Dataset and Algorithms for Interactive Fashion Outfit Recommendation

FANG ZHOU

Master's Program in Informatics
Degree Programs in Comprehensive Human Sciences
Graduate School of Comprehensive Human Sciences
University of Tsukuba
March 2022

Dataset and Algorithms for Interactive Fashion Outfit Recommendation

Name: FANG ZHOU

Fashion stands an important position in our life. It generally showed personal preferences to others. Compared with other items, the sense of fashion items is constantly changing. In recent years, many proposed interactive recommender systems have used the interaction between the user and system to add the factor of preferences moving. However, to the best of our knowledge, there is no existing interactive system in the field of fashion recommendation.

This thesis proposes a new fashion outfit recommendation task where an outfit is recommended after several rounds of user-system interactions. In this task, the system is expected to present a question (a set of fashion items) at each round, and the user is expected to select the best fashion item from the presented items. The system is required to gradually learn users' preferences based on their responses and to generate subsequent questions for recommending a suitable outfit based on the users' selected fashion items. Since there was no dataset available for this task, we developed a new dataset for evaluating interactive fashion outfit recommendations through simulation. Based on the analysis of the developed dataset, we devised an experimental interactive fashion outfit recommendation algorithm based on a learning-to-rank model with hand-crafted features, including category information, similarity, and heterogeneity. Moreover, we proposed a new deep learning structure to achieve higher accuracy in the interactive recommendation task. The structure supports encoding questions by Transformer or Deep Set and estimates the value of questions based on previous questions and given answers. The experimental results demonstrated that the proposed models outperformed the learning-to-rank model, and Set Transformer is a suitable model for embedding questions in the interactive fashion outfit recommendation task. Moreover, we also created a practical training pipeline for proposed models. With the collected comparative data, we found that the pipeline helps to enhance training efficiency with no impact on model performance. Besides, we also conducted an ablation study to fully investiage the effects of hyper-paprameters of the proposed model. The results indicate that the feature of the first question and the corresponding answer is most important in prediction.

Main Academic Advisor: Makoto P. KATO
Secondary Academic Advisor: Haitao YU

Contents

1 Introduction **1**

2 Related Work **4**
2.1 Fashion Recommendation 4
2.2 Interactive Recommender Systems 5
2.3 Bidirectional Long Short-Term Memory for Fashion Recommendation 5
2.4 Visual-Semantic Embedding for Fashion Items 6

3 Interactive Fashion Outfit Recommendation **8**
3.1 Definition 8
3.2 Workflow 8
3.3 Task 10

4 Dataset **11**
4.1 Source Data 11
4.2 Assumption for Simulation 11
4.3 Question Generation 13
4.4 Analysis 14

5 Proposed Methods **19**
5.1 Learning-to-rank Question Selector 19
5.2 Deep Question Selector (DQS) 19

6 Experiments **22**
6.1 Experimental Setup 22
6.2 Results 22
6.3 Training Pipeline 24
6.4 Difference between Aggregators 25
6.5 Case Study 26
6.6 Ablation Study 26

7 Conclusions **29**

Acknowledgements **30**

References **31**

A Experiment Environment **35**

B The URL of Pictures Used in This Thesis **36**

List of Figures

2.1 The architecture of bidirectional LSTMs for fashion outfits. 7

3.1 The workflow of the interactive fashion outfit recommender system. 9

4.1 Percentage of the best and worst questions in each category. 15
4.2 Percentage of the best and worst questions in each category with different categories of the initial question. 16

5.1 The architecture of Deep Question Selector (DQS). 20

6.1 Accuracy of the proposed models in different categories. 23
6.2 The example of question selection. Each question contains five different fashion items; the system is required to select the best question to the answer predicting in the third question. See Appendix B for the URLs of listed item images. 27
6.3 Accuracy, nDCG@1, and nDCG@5 of the ablations of Q_1, (Q_1, a_1), and (Q_1, a_1, Q_2). 28

List of Tables

4.1 Example of questions in our dataset. 12
4.2 Dataset statistics. 14
4.3 Number of question sets with the different number of questions. 14
4.4 Effectiveness of the best and worst questions. "All" indicates that of all the questions. 17
4.5 Average similarity to the previous question and heterogeneity of the best and worst questions. "All" indicates those of all the questions. 17

6.1 The performance of the proposed methods. 23
6.2 The performance of DQS models with the special training pipelines. 24
6.3 Training time of DQS models with the special training pipelines. 24
6.4 The performance of DQS models with the special training pipelines. 25
6.5 Training time of DQS models with the special training pipelines. 25
6.6 The result of representative methods in the case study 26

Chapter 1

Introduction

With the rapid development of information technologies and the internet, we have gradually entered an era of overloaded information. In this era, both producers and consumers face significant challenges. For consumers, finding required or exciting information from a sea of information has been exceedingly arduous. On the other hand, producers are also distressed by making their information stand out and capturing the attention of a vast number of targets. The recommender system is a superior technology for solving these problems. Recommender systems establish the bridge between consumers and producers, navigating the users to the appropriate content. In most practice, deploying recommender systems significantly increases the revenue and improves click-through rate (CTR) and the number of conversations. Nowadays, recommender systems have been broadly used in various online web services, especially online shopping sites.

The recommendation of fashion items, one of the largest markets in online shopping, has drawn the attention of researchers in the last decades. Unlike traditional item recommender systems, fashion recommendation has unique challenges as fashion items are always fitted together with other items. Therefore, when items are recommended to users, their compatibility with other items should be considered by recommender systems. There have been many studies on fashion compatibility, which have been evaluated with real fashion datasets [1, 2, 3, 4].

Although one of the unique characteristics of fashion recommendation, compatibility, has been extensively studied in the literature, there are several aspects ignored in the existing fashion recommendation studies. One of such aspects is *interactivity*. Existing fashion recommendation is passive: the recommendation is based on historical, implicit feedback from users and does not actively interact with users. As can be seen in real fashion shopping interaction between customers and store staff, however, interaction in fashion recommendation is vital to capture the user preferences changing over time, and elicit potential user needs that are often difficult to express. To the best of our knowledge, although various fashion recommendation datasets are available for traditional fashion recommendation settings [5, 1, 6], there are no studies or datasets for interactive fashion recommendation.

To bring interactivity to the fashion recommendation, we propose a new fashion outfit recommendation task where an *outfit*, which consists of multiple fashion items, is recommended after several rounds of user-system interactions. In this task, the system is expected to present a *question*, which comprises multiple fashion items, at each round, and the user

is expected to select the best fashion item from the presented items. The system is required to learn users' preferences based on their responses gradually and to generate subsequent questions for recommending a suitable outfit based on the users' selected fashion items. This setting is inspired by real fashion shopping interaction between customers and store staff: a store staff recommends some fashion items to a customer and tries to find other fashion items that better fit both customers' preferences and their chosen items.

Since there was no dataset available for this task, we developed a new dataset for evaluating interactive fashion outfit recommendations. Our dataset consists of fashion items from Polyvore [6], one of the most used datasets for fashion outfit recommendation, and includes 16,768 question candidates that can be asked in the interactive fashion recommendation task. Using the developed dataset, researchers can simulate users' responses to a given question and evaluate interactive fashion outfit recommendation systems.

We analyzed the developed dataset and evaluated the importance of question selection in interactive fashion outfit recommendations. The analysis showed that (1) the question selection greatly matters for the recommendation performance, (2) the category of fashion items in the question affects the recommendation performance, and (3) effective questions are likely to be similar to the previous question and include similar fashion items. Based on these findings, we extracted several features from questions, and developed a question selection model based on a learning-to-rank algorithm. Moreover, we also proposed *Deep Quesion Selector*, which is a new deep learning model to achieve higher accuracy in the interactive recommendation task. The structure of Deep Quesion Selector can be nested to fit interactive recommendation tasks with different problem settings. Deep Question Selector encodes questions by the aggregator and estimates the value of questions based on previous questions and given answers. The aggregator can be customized for different problem domains and settings, such as Deep Set [7] and Set Transformer [8]. In this thesis, we investigated the performance difference between Deep Question Selector models with different aggregators. Additionally, we present a practical training method for Transformer Deep Question Selector models that reduces the training time while keeping the accuracy.

We conducted experiments with the developed dataset for evaluating our proposed methods. The experimental results demonstrated that the models with proposed structure outperformed the learning-to-rank model with hand-crafted features, and Set Transformer is a suitable model for embedding questions in the interactive fashion outfit recommendation task. Furthermore, the proposed training method for the Transformer Deep Question Selector models successfully reduced the training time without sacrificing the accuracy.

The contributions of this work are summarized as follows.

1. We proposed a new fashion outfit recommendation task that involves user-system interactions.

2. We developed the first interactive fashion recommendation dataset that can be used for simulation-based evaluation.

3. We proposed a learning-to-rank algorithm for the interactive fashion recommendation.

4. We presented a customizable model structure to improve the recommendation accuracy and compared the effectiveness of different feature aggregators.

5. We developed a learning pipeline that reduces the training time without sacrificing accuracy.

Chapter 2 gives a review of the related works. The interactive fashion outfit recommendation task is defined in Chapter 3. Chapter 4 analyzes the developed dataset and reveals the properties of the proposed task and dataset. Chapter 5 introduces our proposed methods. Chapter 6 displays and discusses findings from a series of experiments. Finally, we conclude this work in Chapter 7.

Chapter 2

Related Work

In this chapter, we provide a literature review in the fields related to this work, which are fashion recommendation and interactive recommender systems.

2.1 Fashion Recommendation

Fashion, a form of self-expression and autonomy at a specific time or location, has been a large part of daily life. Generally, clothing is the most crucial part of fashion. People are constantly debating which clothes to wear or purchase. The fashion recommendation task seeks to recommend appropriate fashion items or outfits, as well as a series of fashion items, based on given information such as the user's purchase history and item features.

Kang et al. proposed a support vector machine (SVM) and collaborative-filtering-based fashion recommender systems to improve customer satisfaction [9]. Liu et al. introduced an occasion-oriented clothing recommender system that relies on a latent SVM framework with attributes and categories [10]. With the massive success of online apparel sales in the last decade and a large amount of user data, many fashion recommender systems have been proposed in recent years [11, 12, 13, 14]. Li et al. [14] presented a recurrent neural network (RNN) model with fused features of the item to predict its popularity. Through large-scale experiments, they confirmed that the proposed method is efficient.

In contrast to fashion item recommendation, the outfit recommendation task requires an extra ability to evaluate the compatibility of items [1, 3, 2, 4]. The performances of fashion outfit recommender systems are evaluated in the following two tasks:

Fill in the blank This task is to choose from multiple items to complete an outfit. For instance, given four items, as well as three candidate items, the task is to choose one of the candidate items that best fits as the fifth item to complete an outfit consisting of five items.

Compatibility prediction This task is to predict the compatibility of the given outfit. The compatibility of a fashion outfit measures the suitability of the items to the style of the overall outfit. In most cases, the compatibility of the outfit indicates the affinity between each pair of items.

Notably, fashion is constantly changing. In other words, the opinion toward fashion is unstable. Many studies on fashion recommendation suggest that contextual information

need to be counted in recommendation [15, 16]. When the context is not informative enough to capture rapidly changing users' preferences, users' feedback is necessary to understand the current mood of users. However, there are no studies that actively elicit users' preferences for fashion recommendation.

2.2 Interactive Recommender Systems

Recommender systems have been widely used in various services for increasing commercial revenue. However, there are two problems in the existing recommendation systems: (1) users' experiences are monotonous: users can only passively receive recommended items [17, 18], and (2) systems have limited knowledge about users for effective recommendation [19]. A possible solution to alleviate these two problems is to allow users to interact with the recommender systems. Interactions enable users to feel more involved and intervene in recommendation results to a certain extent, rather than passively receiving results [20]. Moreover, the system can better understand the users' preferences through interactions, and, accordingly, improve the recommendation results.

Christakopoulou et al. [21] proposed a single-round interactive framework with three modules: question generation, user feedback, and item recommendation. The user needs to answer the question generated by the system, and the system predicts relevant items based on the user's feedback.

The effectiveness of interactive recommender systems has been shown in many fields [22, 23, 24, 25]. While the interactive recommendation is thought to be a superior solution in some applications where users are willing to provide feedback, the evaluation of interactive recommendation systems is challenging as it often requires real recommendation services. Another evaluation methodology is a simulation in which users behaviors are simulated based on the observation in real services. While the simulation may not fully reflect real users' behaviors, the evaluation is reproducible and easily repeatable. Thus, simulation-based evaluation can be useful for initial development of interactive systems. We took the latter approach in this study and developed the dataset for evaluating interactive fashion outfit recommender systems through simulations.

2.3 Bidirectional Long Short-Term Memory for Fashion Recommendation

Since deep learning was established, it has surprisingly improved the state-of-the-art in many different artificial intelligent tasks [26]. The recurrent neural network (RNN) is one of the most popular neural networks. In contrast to others, the structure of RNN enables the leveraging of spatial information. Thus, in many sequential learning tasks, RNN leads the top-level performance to others. In recent years, many works have broadly used a special RNN – long short-term memory (LSTM) because of its overwhelming advantage in handling long-term features. Bidirectional long short-term memory (BiLSTM) is an extension of the original LSTM [27]. BiLSTM uses two LSTMs – forward LSTM and backward LSTM to enable the encoding of backward information.

Given a sequence of fashion items $\mathbf{F} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_t$ is the feature representation of t-th fashion item in the outfit. The model uses a forward LSTM to predict the next item images for learning the relationships among fashion items. The objective function of forward LSTM can be written as follows:

$$E_f\left(\mathbf{F}; \mathbf{\Theta}_f\right) = -\frac{1}{N}\sum_{t=1}^{N} \log \mathrm{P}\left(\mathbf{x}_{t+1} \mid \mathbf{x}_1, \ldots, \mathbf{x}_t; \mathbf{\Theta}_f\right) \tag{2.1}$$

where $\mathbf{\Theta}_f$ is the model parameters of forward LSTM, $\mathrm{P}(\cdot)$ is the predicted probability of seeing the $t+1$-th item based on the inputs from previous steps. Similarly, the backward LSTM is trained for predict the previous items, and the objective function is following:

$$E_b\left(\mathbf{F}; \mathbf{\Theta}_b\right) = -\frac{1}{N}\sum_{t=N-1}^{0} \log \mathrm{P}\left(\mathbf{x}_t \mid \mathbf{x}_N, \ldots, \mathbf{x}_{t+1}; \mathbf{\Theta}_b\right) \tag{2.2}$$

where $\mathbf{\Theta}_b$ denotes the parameters of backward prediction model.

Through the training of minimizing the two objective functions, the BiLSTM finally gathered the ability to capture the overall information over the inputs. In our work, we use it for evaluating the compatibility of generated fashion outfits.

In this thesis, we leverage the BiLSTM model proposed by Han et al. [6] to ensure that the items in question are potentially relevant to the initial relevant item. Each fashion outfit is a set of multiple items. The items in the same outfit should share similar features like colors and seasonality. As shown as Figure 2.1, given multiple fashion items, the model can predict the fashion compatibility of these items via bidirectional LSTMs.

2.4 Visual-Semantic Embedding for Fashion Items

Traditional embedding is generally generated with information from one aspect. However, the property of an entity in the real world is always multifaceted, such as text, images, the number of likes. Frome et al. proposed *Deep Visual-Semantic Embedding Model* (DeViSE) [28] to solve the challenge of modeling with rich information. By fine-tuning the visual embedding with the semantic embedding extracted by word2vec, DeViSE provides the embedding of the given input with both visual and textual features.

The kernel of DeViSE is that learning with textual embeddings rather than the one-hot encoded target. After fine-tuning with given textual features, the model gains the ability to generate visual-semantics embeddings (VSE). In order to better utilize the rich information of fashion items, we leverage the VSE of items trained with item images and descriptions.

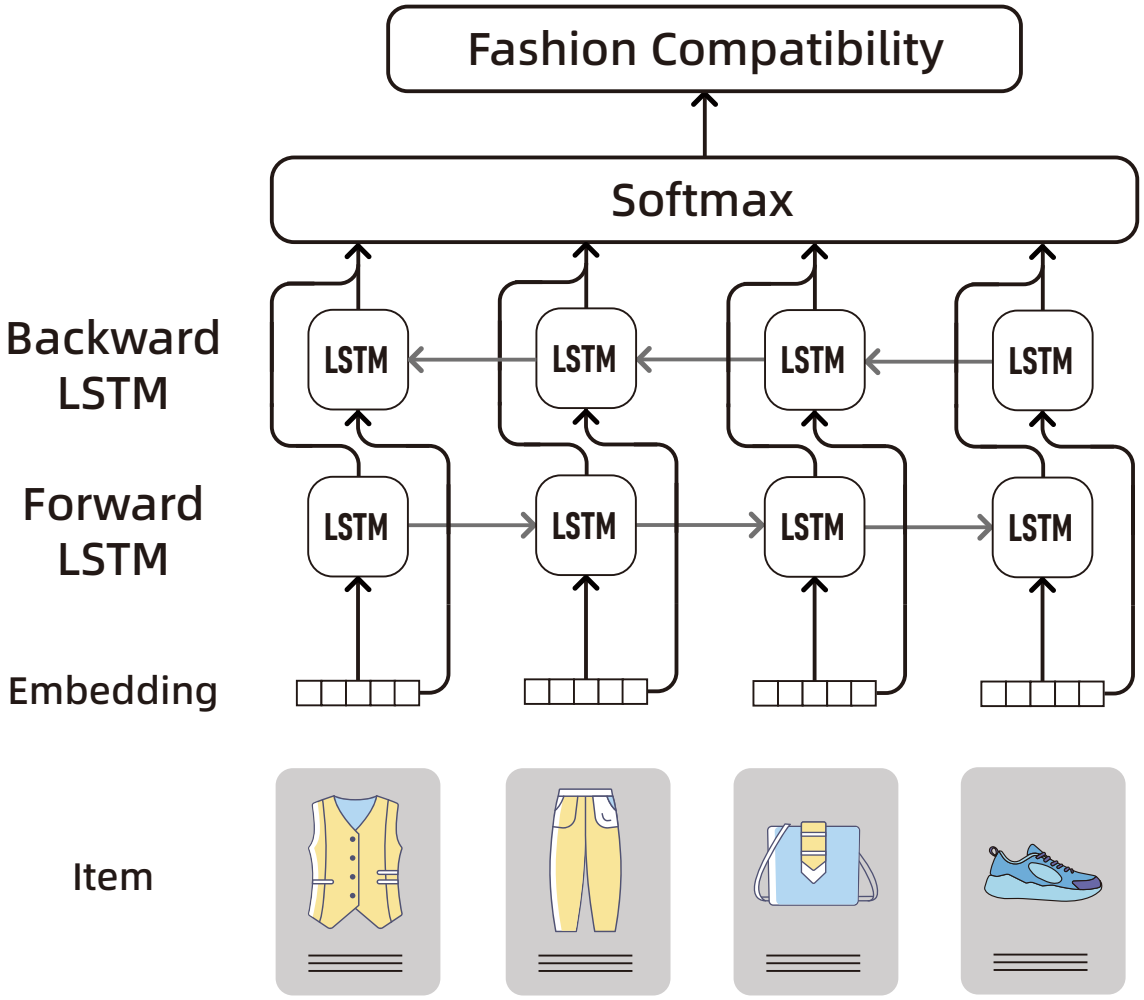

Figure 2.1: The architecture of bidirectional LSTMs for fashion outfits.

Chapter 3

Interactive Fashion Outfit Recommendation

In this chapter, we describe the workflow of an interactive fashion outfit recommender system and provide the details of the interactive fashion outfit recommendation task.

3.1 Definition

Suppose we have a collection I consisting n different fashion items, i.e., $I = \{i_1, i_2, \ldots, i_n\}$. Each fashion item from the collection belongs to a category, e.g., tops, bottoms and dresses etc. The category of a specific item should be one of $C = \{c_1, c_2, \ldots, c_m\}$, where C is the collection of all categories and m denotes the number of categories. The item collection I can be split to m disjoint subsets of different categories, i.e., $I = I_{c_1} \cup I_{c_2} \cup \cdots \cup I_{c_m}$.

A question is defined as a set of fashion items and is used to let the user choose from multiple items. To be comparable each other, fashion items in a question belong to the same category. Thus, a question is defined as $Q = \{i_1, i_2, \ldots, i_k\}$ and is a subset of I_c of a certain category c (k is the number of items in a question).

Similarly, outfit O is defined as a set of items, i.e., $O \subset I$. In contrast to the question Q, O contains items of different categories to form a complete outfit.

3.2 Workflow

Figure 3.1 illustrates the overall workflow of the interactive fashion outfit recommendation system. The recommender system composes and presents a question consisting of multiple fashion items. The user answers the question by choosing the best item from the set of fashion items. After several rounds of question-answering, the system is expected to recommend outfits that include selected items and meet the user's preference.

The interactive fashion outfit recommender system must be able to provide *good* questions for effectively estimating the user's preference, and to recommend outfits compatible with fashion items given as answers to the questions. While the outfit recommendation component can be any existing models proposed in the literature (e.g., [2]), designing a question selection algorithm is an interesting challenge specific to this task. Since items selected by the user in the previous rounds matter to the question selection in the next

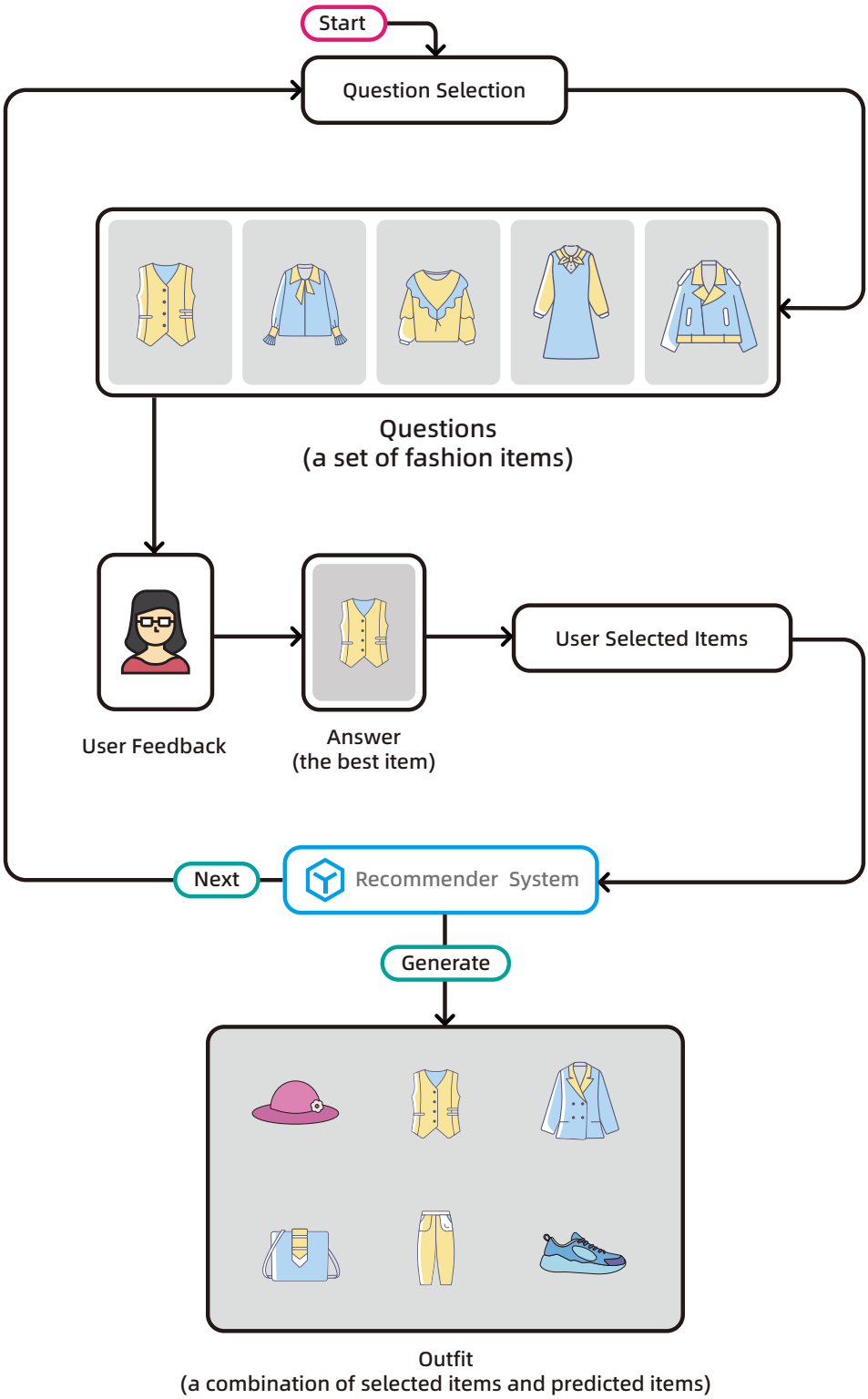

Figure 3.1: The workflow of the interactive fashion outfit recommender system.

round, the question selection algorithm should be able to identify questions that facilitate the following interactions and achieve high accuracy in the outfit recommendation.

3.3 Task

Having described the workflow of the interactive fashion outfit recommender system, we formally define the task of the interactive fashion outfit recommendation.

There are three main components in this task: the user U, question selection algorithm S, and outfit recommendation algorithm R. The user is a function that returns an item for a given question: $U : Q \mapsto i$ where $i \in Q$. Given a question-answering history up to time step t, $H_t = \{(Q_1, a_1), (Q_2, a_2), \ldots, (Q_t, a_t)\}$, the question selection algorithm returns a question based on H_t, i.e., $S : H_t \mapsto Q$. The outfit recommendation algorithm is a function that, given the question-answering history H_t and outfit O, returns a score of O with respect to H_t. A set of outfits are then ranked by the score in descending order, i.e., $R(H_t, O_a) < R(H_t, O_b) \Rightarrow O_a \prec O_b$ where $\prec$ represents the order in the outfit ranking.

With those main components, at each time step t, a question is given by $Q_t = S(H_{t-1})$. The answer at time t is given as follows: $a_t = U(Q_t)$. After T rounds of interactions, the system obtains the question-answering history H_T, and recommends outfits ranked by R. The outfit ranking can be evaluated in terms of any retrieval or recommendation effectiveness measures such as HIT and nDCG.

As the user U is out of score of the system, the goal of this task is to design effective functions for the question selection algorithm S and outfit recommendation algorithm R to achieve better performances in terms of some effectiveness measures. Although one can address both simultaneously, this work primarily focuses on the case where R is given. Therefore, our primary task is, given user U and outfit recommendation algorithm R, to find the question selection algorithm S that can achieve better performances in the outfit recommendation.

For simplicity, we focus on a special case of this task where the question selection algorithm S is defined by a question value function V:

$$S(H_t) = V(H_T, Q) \tag{3.1}$$

where V is a function that measure the value of the question Q given the question-answering history H_T of T interactions, and $\mathcal{Q}$ is a set of candidate questions. This simplification enables us to focus on measuring the goodness of questions and is suitable for the dataset explained in the next chapter.

Chapter 4

Dataset

In this chapter, we introduce a new dataset for the interactive fashion outfit recommendation task. The dataset and some utiliy scripts are publicly available at Github[1]. We also report our findings on the importance of the question selection through the analysis of the developed dataset.

4.1 Source Data

Polyvore[2] was a fashion commerce website, which allowed users to upload outfits they liked and provided several functions to facilitate users' exploration of fashion items and outfits. Since 2015, the data from Polyvore has been widely used in various fashion researches [12, 14, 29, 1]. Our dataset was constructed with the dataset developed by Han et al. [6], which consists of 21,889 outfits with 164,379 items. Each outfit contains multiple items and rich metadata such as the number of likes, hashtags, etc. Moreover, each item also has rich multimodal information, such as the description, image, and category. Note that outfits were developed by designers and used as ground truth in the outfit recommendation tasks such as "fill in the blank" and compatibility prediction.

4.2 Assumption for Simulation

To the best of our knowledge, there is no previous study on interactive fashion outfit recommender systems, and the Polyvore dataset lacks user-system interaction information. Since no dataset exists in the fashion domain that can be used for training and evaluating the interactive fashion recommender system, we intended to create a new dataset that can be applied to the interactive fashion recommender system. The unit of the dataset is a set of interactions that can represent the information of a session of recommendation. With our task set-up, the interaction should be a question-answering pair, and there must have contextual relationships between the interactions within the set. Interactive recommender systems can use these interactions to simulate interactions with the user and eventually make predictions for some of the questions that have not been asked before. The model can

[1] https://github.com/kasys-lab/polyvore-interactive
[2] https://www.polyvore.com/

Table 4.1: Example of questions in our dataset.

Question	Q_1	Q_2	Q_3
Category	Tops	Bottoms	Shoes
Outfit	T1	B1	S1
	T2	B2	S2
	T3	B3	S3
	T4	B4	S4
	T5	B5	S5

be evaluated by comparing the final prediction of the model with the ground-truth users' choices.

To generate a dataset for interactive fashion recommender systems, we propose an assumption that there is a user who desires each outfit in the Polyvore dataset, and the interactive recommender system cannot identify the exact outfit the user is looking for, but can identify candidate items for each category, one of which is a part of the desired outfit. The reason behind our assumption is that when a designer creates an outfit, generally, it also means that for each item in the outfit, there is no better item than the selected one to fit the style with the other items in the outfit. Table 4.1 shows an example of the assumed situations. For three categories "Tops", "Bottoms", and "Shoes", five fashion items are retrieved for a given user, respectively. Each of the fashion items is potentially relevant, but only a fashion item in each category is a part of the desired outfit (e.g., T1, B1, and S1 forms the desired outfit). A set of items from each category can be used as a question, as they are potentially relevant items, and the recommender system needs to identify a question to be asked at each round. In the example of Table 4.1, the recommender system can ask one of three questions $Q_1 = \{\mathrm{T1}, \ldots, \mathrm{T5}\}$, $Q_2 = \{\mathrm{B1}, \ldots, \mathrm{B5}\}$, and $Q_3 = \{\mathrm{S1}, \ldots, \mathrm{S5}\}$. When Q_1 is presented to the user, we assume that T1 is chosen by the user as an answer to Q_1. Therefore, we can simulate users' responses to some questions and automatically evaluate the performance of question selection algorithms. Note that the recommender system cannot identify the desired outfit and which item is answered by the user before the question is asked.

The task definition of the interactive fashion outfit recommendation becomes a little more specific when our dataset is used. Given a user U who prefers to an outfit O as well as a question Q, U returns $a \in Q$ as an answer such that $a \in O$. Moreover, a set of questions $\mathcal{Q}_U$ is prepared for each user U, from which the question selection algorithm is expected to choose the most appropriate question at each time step. Thus, $\mathcal{Q}$ in Equation 3.1 is replaced with $\mathcal{Q}_U$, meaning that there are, say, six candidate questions for each user.

Although this assumption enables us to simulate user-system interactions for fashion outfit recommendation, there are two limitations of this approach. One is the assumption that we can prepare a set of items, one of which is relevant to the user. Its feasibility highly depends on the fashion recommendation algorithm and can be approximated by the HIT@k metric (1 if the top-k items contain a relevant item; otherwise 0). This limitation can be alleviated if the fashion recommendation algorithm is improved. The other limitation is that questions are pre-defined and not purely generated by the interactive recommendation

system. This might prevent us from accurately evaluating the ability of interactive fashion outfit recommender systems, since they may be able to generate better questions and achieve higher recommendation accuracy with better interactions. Whereas, such systems must be able to correctly identify the best question among candidates and achieve relatively higher performances than the others. This type of assumption has been often posed in dialogue generation tasks (e.g., [30]).

4.3 Question Generation

Building a new dataset often requires huge-scale data collection. We found it feasible to use a pretrained user model to make choices instead of actual users by rethinking the purposed task. With the recent developments in fashion recommender systems, many compatibility prediction models can efficiently score outfits close to actual users. For the interactive fashion recommender system, user preference is an unknown distribution, and the system tries to predict the distribution of the user preference and perform prediction based on the last few interactions information. The challenge of the task is catching the preference feature from interactions, not the distribution itself. Both the user model and the actual user are unknown to the system. Consequently, we believe that this has little impact on the system's effectiveness.

Furthermore, as discussed in the previous section, the quality of questions is vital for realistic question-answering simulation. A qualified question should consist of relevant items, any of which can be a part of reasonable outfits. The key idea to guarantee the question quality is to choose potentially relevant items as a part of desired outfits. Since we can access the complete outfit in developing questions, we can find alternatives to a particular item in the outfit by solving the "fill in the blank" task given the other items in the outfit.

Specifically, first, we choose an outfit from the Polyvore dataset, e.g., $O = \{\mathrm{T1}, \mathrm{B1}, \mathrm{S1}\}$ in Table 4.1. To generate a question of "Tops," we input B1 and S1 to a fashion compatibility model and find suitable items that *fill in the blank*, e.g., T2, T3, T4, and T5. A question Q_1 can be composed of the original fashion item T1 as well as the output fashion items T2 – T5. Then, we iterate this process for all the fashion items in the outfit to generate a question for each category. We assume that the fashion compatibility model can identify comparably suitable fashion items for the other items, which are all reasonable options for the user.

Firstly, we initialize a question with one of the fashion items $o \in O$. Secondly, the algorithm finds $k-1$ fashion items to complete the question. A fashion compatibility model f, which takes an incomplete outfit and another fashion item to return the compatibility score of the fashion item, is used to find the $k-1$ most compatible fashion items for $O \setminus \{o\}$. Iterating this procedure, we finally obtain $\mathcal{Q}_U$ containing $|O|$ questions, each of which corresponds to each fashion item in O.

As a fashion compatibility model f, we used the Bidirectional LSTMs model to calculate the compatibility of fashion outfits, which was developed by Han et al. [6]. The number of items in a question, k, is set to 5 for ease of users' choices.

Table 4.2: Dataset statistics.

# Question sets	# Questions	# Unique items
2,758	16,768	16,768

Table 4.3: Number of question sets with the different number of questions.

Size of question sets	# Question sets
4	526
5	580
6	548
7	356
8	748

4.4 Analysis

Table 4.2 shows the basic statistics of the dataset. In the developed dataset, the items are grouped in over 380 different categories. As the number of items in some categories is lower than five, we excluded the categories with small volumes to ensure that each question can contain five items. Finally, we obtained 2,758 question sets, 16,768 questions, and 16,768 unique items. As outfits from the Polyvore dataset may have different numbers of items, the question sets generated based on outfits also have different amounts of questions. Table 4.3 reports the statistics of the size of question sets. Over 59.8% of the question sets contain at least six questions. The average number of questions per question set is 6.08.

We then investigated which questions are likely to be effective in the interactive fashion outfit recommendation task. To this end, we tested various questions and computed the effectiveness measure of the recommended outfit. More specifically, we assumed a situation where only a question had already been asked and answered by a user. All the possible questions were asked in the simulation as the second question. The effectiveness of the second question was evaluated by the fashion outfit recommendation, in which candidate outfits are limited to those consisting of the answers to the first and second questions as well as an item from another question. For example, given a set of questions $\mathcal{Q} = \{Q_1, Q_2, Q_3\}$, let Q_1 be the question already asked. We then ask Q_2 and simulate the user's answer to this question. Based on the question-answering history $H_2 = \{(Q_1, a_1), (Q_2, a_2)\}$, the outfit recommendation algorithm R ranks only the fashion outfits consisting of a_1 and a_2 as well as an item in $Q_3 = \{i_1, i_2\}$, i.e., $\{a_1, a_2, i_1\}$ and $\{a_1, a_2, i_2\}$. The effectiveness of Q_2 is the reciprocal rank of a relevant outfit in the outfit ranking, where the relevant outfit is defined as the one constituting O (an item set from which $\mathcal{Q}$ was generated). To develop the recommendation algorithm R, we used the same fashion compatibility model f used to produce questions in the dataset. The fashion compatibility model solves the "fill in the blank" problem for given items (e.g., a_1 and a_2), and ranks items to form a ranked list of outfits. For example, given a ranked list of items produced by the fashion compatibility model, (i_2, i_1), the outfit ranking is obtained as follows: $(\{a_1, a_2, i_2\}, \{a_1, a_2, i_1\})$.

We first show the average effectiveness of the best and worst questions in Table 4.4. A question is the best question if the recommendation with the question achieved the highest effectiveness among the other questions in a question set. The worst question is defined

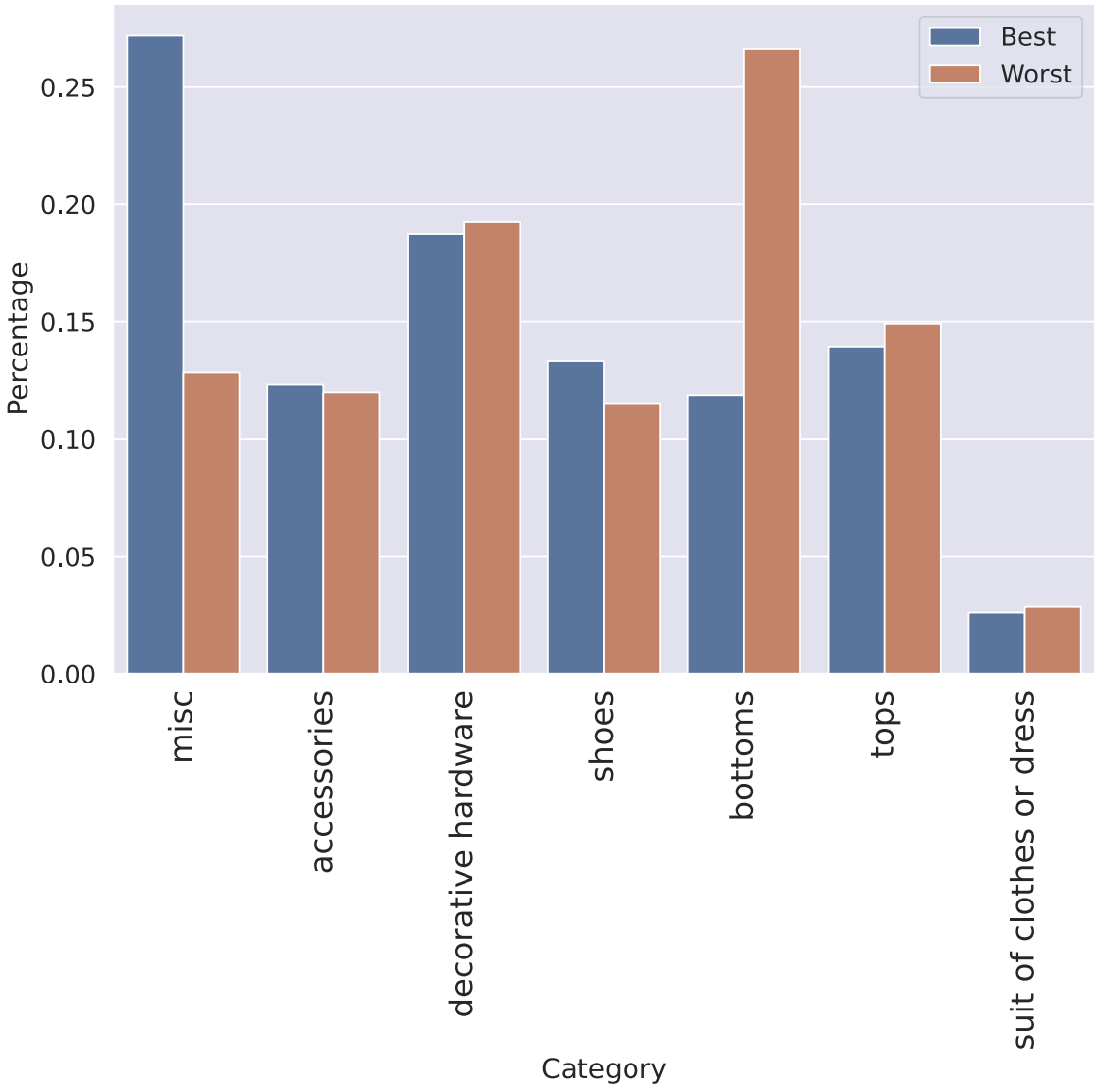

Figure 4.1: Percentage of the best and worst questions in each category.

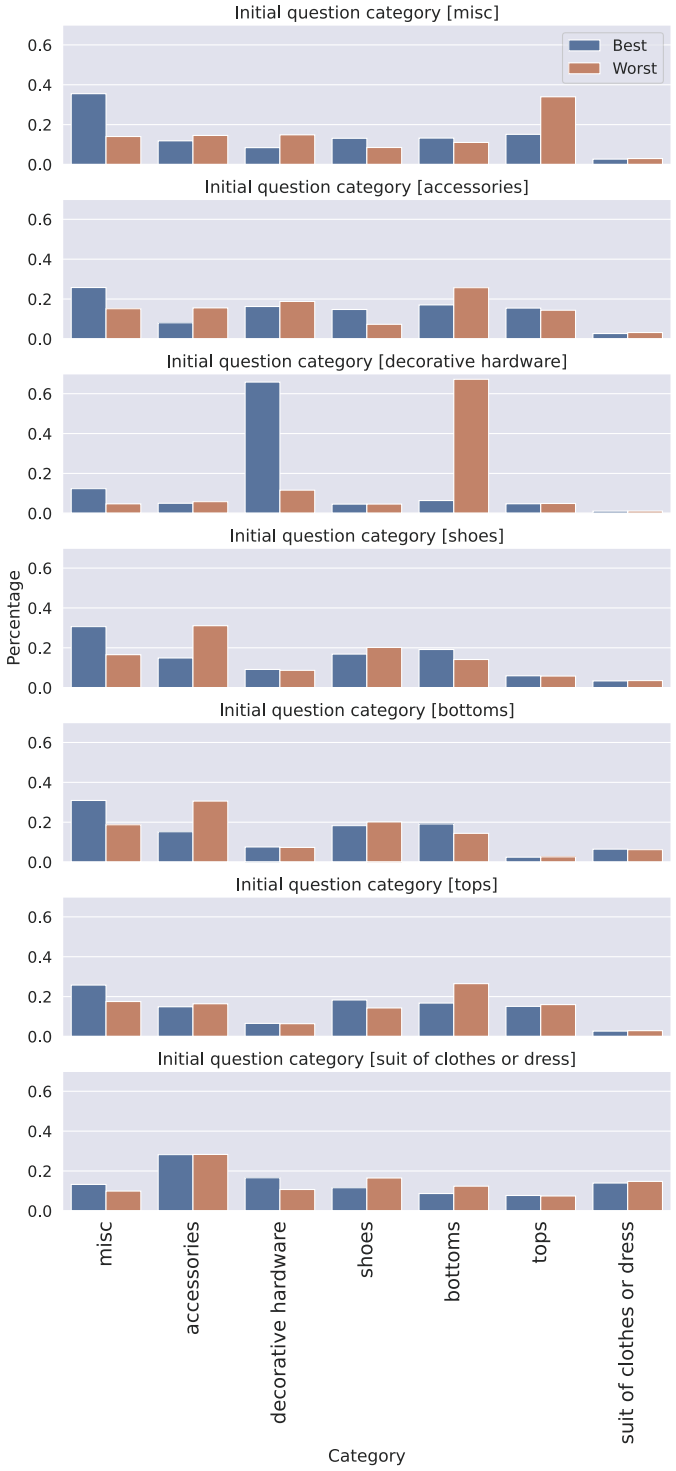

Figure 4.2: Percentage of the best and worst questions in each category with different categories of the initial question.

Table 4.4: Effectiveness of the best and worst questions. "All" indicates that of all the questions.

All	Best	Worst
0.397	0.602	0.243

Table 4.5: Average similarity to the previous question and heterogeneity of the best and worst questions. "All" indicates those of all the questions.

	All	Best	Worst
Similarity	0.352	0.365	0.307
Heterogeneity	2.484	2.460	2.546

similarly. We found that there is a large performance gap between the best and worst questions. This result suggests that the question selection algorithm has a large impact on the outfit recommendation performance.

We then show what category of questions led to effective outfit recommendations. Figure 4.1 shows the percentage of the best and worst questions in each category. Since there are over 380 different item categories in the developed dataset, we manually organized those categories into seven groups for our analysis: *a suit of clothes or dress*, *tops*, *bottoms*, *shoes*, *accessories*, *decorative hardware*, and *miscellaneous*. The figure shows large differences across the categories: the category such as *miscellaneous* shows a high probability of being the best question, while questions from the *bottoms* category are unlikely to receive helpful users' feedback.

Figure 4.2 reports the percentage of the best and worst questions in each category, with different categories of *initial questions*. Recall that we assumed a question had already been asked in this simulation: this question is defined as the initial question. With the initial question in different categories, the effectiveness of questions changes significantly. For example, following a question about decorative hardware with another question may improve the accuracy of future predictions. There are two possible reasons for these uneven performances across categories: (1) some categories (e.g., bottoms) are not informative probably because of a limited number of variations in the category. (2) the consistency of some categories can be strongly required to form good outfits. For example, the style of bottoms and shoes must be the same. In this case, it is not very informative to ask questions regarding shoes when the question about bottoms is already asked.

We further investigated the similarity between the previous question and best/worst question, and the heterogeneity of the best and worst questions. The similarity between two questions is defined as the cosine similarity between the question embeddings:

$$\mathrm{sim}(Q_a, Q_b) = \frac{\mathbf{Q}_a \cdot \mathbf{Q}_b}{\|\mathbf{Q}_a\|\|\mathbf{Q}_b\|} \tag{4.1}$$

Here, a question embedding $\mathbf{Q}_a$ is defined as the mean of the embeddings of items in the question Q_a: $\mathbf{Q}_a = \frac{1}{|Q_a|}\sum_{i \in Q_a} \mathbf{i}$, where $\mathbf{i}$ is an embedding of the item i, which was generated with a visual-semantic embedding model consisting of visual features extracted from ResNet [31] and semantic feature from Word2Vec [32] model.

The heterogeneity of a question indicates to what extent different items constitute the

question, and is defined as the mean of euclidean distances between all item pairs:

$$\mathrm{het}(Q) = \frac{1}{k(k-1)} \sum_{i_a, i_b \in Q} \|\mathbf{i}_a - \mathbf{i}_b\| \tag{4.2}$$

Table 4.5 demonstrates that the best question has higher similarity to the previous question and lower heterogeneity than the average and worst questions. These findings can be explained from the viewpoint of informativeness of questions. In active learning [33], it is more informative to receive labels for similar examples that are difficult to distinguish. This principle could be applied to our case: letting the user choose from similar questions (i.e., questions similar to the previous question or those consisting of similar items) is more effective to distinguish preferred items and the others.

In summary, we discovered that (1) the question selection has a significant impact on recommendation performance, (2) the category of fashion items in the question has an impact on recommendation performance, and (3) effective questions are likely to be homogeneous and similar to the previous question.

Chapter 5

Proposed Methods

To address the challenge in the question selection, we first introduce a learning-to-rank model with features found to be correlated with the question quality, in Section 5.1. In addition, we propose a new deep learning method for achieving a better ability to score questions in Section 5.2.

5.1 Learning-to-rank Question Selector

In the analysis of the developed dataset, we showed a large performance gap when the best and worst questions were asked. Choosing the best question from a set of questions can be converted into a traditional Learning-To-Rank (LTR) problem. The goal of the model in LTR is to rank the elements in the given list. If the model can produce a ranking by scoring the questions in a given question set, it can be treated as a value function V used to select the best question. With the view of LTR, we proposed a method based on LambdaMART [34], which is a widely-used pairwise LTR model.

Based on the analysis of the dataset, we devised the following four features of each question Q_t, as the input of the proposed LTR model:

1. The mean similarity of item pairs in the question Q_t.
2. The similarity between the previous question Q_{t-1} and the question Q_t.
3. The category of fashion items in the previous question Q_{t-1}.
4. The category of fashion items in the question Q_t.

The similarity between two items is defined as the cosine similarity of their visual-semantic embeddings, and the question similarity is defined in Equation 4.1.

5.2 Deep Question Selector (DQS)

Although we included several features that can correlate to the quality of questions in the LTR model, the traditional approach using hand-crafted features may not fully capture the characteristics of questions. Hence, we propose a new deep neural network model for the question selection problem, to model the complex relationship between the previous questions and answers as well as question candidates.

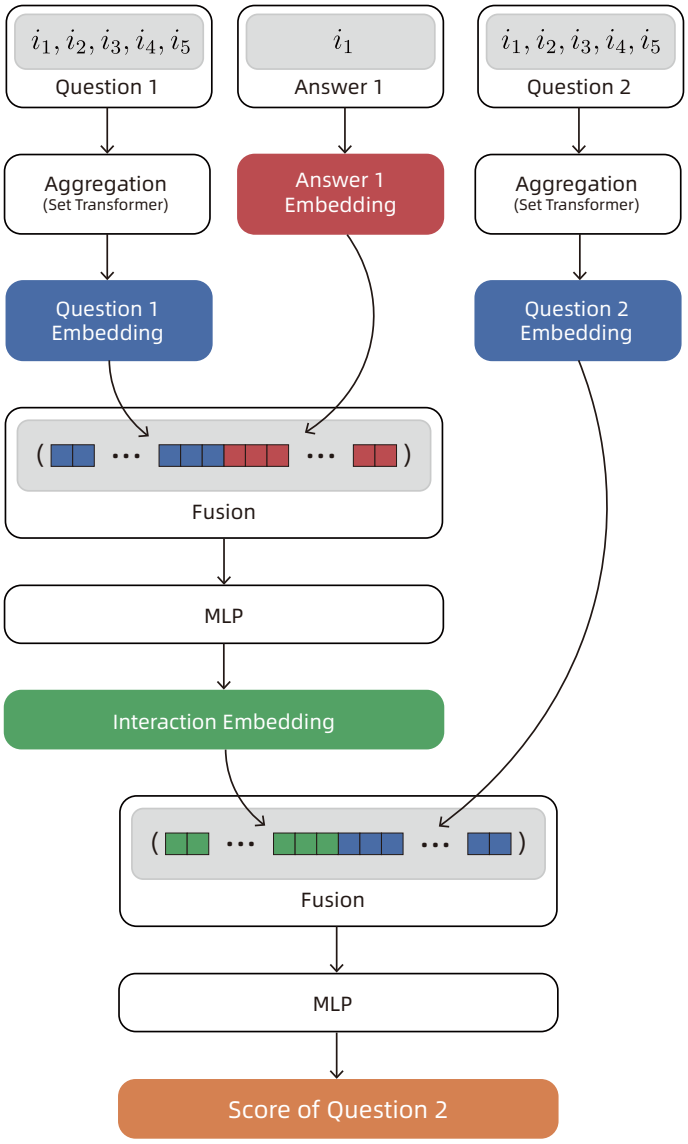

Figure 5.1: The architecture of Deep Question Selector (DQS).

When designing the architecture of this new deep neural network model, there were two desiderata to effectively evaluate the quality of candidate questions:

1. Questions should be modeled as a set of items, and should not be treated as an average or a sequence of items.

2. The past interaction or question-answering history should be modeled properly to avoid duplicate questions or not to receive less informative feedback.

Figure 5.1 illustrates the architecture of the proposed model that can satisfy the two desiderata explained above. As we mentioned in the task setting, the question consists of k items ($k = 5$ in our case). Although the input composed of multiple items is usually treated as a sequence in many machine learning scenarios, ours should be modeled by permutation-invariant models since there is no order for items in a question. Deep Set [7]

and Set Transformer [8] both show good performance with set-structured input data. To better explain the model structure, we assume that the questions are embedded by Set Transformer in the following content.

Generally, question-answer pairs are combined together and used as contextual information to estimate the value of candidate questions. A question-answering history $H_t = \{(Q_1, a_1), \ldots, (Q_t, a_t)\}$ can be embedded by iteratively applying a question-answering fusion and multilayer perceptrons:

$$\mathbf{h}_t = g_t([\mathbf{h}_{t-1}; [\mathrm{ST}_H(Q_t); \mathbf{a}_t]]) \tag{5.1}$$

where $\mathbf{h}_t$ is the embedding for H_t, ST_H is Set Transformer for questions in the history, and $\mathbf{a}_t$ is the embedding of the item given as an answer to the question Q_t. The question-answer pair is combined by the vector concatenation operator $[;]$, and is further combined with the embedding of the question-answering history. A multilayer perceptron g_t is repeatedly applied for keeping the embeddings in low dimensional spaces. Figure 5.1 represents a special case of $t = 1$ where only a question is asked and answered. In this case, the question history $\mathbf{h}_1$ is obtained as follows:

$$\mathbf{h}_1 = g_1([\mathbf{h}_0; [\mathrm{ST}_H(Q_1); \mathbf{a}_1]]) = g_1([\mathrm{ST}_H(Q_1); \mathbf{a}_1]) \tag{5.2}$$

where $\mathbf{h}_0$ is defined as $\emptyset$ and $[\emptyset; x]$ is defined as x.

Given the question answering history H_t, the value of a question conditioned by $\mathbf{h}_t$ is obtained by:

$$V(H_t, Q) = g([\mathbf{h}_t; \mathrm{ST}(Q)]) \tag{5.3}$$

where g is a multilayer perceptron to produce a scalar value based on the embeddings of the question-answering history and question. Note that Set Transformer, ST, can be different from that used for embedding the past questions. If Set Transformer ST_H shares parameters with the other Set Transformer ST, the model is called `DQS-share`. Whereas if the parameters are not shared between two models, we call it `DQS-ind`. These variants of the proposed model were compared in our experiments.

The entire model contains two Set Transformer models ST_H and ST, multilayer perceptrons g_t $(t = 1, 2, \ldots)$, and a multilayer perceptron g for outputting the question value. They are trained with a cross entropy loss of a classification problem where the task is to identify the best question among candidate questions.

Chapter 6

Experiments

We tested the proposed methods on the developed dataset. See Appendix A for detailed information on the experiment environment. Section 6.1 summarizes the data processing and metrics. Section 6.2 presents and discusses the experimental results. Moreover, Section 6.3 demonstrates efficient training pipelines for DQS, and Section 6.6 reports the ablation study.

6.1 Experimental Setup

In our dataset, there are 2,758 question sets, each of which is denoted by $\mathcal{Q}_U$ and was originally generated from the same outfit O. As explained earlier, we assumed that each question set $\mathcal{Q}_U$ is prepared for a user U and the goal is to find the outfit O that satisfies the user U, by choosing effective questions from $\mathcal{Q}_U$. In our experiments, we followed the same experimental protocol as that used in the analysis of the developed dataset. For every question pair (Q_1, Q_3) in a question set $\mathcal{Q}_U$, we assumed a situation where Q_1 had been already asked and answered. The task is to find a question Q_2 from the rest of the questions in $\mathcal{Q}_U$. The performance of the question selection is measured by the accuracy of the best question selection, and the recommendation performance after the question selection. The accuracy is defined as the fraction of cases where the question selection algorithm can identify the best question, as was defined in Section 4.4. The success of the recommendation was measured by nDCG, an effectiveness measure for rankings, where the grade is 1 for a relevant outfit and 0 for the others.

The LTR model was implemented in XGBoost [35]. The DQS models were trained by the Adam optimizer [36]. We split the dataset to a training set and a testing set with a 3:1 ratio. To prevent leakage, it was guaranteed that all the question sets generated from the same outfit were in either the training set or the testing set. Each model was trained only with the train set, while the evaluation ran onto the test set.

6.2 Results

Table 6.1 shows the effectiveness of the question selection models, which include the simplest baseline that selects questions randomly (Random). The proposed models, `DQS-*`, were trained for 200 epochs in this experimental setting. The results show that our DQS

Table 6.1: The performance of the proposed methods.

	Accuracy	nDCG@1	nDCG@5
Random	0.1710	0.5002	0.7173
LTR	0.2202	0.5957	0.7824
`DQS-ST-share-200`	**0.4144**	0.7102	**0.8224**
`DQS-ST-ind-200`	0.3917	0.7010	0.8135

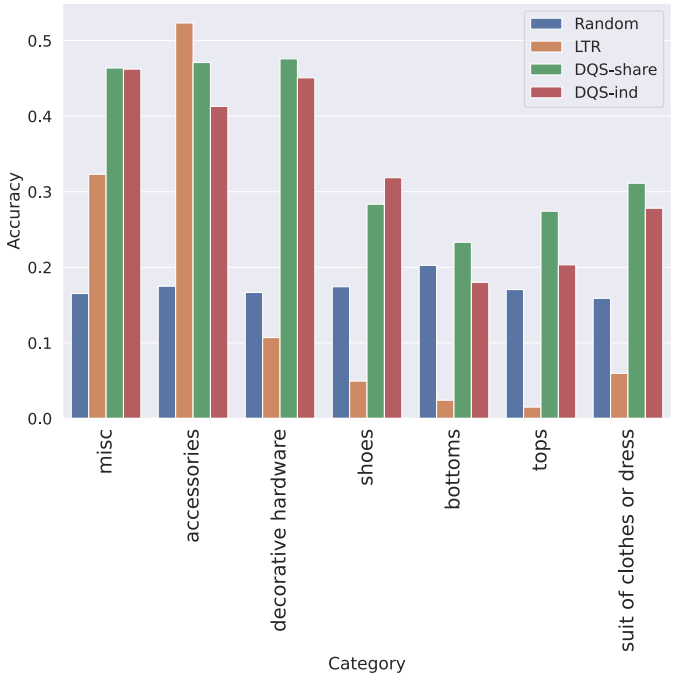

Figure 6.1: Accuracy of the proposed models in different categories.

models outperformed not only the random baseline but also the LTR model, to a large extent. Notably, `DQS-ST-share-200` achieves the 41.44% accuracy of best question prediction. `DQS-ST-ind-200`, which does not share parameters in two Set Transformer models, did not show a better performance than `DQS-ST-share-200`. A two-way ANOVA of nDCG@5 revealed that the system effect is statistically significant ($F(3, 24, 960) = 1329.458, p < 0.05$). A Tukey's HSD test shows that all the differences except for that of `DQS-ST-share-200` and `DQS-ST-ind-200` are statistically significant ($p < 0.05$).

Figure 6.1 reports the accuracy of proposed models in different categories. While LTR, `DQS-ST-share` and `DQS-ST-ind` all achieved substantial improvements over the random baseline, there exist significant differences across categories. The LTR model outperformed in the accessories type question. However, it cannot handle bottoms and tops; the accuracy is even worse than the random baseline. This is probably because the LTR model highly relies on the category information and is likely to select questions in the accessories category. On the other hand, DQS shows its superiority across all the categories against the other methods.

Table 6.2: The performance of DQS models with the special training pipelines.

	Accuracy	nDCG@1	nDCG@5
Random	0.1710	0.5002	0.7173
LTR	0.2202	0.5957	0.7824
`DQS-mean-200`	0.3089	0.6691	0.8053
`DQS-ST-share-200`	0.4144	0.7102	**0.8224**
`DQS-ST-ind-200`	0.3917	0.7010	0.8135
`DQS-DS-share-200`	0.4243	0.7146	0.8210
`DQS-DS-ind-200`	0.3676	0.6757	0.8107
`DQS-ST-share-50-sp`	0.4123	0.7077	0.8140
`DQS-ST-share-100-sp`	0.4269	0.7187	0.8199
`DQS-ST-ind-50-sp`	0.3961	0.7118	0.8179
`DQS-ST-ind-100-sp`	0.4117	0.7072	0.8187
`DQS-ST-com-100-sp`	**0.4282**	**0.7190**	0.8206

Table 6.3: Training time of DQS models with the special training pipelines.

	# Epoch	Time
Set Transformer pre-training	100	53mins
LTR	200	2mins
`DQS-mean-200`	200	1hr 37mins
`DQS-ST-share-200`	200	1d 5hrs 45mins
`DQS-ST-ind-200`	200	1d 15hrs 38mins
`DQS-ST-share-50-sp`	150	8hrs 5mins
`DQS-ST-share-100-sp`	200	15hrs 35mins
`DQS-ST-ind-50-sp`	150	10hrs 29mins
`DQS-ST-ind-100-sp`	200	19hrs 56mins
`DQS-ST-com-100-sp`	200	17hrs 10mins

6.3 Training Pipeline

Furthermore, we propose a practical training pipeline for the DQS models. As shown in the following experimental results, it turned out that the training of Set Transformer is a bottleneck of efficient training. Thus, we aimed to reduce the training time of Set Transformer models without loss of model performances. The idea came from transfer learning [37]: we first pre-train Set Transformer so that its output for a question Q becomes close to the mean of item embeddings in Q. More precisely, we trained ST to minimize the mean absolute error, $\|\mathrm{ST}(Q) - \frac{1}{|Q|}\sum_{i \in Q} \mathbf{i}\|$, for each training question Q. A DQS model then loaded the trained parameters of the Set Transformer and started the learning of the full model. Additionally, to verify the effectiveness of the mean item embedding, we introduced a new model `DQS-mean` in which Set Transformer in DQS is replaced with a mean aggregator. In the following experiments, we pre-trained ST for 100 epochs with the Adam optimizer.

Table 6.2 reports the performance of DQS models, and Table 6.3 shows the training time of each model when using an NVIDIA GeForce GTX TITAN X. DQS models with the pre-trained Set Transformer are suffixed with `*-sp`. Besides, we show the performance of a special model `DQS-com-100-sp`, which is a `DQS-ind` model initialized by parameters of `DQS-share-50-sp` and further trained for extra 50 epochs. These tables indicate that

Table 6.4: The performance of DQS models with the special training pipelines.

	Accuracy	nDCG@1	nDCG@5
DQS-ST-share-200	0.4144	0.7102	**0.8224**
DQS-ST-ind-200	0.3917	0.7010	0.8135
DQS-ST-share-100-sp	0.4269	0.7187	0.8199
DQS-ST-ind-100-sp	0.4117	0.7072	0.8187
DQS-ST-com-100-sp	**0.4282**	**0.7190**	0.8206
DQS-DS-share-200	0.4243	0.7146	0.8210
DQS-DS-ind-200	0.3676	0.6757	0.8107
DQS-DS-share-100-sp	0.4032	0.7058	0.8137
DQS-DS-ind-100-sp	0.3353	0.6848	0.8082
DQS-DS-com-100-sp	0.4022	0.7105	0.8165

Table 6.5: Training time of DQS models with the special training pipelines.

	# Epoch	Time
Set Transformer pre-training	100	53mins
Deep Set pre-training	100	6mins
DQS-ST-share-200	200	1d 5hrs 45mins
DQS-ST-ind-200	200	1d 15hrs 38mins
DQS-ST-share-100-sp	200	15hrs 35mins
DQS-ST-ind-100-sp	200	19hrs 56mins
DQS-ST-com-100-sp	200	17hrs 10mins
DQS-DS-share-200	200	6hrs 36mins
DQS-DS-ind-200	200	8hrs 13mins
DQS-DS-share-100-sp	200	3hrs 28mins
DQS-DS-ind-100-sp	200	4hrs 17mins
DQS-DS-com-100-sp	200	3hrs 57mins

our methods largely reduced the training time without substantial performance loss; some models even achieved better accuracy. Although the training time may vary slightly due to environmental factors, it was turned out that DQS models required more time to train, and the proposed pipeline significantly accelerated the training process. The combination of the two types of DQS models does not show a significant difference. We speculate that the aggregation part might have reached a locally optimal point in the later stage of training. The results also suggest that the mean aggregator for question embedding was effective for pre-training, but not for constituting an effective question selection algorithm.

6.4 Difference between Aggregators

In the previous sections, we only compared the DQS Set Transformer models with traditional machine learning methods. However, as we mentioned before, the question aggregator can be modified for different tasks. Deep Set [7] is a deep learning model which has the same functionality as Set Transformer of embedding set-structured input data. To figure out the difference between question aggregators, we executed aforementioned experiments with DQS Deep Set models.

As Figure 6.4 shown, Set Transformer outperformed Deep Set in most subjects with

Table 6.6: The result of representative methods in the case study

Method	Result
Random	Canditate #4
LambdaMART	Canditate #3
DQS-ST-com-100-sp	Canditate #2
DQS-DS-com-100-sp	Canditate #2

slight approvement. DQS-DS-com-100-sp, which trained with the special pipeline, also achieved very similar performance to DQS-DS-share-100-sp with less training time. Although there exists a slight performance gap between Set Transformer models and Deep Set models, the required training time of Set Transformer models is about five times compared with Deep Set models.

6.5 Case Study

In order to show more clearly the difference between the various methods, we introduce a case study in this section.

Figure. 6.2 shows the basic structure of the dataset. Firstly, the system asks the user to answer the first question and collects the answer. As shown in Figure. 6.2, the user chooses the third item as a favorite item. Secondly, the dataset provides a set of questions. Each question candidate contains five different fashion items, and there is no duplication between items within all questions. Finally, after questioning the user with the specific question, the system will predict the user's favorite item in the third question. According to the influence of each question candidate on the final prediction, the candidate itself can be scored and ranked. The best question candidate is marked as #1 in Figure. 6.2, and the worst question candidate marked as #5. Although each candidate belongs to a different classification, the visual diversity among question candidates seems to influence the scores further.

Table 6.6 reports the result of four representative methods: Random, LambdaMART, DQS-ST-com-100-sp, and DQS-DS-com-100-sp. Unfortunately, none of the models selected the best question candidate for this test case. DQS provides better selection results in the given case. LambdaMART also showed a certain degree of predictive ability in the question selection.

6.6 Ablation Study

To figure out the effect of each component in the DQS model, we conducted an extra experiment for ablation study. We evaluated DQS-ST-com-100-sp by suppressing modules for Q_1, (Q_1, a_1), or (Q_1, a_1, Q_2). Figure 6.3 shows that the results of the ablation study. It can be found that the Set Transformer for the candidate question, Q_2, is the most important for question selection. On the other hand, the first question and answer had almost no effect on the question selection, though there exist small differences between DQS-ST-com-100-sp and $-Q_1$, and DQS-com-100-sp and $-(Q_1, a_1)$. This problem could be further investigated in future work.

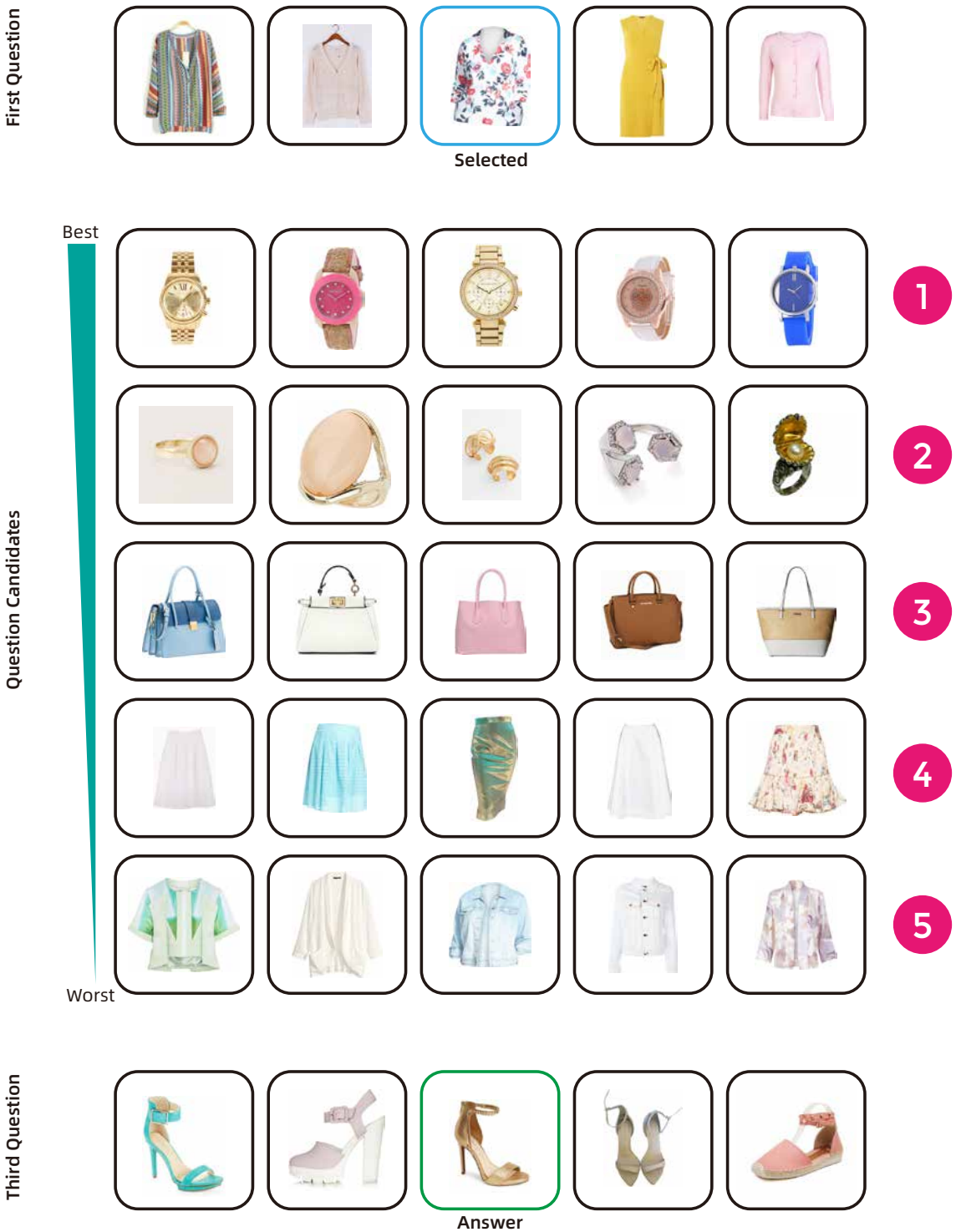

Figure 6.2: The example of question selection. Each question contains five different fashion items; the system is required to select the best question to the answer predicting in the third question. See Appendix B for the URLs of listed item images.

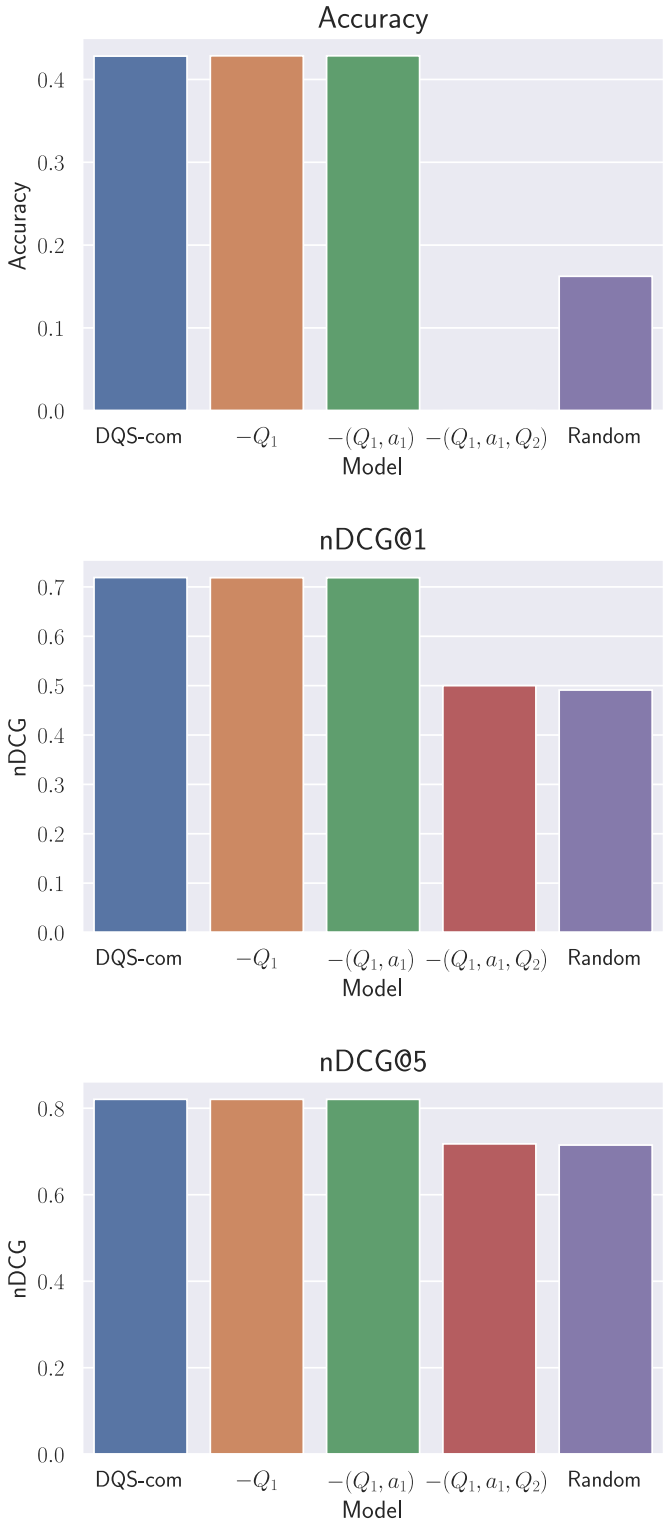

Figure 6.3: Accuracy, nDCG@1, and nDCG@5 of the ablations of Q_1, (Q_1, a_1), and (Q_1, a_1, Q_2).

Chapter 7

Conclusions

This thesis proposed a new fashion outfit recommendation task where an outfit is recommended after several rounds of user-system interactions. We developed a new dataset for evaluating interactive fashion outfit recommendations through simulation. Based on the analysis of the developed dataset, we devised interactive fashion outfit recommendation algorithms based on a learning-to-rank model and Set Transformer. The experimental results demonstrated that the proposed Deep Question Selector models outperformed the learning-to-rank model, and Set Transformer is a suitable model for embedding questions in the interactive fashion outfit recommendation task. We hope that our work and interactive fashion recommendation dataset will draw more attention to interactive fashion recommendation research.

Acknowledgements

I want to acknowledge and give my warmest thanks to my supervisor Professor Makoto P. Kato. His guidance and advice carried me through all the stages of this research project. When I lost sight of my direction in research, he gave me many possible ideas. Whenever I want to get help from him, he always reply to me immediately.

I would also like to give special thanks to everybody in Knowledge Acquisition System Laboratory (Kato Laboratory). Across the long road of the master course, I got too many brilliant suggestions and comments from them.

Finally, I would like to thanks my parents for their love, caring and sacrifices for me.

References

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

[2] Yusan Lin, Maryam Moosaei, and Hao Yang. Outfitnet: Fashion outfit recommendation with attention-based multiple instance learning. In *Proceedings of The Web Conference 2020*, pages 77–87, 2020.

[3] Xingchen Li, Xiang Wang, Xiangnan He, Long Chen, Jun Xiao, and Tat-Seng Chua. Hierarchical fashion graph network for personalized outfit recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 159–168, 2020.

[4] Dhruv Verma, Kshitij Gulati, and Rajiv Ratn Shah. Addressing the cold-start problem in outfit recommendation using visual preference modelling. In *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, pages 251–256. IEEE, 2020.

[5] Kuan-Hsien Liu, Ting-Yen Chen, and Chu-Song Chen. Mvc: A dataset for view-invariant clothing retrieval and attribute prediction. In *Proceedings of the 2016 ACM on international conference on multimedia retrieval*, pages 313–316, 2016.

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

[7] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

[8] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[9] Hanhoon Kang and Seong Joon Yoo. Svm and collaborative filtering-based prediction of user preference for digital fashion recommendation systems. *IEICE transactions on information and systems*, 90(12):2100–2103, 2007.

[10] Si Liu, Jiashi Feng, Zheng Song, Tianzhu Zhang, Hanqing Lu, Changsheng Xu, and Shuicheng Yan. Hi, magic closet, tell me what to wear! In *Proceedings of the 20th ACM international conference on Multimedia*, pages 619–628, 2012.

[11] Yan Wang, Sheng Li, and Alex C Kot. Joint learning for image-based handbag recommendation. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2015.

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

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

[14] Yuncheng Li, Liangliang Cao, Jiang Zhu, and Jiebo Luo. Mining fashion outfit composition using an end-to-end deep learning approach on set data. *IEEE Transactions on Multimedia*, 19(8):1946–1955, 2017.

[15] Edward Shen, Henry Lieberman, and Francis Lam. What am i gonna wear? scenario-oriented recommendation. In *Proceedings of the 12th international conference on Intelligent user interfaces*, pages 365–368, 2007.

[16] Ashay Tamhane, Sagar Arora, and Deepak Warrier. Modeling contextual changes in user behaviour in fashion e-commerce. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 539–550. Springer, 2017.

[17] Rashmi Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems*, pages 830–831, 2002.

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

[19] Michael Jugovac and Dietmar Jannach. Interacting with recommenders—overview and research directions. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(3):1–46, 2017.

[20] Yu Zhu, Yu Gong, Qingwen Liu, Yingcai Ma, Wenwu Ou, Junxiong Zhu, Beidou Wang, Ziyu Guan, and Deng Cai. Query-based interactive recommendation by meta-path and adapted attention-gru. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2585–2593, 2019.

[21] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H Chi. Q&r: A two-stage approach toward interactive recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 139–148, 2018.

[22] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[23] Yu Chen and Pearl Pu. Cofeel: Using emotions for social interaction in group recommender systems. In *First International Workshop on Recommendation Technologies for Lifestyle Change (LIFESTYLE 2012)*, volume 48, 2012.

[24] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Anna Xambó, Emilia Gómez, and Perfecto Herrera. Semantic audio content-based music recommendation and visualization based on user preference examples. *Information Processing & Management*, 49(1):13–33, 2013.

[25] Fedor Bakalov, Marie-Jean Meurs, Birgitta König-Ries, Bahar Sateli, René Witte, Greg Butler, and Adrian Tsang. An approach to controlling user models and personalization effects in recommender systems. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 49–56, 2013.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[27] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[28] Andrea Frome, Greg Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. 2013.

[29] Kristen Vaccaro, Sunaya Shivakumar, Ziqiao Ding, Karrie Karahalios, and Ranjitha Kumar. The elements of fashion style. In *Proceedings of the 29th annual symposium on user interface software and technology*, pages 777–785, 2016.

[30] Lifeng Shang, Tetsuya Sakai, Zhengdong Lu, Hang Li, Ryuichiro Higashinaka, and Yusuke Miyao. Overview of the ntcir-12 short text conversation task. In *NTCIR-12 Conference*, 2016.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[33] Burr Settles. Active learning literature survey. 2009.

[34] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[35] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Appendix A

Experiment Environment

- System: `Ubuntu 18.04.5 LTS`
- Linux Kernel: `3.10.0-1127.19.1.el7.x86_64`
- GPU: single `NVIDIA GeForce GTX TITAN X`
- CUDA: `11.1`
- Docker: `19.03.13`
- Python: `3.8.8`
  - numpy: `1.19.2`
  - scipy: `1.7.1`
  - scikit-learn: `0.24.2`
  - lightgbm: `2.3.1`
  - xgboost: `1.4.2`
  - torch: `1.8.1`
  - torchvision: `0.9.1`
- Python: `2.7.1`
  - tensorflow-gpu: `0.11`

Appendix B

The URL of Pictures Used in This Thesis

1. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=94307864
2. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=88103180
3. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=169354407
4. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=168367314
5. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=154976551
6. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=152368950
7. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=133543185
8. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=133416182
9. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=133348458
10. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=133348447
11. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=132686258
12. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=131275624
13. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=125132715
14. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=106858616
15. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=103912700
16. http://img2.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=101307979
17. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=98664184
18. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=93321499
19. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=85317648
20. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=200740423

21. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=197222045
22. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=173069545
23. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=172337434
24. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=166247947
25. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=164107498
26. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=156113120
27. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=155674974
28. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=129954578
29. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=129461850
30. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=129121319
31. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=123073770
32. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=111583344
33. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=111487583
34. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=110083108
35. http://img1.polyvoreimg.com/cgi/img-thing?.out=jpg&size=m&tid=105426358