Outfit Recommendation based on Styles

Fubao Wu, KungWen Yu, YenSung Chen

Abstract

Recommending outfits for consumers offers the great convenience to their clothing and shopping for fashion items. There are several tasks involved outfit recommendation–missing prediction for compatibility, style extraction and outfit generation for recommendation. Most of current approaches of outfit generation for recommendation focus on style of pair-wise fashion items in an outfit rather than a global style of all outfits. Focusing on global styles of outfits offers more effective and accurate information for outfit recommendation. We propose to explore adversarial auto encoder to extract the global styles from outfits, an unsupervised method to fast learn the global information of outfits. Using the global style information learned into a model, it can effectively generate outfits flexibly and explanatively. We will be doing extensive experiments on a human-generated outfit dataset for the proposed model and test the performances compared with the state-of-the-art methods.

Keywords: Outfit recommendation, unsupervised learning, Adversarial auto encoder, BiLSTM

1. Introduction

Purchasing clothes for compatibility and choosing fashion items play an important role for consumers' lives. The good combinations of fashion item selection require expert knowledge. With the development of artificial intelligence, the recommendation of fashion items for compatibility becomes necessary and has great improvement on consumers' lives. Recommending outfits to users involves two types of questions. The first type question is that given an outfit missing a fashion item(e.g. T-shirt, pant, a bag or sunglasses), we find the best missing item (e.g shoes here) to comprise an outfit to recommend to users. The second type is that given a fashion item(e.g. a green dress), we generate an outfit that has the similar style and compatible with the given item and then recommend to users. We are focusing on the second type of question of recommendation. To achieve the recommendation, there are three main tasks involved. (1)learn some outfit compatibility based on outfit images and descriptions (if available). (2)extract the style of a given fashion item (3)find the outfit that has the similar style with a given fashion item, then recommend it to users.

Learning compatibility is the key for outfit recommendation. An outfit compatibility has assumed to have two properties: (1) items should have visual compatibility or share similar styles. (2) There are existing extensive research on outfit compatibility. Most of them use distance metrics to measure the pair-wise fashion items in an outfit through metric learning [1] or a Siamese network [2].

There are few efforts on learning the styles of outfits. They basically explored the style considering the pairwise items in an outfit. In our work, we learn the compatibility and extract the global styles for outfit recommendation. For the compatibility part, we use a bidirectional LSTM model to learn the sequence of outfits, and propose to use the state-of-the-art generative adversarial encoder to extract the global styles from outfits. Finally, given a fashion item, we can recommend several outfits of different styles to users.

The main contribution of this project is: (1) We propose a framework to learn the outfit recommendation based on multiple modules of neural networks. (2) We propose to do outfit recommendation by extracting the global styles in outfits given a query of fashion item using a generative adversarial autoencoder combined with compatibility modules. (3) We have done extensive experiments and show the result of effectiveness of outfit recommendation in the Polyvore fashion dataset.

2. Related Work

There are extensive studies on automatic fashion analysis including clothing parsing, clothing recognition and retrieval in the community. About the fashion

recommendation and outfit generation, few work [3, 4] explored fashion recommendation.

2.1. Outfit Compatibility

Learning outfit compatibility plays an important role in fashion recommendation. Some initiative work tried to use metric learning [1] to learn the compatibility between a pair of items. However, they neither considered the composition of items to form an outfit nor supported the global styles of an outfit. Another work related to outfit compatibility was to use convolutional neural network to learn the relation between fashion image features [1]. Recent work that uses biLSTM to learn outfit compatibility shows good results [5]. We will refer to the method for our outfit compatibility learning.

2.2. Style Extraction

Fashion style extraction of outfits is the second important problem for providing great recommendation experience for consumers. There exists some work on the extraction of fashion style based on supervised learning. Takagi et al. [6] proposed methods to learn fashion styles by training a neural network using 14 modern styles collected from images. Although the supervised method may be reliable, it is expensive to prepare accurate labels without expert knowledge. Recently, Tkuma et al. [7] tried to use unsupervised methods to extract the style, but they used a simple encoder and decoder method to learn. Our work will focus on extracting the global style with state-of-the-art adversarial autoencoders.

3. Methodology

In this section, we describe methods for generating outfits. An outfit can be represented as a sequence of items, $F = \{x_1, x_2, ..., x_n\}$, where x_i is a fashion item that could be any wearable clothes or accessories. Referring to [5], we decide to use bidirectional long-short term memory (BiLSTM), visual-semantic embedding (VSE), and style embedding (SE) with adversarial autoencoder (AAE). The architecture of the proposed model is shown in Figure 1.

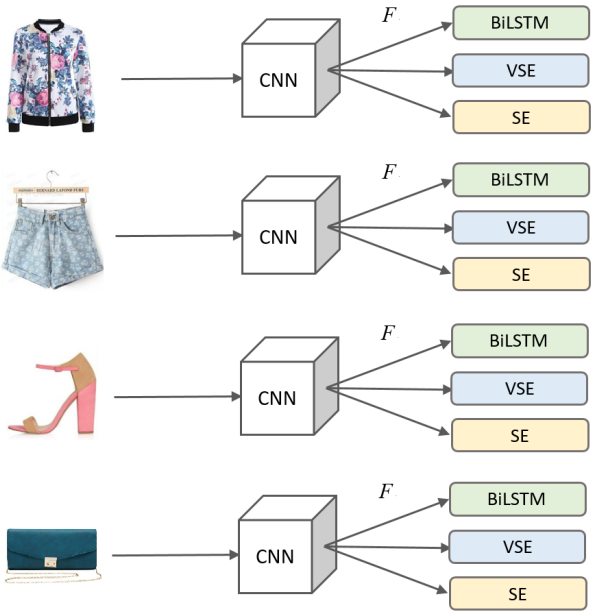

Figure 1: The architecture of the proposed model

3.1. Bidirecitonal Long-Short Term Memory

One property of LSTM is that the model can learn the features of time steps. This is really useful for inputs that have some connections between each element. For the items in an outfit, there is some relationship between items as well. Therefore, in order to let the model to learn the features of sequence, we use LSTM for implementation. However, if we only apply LSTM for one direction, the model may miss some features from the other direction. To solve this drawback, BiLSTM can be applied.

For the BiLSTM, we implement forward and backward LSTM. At each iteration, the BiLSTM model will concatenate the current hidden states from forward and backward LSTM. And since BiLSTM is a joint model, the prediction can be represented as the below equation:

$$P(x_t|x_1, ..., x_{t-1}, x_{t+1}, ..., x_n) = P_{forward}(x_t|x_1, ..., x_{t-1}) \times P_{backward}(x_t|x_n, ..., x_{t+1}) \quad (1)$$

As the representation we described above, the input is an outfit that can be $F = \{x_1, x_2, ..., x_n\}$, each element is an image of the item. During the training, before going to the LSTM model, each image will go through a CNN to get the features. In the part of LSTM, the time step will be the sequence of items. At each iteration, the LSTM will take the previous hidden state h_{t-1} and the input x_t as the current input to calculate the current

hidden state h_t. The following equations are the process that the LSTM computes the current hidden state h_t:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ g_t &= tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ c_t &= f_t c_{t-1} + i_t g_t \\ h_t &= o_t tanh(c_t) \end{aligned} \tag{2}$$

where i_t, f_t, g_t, o_t and c_t represent input gate, forget gate, cell gate, output gate, current cell state, and current hidden state respectively. And $W_{\alpha\beta}$ represents the weight that vector α maps to vector β.

After the current hidden state h_t is obtained, the next item can be predicted by using softmax function:

$$P_{forward}(x_t|x_1, ..., x_{t-1}) = \frac{e^{h_{t-1}x_t}}{\sum_{x \in S} e^{h_{t-1}x}} \tag{3}$$

where S is a set of possible items that can be fit in the outfit. Then the loss function of the outfit F will be:

$$L_{forwrd}(F) = -\frac{1}{n}\sum_{t=1}^{n} P_{forward}(x_t|x_1, ..., x_{t-1}) \tag{4}$$

Similarly, the prediction after using softmax function and the loss function can be represented as following equations:

$$\begin{aligned} P_{backward}(x_t|x_n, ..., x_{t+1}) &= \frac{e^{h_{t+1}x_t}}{\sum_{x \in S} e^{h_{t+1}x}} \\ L_{backward}(F) &= -\frac{1}{n}\sum_{t=n-1}^{0} P_{backward}(x_t|x_n, ..., x_{t+1}) \end{aligned} \tag{5}$$

3.2. Visual-Semantic Embedding

A fashion item usually has a description about its characteristics and attributes. Given an image of a fashion item and its corresponding description, we can utilize Visual-Semantic Embedding (VSE) method to extract a common expression between them. The description of a fashion item can be denoted as $S = \{w_1, w_2, ..., w_M\}$, where w_i represents a word in the description. Mapping a description to an common embedding space can be described as:

$$v = \frac{1}{M}\sum_{t=1}^{M} W_T \cdot e_i$$

where e_i is a one-hot vector corresponding to w_i. W_T represents a word embedding matrix. Similarly, a mapping of image features to an common embedding space can be expressed as:

$$u = W_I \cdot x$$

where W_I denotes the image embedding matrix. The goal is to make sure that v and u are mapped near to each other if they are derived from the same item. Otherwise, it is preferable having no mapping between them. Let U represents the set of word-embedded vectors and $U \setminus u^{(v)}$ be the set of u excluding vectors corresponding to v. That is, $U \setminus u^{(v)}$ are samples different from v. V and $V \setminus v^{(u)}$ have the same definition as U and $U \setminus u^{(v)}$ respectively. The relationship can be expressed as below:

$$\begin{aligned} L_{VSE} = &\sum_{u \in U}\sum_{v' \in V \setminus v^{(u)}} max(0, m_s - d(u, v) + d(u, v')) \\ &+ \sum_{v \in V}\sum_{u' \in U \setminus u^{(v)}} max(0, m_s - d(v, u) + d(v, u')) \end{aligned} \tag{6}$$

where m_s is the margin of hinge function and $d(u, v)$ denotes the cosine distance between u and v.

3.3. Style Embedding with Adversarial Autoencoder (AAE)

To extract the global style of outfits, here we propose to use an unsupervised method–the state-of-the-art generative adversarial autoencoder [8] . We will briefly discuss the model and how we use it in the outfit style extraction. Adversarial autoencoder (AAE) is a probabilistic autoencoder that uses generative adversarial networks (GAN) [9] to perform variational inference. The reason we use the adversarial autoencoder is that it is an unsupervised method (although it can use labels as a supervised adversarial auto-encoder) that can capture the rich distributions of images, and is shown to have better results on disentangling styles of images compared to variational autoencoder [10] and importance weighted autoencoder [11]. The outfit is a sequence of image features, we use AAE to learn the style features without any labeling information. Before we learn the features, we represent an outfit as a sequence of individual items x_t. Because the lengths of different outfit x_t vary, we use a reducing method to represent the style of an outfit as follows:

$$x_s = \frac{1}{N}\sum_{t}^{N} W_s * x_t \tag{7}$$

Here x_s represents the style vector of an outfit. Ws represents a weight matrix that maps an image feature

to a feature vector. The feature vector x_s is the input of an adversarial auto-encoder, then we use it to learn the basis style representation of a variety of outfits. According to the adversarial auto-encoderwe use the loss function:

$$L_{AAE} = E_x[E_{q(z|x)}[-logp(x|z)]] + E_x[L_a] \quad (8)$$

Where $q(z|x)$ is the encoding distribution for compressing the style vector. $p(x|z)$ is the decoding distribution for reconstruction of the style vector. The adversarial loss L_a is obtained by adversarial training procedure that encourages the distribution $q(z)$ of the latent code z to match to the whole distribution of $p(z)$ of positive samples. The reconstructed feature from original sequence x_s is denoted as x'_s.

3.4. Objective Function

Since this is a joint model [7], the total loss should be the loss from biLSTM, VSE, and AAE, the following equation represents the objective function that we have to minimize:

$$min_\theta \sum_F (L_{forward} + L_{backward}) + L_{VSE} + L_{AAE} \quad (9)$$

where θ contains a set of parameter for biLSTM, VSE, and AAE.

4. Experimental Evaluation

4.1. Dataset

There are many fashion websites which enable users to define their own preferred outfits. In this experiment, we use *Polyvore* (www.polyvore.com), which is one of popular fashion websites. All outfits in this website has been arranged by [5]. This dataset contains 164,837 items of clothing grouped in 21,889 outfits. We use 70% of the outfits as training dataset, 20% of them is taken to be as a validation set. 10% of them is used as the test dataset.

4.2. Implementation

We implement our experiment with Pytorch and train on the Google cloud platform with one GPU and 8 CPUs configuration.

4.2.1. Bidirectional LSTM(biLSTM)

In the initial step, we use GoogleNet Inception V3 model [12] to encode each fashion item as a 2048-dimensional feature vector. The dimension of the vector is reduced to 512 with the fully connected layer in the last layer of Inception V3 so that it can be fed into biLSTM.

4.2.2. Visual-semantic Embedding(VSE)

For visual-semantic embedding, we encode the visual description and attributes of available fashion items into a feature vector, The dimension of the joint embedding space is set to 512. Thus, we have $W_I \in R^{2048*512}$ and$W_T \in R^{2757*512}$, where 2757 is the size of the vocabulary. We fix the margin m = 0.2 in Equation 6.

4.2.3. Adversarial AutoEncoder (AAE)

In AAE, we take the feature vector sequence of an outfit as input and encode it to learn the style of the outfit, the learning rate for both encoder and decoder is set to 0.0001.

4.2.4. Joint Training

The loss functions of biLSTM, VSE and AAE are combined together for training jointly. The training learning rate is set to 0.2 initially, then it decays by a factor of 2 for every 2 epochs of biLSTM and VSE. The batch size is set to 10, meaning that each mini batch contains 10 outfit sequences, which has about 65 images and their corresponding descriptions.

Table 1: Parameter Dimensions

Module	Parameter and Variable	Dimension
BiLSTM	x_t	R^{512}
	h_t	R^{512}
VSE	W_T	$R^{2757*512}$
	W_I	$R^{2048*512}$
AAE	W_s	$R^{2048*256}$
	W_t	R^{1024*8}

The Parameter and variable dimension of all modules provided in the formula are shown in the Table 1. W_t is the weight used for the Generative Network in AAE.

The training loss functions of our model are shown in Figure 2. It shows that the three modules of our model converge after 50,000 iterations, even though they are trained for more than 70,000 iterations.

Figure 2: The converged training loss of our model

4.3. Baseline methods

We train our models with two different combinations of modules. There are the following modules we considered here:

- **BiLSTM+VSE**. The models as the baseline model with the bidirectional LSTM and the visual-semantic embedding only
- **BiLSTM+VSE+AAE**. Our full model by jointly learning the bidirectional LSTM, the visual-semantic embedding and the style with AAE.

4.4. Missing Prediction

Here we consider the first task to predict the missing item from outfit. This is an important step to learn the clothing sequences as an outfit. Given a sequence of clothing images and a blank, fill in the blank with a suitable clothing image for prediction. This evaluates a compatibility recognition performance between an item sequence and an item alone. The number of candidate items is set to equation10 in our experiment, and then negative fashion items are sampled from outfits excluding the query outfit.

Therefore, we need to consider the measurement score of compatibility. Before we define the compatibility score, we consider the style similarity inside the compatibility.

We assume the style of each fashion item in an outfit is similar, according to the extracted style from AAE, we define the style similarity as follow:

$$SS(F_{s1}, F_{s2}) = cosSim(x_{s1}, x_{s2}) \tag{10}$$

We use the cosine similarity (cosSim) of the extracted style vector similarity x to measure it.

Then we evaluate the compatibility as follows:

$$x_a = \underset{x \in C}{argmax} \frac{exp(h_{t-1} x_c)}{\Sigma_{x \in C} exp(h_{t-1} x)} + \frac{exp(\tilde{h}_{t+1} x_c)}{\Sigma_{x \in C} exp(\tilde{h}_{t+1} x)} + \gamma SS(F, x_c) \tag{11}$$

where the first and second terms on the right side are the scores calculated by the forward LSTM and backward LSTM, respectively. C represents the set of candidate items. F here represents an input sequence with an item removed and γ is a hyperparameter that represents a weight of style similarity. Here we use $\gamma = 0.2$ according to the best experimental value.

Because we want to predict the outfit compatibility, we design a quantitative measurement of the compatibility to evaluate how good the compatibility is. Through this step, we can avoid the the outfit that has similar type of items appeared multiple times.

Therefore, we can compare the test accuracy of the different models for predicting the missing item and compatiblity AUC in the blank.

Table 2: The comparsion of our model on missing prediction and compatibity accuracy

Models	Missing prediction accuracy
Compatibility AUC	
BiLSTM+VSE	39.6
0.69	
BiLSTM+VSE+AAE	32.5
0.61	

We have used this one dataset to test and find with AAE, it has lower accuracies compared with the baseline model. It is probably this dataset trained is kind of overfitting for our model when training with too many extra iterations done in our experiment. It also does not help with the missing item with the global style when predicting missing items. It needs to be confirmed with more datasets and to be further analyzed for future work.

Here we show some successful missing prediction examples in Figure 3a and 3b. The green circle is the one that has the highest score which is predicted to be the best fit. It shows the missing prediction has successfully recommend the missing item the completed an outfit, which has the high score.

(a) Success case 1 of Missing prediction

(b) Success case 2 of Missing prediction

Figure 3: Successful cases for missing prediction

However, we also have found some bad cases that the missing prediction does not perform well. It also could detect the duplicated items shown in Figure 4a and 4b. It outputs the unsuitable recommendation which will therefore need to use compatibility test to further evaluate whether the outfit is compatible.

(a) Failure 1 of Missing prediction

(b) Failure case 2 of Missing prediction

Figure 4: Failed cases for missing prediction

4.5. Style Extraction

We have assumed the outfit can be represented as mixture of elements of a style basis. Here we define the four basic types of style basis used in this experiment:

Style 1 represents the spring or early summer in pale tone. The main items are skirt or simply designed fashion items.

Style 2 associates with summer, which is dominated by sandals and jeans, which has high contrast with the color of style 1.

Style 3 reminds you of the strength and individuality with lots of dark and black items, many of which are made with leather and metal.

Style 4 is related with winter outfit with made thick fabrics, and also some have high-heel boots and gold accessories.

Here we show some outfit examples of each style in the following figures.

(a) Style 1

(b) Style 2

(c) Style 3

(d) Style 4

Figure 5: Examples of items for different style basis

All the other outfits generated are based on the mix-

ture of the style basis.

4.6. Outfit Generation

After we get the outfit sequence from biLSTM and the global style from AAE, we define a score function to evaluate the likelihood of the sequence and the global style as shown in Equation (12):

$$score(F|s_{target}) = -(E_f(F;\theta_{LSTMf}) + E_b(F;\theta_{LSTMb})) + \beta * cosSim(x_s, s_{target}) \quad (12)$$

where x_s represents the style of the query sequence computed from AAE, and x'_s represents the reconstructed style of the query sequence learned from AAE. The similarity of the style of generated sequence with the target style is calculated in Cosine Similarity. β is a hyperparameter used to control the balance between the sequence likelihood and the style similarity.

Thus for each outfit sequence generated from biLSTM, we can get the outfit style score based on the target style and the generated style. Thus we can recommend the best outfit based on the target styles using a search algorithm. Here we use beam search [13] to speedup the search in our experiment.

We show some examples of (good and bad) outfit generation in the following Figure 6. It shows we can successfully recommend some outfits even with some bad recommendations with duplicated style of items in an outfit existed. Although we use the compatibility evaluation to further evaluate the compatibility score and recommend the outfit with the highest score, it is still not enough to truly filter some bad recommendation. This could may lead to the future work to explore the true reason.

Figure 6: Good (a, b, c) and bad cases (d) for outfit generation. For the bad case, the result may have duplicated items. Green square represents the given item.

5. Conclusion

In this paper, we have researched the outfit recommendation for fashion. We focus on trying to get the global style from the whole outfit with the new Adversarial AutoEncoder combined with the biLSTM compatibility module together to generate outfits. We have experimented in detail the outfit prediction and compatibility test, and the style extraction test to get the outfit generation. The experimental results show our model with AAE is not very good compared with the baseline model, though we can successfully recommend some outfits. For the future work, we will investigate more datasets and find out the real reasons behind why there are some bad outfit generations and thus improve our models.

References

[1] J. McAuley, C. Targett, Q. Shi, A. Van Den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, pp. 43–52.

[2] A. Veit, B. Kovacs, S. Bell, J. McAuley, K. Bala, S. Belongie, Learning visual clothing style with heterogeneous dyadic co-occurrences, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 4642–4650.

[3] Y. Li, L. Cao, J. Zhu, J. Luo, Mining fashion outfit composition using an end-to-end deep learning approach on set data, IEEE Transactions on Multimedia 19 (2017) 1946–1955.

[4] Y. Hu, X. Yi, L. S. Davis, Collaborative fashion recommendation: A functional tensor factorization approach, in: Proceedings of the 23rd ACM international conference on Multimedia, ACM, pp. 129–138.

[5] Y.-G. J. L. S. D. Xintong Han, Zuxuan Wu, Learning fashion compatibility with bidirectional lstms.

[6] M. Takagi, E. Simo-Serra, S. Iizuka, H. Ishikawa, What makes a style: Experimental analysis of fashion prediction, in: Proceedings of the IEEE International Conference on Computer Vision, pp. 2247–2253.

[7] R. G. Takuma Nakamura, Outfit generation and style extraction via bidirectional lstm and autoencoder.

[8] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, arXiv preprint arXiv:1511.05644 (2015).

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, pp. 2672–2680.

[10] D. P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).

[11] Y. Burda, R. Grosse, R. Salakhutdinov, Importance weighted autoencoders, arXiv preprint arXiv:1509.00519 (2015).

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826.

[13] O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: A neural image caption generator, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3156–3164.