# Contrastive Learning for Interactive Recommendation in Fashion

Karin Sevegnani, Arjun Seshadri, Tian Wang, Anurag Beniwal, Julian McAuley, Alan Lu, Gerard Medioni
Amazon
San Francisco, USA
karin.sevegnani@hw.ac.uk,{sesarjun,wangtan,beanurag,jumcaule,alalu,medioni}@amazon.com

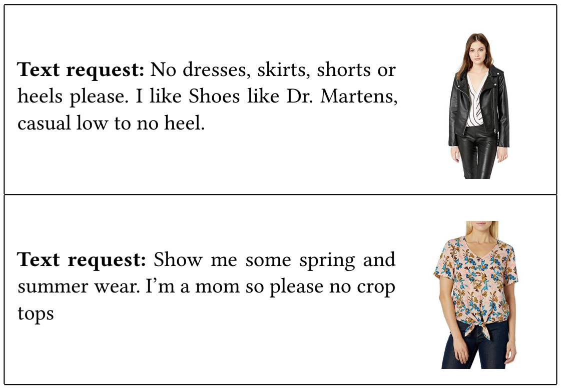
## ABSTRACT

Recommender systems and search are both indispensable in facilitating personalization and ease of browsing in online fashion platforms. However, the two tools often operate independently, failing to combine the strengths of recommender systems to accurately capture user tastes with search systems' ability to process user queries. We propose a novel remedy to this problem by automatically recommending personalized fashion items based on a user-provided text request. Our proposed model, WhisperLite, uses contrastive learning to capture user intent from natural language text and improves the recommendation quality of fashion products. WhisperLite combines the strength of CLIP embeddings with additional neural network layers for personalization, and is trained using a composite loss function based on binary cross entropy and contrastive loss. The model demonstrates a significant improvement in offline recommendation retrieval metrics when tested on a real-world dataset collected from an online retail fashion store, as well as widely used open-source datasets in different e-commerce domains, such as restaurants, movies and TV shows, clothing and shoe reviews. We additionally conduct a user study that captures user judgements on the relevance of the model's recommended items, confirming the relevancy of WhisperLite's recommendations in an online setting.

**ACM Reference Format:**
Karin Sevegnani, Arjun Seshadri, Tian Wang, Anurag Beniwal, Julian McAuley, Alan Lu, Gerard Medioni. 2022. Contrastive Learning for Interactive Recommendation in Fashion. In *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom'22).* ACM, New York, NY, USA, 5 pages.

## 1 INTRODUCTION

Recommendation systems are at the core of e-commerce websites such as Amazon, Netflix, and Ebay. These systems recommend products based on the user's historical preferences and purchases [18, 26]. The goal is to recommend products to users that they might like and improve the experience to make it enjoyable and satisfactory.



**Text request:** No dresses, skirts, shorts or heels please. I like Shoes like Dr. Martens, casual low to no heel.

**Text request:** Show me some spring and summer wear. I'm a mom so please no crop tops

**Figure 1: Examples of personalized text requests written by the user browsing a clothing retail online shop on the left, and an accordingly recommended item on the right.**

The goal of this work is to recommend fashion items focusing specifically on the user's free-form text request provided to a fashion stylist while using a personalized styling subscription service. In our problem setting, we only rely on the text request provided by the user, and do not assume knowledge of user click-stream information or prior purchases. Although prior purchases are often used to model user preferences [19], the presence of such information cannot be assumed for so-called "cold start" users that are using a service for the first time and have no historical purchase or click stream information associated with them. Moreover, text requests often reveal more nuanced aspects of user preferences at the time of placing a styling request, making our work essential to improving existing recommender systems.

Our primary dataset consists of three parts: text requests from users of a personalized styling service, items chosen by a human stylist for the user based on those text requests, and items the user chooses to try from the stylist's recommended items. Figure 1 shows two stylized examples, where users communicate a preference towards specific clothing items through a text request, and professional fashion stylists respond with a product recommendation pertaining to the customer request.

Our modeling task is challenging on several levels. First, text requests are more elaborate than a search query, and contain both useful contextual information that captures high-level user preferences and less helpful information that is irrelevant for retrieving fashion items e.g. "*I am a school teacher and I don't wear yoga pants.*" Second, each item in the recommended set of items might only meet a subset of criteria specified by the user in the text request, creating opportunities for model misattribution and overfitting. Even the recommended set of items as a whole might not cover

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*SIGIR eCom'22, July 15, 2022, Madrid, Spain*
© 2022 Copyright held by the owner/author(s).

all of the criteria mentioned by the user. Third, there are items in the dataset selected by stylists that the user could potentially like despite the items not having the exact attributes requested by the user, further exacerbating the misattribution problem. Finally, there could be items that the user selected despite being irrelevant to the text request, or sometimes even contradictory to their initial text request. Each of these factors introduce noise in the dataset labels, and together create a challenging modeling problem.

The contributions of this paper can be summarized as follows:

- Defining and addressing a real-world recommendation task for online fashion retail in order to improve the user experience (Section 3.1),
- Validating the multi-domain generalization of the system offline using a real-world dataset and three other publicly available data sources (Section 4.3),
- Validating the online performance of the system through a user study on Amazon Mechanical Turk (Section 4.4).

## 2 RELATED WORK

The literature of recommender systems can be broadly segmented into collaborative filtering [10, 20] and content-based methods [2, 16]. While collaborative filtering techniques only exploit past user purchases and product interactions to generate new recommendations [7], content-based methods rely on information about the users or the items [3]. Recommendation systems for fashion retail e-commerce, the domain of our work, frequently fall into the content-based category, and are often paired with images and textual descriptions of the products [1, 4, 5, 8, 9, 21, 25]. Our work is similar in being content-based to other works in fashion retail e-commerce, as it leverages textual user requests as the primary representation of the user. Within the broader literature of recommendation systems, our work most closely aligns with session-based recommendation approaches, which focus on predicting the user's immediate next actions using explicit feedback of the customer during the current session. In our setting, the user's text request serves as the representation of the customer's preferences in the current session. We refer the reader to [12], which provides a comprehensive review of the session-based recommendation literature.

## 3 METHODOLOGY

### 3.1 Problem Formulation

Our work approaches the recommendation problem by trying to build affinity between a natural language text query expressed by the user and an item description. The natural language query $\mathbf{x} = (x_1, x_2, ..., x_N)$ can consist of several sentences, and the item description $\mathbf{z} = (z_1, z_2, ..., z_M)$ can include several item attributes and characteristics, where $N$ and $M$ are the number of tokens in the request and the item description, respectively. The similarity between the request $\mathbf{x}$ and the description $\mathbf{z}$ is determined by the cosine distance between the two. Once similarity scores are computed, the various similarities are then ranked and used to perform a retrieval task. The success of the task is evaluated with different retrieval metrics, which we describe in Section 4.2.

We present two solutions to our problem setting:

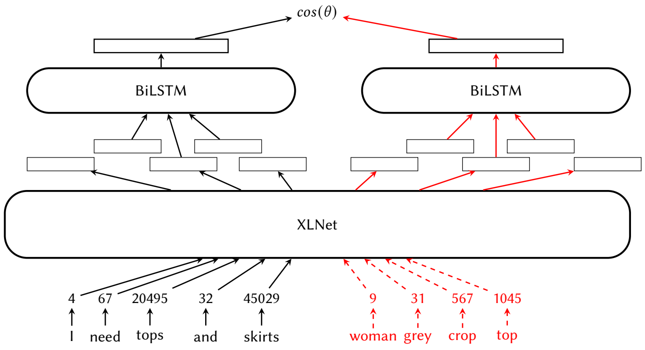


**Figure 2: Architecture of the Whisper model. The dashed lines mean that the weights for XLNet are frozen when computing the embeddings for the product description.**

- The Whisper approach: training a model that refines word embeddings for text notes and item descriptions in order to closely pair requests and products in the word embeddings space and treat them as a match for recommendation (Section 3.2); and
- The WhisperLite approach: derive embeddings from a pre-trained system and train a model only to learn which items to recommend using contrastive loss (Section 3.3).

### 3.2 Whisper

Large pre-trained transformers such as BERT [6], RoBERTa [11], or XLNet [24] have proven particularly effective at the generation of contextual word embeddings. The architecture of the Whisper model is planned and developed in a modular way. We initially leverage one of the state-of-the-art large language models for generating word embeddings. XLNet is the transformer of our choice, since it has been proven to outperform BERT on a variety of tasks [14, 24].

In order to prepare the initial text requests and item descriptions, they are first split into tokens and assigned input ids based on the model tokenizer's built-in vocabulary. Moreover, we perform a keyword search for each text request to highlight categorical tokens and strengthen the division of the text requests into the main clothing categories.

The pre-processed inputs are then passed into XLNet, which generates contextual word embeddings in an auto-regressive way. The embeddings of the words are one-dimensional tensors retrieved through the last hidden state of the language model. Then, the embeddings are fed into a bidirectional LSTM, which produces one feature vector for each text request received as input. A second bi-LSTM is used in order to generate different feature vectors for the product descriptions in a two-tower fashion [23], since the semantics of user requests are very different from those of product descriptions.

As shown in Figure 2 of the Whisper architecture, the transformer's task is to generate contextual word embeddings for both the text request and the item descriptions. The two are then separated and passed as input in the two LSTMs as mentioned above, producing the two corresponding feature vectors. The model is trained using Cosine Embedding Loss

$$loss(x, y) = \begin{cases} 1 - cos(x_1, x_2), \text{if } y = 1 \\ max(0, cos(x_1, x_2)), \text{if } y = -1 \end{cases} \tag{1}$$

## 3.3 WhisperLite

WhisperLite is a lighter version of the Whisper model described in the previous section, where the text embeddings are calculated through the pre-trained CLIP text-only model [17] as opposed to XLNet. Moreover, WhisperLite uses MLPs instead of bi-LSTM layers, and is trained using a customized loss function, combining binary cross entropy loss for classification

$$\mathcal{L}_{BCE} = -(ylog(\hat{y}) + (1 - y)log(1 - \hat{y})) \tag{2}$$

with a contrastive loss. In (2), $\hat{y}$ is the predicted probability of observation $o$ being in class $c$, and $y$ represents whether $o$ actually belongs in $c$.

Additionally, we exchanged the initial cosine similarity to the dot product between the request and the product description vectors since the two metrics are comparable [13, 15]. The CLIP parameters are held frozen, and only the MLPs are trained, significantly reducing the training time of WhisperLite.

# 4 EXPERIMENTS

## 4.1 Compared models

We consider two baseline models to compare with our proposed approach.

- A **random** baseline, where the `(request,item)` pairs are assigned a random ranking.
- **OkapiBM25**, a function that ranks the item description based on its estimated IDF to the text request that the user provides.

Since the task addressed in this project deals with query-to-item recommendations without additional information about the user, to the best of our knowledge there are no other baselines that fit the problem exactly. As a result, our current baselines for comparison are elementary, and adapting existing work to suit our problem setting is an important direction for future work.

## 4.2 Datasets and Evaluation

| | Text requests | Item descriptions |
|---|---|---|
| Number | 4.5M | 4.5M |
| Unique | 245k | 218k |
| Sentiment (pos-neutr-neg) | 74.26% - 17.42% - 8.32% | N/A |
| Word length (min-max-avg) | 3 - 221 - 27.57 | 9 - 52 - 20.68 |
| Word length (50th-90th perc.) | 23 - 54 | 20 - 25 |

**Table 1: Data analysis of the WhisperD dataset.**

The main dataset used in this work, WhisperD, contains text requests from a personalized online fashion recommendation system, collected from real interactions between users and fashion stylists. We additionally consider adaptations of three open source datasets to demonstrate the broad applicability of our methodology. The statistics of the datasets are shown in Table 2.

### 4.2.1 *WhisperD.*

The text requests in the dataset contain both positive preferences, e.g., *Spring and summer wear*, as well as negative ones, e.g., *No dresses, skirts*, and encompass a wide variety of writing styles. Alongside every request in the dataset is also a collection of fashion stylists' item recommendations catering to the request. These item recommendations are categorized in three ways: items that the user decided to try on (TRYs), items that the user bought (KEEPs), and items that the user did not even try (NOTTRYs). Table 4.2 describes the details of the WhisperD dataset. All of the items associated with a user's request are chosen by a fashion stylist, and are therefore classified as positive examples. On the other hand, the negative examples are sampled randomly from the dataset.

### 4.2.2 *Open-source Datasets.*

The open source datasets are obtained through slight modifications of the existing **Yelp** (restaurants domain) and **Amazon** (movies&TV and clothing) datasets. Whereas the original datasets contain customer reviews with binary sentiment annotations, we instead transform the datasets to match customer reviews to item product descriptions [9, 22], aligning the datasets to the modality of WhisperD. With this transformation, given a review and a set of possible targets, the task is to pair the review with the correct target description. As described in [9, 22], the content expressed through reviews can be interpreted as personal user preferences.

### 4.2.3 *Evaluation.*

The evaluation measures used on the aforementioned datasets are the standard in recommender systems literature. The main goal is to determine whether the items retrieved by the model satisfy the user's interest based on their initial query.

- Precision@$k$ measures the number of recommended items in the top-$k$ that are relevant,
- Recall@$k$ calculates the number of relevant products that are in the top-$k$ recommended ones,
- Normalized Discounted Cumulative Gain (NDCG) measures the relevancy of the item based on its position in the list of recommended products.

For precision@$k$ and recall@$k$ we chose $k \in \{1, 2, 3, 4\}$ because of the varying number of relevant items for example.

| Tasks | Train | Dev | Test |
|---|---|---|---|
| WhisperD | 1.58M | 197k | 197k |
| Yelp | 11.3M | 1.4M | 1.4M |
| Amazon Clothing | 2.22M | 278k | 278k |
| Amazon Movies | 13.52M | 1.69M | 1.69M |

**Table 2: Statistics of all datasets used for this work.**

## 4.3 Results

Table 3 shows the results of the experiments on the previously presented datasets. The `WhisperLite` model (`WLite`) substantially outperforms the other baselines on the WhisperD data on almost all metrics. There is also a considerable gap between the `Whisper` and `WhisperLite` models, which only appears in the results on WhisperD. One explanation is the WhisperD dataset consists of noisier text requests and label when compared to public datasets. Higher noise settings could lead to a great degree of overfitting in

| | | PREC@1 | PREC@2 | PREC@3 | PREC@4 | REC@1 | REC@2 | REC@3 | REC@4 | NDCG |
|---|---|---|---|---|---|---|---|---|---|---|
| WhisperD | Random | 0.5010 | 0.4667 | 0.4200 | 0.3697 | 0.2968 | 0.5250 | 0.6730 | 0.7571 | 0.6958 |
| | OkapiBM25 | 0.5044 | 0.5069 | 0.5067 | **0.5072** | 0.0321 | 0.0646 | 0.0968 | 0.1293 | 0.7864 |
| | Whisper | 0.5376 | 0.4917 | 0.4343 | 0.377 | 0.3217 | 0.5515 | 0.6916 | 0.7678 | 0.712 |
| | WLite | **0.7633** | **0.6294** | **0.5111** | 0.4172 | **0.4707** | **0.6884** | **0.7817** | **0.8188** | **0.8088** |
| Yelp | Whisper | 0.8532 | 0.4316 | 0.289 | 0.2173 | **0.8473** | 0.8514 | 0.8524 | 0.8528 | 0.8533 |
| | WLite | **0.8554** | **0.4377** | **0.2943** | **0.2218** | 0.8440 | **0.8527** | **0.8547** | **0.8555** | **0.8562** |
| Amazon movies&TV | Whisper | 0.7695 | 0.3881 | 0.2596 | 0.1951 | 0.7656 | 0.7683 | 0.7689 | 0.7692 | 0.7696 |
| | WLite | **0.7761** | **0.3953** | **0.2652** | **0.1996** | **0.7682** | **0.7746** | **0.7759** | **0.7765** | **0.7768** |
| Amazon clothes, shoes & jewelry | Whisper | **0.8793** | 0.4516 | 0.3043 | 0.2298 | **0.8648** | **0.8743** | **0.8765** | **0.8776** | **0.8794** |
| | WLite | 0.8719 | **0.4587** | **0.3121** | **0.2368** | 0.845 | 0.8637 | 0.8685 | 0.8705 | 0.8732 |

**Table 3: Results table.**

`Whisper` model relative to the `WhisperLite` model, as the `Whisper` model optimizes over feature vectors for the text requests by users whereas the `WhisperLite` model does not. One possible interpretation of such a difference could be that a model trained on generating embeddings for 'generic' text requests, such as '*Looking to update my wardrobe*' or '*Going to Vegas next month*', diverts its ability to generate meaningful feature vectors for the description of the clothes that are paired with such requests. Instead, `WLite` exploits the feature vectors generated by the pre-trained CLIP model to train only the two perceptrons with a classification objective function, which could explain the meaningful difference in performance on the WhisperD data.

On the open-source datasets the two models perform very similarly with a generally higher performance on all metrics compared to the results on WhisperD, hinting at the difficulty of the clothing recommendation task. The results on the Yelp and Amazon clothing datasets are similar, whereas the performance on WhisperD is closer to the Amazon movies dataset.

## 4.4 Human Evaluation

The evaluation compared the `WhisperLite` model with the `random` baseline through a user study on Amazon Mechanical Turk. A total of 1500 data points are randomly sampled from the outputs of the two respective models on the WhisperD dataset and submitted to the participants of the study for evaluation. Each example is associated with $k$ recommended items, where $k \in \{3, 5, 7\}$. In order to increase accountability and discard possible noise, every data point has been rated by 3 unique workers.

The set up of this study is as follows: users are shown one text request and $k$ recommended products associated with it. They then answer the question "*How relevant are these items based on the text request?*" on a likert-scale from 1 to 5, where 1 represents 'Definitely irrelevant', 5 means 'Definitely relevant', and 3 is left as the option for 'Not sure'. The studies for `WhisperLite` and `random` are designed to be separate and not as a comparison between the two, with the purpose of avoiding an introduction of bias on workers completing the tasks. Furthermore, users were required specific qualifications to participate in this study and ensure an acceptable level of quality to the results, namely: being master workers, having completed and submitted at least 500 HITs on MTurk with an acceptance rate greater than 80%, and being located in an English-speaking country. Table 4 shows the impact of both systems on real-world users. We first classified 'negative' ratings such as 1 and 2 as '$-1$', and 'positive' ratings like 4 and 5 as '+1', whereas 0 has been assigned to 'neutral' judgements.

For all three values of $k$, where $k$ items are recommended given a specific request, users preferred the recommendations made by the `WhisperLite` model, with an average rating much higher than the recommendations from the `random` baseline. The error bars computed on the average scores show the variability of the collected judgements from human judges. Instead, the error bars for the `random` baseline indicate that the human ratings are less reliable than the ones collected for `WLite`, as they are bigger (or almost as big, for $k = 7$) than the mean values. This shows a net preference towards the items recommended by `WLite`.

| | | Average |
|---|---|---|
| $k = 3$ | Random | $0.31 \pm 0.37$ |
| | WLite | $\mathbf{0.52} \pm 0.28$ |
| $k = 5$ | Random | $0.31 \pm 0.38$ |
| | WLite | $\mathbf{0.72} \pm 0.24$ |
| $k = 7$ | Random | $0.38 \pm 0.35$ |
| | WLite | $\mathbf{0.59} \pm 0.27$ |

**Table 4: Analysis of the human evaluation of the `random` baseline and the `WLite` model.**

# 5 CONCLUSION AND FUTURE WORK

In this work we describe recommending fashion items for a customer using a single text request. We propose two different approaches, Whisper and WhisperLite, that each address this session-based recommendation problem. While both approaches are competitive in several open source datasets spanning multiple domains, WhisperLite significantly outperforms all other baselines on the WhisperD dataset. We attribute the difference in model performance to the higher level of noise in the WhisperD dataset, where fine-tuning lower, language-model layers of model is highly sub-optimal, as it can confuse the system and negatively influence the generation of the features for the products (see Section 4.3). A meaningful extension of the proposed recommendation approach is to consider several other features like images of the clothing items or the body shape of the customer, in order to recommend items that could complement specific body shapes. Possible features that can also be included are user behavioral data from previous browsing sessions or historical data. We leave exploring the impact of these additional features as future work.

# REFERENCES

[1] Stuti Ajmani, Hiranmay Ghosh, Anupama Mallik, and Santanu Chaudhury. 2013. An ontology based personalized garment recommendation system. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Vol. 3. IEEE, 17–20.

[2] Marko Balabanović and Yoav Shoham. 1997. Fab: content-based, collaborative recommendation. *Commun. ACM* 40, 3 (1997), 66–72.

[3] Chumki Basu, Haym Hirsh, William Cohen, et al. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Aaai/iaai*. 714–720.

[4] Samit Chakraborty, Md Hoque, Naimur Rahman Jeem, Manik Chandra Biswas, Deepayan Bardhan, Edgar Lobaton, et al. 2021. Fashion Recommendation Systems, Models and Methods: A Review. In *Informatics*, Vol. 8. Multidisciplinary Digital Publishing Institute, 49.

[5] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2019. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 765–774.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. 241–250.

[8] Hyunwoo Hwangbo, Yang Sok Kim, and Kyung Jin Cha. 2018. Recommendation system development for fashion retail e-commerce. *Electronic Commerce Research and Applications* 28 (2018), 94–101.

[9] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian McAuley. 2017. Visually-aware fashion recommendation and design with generative image models. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 207–216.

[10] Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. *Recommender systems handbook* (2015), 77–118.

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[12] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 28, 4 (2018), 331–390.

[13] Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. 2018. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *International Conference on Artificial Neural Networks*. Springer, 382–391.

[14] Christopher Malon. 2021. Overcoming Poor Word Embeddings with Word Definitions. *arXiv preprint arXiv:2103.03842* (2021).

[15] L Naveen, Ananda Raj, S Rajalakshmi, et al. 2021. Abstractive Text Summarizer: A Comparative Study on Dot Product Attention and Cosine Similarity. In *2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, 1–8.

[16] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020* (2021).

[18] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender systems: introduction and challenges. In *Recommender systems handbook*. Springer, 1–34.

[19] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[20] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009).

[21] Qingqing Tu and Le Dong. 2010. An intelligent personalized fashion recommendation system. In *2010 International Conference on Communications, Circuits and Systems (ICCCAS)*. IEEE, 479–485.

[22] Shoujin Wang, Chenlu Yang, Tong Qu, Kai Yang, and Wanggen Wan. 2022. Learning Users' Visual Preferences for Improving Recommendations. *IEEE Access* (2022).

[23] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. 2020. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion Proceedings of the Web Conference 2020*. 441–447.

[24] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[25] Ruiping Yin, Kan Li, Jie Lu, and Guangquan Zhang. 2019. Enhancing fashion recommendation with visual compatibility relationship. In *The World Wide Web Conference*. 3434–3440.

[26] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.