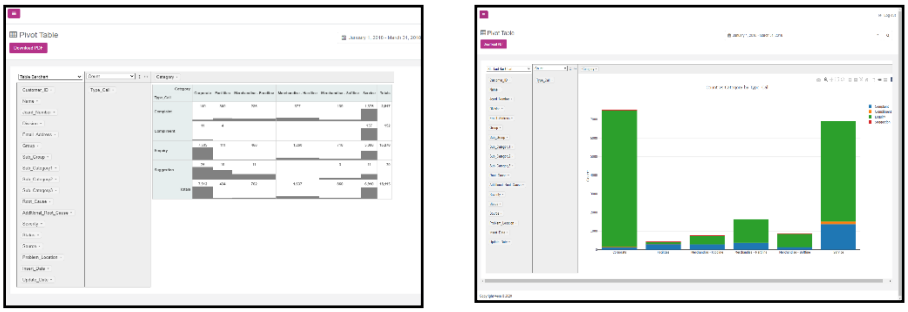


Fig. 11: Advanced search.

3.5.4. Pivot table and pivot chart

A pivot table is a table of statistics that summarizes the data by sums, averages, or other methods that make the data have a meaningful summary. With a pivot table, users can summarize the data up to certain conditions that might not be included in the highlight dashboard mentioned before. The pivot table we implemented here enables the user to drag and drop the data column to form the pivot table (see Fig. 12(a)). Besides forming the table, we also enable users to generate the chart by using the data summarized by the pivot table (see Fig. 12(b)). All the charts and tables generated in this pivot table can be exported to PDF.



(a) (b)

Fig. 12: Chart visualization: (a) pivot table (b) pivot chart.

3.5.5. Report

The report function here enables users to export the data to get the summarize of the data within the desired period. It has several predefined report layouts that provide different insights into the data. Users can select their desired period and report layout

then download it as a PDF. All the PDFs are generated at the backend of the system and then served to the user. Besides downloading as a PDF, the email feature has been integrated inside this module as depicted in Fig. 13. With the email feature integrated, a selected group of people can receive the report periodically.

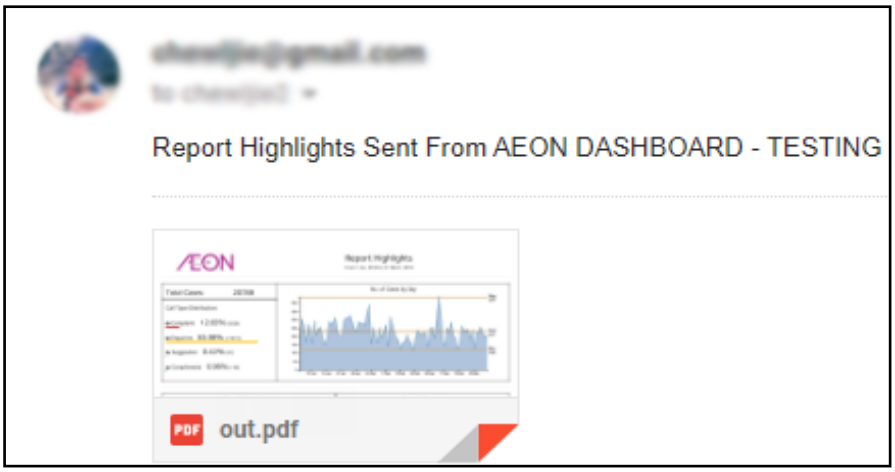


Fig. 13: Report email sent by email notification.

4. Evaluation and discussion

The similarity algorithm used in the CB method is Jaccard Similarity Algorithm. It can be used to measure similarities between attributes. This algorithm is defined as the size of the intersection divided by the size of the union of two sets of data (see E1).

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad \text{(E1)}$$

where A and B indicate a different set of data, A ∩ B is the intersection of two sets of data, and A ∪ B is the union of two sets of data. The higher the Jaccard similarity, the more similar the data.

Fig. 14 shows the Jaccard similarity calculation result from the database. One of the benefits of the graph database we use is that it can calculate the similarity result in the query. From the result above, the s1 is the user preference, while the s2 items category, intersection count is the count of interaction between user preference and item category and the size is the total union count of the user preference and item category.

Fig. 15 is the CB recommendation result for a particular user logged in with Jcard number 1000000745, who has requested an enquiry service request in home fashion, and user preferences have been set to the grocery. We can see that from the recommendation list displayed to the user, both categories of items have been

suggested to the user. This suggestion list is located on the home page and user profile page.

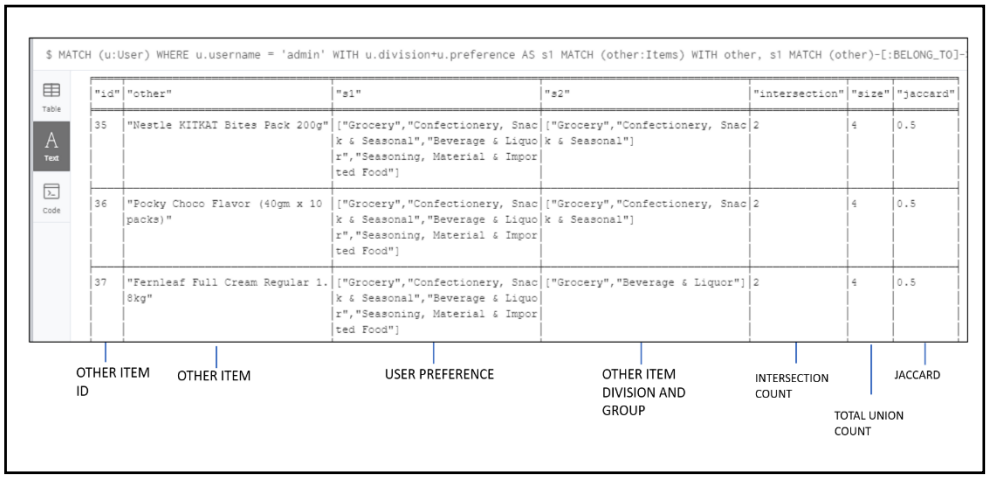


Fig. 14: Jaccard similarity calculation result in database.

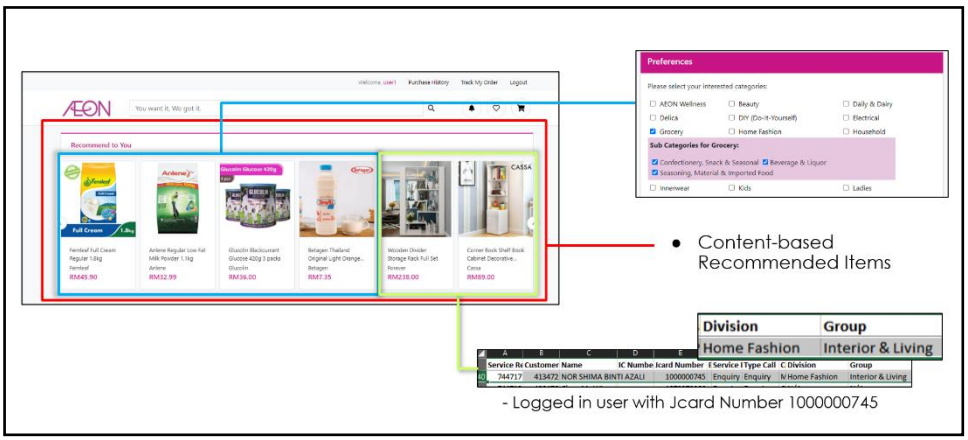


Fig. 15: Jaccard similarity calculation result in database.

5. Conclusion and future work

In this paper, we proposed and developed an intelligent CB recommender system that allows users to not only access the product recommendations but also the dashboard systems. The descriptive analytics dashboard allows users to have insight into past data. It consists of 5 features: (1) Highlight Dashboard; (2) Customer Dashboard; (3) Advanced Search; (4) Pivot Table; and (5) Report. These five features help a company to drill down the past data from different degrees of view, but also enable them to summarize the data in the way they would like to. They could generate the summarized table by just drag and drop the column they desire instead of just getting

the insight from the graph we created. The periodic email function in reports also helps the company to get periodic insight from the data.

In future work, we intend to extend the dashboard to include more assorted recommendations by using a hybrid approach. In addition, the dashboard will be extended by incorporating some prediction models.

References

Abhijit, A., Biswanath, D. & Animesh, D. (2022), Finding most informative common ancestor in cross-ontological semantic similarity assessment: An intrinsic information content-based approach. *Expert Systems with Applications*, 192. DOI:https://doi.org/10.1016/j.eswa.2021.116281.

Arafeh, M., Ceravolo, P., Mourad, A., Damiani, E., & Bellini, E. (2021). Ontology based recommender system using social network data. *Future Generation Computer Systems*, 115, 769-779. DOI: https://doi.org/10.1016/j.tele.2017.08.008.

Bagherifard, K., Rahmani, M., Nilashi, M. & Rafe, V. (2017). Performance improvement for recommender systems using ontology. *Telematics and Informatics*. 34(8), 1772-1792. DOI: https://doi.org/10.1016/j.tele.2017.08.008.

Bancu, C., Dagadita, M., Dascalu, M., Dobre, C., Trausan-Matu, S., Florea, A.M. (2012). ARSYS--article recommender system. *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 349-355.

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2015) .Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4), 305-338. DOI: http://doi.org/10.1007/s00799-015-0156-0.

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyption Informatics Journal*, 16(3), 261-273.

Javed, U., Shaukat, K.A., Hameed, I., Iqbal, F., Mahboob Alam, T., & Luo, S. (2002). A review of content-based and context-based recommendation systems. *International Journal of Emerging Technologies in Learning*, 16(3), 274-306. Retrieved June 1, 2022 from https://www.learntechlib.org/p/219036/.

Khojamli, H. & Razmara, J. (2021), Survey of similarity functions on neighborhood-based collaborative filtering. *Expert Systems with Applications*, 185, 1-28. DOI: https://doi.org/10.1016/j.eswa.2021.115482.

Le, N. L., Abel, M. H., & Gouspillou, P. (2022). Towards an ontology-based recommender system for the vehicle sales area, lecture notes in networks and systems, 441, Springer. DOI: https://doi.org/10.1007/978-3-030-98531-8_13.

Li, M., Wen, L., Chen, F. (2021). A novel collaborative filtering recommendation approach based on soft co-clustering. *Physica A: Statistical Mechanics and its Applications*, 561.

Li, Y., Lin, L., & Ho, C. (2017), A social route recommender mechanism for store shopping support. *Decision Support Systems*, 94, 97-108. DOi: http://doi.org/10.1016/j.dss.2016.11.004.

Sharma, M., Mittal, R., Bharati, A., Saxena, D., & Singh, A.K. (2021). A survey and classification on recommendation systems .*International Conference on Big Data, Machine Learning and Applications*, 1-17.

Sharma, R. & Singh, R. (2016). Evolution of recommender systems from ancient times to modern era: A survey. *Indian Journal of Science and Technology*, 9(20), 20. DOI: http://doi.org/10.17485/ijst/2016/v9i20/88005.

Sridevi, M., Rao, R. & Rao, M. (2016). A survey on recommender system. *International Journal of Computer Science and Information Security*, 14(5), 265-272.

Sulthana, A.R. & Ramasamy, S. (2019). Ontology and context based recommendation system using neuro-fuzzy classification. *Computers & Electrical Engineering*, 74, 498-510. DOI: http://doi.org/10.1016/j.compeleceng.2018.01.034.