Content-based recommender system architecture for similar e-commerce products

Ari Nurcahya [a,1], Supriyanto [a,2,*]

[a] Department of Informatics, Faculty of Industrial Technology, Universitas Ahmad Dahlan, Indonesia
[1] *nurcahyaari@gmail.com* [*]; [2] *supriyanto@tif.uad.ac.id*
* *corresponding author*

ABSTRACT

Recommendation systems are quite famous and are increasingly being used on e-commerce platforms for a variety of purposes. The recommendation system technique used also varies greatly depending on the scope and Item of recommendation. Content-based filtering, for example, is used to recommend related product items based on user preferences. However, how the recommendation system architecture should be built starts by creating a data model for bringing up related product items. This paper offers a system architecture by considering the initial problem usually faced by recommendation systems, namely the cold start problem. The problem of lack of user preferences data is trying to be overcome by utilizing product item documents. Product item documents are processed using the TF-IDF algorithm and Vector Space Model to generate a data model. Then a query can be applied to find similarities to items that the user has seen. In the end, the recommendation system architecture that was built produced excellent Precision using Recall and Precision testing. Tests are carried out for data using the weighting of product names and product labels. The result obtained 0.84 for the average value of Recall and 0.78 for the average value of Precision.

Keywords:
Recommender system
Content-based filtering
Vector-Space Model
TF-IDF

I. Introduction

E-commerce is not only used by large traders but is also used by traders who are pioneering efforts. The types of products also vary, ranging from building materials to food materials such as agricultural products [1]. Growing with more and more platforms [2]. E-commerce is overgrowing, and a study predicts that there will be an increase in the e-commerce market amounted to 47.5% in 2022 compared to 2019 [3]. Many products sold through e-commerce are a challenge for sellers to offer their products. Products with the same name and type may number in the dozens in a single e-commerce platform. E-commerce platforms should provide convenience to the user's relevant product search [4][5]. Moreover, with the increasing number of products sold, buyers will take a long time to evaluate products purchased by the user [6].

One of the conveniences provided is by providing other product recommendations based on products that have been searched by users [7]. The recommendation system has been widely applied in various companies, such as Amazon, Netflix, and Tokopedia, to support business[8]. Netflix uses the recommendation system for various purposes such as top movies, trending, movie similarity, and search [9][10]. In the tourism sector, the recommendation system is also used to make it easier for prospective tourists to determine their tour trips [11]. The utilization of a recommendation system in e-commerce produces several benefits: increased sales, increased user loyalty, and others [12]. The recommendation system can make it easier for users to find, get, and determine products online [13].

The recommendation system has several methods that can be used in its design, namely, Content-Based Filtering, Collaborative Filtering, and Hybrid[14]. Content-Based Filtering (CBF) is a recommendation that uses the content attribute of an item, which is then used to determine the similarity between items in determining recommendations [15][16][17]. Collaborative filtering is a recommendation system whose attributes are not from the content of an item or product but the similarity or relationship of user data; there are two categories of Collaborative Filtering: User-Based and Item Based [18]. In User-Based, the approach is to see the similarities between one user and another. In Item Based, the approach is seen from the similarity between items that interact or are

assessed by users. The third approach is Hybrid which combines the recommendation system's filtering method [19].

This paper discusses the recommender system architecture in e-commerce to determine which products are related and similar to produce items that have been searched for and seen by users. The aim is to determine the appropriate system architecture and mechanism for implementing a recommendation system on an e-commerce platform. This paper uses an example of an e-commerce platform that is made simple using a data set taken from one of the platforms. Data were taken only general data of a document item product named the product name and product description [20].

The recommendation system architecture that is made will only use content-based filtering techniques. It is based on the possible problems that the recommendation system will inevitably face when only the product documents are available [21]. This problem is commonly called the cold start problem, making content-based filtering unable to run correctly because user preferences have not been obtained [22].

The cold start problem can be solved with a simple mechanism: to use log data of user activities when searching and finally see the Item [23]. This data can be used as a simple preference so that its similarity can be found with the product item model data. The recommendation system architecture built includes how the model data is stored and how it can be used as a source to find similarities between items with an adequate Precision level.

II. Method

This paper proposes a content-based recommendation system architecture to produce similar product items and according to e-commerce platform users' preferences. The text mining methodology [24] is shown in Fig. 1, namely data collection, data preprocessing, weighting using TF-IDF, forming data using a vector space model, and checking similarities with cosine similarity[25]. The resulting data model is stored in JSON form, and item similarity checks are made in Restful API.

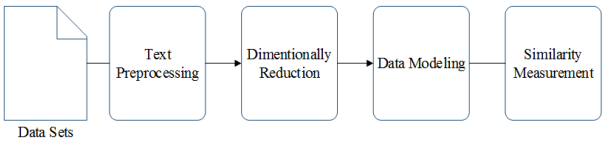


Fig. 1.Methodology process.

Starting the development process, previously prepared an e-commerce platform consisting of 2 pages. The first page is the front of the e-commerce platform, and the second is the product detail page. The system scenario that is created assumes that the user explores the system by seeing the products on the front page, continuing to the product detail page. The product item data used is sample data taken from an e-commerce site in Indonesia. The data taken are product names, product prices, pictures, and product descriptions.

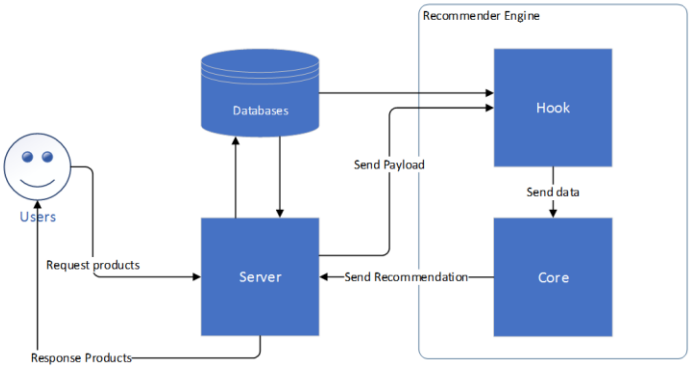


Fig. 2.Recommendation system architecture.

Fig. 2 shows an outline of the workflow system recommendation on the product detail page starting from the system's user browse products. The system will send a product request that the user sees. The

product id will then be sent to the Recommender Engine, after which the product id will be stored in the database. The Recommender Engine will calculate the similarity between the product data and other products from the database. Recommender Engine contains two parts of Hook and Core. Hook section in charge of receiving data from the database and receive payload Daris server. Data were then submitted to the cores to do the stages of the process of determining a recommendation. Core performs the Text preprocessing process followed by weighting TF-IDF. The data is converted into a vector space model to calculate the value of its similarity to the Cosine Similarity, having obtained the weighting value. Responses are in the form of products that users see and other product recommendations. When the user returns to the home page, the system will display the recommendations obtained from previous activities on the product detail page.

When users search for a product, the server will accept the request, and the server will search for the product the user is looking for from the database. After searching for data from the database, the server receives the data and displays it to users. The server will send the user's search data to the Recommender Engine, which Hook will capture data. The data in the form of user id and product id being searched will be stored in the database, and the Recommender Engine will retrieve data from the database to process the similarity of the data sought with the data in the database with the TF-IDF algorithm and the Vector Space Model. Then the results of these recommendations will be sent to users. So every user sees a product, a list of other products similar to the product being viewed will be displayed. Moreover, when users see the feed menu, recommendations will appear based on products that have been seen before.

System testing conducted in this study aims to determine how the system's suitability of products is recommended. The system testing is done by calculating the recall and Precision [26] values of several randomly selected product samples then calculating the average value obtained. The test system is based on three schemes testing systems. In the first test, the document attributes used for the recommendation are the product name and description. In the second test, the document attribute used for recommendation is the name of the product. In the third test, the document attribute used for recommendation is the product name with the product label.

III. Results and Discussion

A. Datasets Collection Results

The data taken is data on the type of foodstuffs, agricultural products, or livestock products, such as vegetables, fruits, spices, and the like. It has been obtained approximately 868 data as data sets. Furthermore, each data is labeled according to its category. Table 1 shows the 20 labels of datasets obtained. The label distinguishes specific foods such as vegetables distinguished from the shape, parts, and plant species.

Table 1. Datasets

Label	Total
sayuran daun	81
Garnish	54
sayuran buah	51
lauk nabati	131
sayuran akar	37
sayuran jamur	23
sayuran bunga	14
Seafood	8
sayuran polong	13
sayuran umbi	30
bumbu masak	126
rempah daun	6
Lalapan	3
Buah	137
rempah akar	22
lauk hewani	4
Daging	14
rempah biji	48
Kacang	61
rempah batang	5

Information:
- Total: is the total document in the database record
- Label: is the document label

B. Text Preprocessing

The preprocessing stage aims to clean the product name data and product descriptions from unnecessary words [27]. In-text preprocessing four stages will be followed by the system. The first stage is case folding which aims to convert all Text into a standard format, where the text format is used in lowercase. So that all Text that has a capital format will be changed first to lowercase. Changes in original data and case folding results are shown in Fig. 3. At this stage, the toLowerCase() function is used, which is a JavaScript string manipulation.

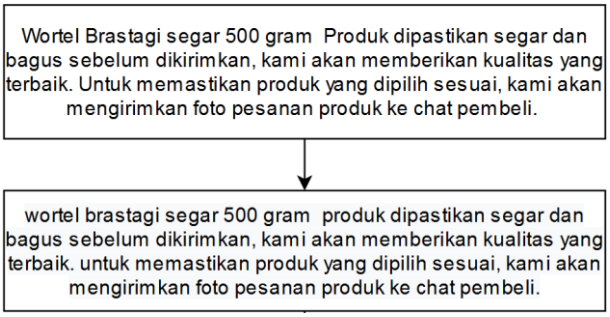


Fig. 3.Text preprocessing example preprocessing original data-case folding.

The second stage is tokenization, which functions to convert documents into tokens or words. At this stage, use the WordTokenize() function in the Natural.js library. The change in form from the tokenization process can be seen in Fig. 4, which shows that the description sentence has changed to 1 separate word from each other.

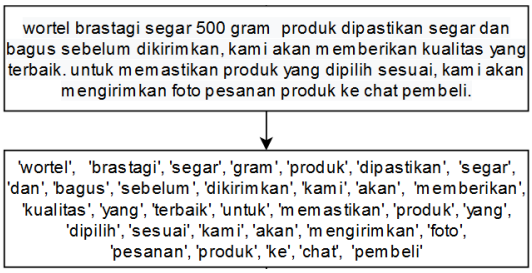


Fig. 4.Text preprocessing example preprocessing case folding-okenization.

The third stage is stopword removal which aims to remove meaningless words. At this stage, use the stopword list datasets obtained from Kaggle. Then the dataset will be matched with words in the document. If the word is on the stopword list, it will be deleted. The last step is stemming, which aims to remove the affixes to words to produce only the root words. At this stage, the literary algorithm[28] is used for Indonesian Text and Porter's algorithm[29] for English Text. The final results of this preprocessing stage are shown in the last part of Fig. 5.

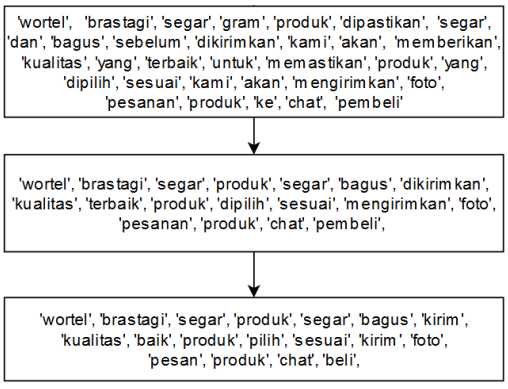


Fig. 5.Text preprocessing example tokenization-stopword removal-stemming.

C. TF-IDF process

The TF-IDF process was carried out to count and determine how important these words were in the data set[30]. TF-IDF starts from calculating each Term in the document (TF), wherein this process bag-of-words will be generated. Then the process continues by counting the number of documents that have a specific term (DF). After that, calculating the Inverse Document Frequency (IDF) and finally, the TF value is multiplied by the IDF. Document weighting is done using the TfIdf () function in the Natural.js library.

An example of calculating TF-IDF will be explained in the following steps. Suppose there are four documents, namely:
Document 1: "Bayam Segar,Bayam segar /ikat +-300gr Note : Tersedia bumbu dan sayuran packing"
Document 2: "Daun Bawang Segarikat,Daun bawang segar fresh.. harga untuk 1ikat.kurleb 70gr"
Document 3: "Tempe Per 1 Papan,Tempe per 1 Papan"
Document 4: "Wortel Brastagi segar 500 gram Produk dipastikan segar dan bagus sebelum dikirimkan"

The four documents will be calculated TF-IDF, starting from the Text preprocessing and then calculating the TF-IDF.

Step 1 is to calculate the frequency of the Term's appearance on each existing document (TF). Then the results will be like Table 2.

Table 2. TF Results

Token	tf				
	Q	**D1**	**D2**	**D3**	**D4**
bayam	2	2	0	0	0
segar	2	2	1	0	2
note	1	1	0	0	0
sedia	1	1	0	0	0
bumbu	1	1	0	0	0
sayur	1	1	0	0	0
pack	1	1	0	0	0
ikat	1	1	0	0	0
daun	0	0	0	0	0
bawang	0	0	1	0	0
Segarikat	0	0	2	0	0
Fresh	0	0	2	0	0
harga	0	0	1	0	0
kurleb	0	0	1	0	0
tempe	0	0	0	1	0
papan	0	0	0	1	0
wortel	0	0	0	0	1
brastagi	0	0	0	0	1
produk	0	0	0	0	1
bagus	0	0	0	0	1
kirim	0	0	0	0	1

Result infromation:
- bayam (D1) in document 1 is 2 words
- segar (D1) in document 1 is 2 words
- note (D1) in document 1 is 1 word
- sedia (D1) in document 1 is 1 word
- bumbu (D1) in document 1 is 1 word

Step 2, namely calculating df, df is the number of documents that contain specific words. The df calculation is based on Table 2. The results of the df calculation are shown in Table 3.

Table 3. DF Results

Token	tf					df
	Q	***D1***	***D2***	***D3***	***D4***	
bayam	2	2	0	0	0	1
segar	2	2	1	0	2	3
note	1	1	0	0	0	1

Token	tf					df
	Q	*D1*	*D2*	*D3*	*D4*	
sedia	1	1	0	0	0	1
bumbu	1	1	0	0	0	1
sayur	1	1	0	0	0	1
pack	1	1	0	0	0	1
ikat	1	1	1	0	0	2
daun	0	0	2	0	0	1
bawang	0	0	2	0	0	1
segarikat	0	0	2	0	0	1
fresh	0	0	1	0	0	1
harga	0	0	1	0	0	1
kurleb	0	0	1	0	0	1
tempe	0	0	0	1	0	1
papan	0	0	0	1	0	1
wortel	0	0	0	0	1	1
brastagi	0	0	0	0	1	1
produk	0	0	0	0	1	1
bagus	0	0	0	0	1	1
kirim	0	0	0	0	1	1

Result information:
- Df "bayam" has a value of 1 because the number of words "bayam" in the four documents is only 1, namely in document 1.
- Df "segar" has a value of 3 because the number of "segar" words in the four documents is 3, namely in documents 1,2, and 4.
- Df "note" has a value of 1 because the number of words "note" in the four documents is only 1, namely document 1.
- Df "sedia" has a value of 1 because the number of words "sedia" in the four documents is only 1, namely document 1.

Step 3 is to calculate the IDF. The equation used is as (1)

$$1 + ln(\frac{N}{1+df}) \quad (1)$$

Ln: Natural logarithm. Where this logarithm is based on e, which is Euler's constant. The quantity of Euler's constant is e = 2.718281828459
N: Number of documents
Df: Document Frequency, or the number of occurrences of Term in the document
The results of the IDF calculation are shown in Table 4.

Table 4. IDF Results

Token	tf					df	idf
	Q	*D1*	*D2*	*D3*	*D4*		
bayam	2	2	0	0	0	1	1.693147
segar	2	2	1	0	2	3	1
note	1	1	0	0	0	1	1.693147
sedia	1	1	0	0	0	1	1.693147
bumbu	1	1	0	0	0	1	1.693147
sayur	1	1	0	0	0	1	1.693147
pack	1	1	0	0	0	1	1.693147
ikat	1	1	1	0	0	2	1.287682
daun	0	0	2	0	0	1	1.693147
bawang	0	0	2	0	0	1	1.693147
segarikat	0	0	2	0	0	1	1.693147
fresh	0	0	1	0	0	1	1.693147
harga	0	0	1	0	0	1	1.693147
kurleb	0	0	1	0	0	1	1.693147
tempe	0	0	0	1	0	1	1.693147
papan	0	0	0	1	0	1	1.693147
wortel	0	0	0	0	1	1	1.693147
brastagi	0	0	0	0	1	1	1.693147
produk	0	0	0	0	1	1	1.693147
bagus	0	0	0	0	1	1	1.693147

Token	tf					df	idf
	Q	*D1*	*D2*	*D3*	*D4*		
kirim	0	0	0	0	1	1	1.693147

Calculation:
- bayam (D1): 1 + ln(4/(1 + 1) = 1,693147181
- segar (D1): 1 + ln(4/(1 + 1) = 1
- note (D1): 1 + ln(4/(1 + 1) = 1,693147
- sedia (D1): 1 + ln(4/(1 + 1) = 1,693147
- bumbu (D1): 1 + ln(4/(1 + 1) = 1,693147

Step 4, namely calculating the TF-IDF. TF-IDF calculations use (2).

$$W_{i,j} = tf_{i,j} \; x \; idf_i \quad (2)$$

Here is an example of calculating tf-idf (Table 5).

Table 5. TF-IDF Results

Token	tf					df	Idf	W				
	Q	D1	D2	D3	D4			Q	D1	D2	D3	D4
bayam	2	2	0	0	0	1	1,6931	3,386	3,386	0	0	0
segar	2	2	1	0	2	3	1	2	2	1	0	2
note	1	1	0	0	0	1	1,6931	1,693	1,693	0	0	0
sedia	1	1	0	0	0	1	1,6931	1,693	1,693	0	0	0
bumbu	1	1	0	0	0	1	1,6931	1,693	1,693	0	0	0
sayur	1	1	0	0	0	1	1,6931	1,693	1,693	0	0	0
pack	1	1	0	0	0	1	1,6931	1,693	1,693	0	0	0
ikat	1	1	1	0	0	2	1,2877	1,288	1,288	1,288	0	0
daun	0	0	2	0	0	1	1,6931	0	0	3,386	0	0
bawang	0	0	2	0	0	1	1,6931	0	0	3,386	0	0
segarikat	0	0	2	0	0	1	1,6931	0	0	3,386	0	0
fresh	0	0	1	0	0	1	1,6931	0	0	1,693	0	0
harga	0	0	1	0	0	1	1,6931	0	0	1,693	0	0
kurleb	0	0	1	0	0	1	1,6931	0	0	1,693	0	0
tempe	0	0	0	1	0	1	1,6931	0	0	0	1,693	0
papan	0	0	0	1	0	1	1,6931	0	0	0	1,693	0
wortel	0	0	0	0	1	1	1,6931	0	0	0	0	1,693
brastagi	0	0	0	0	1	1	1,6931	0	0	0	0	1,693
produk	0	0	0	0	1	1	1,6931	0	0	0	0	1,693
bagus	0	0	0	0	1	1	1,6931	0	0	0	0	1,693
kirim	0	0	0	0	1	1	1,6931	0	0	0	0	1,693

Calculation:
- bayam (D1): 2 * 1,693147181 = 3,386294362
- segar (D1): 2 * 1 = 2
- note (D1): 1 * 1,693147 = 1,693147
- sedia (D1): 1 * 1,693147 = 1,693147
- bumbu (D1): 1 * 1,693147 = 1,693147

This weighting process produces a value or weight given to each token. The high-weight tokens are bayam, segar, daun, bawang, segarikat.

D. Cosine Similarity

Every word in the query and document is converted into a Vector Model to calculate Cosine Similarity. The calculation result from Cosine Similarity will produce a similarity value between the query and the available documents. Each document is compared with other documents to get a Similarity between one document and another. Table 6 is an example of calculating the cosine similarity.

Table 6. TF-IDF Results

Token	W					Q2	D12	D22	D32	D42	Q * D1	Q * D2	Q * D3	Q * D4
	Q	D1	D2	D3	D4									
bayam	3,39	3,39	0	0	0	11,5	11,5	0	0	0	11,5	0	0	0
segar	2	2	1	0	2	4	4	1	0	4	4	2	0	4
note	1,69	1,69	0	0	0	2,87	2,87	0	0	0	2,87	0	0	0

Token	W					Q2	D12	D22	D32	D42	Q * D1	Q * D2	Q * D3	Q * D4
	Q	D1	D2	D3	D4									
sedia	1,69	1,69	0	0	0	2,87	2,87	0	0	0	2,87	0	0	0
bumbu	1,69	1,69	0	0	0	2,87	2,87	0	0	0	2,87	0	0	0
sayur	1,69	1,69	0	0	0	2,87	2,87	0	0	0	2,87	0	0	0
pack	1,69	1,69	0	0	0	2,87	2,87	0	0	0	2,87	0	0	0
ikat	1,29	1,29	1,29	0	0	1,66	1,66	1,66	0	0	1,66	1,66	0	0
daun	0	0	3,39	0	0	0	0	11,5	0	0	0	0	0	0
bawang	0	0	3,39	0	0	0	0	11,5	0	0	0	0	0	0
segarikat	0	0	3,39	0	0	0	0	11,5	0	0	0	0	0	0
fresh	0	0	1,69	0	0	0	0	2,87	0	0	0	0	0	0
harga	0	0	1,69	0	0	0	0	2,87	0	0	0	0	0	0
kurleb	0	0	1,69	0	0	0	0	2,87	0	0	0	0	0	0
tempe	0	0	0	1,69	0	0	0	0	2,87	0	0	0	0	0
papan	0	0	0	1,69	0	0	0	0	2,87	0	0	0	0	0
wortel	0	0	0	0	1,69	0	0	0	0	2,87	0	0	0	0
brastagi	0	0	0	0	1,69	0	0	0	0	2,87	0	0	0	0
produk	0	0	0	0	1,69	0	0	0	0	2,87	0	0	0	0
bagus	0	0	0	0	1,69	0	0	0	0	2,87	0	0	0	0
kirim	0	0	0	0	1,69	0	0	0	0	2,87	0	0	0	0
						31,5 5,61	Sqrt(Di)				SUM(Q*D1)			
							5,61	6,76	2,39	4,28	31,5	3,66	0	4
									Rank Score		1	0,1	0	0,17

Q = Query (The query used is document 1).
D12 = the rank of Document 1.
D22 = the rank of Document 2.
D32 = the rank of Document 3.
D42 = the rank of Document 4.
Q*D1 = result Q * D1
Q*D2 = result Q * D2
Q*D3 = result Q * D3
Q*D4 = result Q * D4

After the data in the document is calculated its weight using the TF-IDF algorithm. Next is to rank the weight of each Term in the document. Here is an example of a calculation:

- "Bayam" (D1)
  W = 3,386.
  W^2 = 3,386 * 3,386 = 11,467
- “Segar” (D1)
  W = 2
  W^2 = 2 * 2 = 4
- “Note” (D1)
  W = 1,693
  W^2 = 1,693 * 1,693 = 2,86675
- “Sedia” (D1)
  W = 1,693
  W^2 = 1,693 * 1,693 = 2,86675

The next step is to calculate the total overall value of the query and the document that is:

After getting the query's total value and document weights, the next step is to calculate the square root of the total value.

The sum root of the query weight = $\sqrt{31,5} = 5,61$

The next step is to multiply the weight value (w) by the query weight.

Document 1:

- "Bayam"
  Q = 3,386.
  W = 3,386.

Q*W = 3,386 * 3,386 = 11,467

Document 2:

- "Bayam"
  Q = 3,386.
  W = 0
  Q*W = 3,386 * 0 = 0

Document 3:

- "Bayam"
  Q = 3,386.
  W = 0
  Q*W = 3,386 * 0 = 0

Document 4:

- "Bayam"
  Q = 3,386.
  W = 0.
  Q*W = 3,386 * 0 = 0

After obtaining the weight * query weight, the next step is to calculate these values' total number. The next step is to calculate the score ranking value, with the following (3).

$$\frac{total\ Q*value\ of\ the\ document}{(root\ to\ the\ number\ of\ documents^2 * root\ of\ the\ total\ number\ of\ query^2)} \quad (3)$$

- D1 = 31,5 / (5,61 * 5,61) = 1
- D2 = 3,66 / (5,61 * 6,76) = 0,1
- D3 = 0 / (5,61 * 2,39) = 0
- D4 = 4 / (5,61 * 4,28) = 0,17

After the data has been successfully calculated for Cosine Similarity, the rank score results are taken and sorted to get the recommendation. The rank score on the Cosine Similarity results is between 0 and 1. The higher the value, it proves the document similarity between the query document and the recommended document.

E. Testing

Testing was done by making a recommendation model from the data set of 868 data, which then calculated the similarity to Cosine Similarity. The test that will be done is to calculate the recall and precision values. The greater the recall and precision values, the better the recommendation system with the Content-Based Filtering method on product recommendations can provide appropriate recommendation results. There are 3 test schemes. The first recommendation system will recommend a product based on the description and name of the document. The second is the recommendation system will recommend products based on the name of the document. The third is the recommendation system will recommend products based on the document's name and product label. The first schematic test can be seen in Table 7.

Table 7. Testing the first schema

Document ID	*Document Name*	*Recall*	*Precision*
14	Bayam Hijau Per Ikat	0,38	0,41
222	Wortel Brastagi 500 Gram	0,4	0,25
154	Jamur Shitake Mushroom Import Beli 200 Gram Sayuran Segar - Goodfruits	0,05	0,4
31	Tempe 1 Papan	0,49	0,97
666	Asam Jawa Tanpa Biji 150gr	0,33	0,73
42	Kentang Segar	0,42	0,56
452	Daun Ketumbar wansui 250gr	0,14	0,44
30	Ikan Cue Keranjang	0,25	1
	Average	0,30	0,59

In the second scheme test, where the data used to calculate the weighting is the product's name only. The test results can be seen in Table 8.

Table 8. Testing second schema

Document ID	*Document Name*	*Recall*	*Precision*
14	Bayam Hijau Per Ikat	0,97	0,79
222	Wortel Brastagi 500 Gram	0,97	0,65
154	Jamur Shitake Mushroom Import Beli 200 Gram Sayuran Segar - Goodfruits	0,91	0,81
31	Tempe 1 Papan	0,76	1
666	Asam Jawa Tanpa Biji 150gr	0,57	0,78
42	Kentang Segar	0,96	0,5
452	Daun Ketumbar wansui 250gr	0,85	0,71
30	Ikan Cue Keranjang	0,75	1
	Average	0,84	0,78

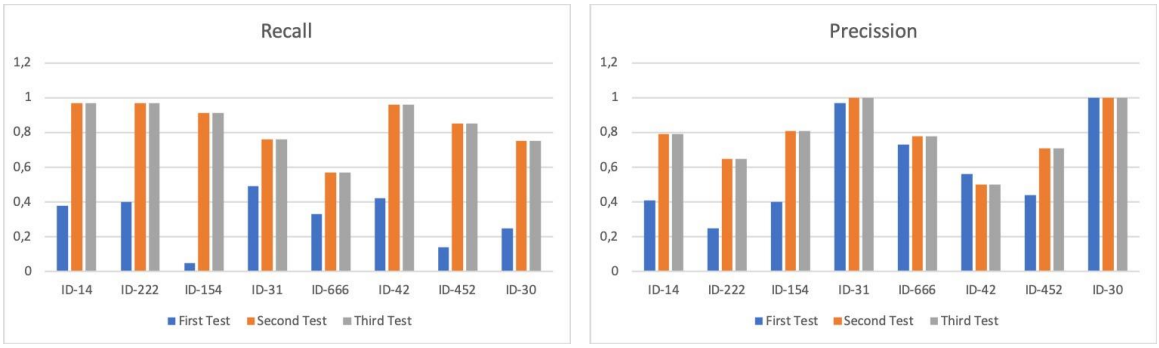
In the third schema testing, the data used for weighting are product names and product labels, which these labels will later be used as product categories. The test results can be seen in Table 9.

Table 9. Testing second schema

Document ID	*Document Name*	*Recall*	*Precision*
14	Bayam Hijau Per Ikat	0,97	0,79
222	Wortel Brastagi 500 Gram	0,97	0,65
154	Jamur Shitake Mushroom Import Beli 200 Gram Sayuran Segar - Goodfruits	0,91	0,81
31	Tempe 1 Papan	0,76	1
666	Asam Jawa Tanpa Biji 150gr	0,57	0,78
42	Kentang Segar	0,96	0,5
452	Daun Ketumbar wansui 250gr	0,85	0,71
30	Ikan Cue Keranjang	0,75	1
	Average	0,84	0,78

The Recall and Precision values have been produced from the above test results, which differ significantly between the first test scheme and the second and third test schemes. Where the first test scheme produces an average value of Recall, and Precision is below 0.60. Moreover, the second and third test schemes produce identical mean values. It happens because the product description in the online store is not relevant to the name of the product. For example, usually, a merchant writes a product description with a shop description or conditions in the store, such as delivery time or the method of delivery that can be done in the store. It will make the results of the recommendations unsuitable. So it can be concluded that it produces a large text value, so the product data must have an exact name and description, and there is no need to add other sentences that are useless.

Testing this recommendation system produces different Recall and Precision values between the first, second, and third test schemes. The test value of the first scheme, as shown in Fig. 6, produces an average Recall value of 0.30 and a Precision of 0.59, and in the second and third tests, it produces an average Recall value of 0.84 and Precision of 0.78.



(a) (b)

Fig. 6.(a) Recall and (b) Precision test results

IV. Conclusion

A product recommendation system has been made using the Content-Based Filtering method. The system architecture must consider the availability of datasets that are not ideal. As the dataset used in this study, some words should not be in the product naming and product descriptions. It will affect the computational load because the system architecture required a preprocessing stage which is likely to provide more load on the computing performance. This product recommendation system using the Content-Based Filtering method certainly still has many shortcomings. So that for development for further research, there are suggestions that can be used in this study. The system in this study is still less efficient in power management. Where to make a recommendation model takes a long time. If the document used for weighting is name and description, it will take an average of about 10 minutes. The results of generating these recommendations will be saved in a file with the .json format, a huge file. The average file size is around 40-100MB. It will be a problem if more data is used because before the data is stored in the file, the data will be stored in variables. If the data is still stored in variables, it will take up quite a lot of RAM space.

The recommendation results are not Real-Time. It means that if the database's data is added, the data is not directly included in the recommendation. The system must first re-create the recommendation model so that the data is included in the recommendation. The recommendation system does not yet support more than one language. In this recommendation, two stemming methods are used, namely Sastrawi and Porter. However, the results are not satisfactory. First, the word will be stemmed with Literary, then checked whether it had been tuned or not. If not, then it will be followed by stemming using Porter. Even though it can stem from English, when the word that should be Indonesian and is not successful in stemming with the Literary algorithm, it is systemized using Porter's algorithm.

References

[1] I. P. Gerachenko, A. A. Kuldiaeva, J. P. Dus, S. Dyrka, and N. L. Seitakhmetova, "FORECAST OF DEVELOPMENT OF THE GLOBAL E-COMMERCE MARKET," *Bull. Natl. Acad. Sci. Repub. KAZAKHSTAN*, vol. 4, no. 386, pp. 157–164, Aug. 2020.

[2] H. D Tran, "From E-Commerce to M-Commerce," *J. Text. Sci. Fash. Technol.*, vol. 6, no. 1, pp. 1–2, 2020.

[3] E. S. Soegoto and A. Nugraha, "E-Commerce for Agriculture," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 879, no. 1, 2020.

[4] I. Almarashdeh *et al.*, "Search Convenience and Access Convenience: The Difference Between Website Shopping and Mobile Shopping," in *Proceedings of the Tenth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2018)*, 2020, pp. 33–42.

[5] N. Vaidya and A. R. Khachane, "Recommender systems-the need of the ecommerce ERA," in *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, 2017, no. Iccmc, pp. 100–104.

[6] L. Wu, D. Hu, L. Hong, and H. Liu, "Turning Clicks into purchases: Revenue optimization for product search in e-commerce," *41st Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval, SIGIR 2018*, no. 1, pp. 365–374, 2018.

[7] D. H. Manjaiah and F. Mohammed, "A survey on recommendation systems for social media using Big Data analytics," *Int. J. Latest Trends Eng. Technol.*, pp. 48–58, 2017.

[8] R. Burke, A. Felfernig, and M. H. Göker, "Recommender Systems: An Overview," *AI Mag.*, 2017.

[9] C. A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System," *ACM Trans. Manag. Inf. Syst.*, vol. 6, no. 4, pp. 1–19, Jan. 2016.

[10] Prince Praveen, Sagar Parmar, and Praveen Goud, "Movie Recommendation System Approaches," *Int. J. Eng. Res.*, vol. V9, no. 04, 2020.

[11] O. Artemenko, O. Kunanets, and V. Pasichnyk, "E-tourism recommender systems : a survey and development perspectives," *Econtechmod. An Int. Q. J.*, vol. 6, no. 2, pp. 91–96, 2017.

[12] F. Karimova, "A Survey of e-Commerce Recommender Systems," *Eur. Sci. Journal, ESJ*, vol. 12, no. 34, p. 75, 2016.

[13] V. G. Student and C. Science, "An Effective Product Recommendation System for E-Commerce Website Using Hybrid Recommendation Systems," *Int. J. Comput. Sci. Commun.*, vol. 8, no. 2, pp. 81–88, 2017.

[14] M. H. Mohamed, M. H. Khafagy, and M. H. Ibrahim, "Recommender Systems Challenges and Solutions Survey," *2019 Int. Conf. Innov. Trends Comput. Eng.*, no. February, pp. 149–155, 2019.

[15] J. Son and S. B. Kim, "Content-based filtering for recommendation systems using multiattribute networks," *Expert Syst. Appl.*, vol. 89, pp. 404–412, 2017.

[16] S. Jain, A. Grover, P. S. Thakur, and S. K. Choudhary, "Trends, problems and solutions of recommender system," *Int. Conf. Comput. Commun. Autom. ICCCA 2015*, no. May, pp. 955–958, 2015.

[17] [R. Wita, K. Bubphachuen, and J. Chawachat, "Content-Based Filtering Recommendation in Abstract Search Using Neo4j," *ICSEC 2017 - 21st Int. Comput. Sci. Eng. Conf. 2017, Proceeding*, vol. 6, pp. 136–139, 2018.

[18] Sangeeta and N. Duhan, "Collaborative filtering-based recommender system," in *Advances in Intelligent Systems and Computing*, 2018.

[19] E. Çano and M. Morisio, "Hybrid recommender systems: A systematic literature review," *Intell. Data Anal.*, vol. 21, no. 6, pp. 1487–1524, 2017.

[20] F. Pourgholamali, M. Kahani, E. Bagheri, and Z. Noorian, "Embedding unstructured side information in product recommendation," *Electron. Commer. Res. Appl.*, vol. 25, pp. 70–85, 2017.

[21] J. Supriyanto; Fahana, "Possible System Architecture for Travel Recommender," vol. 5, no. 1, pp. 1–8, 2020.

[22] L. Sharma and A. Gera, "A Survey of Recommendation System : Research Challenges," *Int. J. Eng. Trends Technol.*, vol. 4, no. 5, pp. 1989–1992, 2013.

[23] M. Zhou, Z. Ding, J. Tang, and D. Yin, "Micro behaviors: A new perspective in E-commerce recommender systems," *WSDM 2018 - Proc. 11th ACM Int. Conf. Web Search Data Min.*, vol. 2018-Febuary, pp. 727–735, 2018.

[24] G. Orellana, M. Orellana, V. Saquicela, F. Baculima, and N. Piedra, "A text mining methodology to discover syllabi similarities among higher education institutions," *Proc. - 3rd Int. Conf. Inf. Syst. Comput. Sci. INCISCOS 2018*, vol. 2018-Decem, pp. 261–268, 2018.

[25] I. Indriyanto and I. D. Sumitra, "Measuring the Level of Plagiarism of Thesis using Vector Space Model and Cosine Similarity Methods," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 662, no. 2, 2019.

[26] D. M. Berry, A. Ferrari, and S. Gnesi, "Assessing tools for defect detection in natural language requirements: Recall vs precision," *Sch. Comput. Sci. Univ. Waterloo, Tech. Rep*, 2017.

[27] L. Hickman, S. Thapa, and L. Tay, "Text Preprocessing for Text Mining in Organizational Research : Review and Recommendations In-press at Organizational Research Methods," no. October, 2020.

[28] M. A. Rosid, A. S. Fitrani, I. R. I. Astutik, N. I. Mulloh, and H. A. Gozali, "Improving Text Preprocessing for Student Complaint Document Classification Using Sastrawi," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 874, no. 1, 2020.

[29] F. Riza, S. Rifai, A. Dirgantara, Sfenrianto, Rasenda, and S. Herdyansyah, "Information Retrieval Technique for Indonesian PDF Document with Modified Stemming Porter Method Using PHP," *J. Phys. Conf. Ser.*, vol. 1477, no. 3, 2020.

[30] M. Artama, I. N. Sukajaya, and G. Indrawan, "Classification of official letters using TF-IDF method," *J. Phys. Conf. Ser.*, vol. 1516, no. 1, 2020.