Two-Stage Approach for Predicting Fashion Compatibility

Adilzhan Ismailov
adilzhan.ismailov@depop.com
Depop
London, United Kingdom

ABSTRACT

Predicting whether a set of items of clothing goes well together and can be combined into an outfit is a challenging task, with solutions usually relying on content information, such as images and text descriptions in an attempt to capture visual aesthetics, styles and trends. This paper presents the winning solution to the SIGIR 2022 Workshop on eCommerce Fashion Outfits Challenge that proposes a two-stage approach that leverages both context and content information by combining a compatibility measure derived from an item-to-item collaborative filtering model with features based on product metadata.

CCS CONCEPTS

• **Information systems** → *Information retrieval*; • **Computing methodologies** → *Supervised learning*.

KEYWORDS

fashion recommender systems, outfit recommendations, collaborative filtering, gradient boosting

ACM Reference Format:
Adilzhan Ismailov. 2022. Two-Stage Approach for Predicting Fashion Compatibility. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (SIGIR eCom'22).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Predicting whether a set of items of clothing goes well together and can be combined into an outfit is a challenging task, with solutions usually relying on image and text information to capture visual aesthetics, styles and trends. The focus on content-based model as opposed to a collaborative-filtering based model popular for fashion item recommendations tasks can be explained by scarcity of outfit interaction data [4] as data on outfits composed by experts is scarce and larger datasets could be noisier due to being bootstrapped from user activity data.

The dataset provided for the Fashion Outfits Challenge of the SIGIR 2022 Workshop on eCommerce consisting of around 300,000 outfits composed by stylists and fashion experts is different in this regard and in this paper we present the winning solution to the SIGIR 2022 Workshop on eCommerce Fashion Outfits Challenge that proposes a two-stage approach that relies more heavily on contextual data, i.e. what items tend to be put in similar outfits more often.

The challenge focused on the Fill in the Blank (FITB) task where a missing item of a real outfit is required to be predicted from a list of candidates. Our proposed solution for this task consists of two models. The first model is an item-to-item recommendation model that treats every outfit as a unit of observation and allows us to produce pairwise compatibility scores between every item in the target outfit and every item in the candidate set.

The second model—a gradient boosting machine model—ranks candidates based on a set of hand-crafted features. There are four main groups of features: features based on the output of the first model, features based on outfit attributes, features based on candidate attributes, and features based on matching attributes between the candidates and the items comprising the target outfit. The features are described in more detail in Section 8.

The model achieves an accuracy score of 0.799 on this task with the second-place solution scoring 0.764.

2 RELATED WORK

According to the review by Deldjoo et al. fashion outfit recommendations tasks have been predominantly approached by leveraging content-based methods, in particular, visual features, which can be partly explained by scarcity of outfit interaction data. Apart from metric-learning-based solutions, these approaches, for instance, include multi-modal neural nets that mask certain items [2] or model outfits as a sequence of items and feed them into a LSTM model [5].

Other approaches attempt to combine both contextual information and product features. A graph neural network for products conditioned on their features has been shown to achieve high score on the FITB tasks [3] and other approaches such as tensor factorization [6] also were proposed in the literature for personalised outfit recommendations. Our solution is conceptually closer to these approaches, although it is not an end-to-end training approach, but a multi-stage one.

3 THE CHALLENGE

The main goal of the Fashion Outfits Challenge of the SIGIR 2022 Workshop on eCommerce was to develop a model that is able to generate outfits for individual products. The challenge focused on the Fill in the Blank task where a missing item of a real outfit is required to be predicted from a list of candidates and the final score is defined as an average accuracy of these predictions. The challenge consisted of a development phase and a test phase and the standing in the development phase was available on a public leaderboard (although participants had an ability to hide their submissions). The results of the test stage determined the final standing.

SIGIR eCom'22, July 11–15, 2022, Madrid, Spain
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

3.1 Competition Dataset

The competition dataset was provided by FARFETCH and consisted of over 300,000 outfits created by experts from about 400,000 products. It consisted of three parts: the outfits themselves, product metadata, including both categorical and textual data, and product images—with the items photographed on a white background.

Test and development sets consisted of an "incomplete outfit"—a list of products that constituted an outfit but with one item we need to predict omitted—and a list of candidate items. An example of an incomplete outfit is shown on Figure 1.

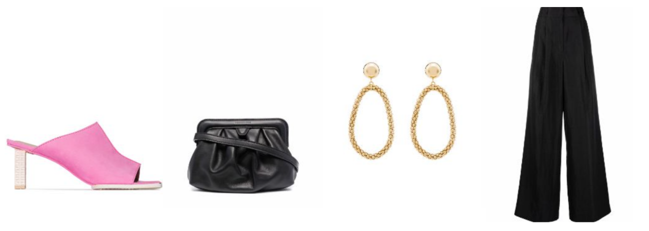

Figure 1: Example of an incomplete outfit.

The most popular items, such as a pair of white low-top sneakers appeared in about 1% of training outfits. It is possible that existence of such "anchor" items helped a collaborative filtering approach to achieve a high score on this dataset. On the other hand, about 14% of candidate items never appeared in an outfit in the training set, which means there is an upper bound on purely context-based approaches.

4 VALIDATION

Ten-fold cross-validation was used to develop the solution, as using this number of folds resulted in each validation set being about the size of the development and test sets. The folds were determined by randomly splitting the training set by outfits. In fact, the score of a single fold correlated well with the public leaderboard scores, which allowed for faster experimentation.

Selecting a right set of candidates was crucial for achieving this correlation. The candidate-generation process was based on a few observations about the development set. We noticed that the number of candidates for each incomplete outfit was uniformly distributed between 10 and 40, and for about a half of the outfits all the candidates came from a single category whereas for another half the categories were randomly distributed. Hence we constructed the list of candidate for each outfit in the training set using the following procedure:

(1) Sample an item from the outfit and assign to be the label to predict
(2) Sample an integer n from $[10, 40]$ uniform distribution
(3) To construct the list of negatives, with probability 0.5 either sample $n - 1$ items from the same category as the label or sample $n - 1$ items from all products in the training set

5 PRE-PROCESSING AND COLLISIONS

We noticed that about 1% of candidate lists in development and test datasets included a collision, i.e. an item occurring twice in the list. We hypothesised that this item was the true label and the collision occurred due to a sampling procedure similar to the one described above performed by the creators of the dataset. Submitting predictions based on this hypothesis improved the score, and was a part of post-processing the predictions. The gain from this post-processing technique was 0.0048 on the development stage using the first-stage model only (see the following sections for details on the model). The effect was not measured using the full two-stage approach on the public leaderboard but it is likely to be much smaller as the majority of these collisions overlapped with the model predictions.

We also removed one observation from the development set that had anomalously long list of items in the incomplete outfit field. Visualizing the items confirmed that they are unlikely to constitute an outfit.

6 TWO-STAGE APPROACH

The approach consisted of two stages, as depicted in Figure 2. In the first stage we used an item-to-item recommendation model based on Alibaba's Swing algorithm [9]. This model used complete outfits from the training set and incomplete outfits from the development and test sets to learn item compatibility. The features derived from these compatibility scores were combined with features based on product metadata and comparisons between incomplete-outfit items and candidates. The full set of features was then used in a gradient boosting machine model based on LightGBM framework [7] to predict whether a candidate item is the target missing item and the item with the highest predicted probability is the final prediction.

7 ITEM COMPATIBILITY SCORE

We derived item compatibility using an algorithm based on a version of Alibaba's Swing algorithm [9]—with parameter values that are based on heuristics [8]. It was implemented as follows:

(1) Each outfit k consisting of a set of items $items_k$ is assigned a score ω_k inversely proportional to the number of items in the outfit:

$$\omega_k = \frac{1}{(|items_k| + 5)^{0.35}}$$

(2) For any pair of items (i, j) we want to calculate a similarity score we collect two sets of outfits in which either of the items occur[1]:

$$outfits_i = \{k \mid i \in items_k\}$$

$$outfits_j = \{k \mid j \in items_k\}$$

(3) We compare outfits from these sets pairwise. If an outfit $a \in outfits_i$ has common items with an outfit $b \in outfits_j$ we calculate an outfit-pair score as follows:

$$outfit\_score(a, b) = \frac{\omega_a * \omega_b}{1 + |items_a \cap items_b|}$$

[1]Original algorithm only considers sets that contain both i and j [9].

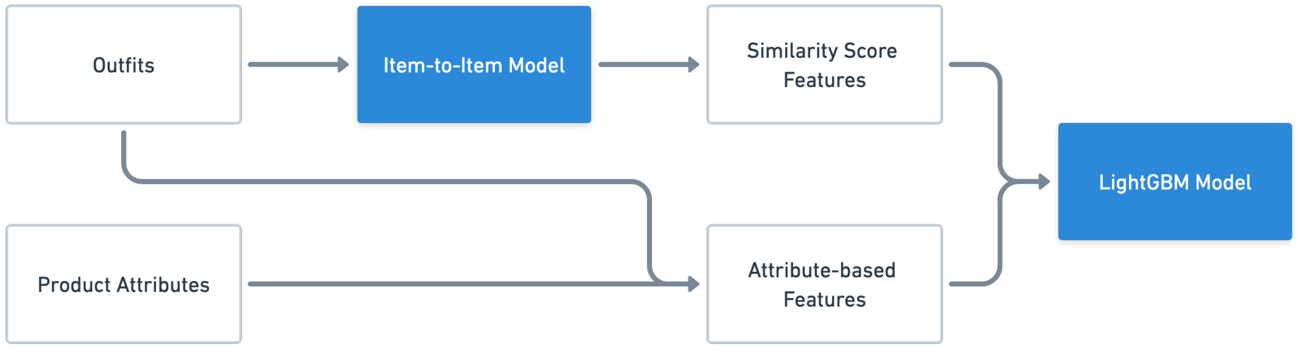

Figure 2: Two-stage solution architecture.

where $|items_a \cap items_b|$ is the size of the set of common items between outfits a and b[2]. The score is zero if the outfits do not have common items.

(4) The compatibility score for the pair of items (i, j) is the sum of scores for all such pairs of outfits:

$$score(i, j) = \Sigma_{a \in outfits_i, b \in outfits_j} outfit\_score(a, b)$$

This collaborative-filtering model was implemented in PySpark. Despite being based on heuristics it outperformed other models such as an ALS model and a matrix factorization model estimated with gradient decent, both fine-tuned on a validation set. With a development-phase accuracy of 0.68 it would have likely placed in top 3–4 on the final leaderboard by itself.

However, this model, being based on interactions, cannot address the cold-start problem. We can estimate an upper bound for this approach using our validation set. We calculate the share of labels in a validation fold that do not appear in the remaining folds' outfits. This number is 0.784, averaged across ten folds. This is about 10 percentage points higher than the score item-to-item model achieves, however it is lower than 1 by a large margin. Hence for the second stage we combined compatibility score features with other features based on product attributes.

7.1 Incomplete Outfits

We found that even incomplete outfits improved the performance of the first-stage model. The FITB score of the item-to-item model was improved by adding incomplete outfits to the training set. Using additional data from test and development phases we measure an increase in accuracy of 0.01 on validation folds.

8 FEATURE ENGINEERING AND RANKING

8.1 Outfit-Candidate Compatibility Score

The most important set of features for the ranking model was based on the output of the item-to-item model. First we calculated pairwise compatibility scores between each item in an incomplete outfit and a candidate. Then we aggregated them by calculating max, mean, min, standard deviation of these scores at an outfit-candidate level.

We also calculated these summary statistics for non-zero scores only, i.e. for the pairs of items that had some overlap in the training set.

8.2 Outfit Features

The second set of features was based on the metadata of the items in incomplete outfits. We one-hot-encoded hierarchical categorical features, such as category and subcategory and other categorical features, such as gender, and sum them over the items. For each of these variables we omitted the categories that represent less than 0.5% of the items and encode them as "Other".

To represent the colour of the outfit we one-hot-encoded both main and second colours and sum them over outfits with a weight of 0.5 assigned to the secondary colour. Similar to other categorical features we omitted unpopular categories.

8.3 Candidate Features

Hierarchical categorical features, gender, colour and brand for each candidate item were also added to the feature set. We did not one-hot-encode them and used the categorical feature encoder built into LightGBM. We did omit unpopular categories, similar to other categorical features.

8.4 Item-Candidate Features

The third set of features was constructed by comparing values of categorical features between items in incomplete outfits and candidates pairwise. A binary variable was assigned a value of 1 if both an item and a candidate belong to the same category, e.g. to the same category of products, and 0 otherwise. This was done for hierarchical features, colours, brands and gender. These binary variables were then averaged over outfits, giving us an outfit-candidate level feature.

8.5 Ranking Model

The final set of features consisted of 112 elements that are listed in Section A. The ranking of the candidate items was formulated

[2]Intuitively, if two outfits contain many common items, a wide range of items could complement either of them, so we learn little about how items i and j are similar. If they only have few items in common (and contain few items themselves), this implies much stronger relationship between items i and j.

as a binary classification problem and solved using LightGBM implementation of the gradient boosting tree algorithm. The hyperparameters were tuned with the help of the Optuna library [1] using FITB score on held-out validation set as a target metric. The resulting hyparameters are listed in Table 1.

Table 1: LightGBM hyperparameters.

Parameter	Value
boosting_type	gbdt
objective	binary
num_boost_round	10000
early_stopping_rounds	100
metric	binary_logloss
learning_rate	0.02
bagging_fraction	0.94
bagging_freq	2
feature_fraction	0.426
lambda_l1	0
lambda_l2	0
min_child_samples	46
num_leaves	128

The most important features as measured by gain are shown on Figure 3. The top is dominated by features based on summary statistics of compatibility score that occupy five out of the top seven spots that stand out from the rest of the features as measured by gain. The other two features in the top seven are related to brands: categorical encoding of candidate items' brands and a measure of candidate item belonging to the same brand as other items in the outfit. Outfit features seem to be the least important for the ranker model.

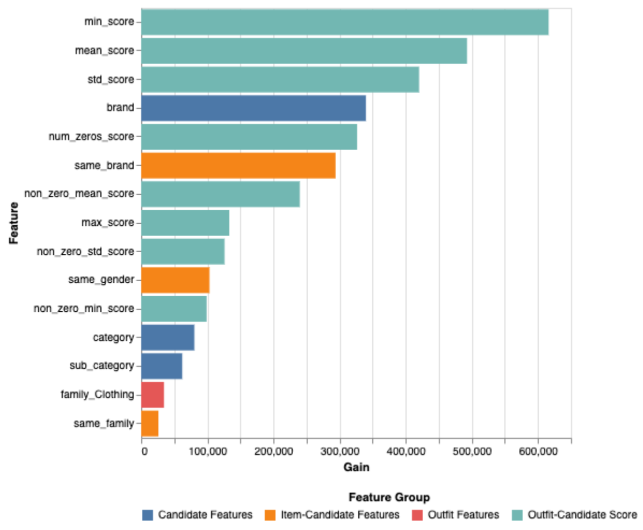

Figure 3: Top features by gain.

9 ENSEMBLING AND RESULTS

9.1 Results Across Stages and the Cold-Start Problem

We can compare results across the stages using our validation setup. Table 2 contains results for the first-stage model (item-to-item model), the two-stage approach and the ranking model trained only on aggregated product attributes. The accuracy numbers are averaged over ten folds and presented together with 95% confidence intervals. We can see that combining item-to-item model with attribute-based features adds around 10 percentage points to the item-to-item model. We can also see that the attribute-based-features model performs worse than the item-to-item model by about 8 percentage points, however it is likely that more features can be constructed to close this gap.

Table 2: Ablation study.

Model	Accuracy
Attribute-based features only	0.606 ±0.018
Item-to-item model	0.689 ±0.007
Two-stage approach	0.796 ±0.012

The final model is likely to underperform for new items and candidates that are not well-represented in training data due to the first-stage model's reliance on interactions. However, the attribute-based-features model still achieves the score of 0.606, which is about 76% of the performance of the final model. This implies that the model can be used to address the cold-start problem in these cases, given that we have access to the candidate's attribute data. Moreover, the ranking model does not use any features based on description, material or features extracted from images, which leaves potential to push the score higher.

9.2 Final Submission

The two-step estimation procedure described above was conducted for each of the ten folds with the final output of the ranking model retained. The scores were ensembled by taking the mean of the output scores and picking the candidate with the highest average scores as the final prediction. This approach outperformed averaging over ranks when using a single validation fold as a test set and ensembling the remaining nine. Therefore the score averaging approach was used in the final model as well.

As a part of post-processing the collision items were assigned as predictions in a few cases where they did not match the predictions from the model.

The final score of 0.799 on the test set ranked first on the private leaderboard with the second-place score of 0.764.

10 CONCLUSION

This paper describes the winning solution for the Fashion Outfits Challenge of the SIGIR 2022 Workshop on eCommerce. The proposed solution consists of two stages and leveraged both contextual information and product metadata.

Despite achieving the highest score on the leaderboard we believe there is a large room for improvement, with improvements potentially coming from leveraging both content and context information better. For instance, for simplicity, we did not utilise any text fields such as item descriptions and materials or any features based on images. However these features are usually the most informative features in fashion recommendation tasks. Hence creating more hand-crafted features even within the proposed framework (for example, by calculating TF-IDF scores for description fields and averaging them over outfits) could improve the accuracy. A more promising approach could be combining contextual information and multi-modal inputs in a single deep neural network—for instance, a transformer model trained with an MLM task that resembles FITB task very closely.

We believe that addressing the cold-start problem in the context of this task is the most impactful direction for further development as proposing matching outfits for new items is likely the main industry application for these models.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[2] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: personalized outfit generation for fashion recommendation at Alibaba iFashion. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2662–2670.

[3] Guillem Cucurull, Perouz Taslakian, and David Vazquez. 2019. Context-aware visual compatibility prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12617–12626.

[4] Yashar Deldjoo, Fatemeh Nazary, Arnau Ramisa, Julian Mcauley, Giovanni Pellegrini, Alejandro Bellogin, and Tommaso Di Noia. 2022. A review of modern fashion recommender systems. *arXiv preprint arXiv:2202.02757* (2022).

[5] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. 2017. Learning fashion compatibility with bidirectional lstms. In *Proceedings of the 25th ACM international conference on Multimedia*. 1078–1086.

[6] Yang Hu, Xi Yi, and Larry S Davis. 2015. Collaborative fashion recommendation: A functional tensor factorization approach. In *Proceedings of the 23rd ACM international conference on Multimedia*. 129–138.

[7] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[8] Eugene Yan. 2021. *Real-time Machine Learning For Recommendations*. Retrieved June 27, 2022 from https://eugeneyan.com/writing/real-time-recommendations/

[9] Xiaoyong Yang, Yadong Zhu, Yi Zhang, Xiaobo Wang, and Quan Yuan. 2020. Large Scale Product Graph Construction for Recommendation in E-commerce. *arXiv preprint arXiv:2010.05525* (2020).

A LIGHTGBM MODEL FEATURES

The full list of features used in the ranking model:

Outfit-Candidate Compatibility Score features. mean_score, max_score, min_score, std_score, num_zeros_score, non_zero_mean_score, non_zero_min_score, non_zero_std_score.

Outfit Features. family_Accessories, family_Activewear, family_Bags, family_Clothing, family_Jewellery, family_other, family_Shoes, category_Backpacks, category_Belt_Bags, category_Belts, category_Boots, category_Bracelets, category_Clutch_Bags, category_Coats, category_Denim, category_Dresses, category_Earrings, category_Hats, category_Jackets, category_Knitwear, category_Loafers, category_Messenger_Bags, category_Mini_Bags, category_Mules, category_Necklaces, category_other, category_Pumps, category_Rings, category_Sandals, category_Satchels_&_Cross_Body_Bags, category_Scarves, category_Shirts, category_Shorts, category_Shoulder_Bags, category_Skirts, category_Sunglasses, category_Sweaters_&_Knitwear, category_T-Shirts_&_Vests, category_Tops, category_Tote_Bags, category_Trainers, category_Trousers, sub_category_Blazers, sub_category_Blouses, sub_category_Cardigans, sub_category_Cropped_Trousers, sub_category_Day_Dresses, sub_category_Flared_Trousers, sub_category_Hi-Tops, sub_category_High-Waisted_Trousers, sub_category_Hoodies, sub_category_Jumpers, sub_category_other, sub_category_Knitted_Tops, sub_category_Low-Tops, sub_category_N/D, sub_category_Regular_&_Straight-Leg_Jeans, sub_category_Regular_&_Straight-Leg_Trousers, sub_category_Shirts, sub_category_Straight_Trousers, sub_category_Straight-Leg_Jeans, sub_category_Sweatshirts, sub_category_T-Shirts, sub_category_T-shirts_&_Jerseys, sub_category_Tailored_Trousers, sub_category_Track_Pants, sub_category_Vests_&_Tank_Tops, gender_MEN, gender_UNISEX, gender_WOMEN, colour_BLACK, colour_BLUE, colour_BROWN, colour_DO_NOT_USE_-_Beige, colour_DO_NOT_USE_-_IVORY, colour_DO_NOT_USE_-_NAVY, colour_GOLD, colour_GREEN, colour_GREY, colour_METALLIC, colour_MULTICOLOUR, colour_N/D, colour_NEUTRALS, colour_ORANGE, colour_PINK, colour_PURPLE, colour_RED, colour_SILVER, colour_WHITE, colour_YELLOW.

Candidate Features. family, category, sub_category, gender, main_colour, second_color, brand.

Item-Candidate Features. same_family, same_category, same_sub_category, same_gender, same_main_colour, same_second_color, same_brand.