E-Commerce Content and Collaborative-based Recommendation using K-Nearest Neighbors and Enriched Weighted Vectors

Bardia Rafieian[*] **Marta R. Costa-jussà**
TALP Research Center, Universitat Politècnica de Catalunya, Barcelona
`{bardia.rafieian,marta.ruiz}@upc.edu`

Abstract

In this paper, we present two productive and functional recommender methods to improve the accuracy of predicting the right product for the user. One proposal is a survey-based recommender system that uses k-nearest neighbors. It recommends products by asking questions from the user, efficiently applying a binary product vector to the product attributes, and processing the request with a minimum error. The second proposal uses an enriched collaborative-based recommender system using enriched weighted vectors. Thanks to the style rules, the enriched collaborative-based method recommends outfits with competitive recommendation quality. We evaluated both of the proposals on a Kaggle fashion-dataset along with iMaterialist and, results show equivalent performance on binary gender and product attributes.

1 Introduction

The demanding market of the fashion industry has led to a challenging environment for recommender systems (Nenni et al., 2013). To say, increasing information in the fashion shopping brings confusion to users who have to select the right choice among a huge number of available fashion products. Furthermore, to satisfy the excessive desire of users in the fashion industry, recommendation engines are playing an important role by automating the product selection procedure (Luce, 2018).

A recommender system targets to predict user's tastes and recommend product items that exceptionally are interesting for them. In recommendation engines, the product and the user information is gathered to predict the score or choice of a user to a product. The information collected from a user is either directly or indirectly while the data for products is collected explicitly. In content-based filtering recommendations (Schafer et al., 2007), we explicitly gather information for both users and products and then compare the similarity to recommend the best choice. On the other hand, the collaborative-based algorithm (Bhagavatula et al., 2018) uses "User Behavior" for recommending items. They explore the behavior of users and products in terms of rating, selection, purchase history, and cookies (Isinkaye et al., 2015).

We introduce two settings of recommendation systems for the fashion industry using both types of collaborative and content-based filtering. In the first approach, we propose a survey content-based recommender system and as a second, we introduce an enriched collaborative-based recommender using a novel weighted system. Finally, we initiate style rules that bring simplicity in selecting outfits based on available products.

The rest of the paper is organized as follows. The next section covers an overview of available approaches in fashion recommending systems. Section 3 describes our proposed methods in detail. Section 4 shows the experimental framework and results and finally, section 5 discusses the conclusions and future work of the proposed system.

2 Related Works

The wide fashion domain requires studies in the research area to assist the shopping experience. For a long time, the limitation on computing resources was a drawback for companies to apply deep learning algorithms on their platform. Fortunately, now technology allows processing a vast amount of data in a short time. The machine learning field is impacting the fashion industry in several ways including Apparel designing, Manufacturing process and, Virtual merchandising [1]. Several fashion retailers are using question Answering systems (Santoro et al., 2017), Visual search, and automatic product tagging (Hiriyannaiah et al., 2020). Nowadays, NLP as a machine learning technology improves the field of personalization by extracting the product characteristics such as attributes, reviews, feedback, and other information [2].

Recently, the fashion industry is developing outfit recommenders by using machine learning techniques to decrease the time for exploring and combining products along with trended styles.

Consequently, in (Lin et al., 2018) they addressed the task of outfit recommendation by suggesting a bag of lower body clothes to a selected top. That is to say, a neural multi-task learning framework, called neural outfit recommendation (NOR) has been introduced consisting of two: a convolutional neural network with an attention mechanism to mine visual attributes of products and then a recurrent neural network to translate the visual information into text format. As a result, the mutual information among products is discovered and the system can recommend outfits using these features.

In (Akshaya et al., 2018) they used K-Nearest Neighbour algorithm to detect the nearest neighbor or a cluster based on the k value. There is also a considerable amount of work on recommending different types of categories such as dresses, backbags, heels and handbags along with rating top k products to display (Li et al., 2010). One implemented solution uses a text mining approach to improve ranked based recommender systems. Also, in (Ahuja et al., 2019) a combination of k-means Clustering (Singh et al., 2020) (Jin and Han, 2010) along with k-Nearest Neighbor is implemented on the movielens dataset (Harper and Konstan, 2015) to achieve an improving result. In their proposed technique, the recommender system predicts the user's preference for a movie based on different parameters which also can be applied to product categories in the fashion industry.

3 Proposed Methods

This section describes the two proposed methods of our recommender system. The first subsection explains the survey content-based recommender system architecture within an example. The second subsection provides the enriched collaborative-based recommender system with the stylist, including architecture and examples.

3.1 Survey content-based recommender system

In a survey content-based recommender systems, we rely on replies to product survey questions and try to locate the most similar products. To address this, we require a dataset of product vectors and a replied-on-survey vector with the same dimension as the product vectors. The product dataset includes product information in the form of binary vectors. Table 1 shows an example of the database format of the product table. In this example, there are two different classes each has three sub-classes for totally three products:

Product_id	**Class1**	**Class2**
Product_1	Subclass1	Subclass1
Product_2	Subclass2	Subclass2
Product_3	Subclass3	Subclass3

Table 1: Database format of the product table.

[1]`https://medium.com/datadriveninvestor/`

[2]`https://www.information-age.com/natural-language-processing-explained-123480247/`

To prepare the data for training, we created a binary table from the original one, including only 0 and 1 values. As a result, we extended columns to classes.sub-classes values. Table 2 shows the transformed format of the original table 1.

Product_id	**Class1.Sub1**	**Class1.Sub2**	**Class1.Subc3**	**Class2.Sub1**	**Class2.Sub2**	**Class2.Sub3**
Product_1	0/1	0/1	0/1	0/1	0/1	0/1
Product_2	0/1	0/1	0/1	0/1	0/1	0/1
Product_3	0/1	0/1	0/1	0/1	0/1	0/1

Table 2: Modified database format of product table.

Let's consider $\overrightarrow{P_i}$ as product vectors and $\overrightarrow{R}$ as a reply vector. As a first step, we train unsupervised K-Nearest Neighbors (Knn) (Peterson, 2009) on all $\overrightarrow{P_i}$ = { $\overrightarrow{P_1}$, ... ,$\overrightarrow{P_n}$} vectors using Euclidean distance so that for each $\overrightarrow{P_i}$ the distance is calculated. At this step, we train all categories of products. Considering $\overrightarrow{P_1}$ and $\overrightarrow{P_2}$ as two sample vectors, the distance D is calculated as shown in equation 1.

$$D(\overrightarrow{P_1}, \overrightarrow{P_2}) = ||\overrightarrow{P_1} - \overrightarrow{P_2}|| = \sqrt[2]{(\overrightarrow{P_{1_1}} - \overrightarrow{P_{2_1}})^2 + (\overrightarrow{P_{1_2}} - \overrightarrow{P_{2_2}})^2 + ... + (\overrightarrow{P_{1_n}} - \overrightarrow{P_{2_n}})^2} \tag{1}$$

Afterward, through a survey, we receive information and create a reply vector equivalent to the product vector in terms of size. More precisely, in the survey, there are questions for each sub-class equivalent to the product classes. Then, at each session, the user selects the required sub-class in each class. Finally, for each reply, recommendations are chosen from its n best predictions. The n number of predicted recommendations can be defined or modified at each training by the e-commerce platform manager.

3.2 Enriched collaborative-based recommender system

The enriched collaborative-based recommender system pretends to take advantage of extra user external information. Traditional collaborative-based recommender systems reply to users' historical preferences on a set of items. Because they are based on historical data, the main assumption is users who have agreed in the past items to also agree in the future ones (Schafer et al., 2007). we propose to inject: the history of clicks and feedback, the clicked product and, styles.

3.2.1 Injected information

History of clicks: We require the set of past clicked products by each user to track their preference, as shown in equation 2.

$$\overrightarrow{C} = \{\overrightarrow{C_i} | \overrightarrow{C_i} \in \overrightarrow{P}, and \langle c, c \rangle = \langle p, p \rangle\} \tag{2}$$

where $\overrightarrow{C}$ is the set of clicked products and $\overrightarrow{P}$ is the vector of products. Then, we sum up all vectors in $\overrightarrow{C_i}$ as $\overrightarrow{SC}$ by considering n as a number of product vectors in the history of clicks, as shown in equation 3.

$$SC = \sum_{i=1}^{n} \overrightarrow{C_i} \tag{3}$$

where $\overrightarrow{SC}$ includes sum of all vector elements. Then, executing the algorithm 1, we select maximum values in all elements of $\overrightarrow{SC}$ as 1 and the rest as 0.

```
Result: Return SC
for element in SC do
    if element is Maximum then
        element = 1;
    else
        element = 0;
    end
end
```

Algorithm 1: Creating a binary vector from maximum values of the history of clicks

History of feedback (rating): We assume each user is able to rate each product by its own preference as shown in equation 4.

$$F_{p_x} = \begin{cases} 1, & \text{like} \\ 0, & \text{not rated} \\ \text{-1}, & \text{dislike} \end{cases} \tag{4}$$

where F_{p_x} is feedback on product P_i by user X_i. Further, we discuss in detail the way we use this data.

Clicked product: The clicked product (CP) returns the corresponded vector of the product so that we use it to find similar products into our trained model. At this step, we already have trained models varied by main product categories. Next, n number of similar products to the clicked one is detected by Euclidean distance.

Styles: Styles are predefined patterns to combine products as an outfit. To say, each style includes several clothes with predefined category, subcategory and, color. We have styles that ranged from two to five products to complete any outfit. Table 3 gives a sample of style including three products with predefined attributes which complete an outfit. Table 3 gives a sample of style including three products with predefined attributes which complete an outfit.

{ ”Product1”:[{ ”Category”:”Dress”, ”Subcategory”:”Nightdress”, ”Color”:”Blue” }]
,”Product2”: [{ ”Category”:”Footwear”, ”Subcategory”:”Sandals”, ”Color”:”Black”}]
”Product3”:[{ ”Category”:”Accessory”, ”Subcategory”:”Belt”, ”Color”:”Black” }]}

Table 3: Sample of style including 3 products with predefined attributes.

After retrieving a bag of similar products to the clicked one, we look for the best style to create an outfit.

3.2.2 Method to inject information

In the next, we explain how the communication between injected parts is done. The proposed system works using three main steps to recommend an outfit to a specific user.

Step1 : In this step, the product information is taken as input and we create the new data by transforming the raw data into a binary matrix. In tables 4 and 5 we illustrate a real example of normal and transformed matrix table of products respectively.

Product_id	**Cat**	**Subcat**	**Color**
Product_1	shirt	tshirt	red
Product_2	dress	blazer	black
Product_3	footwear	sanadals	white
Product_4	pants	boots	black
Product_5	shirt	boots	white

Table 4: A normal form of the product table

Product_id	**Cat.Shirt**	**Cat.Dress**	**Cat.Footwear**	**Cat.Pants**	**Subcat.Tshirt**	**...**	**Color.Red**	**Color.Black**	**Color.white**
Product_1	0/1	0/1	0/1	0/1	0/1		0/1	0/1	0/1
Product_2	0/1	0/1	0/1	0/1	0/1		0/1	0/1	0/1
Product_3	0/1	0/1	0/1	0/1	0/1		0/1	0/1	0/1
Product_4	0/1	0/1	0/1	0/1	0/1		0/1	0/1	0/1
Product_5	0/1	0/1	0/1	0/1	0/1		0/1	0/1	0/1

Table 5: A transformed form of the product table 4

Then, a history of feedback for each user is stored in the database by each feedback (like or dislike). As mentioned, a product can be rated by each user so that its user based to respect the other user’s feedback. In the initial point, when there is no feedback, we select all of the products to perform similarity detection, but once we reach a considerable amount of feedback on a product, we select 80% of liked and 20% of unrated products to recommend. The disliked products are ignored and marked as a blacklist in the database. using user-based feedback, we ensure disliked products are still available for other users. Later, the history of clicks is another input stored in the database to keep a track of user preferences. By using this information, we give a higher weight to the features in which the user has a higher interest. As an example, user a has clicked more on products with black color and leather material, so we try to recommend products including these features. However, these features are being changed during the lifetime of a user in the system so that their preferences are dynamic for different categories. For instance, the recommended color will change once the weight of the current color is less than the previous one due to the clicks on the other type of colors. Afterward, the styles are injected into the database to combine recommended products as outfits. we started with 1000 style patterns with the mentioned format. In the discussion part, we explain the future available tasks for this part. And finally, once a user clicks on a product, we keep the vector of this product as a temporary value to detect its similar products. The important point at this step is the way we store the product information. That is, in order to prevent recommending the same category (for example, recommending always shirt to a clicked shirt product), we store products by different categories and detect similar products in different categories. In the next step, we explain the algorithm we used.

Step 2: After storing the recommended information, we follow Algorithm 2 procedure to prepare the recommended products. Considering k as a number of required products in each category, CP as a current clicked product by the user, P_c as a list of products divided by categories(c), SC as a binary list of history of clicks and F as a list of user's feedback.

```
Result: $ReturnRecommendedList(recommendedproductsbycategory)$
initialization;
$input(k)$;
$open(P_c)$;
$open(SC)$;
$open(F)$;
$open(CP)$;
for $(categoryinP_c)$ do
    for $productincategory$ do
        if $((F_p == 1)and(SCisinproduct))$ then
            $ListLiked = add(product)$;
        end
        if $((F_p == 0)and(SCisinproduct))$ then
            $ListUnrated = add(product)$;
        end
    end
    $Trainset = 80\%(k)(ListLiked) + 20\%(k)(ListUnrated)$;
    $RecommendedList = insert(Knearestneighbors(Trainset))$;
end
```

Algorithm 2: Creating a list of the recommended products per category

Step3: Having the list of recommended products we try to fit products in style rules. We look for a match between recommended products in styles in terms of $Category$, $Subcategory$ and $Color$, as soon as a match is found, we return that list as a total look. Algorithm 3 describes the approach we produce outfit out of recommended products. Table 6 shows an example of a recommended outfit including four products:

```
{"Product133":[{ "Category":"tops", "Subcategory":"formaltop", "Color":"red" "Fabrics:"Cotton" }]
"Product87": [{ "Category":"Footwear", "Subcategory":"boot", "Color":"Blue", "Fabrics:"Foam" }]
"Product91": [{ "Category":"pants", "Subcategory":"jeans", "Color":"Blue", "Fabrics:"Denim" }]}
```

Table 6: Example of recommended outfit.

```
Result: $Alistofclothesasoutfit$
$initialization$;
$open(Styles)$;
$open(RecommendedList)$;
for $styleinStyles$ do
    if $(RecommendedListisinstyle)$ then
        $Returnlistofproducts$;
    end
end
```

Algorithm 3: Creating outfit out of recommended products with Style rules

4 Experiments

Data We perform our experiment on Kaggle fashion-dataset[3] along with iMaterialist[4] with totally around 90k rows and 8 columns. The information of each of the columns included gender, master category, subcategory, article type, base color, year, season, and usage. In total, there are 15 product categories: accessories, bag, beachwear, coats, hat, hosiery, footwear, apparel set, bottom wear, dress, innerwear, loungewear, saree, socks, and topwear. This is a gender-balanced dataset (considering binary classification), we have approximately 48% women and 52% men proportion entries. We are aware of the limitation of the binarization in gender, where other communities from LGTB+ are disregarded, unfortunately non-binary gender is not currently included in this data.

Data Privacy We undertake to comply with the legislation in force regarding the protection of personal data contained in the RGPD 2016/679 UE. The basis for legitimizing the processing of the data is the consent of the interested party. We carry out the treatments related to the scope of the consent and informed purposes. The data provided will be kept for the duration of the contractual relationship and during the years necessary to comply with legal obligations. The data will not be transferred to third parties, except for group companies. We perform the recommender on the data provided by the business to business partners with all of the privacy respects and regulations accepted by them and their final users.

Survey content-based recommender system results In our first experiment we evaluated our survey content-based recommender described in Section 3.1 with 1,000 random answered surveys. Then, we compared the results by Root Mean Square Error (RMSE) based on equation 5.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - Y_i')^2 \tag{5}$$

We compared the MSE between requested and queried products when assigning k number of nearest neighbors to the surveyed product. Figure 1 shows the MSE difference for two settings with $k = 5$ and $k = 10$.

[3]https://www.kaggle.com/paramaggarwal/fashion-product-images-dataset?

[4]https://www.kaggle.com/c/imaterialist-fashion-2019-FGVC6/data?select=train

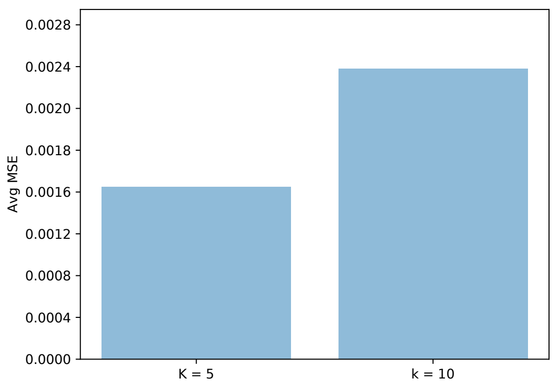

Figure 1: Mean square error for 1,000 surveys on 90k data in two settings: k = 5 and k = 10

To visualize results, we provided five random surveys (expected) and its 5 nearest neighbors found with knn recommender system from section 3.1. We selected random products as input with related vector and performed the experiment. Figure 2 illustrates the expected products in the first row and the recommended ones in the rest in five different requests per column. We can see the category and other attributes in the recommendations are close to the expected product.

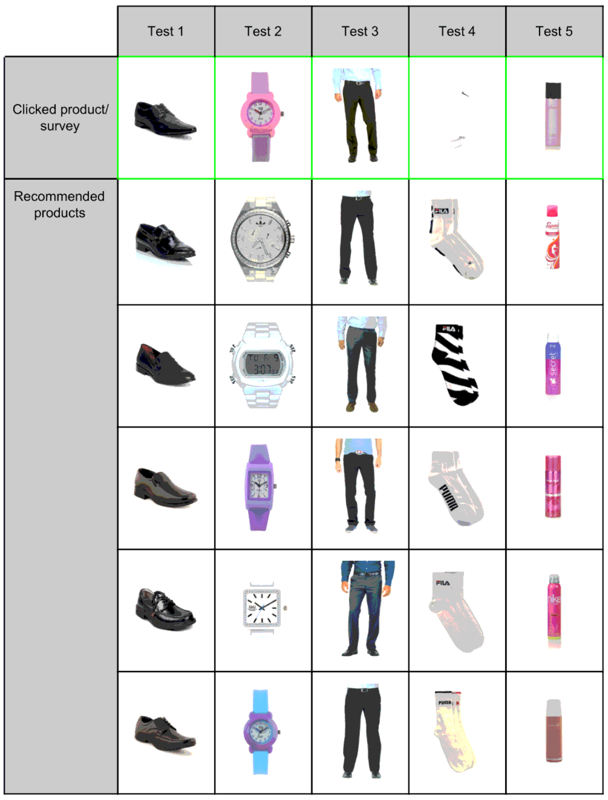

Figure 2: A visualization of k = 5 recommended products for random requests

Collaborative-based recommender system results. We evaluated our second recommender system explained in the section 3.2. In this experiment, we grouped the dataset by product categories such that we find similar products to each clicked-product in different product catalogs. To perform our analysis, totally we created 8 categories out of 10 available in the dataset. Next, the proposed approach performed on the clicked-product and outfits was achieved by using our pre-defined style patterns. We selected five random output samples from the proposed recommender in figure 3.

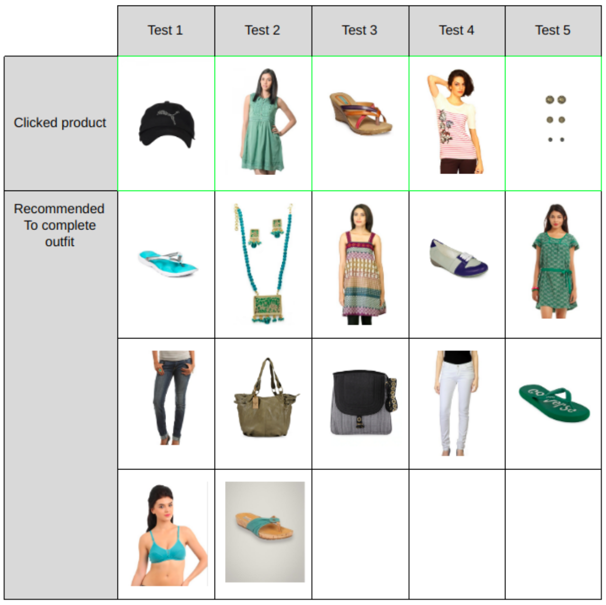

Figure 3: A visualization of full outfit recommendations for the clicked-product using the style rules for five random cases

As shown in Figure 4, the average MSE has decreased by 4.5%, which is a significant decrease in the error.

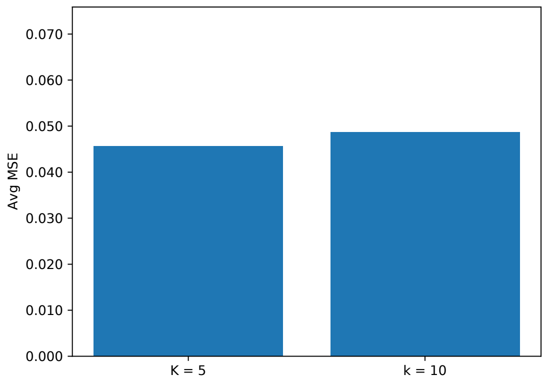

Figure 4: Average MSE in the Collaborative-based recommender in two different settings

Gender analysis. In our setting, once a user clicks on a product should get some recommendations. The system is not aware of any information about the gender of this person and there is no gender filter on products. We want to evaluate the performance of our system desegregated by gender. Referring to the details in 5, we achieved more than 99% accuracy in recommending the product to the correct gender. The results indicated on the bar chart shows similar performance on men-women which is due to a balanced dataset. Moreover, we have improved the gender selection by up to 100% by giving a higher weight to the gender vectors in the clicked product.

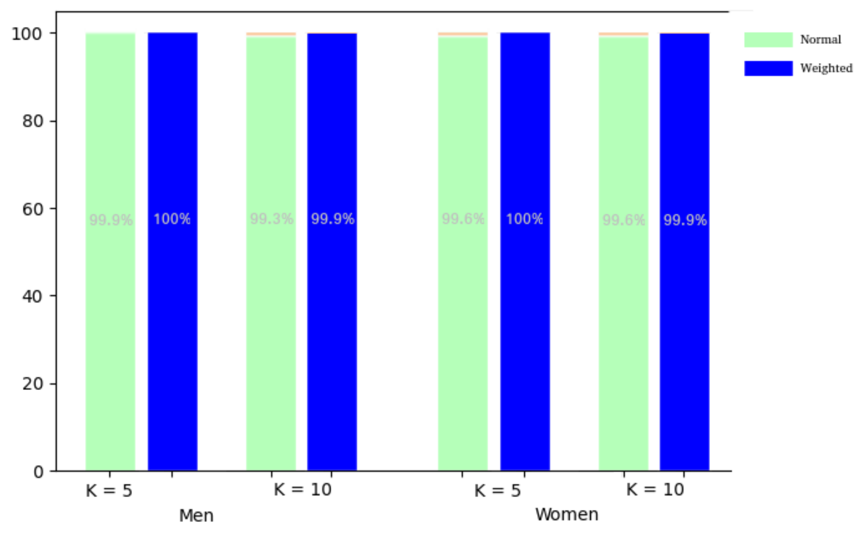

Figure 5: Results of recommending the correct product gender to men and women in normal and weighted modes of vectors

5 Conclusions

Recommender systems are becoming one of the main key points in e-commerce and today customers are demanding more personalized products. In our approach, two recommender systems were introduced where each one has specific usage in the e-commerce platform. The survey content-based recommender system suggests the nearest product based on the replies on the survey with the least distance. On the other hand, the enriched collaborative-based recommender system uses more filters and style rules to create outfits based on the clicked product with similar performance on a gender basis (assuming binary gender). However, there are remaining challenges in updating style rules or missing some products in case of using a few numbers of neighbors. As future work, we plan to develop an enriched collaborative-based recommender system by using more external information and dynamically updating styles based on the local trends.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work is supported in part by the Viume Company through research and development of the online e-commerce platform and by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación, through the postdoctoral senior grant Ramón y Cajal.

References

Rishabh Ahuja, Arun Solanki, and Anand Nayyar. 2019. Movie recommender system using k-means clustering and k-nearest neighbor. pages 263–268, 01.

Srikanth Akshaya, Parvaneh Kamali, and P.Sudha. 2018. Outfit recommender system using knn algorithm. *International journal of engineering research and technology*, 6.

Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. 2018. Content-based citation recommendation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Louisiana, June.

F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, 5(4), December.

Srinidhi Hiriyannaiah, Siddesh G M, and K. Srinivasa. 2020. Deep visual ensemble similarity (dvesm) approach for visually aware recommendation and search in smart community. *Journal of King Saud University - Computer and Information Sciences*, 04.

Folasade Olubusola Isinkaye, Yetunde Folajimi, and Bolanle Adefowoke Ojokoh. 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, 16:261–273.

Xin Jin and Jiawei Han, 2010. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA.

Yize Li, Jiazhong Nie, Yi Zhang, Bingqing Wang, Baoshi Yan, and Fuliang Weng. 2010. Contextual recommendation based on text mining. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, page 692–700, USA. Association for Computational Linguistics.

Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2018. Explainable fashion recommendation with joint outfit matching and comment generation. *CoRR*, abs/1806.08977.

Leanne Luce. 2018. *Artificial Intelligence for Fashion: How AI is Revolutionizing the Fashion Industry*. Apress.

Maria Elena Nenni, Luca Giustiniano, and Luca Pirolo. 2013. Demand forecasting in the fashion industry: A review. *International Journal of Engineering Business Management*, 5.

L. E. Peterson. 2009. K-nearest neighbor. *Scholarpedia*, 4(2):1883. revision #137311.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Timothy P. Lillicrap. 2017. A simple neural network module for relational reasoning. *CoRR*, abs/1706.01427.

J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen, 2007. *Collaborative Filtering Recommender Systems*, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tarana Singh, Anand Nayyar, and Arun Solanki. 2020. Multilingual opinion mining movie recommendation system using rnn. In Pradeep Kumar Singh, Wiesław Pawłowski, Sudeep Tanwar, Neeraj Kumar, Joel J. P. C. Rodrigues, and Mohammad Salameh Obaidat, editors, *Proceedings of First International Conference on Computing, Communications, and Cyber-Security (IC4S 2019)*, pages 589–605, Singapore. Springer Singapore.