Dressing as a Whole: Outfit Compatibility Learning Based on Node-wise Graph Neural Networks

Zeyu Cui
Institute of Automation, Chinese Academy of Sciences
University of Chinese Academy of Sciences
zeyu.cui@nlpr.ia.ac.cn

Zekun Li
Institute of Information Engineering, Chinese Academy of Sciences
University of Chinese Academy of Sciences
lizekunlee@gmail.com

Shu Wu
Institute of Automation, Chinese Academy of Sciences
University of Chinese Academy of Sciences
shu.wu@nlpr.ia.ac.cn

Xiaoyu Zhang
Institute of Information Engineering, Chinese Academy of Sciences
zhangxiaoyu@iie.ac.cn

Liang Wang
Institute of Automation, Chinese Academy of Sciences
University of Chinese Academy of Sciences
wangliang@nlpr.ia.ac.cn

ABSTRACT

With the rapid development of fashion market, the customers' demands of customers for fashion recommendation are rising. In this paper, we aim to investigate a practical problem of fashion recommendation by answering the question "which item should we select to match with the given fashion items and form a compatible outfit". The key to this problem is to estimate the outfit compatibility. Previous works which focus on the compatibility of two items or represent an outfit as a sequence fail to make full use of the complex relations among items in an outfit. To remedy this, we propose to represent an outfit as a graph. In particular, we construct a *Fashion Graph*, where each node represents a category and each edge represents interaction between two categories. Accordingly, each outfit can be represented as a subgraph by putting items into their corresponding category nodes. To infer the outfit compatibility from such a graph, we propose *Node-wise Graph Neural Networks* (NGNN) which can better model node interactions and learn better node representations. In NGNN, the node interaction on each edge is different, which is determined by parameters correlated to the two connected nodes. An attention mechanism is utilized to calculate the outfit compatibility score with learned node representations. NGNN can not only be used to model outfit compatibility from visual or textual modality but also from multiple modalities. We conduct experiments on two tasks: (1) Fill-in-the-blank: suggesting an item that matches with existing components of outfit; (2) Compatibility prediction: predicting the compatibility scores of given outfits. Experimental results demonstrate the great superiority of our proposed method over others.

CCS CONCEPTS

• **Applied computing → Online shopping**.

KEYWORDS

Compatibility learning, graph neural networks, multi-modal

ACM Reference Format:
Zeyu Cui, Zekun Li, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Dressing as a Whole: Outfit Compatibility Learning Based on Node-wise Graph Neural Networks. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313444

1 INTRODUCTION

Clothing plays an increasingly significant role in human's social life, as a proper outfit can enhance one's beauty and display the personality. However, not everyone has a strong fashion sensitivity. "How to make a suitable and beautiful outfit?" has become a daily headache for many people. Solving this problem requires modeling human notions of the compatibility. Here we aim to propose an effective method to predict the compatibility of outfits and thus help people make proper outfits.

Previous works on compatibility learning mainly focused on the compatibility of two items. These works followed the idea to map the items into a style space and estimate the distance between style vectors of items. Veit *et.al* used SiameseCNNs [1, 38] to transform items from the image space to the style space. Mcauley *et.al* [23] proposed to use Low-rank Mahalanobis Transformation to map items into the style space. Lately, some works proposed to model the compatibility of items by mapping items into several style spaces [11, 31]. Recently, there appear a few works directly measure the compatibility of outfit. Li *et.al* [17] proposed to represent an outfit as a sequence and adopt Recurrent Neural Network (RNN) [3, 6, 12, 21, 24, 25, 33] to model the compatibility. Similarly, another work [9] further represented an outfit as a bidirectional sequence with a specific order (i.e., top to bottom and then to accessories), and used Bidirectional Long Short Term Memories (Bi-LSTMs) to predict the next item in an outfit.

The first two authors Zeyu Cui and Zekun Li contribute to this work equally and are listed as joint first authors. Shu Wu and Xiaoyu Zhang are both corresponding authors.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313444

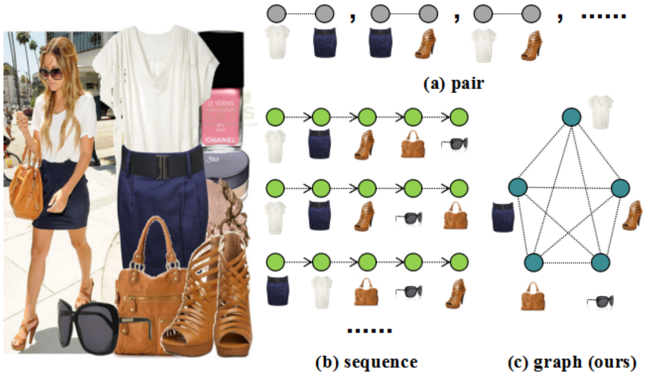


Figure 1: Different representations of an outfit. Left is the outfit. (a), (b) and (c) in the right are its pair, sequence and graph representations respectively. Circles represent the items and links represent the relations.

Solving the task of modeling outfit compatibility relies on an appropriate outfit representation. The previous works take advantage of two kinds of outfit representations: *pair representation* [11, 23, 31, 38] and *sequence representation* [9]. We show different outfit representations in Figure 1. From (a) we can see that representing the outfit as pairs separately cannot reflect the complex relations among multiple items. In term of *sequence representation* in (b), there doesn't exist a fixed order of items. More importantly, the relations among items in an outfit are not ordered since an item has relations with not only its prior or rear item in the sequence. To remedy this, we propose a new outfit representation: *graph representation*. As shown in (c), it can better reflect the dense and complex relations among multiple items in an outfit.

In particular, we construct a *Fashion Graph* based on Polyvore dataset, where each node represents a category and each edge represents the interaction between two nodes. If two categories have matching relation (i.e., they appear in the same outfit), there should be interactions between them. We have two directed edges between them in *Fashion Graph* since the interactions in two directions ought to be different. For example, one may select a pair of socks to match the shoes but not select shoes for socks in particular. A compatible outfit should have a key property: the categories of items in it are not overlapped (e.g., there won't be two pairs of shoes in an outfit). Thus, by putting each item into its corresponding node, an outfit can be represented as a subgraph of *Fashion Graph*. The task of modeling outfit compatibility can be accordingly converted to a graph inference problem.

A possible method is using Graph Neural Networks (GNN) to model the node interactions and infer compatibility from the graph. GNN is an approach to applying neural networks to graph-structured data, first proposed by Gori *et al.* [7]. It can be applied on most kinds of graphs, including directed, undirected and cyclic graphs. However, it has trouble in propagating information across a long range in a graph. To address this, Li *et al.* [18] proposed Gated Graph Neural Network (GGNN) by introducing a Gated Recurrent Units (GRU) for updating. Nevertheless, in GGNN, the nodes interact with others by communicating their state information on the edges in a fixed way, which leads to a limitation that it has difficulty in modeling flexible and complex node interactions.

To address this limitation, we propose a novel model NGNN to better model node interactions and better infer the compatibility from graph. In NGNN, the communication of state information on each edge is different. Specifically, when two nodes communicate state information on the edge, the state information will first be transformed by a transformation function determined by parameters correlated to the two nodes. In this way, NGNN can model edge-wise interactions with node-wise parameters. Since the parameters are correlated to nodes instead of numerous edges, it can greatly reduce the parameter space. During training process, NGNN only deals with the nodes in the subgraph each time, which reduces the time complexity as well. Moreover, when there are more data to train the model, we only need to update the parameters of these concerned nodes without changing the edges or other nodes, which is effective in real applications. With the learned node representations, we utilize an attention mechanism to calculate a graph-level output to serve as the outfit compatibility score.

The framework of our proposed method is shown in Figure 2. We first construct a *Fashion Graph* where an outfit can be represented as a subgraph. Then we use the proposed novel model NGNN to model node interactions and learn the node representations. Finally, we predict the compatibility score by calculating a graph-level output using an attention layer. This proposed method can model outfit compatibility from visual, textual or any single modality with a channel of input features. By using two channels of input to NGNN, it can also be used to jointly model the outfit compatibility from visual and textual modality.

We conduct experiments on two tasks: (1) Fill-in-the-blank: suggesting an item that matches with existing components of outfit; (2) Compatibility prediction: predicting the compatibility scores of given outfits. Experiment results on the Polyvore dataset demonstrate the great superiority of our method over others. The code and dataset of our work have been released[1].

Our main contributions can be summarized in threefold:

- To the best of our knowledge, this is the first attempt to represent an outfit as a graph, which can better capture the complex relations among multiple items in an outfit.
- We propose a novel model NGNN which can better model node interactions in the graph and learn better node representations. Taking advantage of NGNN, we can not only model outfit compatibility from visual or textual modality, but also from multiple modalities.
- Experimental results on a real-world dataset (Polyvore dataset) demonstrate the great superiority of our method over others.

2 RELATED WORK

2.1 Fashion Compatibility Modeling

There have been many studies focusing on modeling the compatibility of clothing recently, since it is the key to fashion recommendation. McAuley *et al.* assumed that there existed a latent style space where compatible items stayed close. They used Low-rank Mahalanobis Transformation (LMT) to map items into a latent space and

[1] https://github.com/CRIPAC-DIG/NGNN

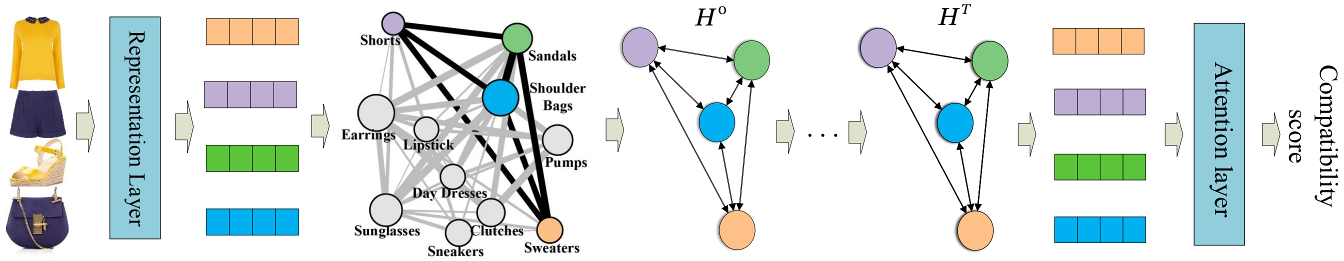


Figure 2: Overview of the proposed method. Based on the dataset, we first construct a *Fashion Graph*, where each node represents a category and each edge represents the interaction between two nodes. An outfit (e.g. sweater, short, sandals, shoulder bag) can be represented as a subgraph. We then design NGNN to model the node interactions and learn node representations. An attention layer is finally utilized to calculate the compatibility score.

measured the distance there [23]. Following that, Veit *et al.* proposed to learn the distance metric with an end-to-end SiameseCNNs [38]. Some other works mapped items into several latent spaces and jointly modeled the distances in these latent spaces, which can measure the compatibility of items in different aspects [2, 11, 31]. However, these works only focus on the compatibility of two items instead of a whole outfit.

Recently, some studies have been trying to model the compatibility of a whole outfit directly. Li *et al.* extracted multi-modal information from items and evaluated the outfit compatibility with an Recurrent Neural Network (RNN) [17]. Then they constructed an outfit recommendation by selecting the items which achieve the highest compatibility scores along with the given items. Han *et al.* represented an outfit as a sequence with a specific order. Then they used bidirectional LSTMs to predict the next item given a set of items and also the compatibility scores of outfits [9]. Modeling outfit compatibility relies on a proper representation of the outfit, which can reflect the complex relations among multiple items. In this work, we represent an outfit as a graph, which is proved to be more effective than pairs and sequence in the later experiments.

Categorical information plays a dominant role in the representation of an item. In the above conventional methods modeling items in a common visual latent space, the items of same categories tends to be close, which makes it harder to reflect the style information. For example, in the common visual latent space, the similarity between suit pants and leather shoes is much smaller than the similarity between suit pants and jeans. In Sherlock [10], the embedding matrices for transferring visual features to style features vary among different categories. Liu *et al.* proposed that items consist of two components: style and category [20]. In this work, we utilize the categorical information directly by putting items into its corresponding category nodes in the *Fashion Graph*.

2.2 Graph Neural Networks

There have been various ways to deal with graph-structured data. Some works convert graph-structured data into sequence-structured data to deal with. Inspired by word2vec [26], Perozzi *et al.* proposed an unsupervised DeepWalk algorithm to learn node embedding in graph based on random walks [27]. After that, Tang *et al.* proposed a network embedding algorithm LINE [36], followed by node2vec [8]. Some works apply neural networks on the graph-structured data directly. Duvenaud *et al.* designed a special hashing function so that Convolutional Neural Network (CNN) can be used on the graphs [5]. Kipf *et al.* proposed a semi-supervised method on graphs based on an efficient variant of CNN [14].

Gori *et al.* first proposed Graph Neural Networks (GNN) [7], another approach to applying neural networks directly to graph-structured data. GNN can be applied on most kinds of graphs, including directed, undirected and cyclic graphs. Feed-forward neural networks are applied to every node in the graph recurrently. The nodes interact with others by communicating their state information. Thus, the hidden state of nodes will update dynamically. Scarselli *et al.* utilized multi-layer perceptions (MLP) to update the hidden state of nodes [29]. However, there exists problem in propagating information across a long range in a graph. To remedy this, Li *et al.* proposed Gated Graph Neural Network (GGNN) by introducing a Gated Recurrent Units (GRU) for updating [18]. GGNN has been utilized for various tasks, such as image classification [22], situation recognition [16], recommendation [39] and script event prediction [19]. Nevertheless, it has difficulty in modeling complex node interactions since the communication of state information on the edges are in a fixed way. To address this limitation, we propose a novel model NGNN, which can better model node interactions so as to better infer outfit compatibility from the graph. In NGNN, the communication of state information on each edge is different. It can model edge-wise interactions with node-wise parameters.

3 DATASET AND FEATURES

In this section, we introduce the dataset we use and how we extract the visual and textual features of items, which are used as input of our proposed method.

3.1 Polyvore Dataset

Polyvore dataset released by [9] was collected from a popular fashion website Polyvore.com[2], where fashion stylists can share the outfits they created to the public. As shown in Figure 3, items in the outfits have rich information including clear images, category information, titles, numbers of likes, etc. We only use the images, titles and category information of items in this paper. The dataset has been utilized by several works on fashion analysis [13, 17, 32].

[2] http://www.polyvore.com/.

	Category	Image	Title
	Coats		Irregular lapel blue trench coat
	Knee Length Skirts		Beautiful cotton lace skirt yellow
	Sandals		Mcq alexander mcqueen mirrored leather sandals
	Necklaces		Navy blue yellow statement bib necklace earrings set

Figure 3: Examples of outfit composition on Polyvore.com. Each item has a clear image, titles and category information.

There are totally 21889 outfits covering 380 categories in the Polyvore dataset, which was split into three non-overlapping sets, 17316 for training, 1497 for validation and 3076 for testing. Following [32], there are many categories appearing too few times in the dataset such as *Toys*, *Furniture* and *Cleaning*, which are not actually fashion categories. To ensure the training quality, we keep categories which appear more than 100 times in the dataset and 120 categories remain. To ensure the integrity of outfits, we filtered out the outfits consisting of less than 3 items. Finally, there remain 16983 outfits in the training set, 1497 in the validation set and 2697 in the test set, with totally 126054 items covering 120 categories. The maximum size of outfits is 8 and the average size is 6.2.

3.2 Features

We extract the visual features using the images of items and textual features using titles. We will model the outfit compatibility using these features of items as described in the following.

3.2.1 Visual Feature. We utilize the advanced deep convolutional neural network GoogleNet InceptionV3 model [34] to extract the visual features, which has been proved to be effective in image representation learning [4, 9, 15, 30]. In particular, we feed the images of items into the model and adopted the linear layer output as the visual feature. The visual feature of each item is a 2048-dimensional vector.

3.2.2 Textual Feature. Following [32], we first construct a vocabulary based on the words of the titles in the dataset. Due to the noise of these user-generated titles, we first filter out the words which appear in less than 5 items. Since there are many meaningless words such as ‘a’, ‘an’ and ‘de’, we then filter out words with less than 3 characters. Finally we obtain a vocabulary with 2757 words and therefore we represented the textual feature of each item as a 2757-dimensional Boolean vector.

4 PROPOSED METHOD

We first formulate the problem. Then we introduce how to construct the *Fashion Graph* and model outfit compatibility from the graph with NGNN. We also introduce how to jointly model outfit compatibility from multiple modalities with NGNN. Finally, we describe the training strategy and analyze our model.

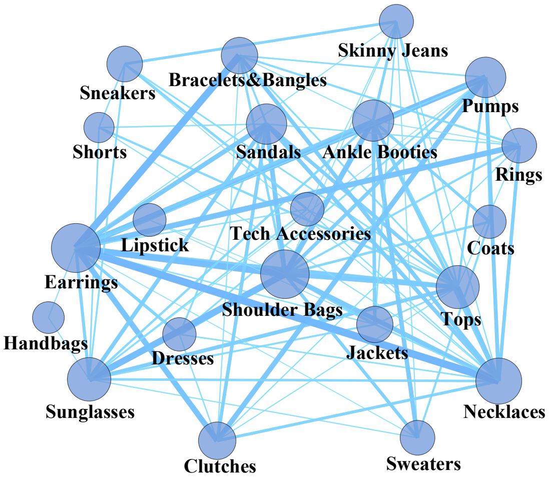


Figure 4: Illustration of the most popular categories in Polyvore dataset. Each circle denotes a category, and each link denotes the matching relation between two categories (i.e., they have appeared in the same outfit). The areas of circles and the widths of the links are proportional to the number of fashion items in the corresponding categories and the co-occurrence frequency between categories.

4.1 Problem Formulation

In this paper, we aim to model the compatibility of outfits. We have a set of outfits $\mathcal{S} = \{s_1, s_2, s_3, \ldots\}$ in the training set. The total item set is $\mathcal{V}$. Given an outfit s consisting of $|s|$ items (each item has an image and textual description), we aim to predict the compatibility score x_s of the outfit. The score can be used for various tasks of fashion recommendation.

We introduce some notations which will be used in the following. For each item $v_i \in \mathcal{V}$, its feature is f_i and the representation of f_i in the latent space is r_i. The category of v_i is c_i. The corresponding node of c_i in the *Fashion Graph* is n_i, while the state of node n_i in NGNN is h_i.

4.2 Graph Construction

In Figure 4, we illustrate the most popular categories in the Polyvore dataset. Each circle denotes a category, and each link denotes the matching relation between two categories (i.e., the two categories have appeared in the same outfit). The areas of circles and the widths of the links are proportional to the number of fashion items in the corresponding categories and the co-occurrence frequency between categories.

Similar to Figure 4, we construct a *Fashion Graph* $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ according to the training dataset. Each node in the graph $n_i \in \mathcal{N}$ represents a category $c_i \in C$, so that $|\mathcal{N}| = |C|$. The edge represents interaction between nodes. If two categories have matching relation in the training dataset, there are two directed edges in reverse directions between the two corresponding nodes in *Fashion Graph*, since the interactions in two directions ought to different.

For example, one may select a pair of socks to match the shoes but less likely to select a pair of shoes to match socks particularly.

According to the training dataset, we calculate the weights of directed edges as follow,

$$w(n_i, n_j) = \frac{Count_{c_i,c_j}/Count_{c_j}}{\sum_k Count_{c_i,c_k}/Count_{c_k}}, \tag{1}$$

where $w(n_i, n_j)$ denotes the weight of edge from n_i to n_j, $Count_{c_i,c_j}$ is the co-occurrence frequency of category c_i and c_j, and $Count_{c_j}$ is the occurrence frequency of category c_j.

The constructed *Fashion Graph* has 120 nodes and 12500 directed and weighted edges. By putting each item into its corresponding node, an outfit can be represented as a subgraph of *Fashion Graph*. Then we use NGNN to infer the compatibility from the subgraph as described in the next section.

4.3 Node-wise Graph Neural Network

Our proposed method NGNN consists of three steps. The first step is to learn the initial node state. The second step is to model node interactions and update node state. The third step is to calculate the compatibility score with attention mechanism.

4.3.1 Learning Initial Node State. In NGNN, each node n_i is associated with a hidden state vector h_i, which will be updated dynamically. The input of NGNN is features (either visual or textual features) of items, which can be used to initialize the state of their corresponding nodes. For each item $v_i \in s$, we first map its feature f_i to a latent style space with the size of d. Considering the difference among the categories, we have a linear mapping matrix W_h^i for each category c_i. Thus, we can obtain the representation of item v_i in the latent space as follows,

$$r_i = W_h^i f_i. \tag{2}$$

We then use the representation to initialize the hidden state h_i^0 of its category corresponding node n_i as,

$$h_i^0 = tanh(r_i). \tag{3}$$

Thus, the size of node state is also d.

4.3.2 Modeling Node Interactions. Each time the nodes communicate (i.e., propagate their state information to others and receive others' information), which is called a propagation step, the state of nodes will be updated.

In the traditional GGNN, at propagation step t, the nodes will receive sum of neighbors' state as,

$$a_i^t = \sum_{n_j \to n_i \in \mathcal{E}} \mathrm{A}[n_j, n_i] W_p h_j^{t-1} + b_p, \tag{4}$$

where W_p and b_p are weights and biases of a shared linear transformation on all the edges. $\mathbf{A}$ is the adjacency matrix:

$$\mathbf{A}[n_i, n_j] = \begin{cases} w(n_i, n_j), \text{ if } n_i \to n_j \in \mathcal{E}, \\ 0, \text{ others .} \end{cases} \tag{5}$$

The traditional GGNN use the same W_p and b_p to model interactions between different nodes. However, it fails to model the complex and flexible interactions between them. Since the node interaction on each edge ought to be different, we aim to model edge-wise interaction. There is a straight-forward way to achieve edge-wise node interactions, that is to let each edge has its own transformation weight and bias W_p and b_p. However, it will cost a great parameter space with numerous edges in a graph and cannot be applied on large scale graphs.

To remedy the problem above, we have an output matrix W_{out}^i and an input matrix W_{in}^i for each node n_i in NGNN. The transformation function of edge $n_i \to n_j$ from node n_i to node n_j could be written as

$$W_p^{n_i \to n_j} = W_{out}^i W_{in}^j. \tag{6}$$

Thus, the Equation 4 could be rewritten as,

$$a_i^t = \sum_{n_j \to n_i \in \mathcal{E}} \mathbf{A}[n_j, n_i] W_{out}^j W_{in}^i h_j^{t-1} + b_p. \tag{7}$$

After receiving state information a_i^t, the state of node n_i will be updated as following,

$$z_i^t = \sigma(W_z a_i^t + U_z h_i^{t-1} + b_z), \tag{8}$$

$$r_i^t = \sigma(W_r a_i^t + U_r h_i^{t-1} + b_r), \tag{9}$$

$$\tilde{h}_i^t = tanh(W_h a_i^t + U_h(r_i^t \odot h_i^{t-1}) + b_h), \tag{10}$$

$$h_i^t = \tilde{h}_i^t \odot z_i^t + h_i^{t-1} \odot (1 - z_i^t), \tag{11}$$

where, $W_z, W_r, W_h, b_z, b_r, b_h$ are weights and biases of the updating function, which are similar to Gated Recurrent Unit (GRU) [18]. z_i^t and r_i^t are update gate vector and reset gate vector, respectively.

After T propagation steps, we can obtain the final state of nodes, which are also the final node representations. Then we can generate a graph-level output to serve as the compatibility score x_s of outfit s. We adopt the attention mechanism described in next section to compute the graph-level output.

4.3.3 Compatibility Calculating with Attention Mechanism. We contend two viewpoints on how fashion item influence the outfit compatibility here. The first viewpoint is that items have different influence on an outfit. For example, an improper top and an improper necklace have different influence to the outfit influence. The second is that the same item plays different roles in different outfits. For example, a coat maybe suitable for an outfit and promote the compatibility, while it will decrease the compatibility together with summer clothes. Thus, we aim to design an attention mechanism to model the influence of items on the outfit compatibility. Since a node receives state information from other nodes, their node representations are aware of the global information (i.e., other nodes' state). Similar with [18], we utilize self-attention [35, 37] to calculate the graph-level output and also the outfit compatibility score as following,

$$x_s = \sum_{i=1}^{|s|} \sigma(\theta(h_i^t)) \cdot \alpha(\delta(h_i^t)). \tag{12}$$

$\theta(\cdot)$ and $\delta(\cdot)$ are two perception networks to output a real value. $\theta(\cdot)$ is used to model the weights of items (i.e., importance of items' influence on the outfit compatibility), and $\delta(\cdot)$ is used to model the compatibility score of items (i.e., how the item goes with the other items in the outfit). $\alpha(\cdot)$ and $\sigma(\cdot)$ are leaky relu and sigmoid activate functions $\alpha(x) = max(0.01x, x), \sigma(x) = 1/(1 + e^{-x})$.

4.4 NGNN with Multi-modal Input

As described above, with the input of visual features or textual features of items, NGNN can be used to model the outfit compatibility from visual or textual modality. We here introduce how to jointly model the outfit compatibility from both visual and textual modality with NGNN.

We use two channels of NGNN, the input of a channel is visual features, and the input of another channel is textual features. Specifically, for each outfit s, we input the visual features and textual features to the two channels of NGNN respectively and obtain a visual compatibility score x_s^{vis} and a textual compatibility score x_s^{txt}. We calculate the final outfit compatibility score x_s as follow,

$$x_s = \beta x_s^{vis} + (1 - \beta) x_s^{txt} \tag{13}$$

where β is a non-negative trade-off parameter.

As described in Section 4.3.1, for each item v_i, its visual feature f_i^{vis} and textual feature f_i^{txt} are mapped into a shared latent space respectively. We denote their representations in latent space as r_i^{vis} and r_i^{txt}. Considering the coherent relation between the visual and textual information of items, we use a regularization to ensure the consistency between the visual and textual feature of the same item in the shared latent space as follow,

$$\mathcal{L}_{con} = \sum_{i=1}^{|s|} \left\| r_i^{vis} - r_i^{txt} \right\|^2 \tag{14}$$

4.5 Training Strategy

We denote outfits in the training dataset as positive outfits. For each outfit $s = \left\{ v_1, v_2, ..., v_p, ... \right\} \in \mathcal{S}$, we randomly select an item v_p and replace it with a random item $v_q \in \mathcal{V}$ to form a negative outfit $s^- = \left\{ v_1, v_2, ..., v_q, ... \right\}$. Thus, we can obtain a dataset $\mathcal{P}_{train}$ consisting of these pairs (s, s^-). Similar to Bayesian Personalized Ranking [28], we assume that the positive outfit has higher compatibility than the negative one. Therefore, we have the following objective function,

$$\mathcal{L}_{bpr} = \sum_{(s,s^-) \in \mathcal{P}_{train}} - \ln \sigma(x_s - x_{s^-}). \tag{15}$$

The objective function of NGNN is:

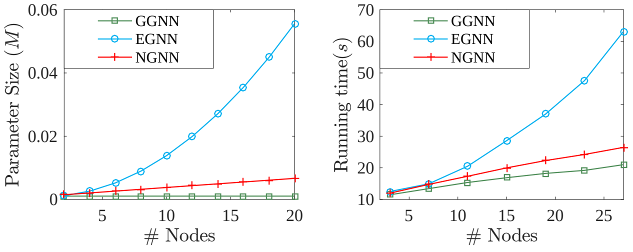
$$\mathcal{L}_1 = \mathcal{L}_{bpr} + \frac{\lambda}{2} \left\| \Theta \right\|^2, \tag{16}$$

where Θ refers to the set of parameters and λ is the hyper-parameter of L2 regularizer, which is designed to avoid overfitting.

When it comes to NGNN with multi-modal input, we have an additional loss function $\mathcal{L}_{con}$ as described in Section 4.4. Thus, we have the following objective function,

$$\mathcal{L}_2 = \mathcal{L}_{bpr} + \mathcal{L}_{con} + \frac{\lambda}{2} \left\| \Theta \right\|^2. \tag{17}$$

We only train NGNN on the subgraph of outfit each time. That is to say, only the nodes in the subgraph are trained and only their parameters are updated each time, which greatly reduces time complexity. When we have more data to train the model, we only need to update the parameters of concerned nodes without changing the parameters of edges or other nodes, which is effective in real applications.



(a) Parameter Size

(b) Running Time

Figure 5: The parameter sizes and running time of GGNN, EGNN and NGNN on graphs with the number of nodes varying from 2 to 30. EGNN is a simple variant of GGNN, in which every edge has a different dtransformation matrix for propagating state information.

4.6 Model Analysis

In this section, we analyze the parameter space and time complexity of our proposed method NGNN.

4.6.1 Parameter Space. The parameter space of our proposed method consists of two parts: the parameters correlated to nodes and the perception networks in attention mechanism. For each node n_i, we have a linear mapping matrix W_h^i to map the input features into the latent space, an input matrix W_{in}^i and an output matrix W_{out}^i to propagate state information. Totally we have $3 \left| \mathcal{N} \right|$ matrices, which are proportional to the number of nodes $|\mathcal{N}|$. In addition, we have two matrices of perception networks in the attention mechanism. Overall, we have $3 \left| \mathcal{N} \right|$ matrices. Therefore, we say that the parameters of NGNN is node-wise.

4.6.2 Time Complexity. For each input outfit s, the time complexity of training process in NGNN consists of three parts. The first part is calculating the transformed state information nodes receive, which is $O(|s|)$. The second part is updating the state of the concerned nodes, which is also $O(|s|)$. The last part is updating the parameters of concerned nodes. Since the number of parameters correlated to the concerned nodes is $3\,|s|$ as described above, its complexity is also $O(|s|)$. According, the total complexity is $O(|s|)$.

We compare NGNN with GGNN [18] and Edge-wise GNN (EGNN). EGNN is a straight-forward way to achieve edge-wise interactions. In EGNN, each edge has an unique transformation matrix for propagating state information. We train the three models on a toy dataset consisting of outfits with the number of items varying from 2 to 30. In another word, the three models are trained on graphs with the number of nodes varying from 2 to 30. The parameter size and running time are shown in Figure 5. As can be seen, the parameter size and running time of EGNN increased quadratically while those of NGNN and GGNN increased linearly.

In summary, NGNN can better model edge-wise interactions and learn better node representations with appropriate parameter space and time complexity. When there are more training data, we can only update the parameters of concerned nodes offline and upload them online without changing the edges or other nodes, which is effective in real applications.

5 EXPERIMENTS

As the main contribution of this work is to model outfit compatibility, we aim to answer the following research questions via experiments.

RQ1 How does our proposed NGNN perform compared with other pair or sequence representation models?
RQ2 How do different components in NGNN effect the performance?
RQ3 How do different modalities influence the inference of compatibility?
RQ4 How do the hyper-parameters d and T effect the performance?

Next, we first describe the two tasks we conduct experiments on and the experimental settings. We then report the results by answering the above research questions in turn. Finally, we analyze the parameter space and time complexity of our proposed model.

5.1 Task Description

We conduct experiments on two tasks: fill-in-the-blank fashion recommendation and compatibility prediction on the polyvore dataset described in Section 3.1.

5.1.1 Fill-in-the-blank Fashion Recommendation. The task of fill-in-the-blank (FITB) in fashion recommendation is first introduced in [9]. In this task, Given a set of fashion items and a blank, we aim to find the most compatible item from the candidates set to fill in the blank as shown in Figure 6. This is a practical scenario in people's daily life. For example, a person wants to select clothes to match the pants, hat and shoes. Following [9], we create a fill-in-the-blank dataset based on the Polyvore test dataset. For each outfit in the test dataset, we randomly select an item from it and replace the item with a blank. By randomly select 3 negative items from $\mathcal{I}$ we can form a 4-length candidates set along with the positive one, assuming the negative items are less compatible than the positive one. In our work, we respectively put the four candidate items into the blank to form an outfit and calculate their compatibility scores with our model. The one which achieves the highest outfit compatibility score is chosen as the answer. The performance is evaluated by the accuracy of choosing the right answer from the four candidates (FITB accuracy), which is 25% by randomly selecting.

5.1.2 Compatibility Prediction. This task is to predict the compatibility score for any given outfit. To evaluate our model, we first construct a dataset $\mathcal{P}_{test}$ using the Polyvore test dataset. Specifically, for each positive outfit s in the Polyvore test dataset, we create a negative outfit s^- by randomly selecting $|s|$ items from $\mathcal{V}$. Thus, we can obtain a dataset $\mathcal{P}_{test}$ consisting of evaluation pairs (s, s^-). We then adopt the widely used metric AUC (Area Under the ROC curve) to evaluate the performance, which is defined as,

$$AUC = \frac{1}{|\mathcal{P}_{test}|} \sum_{(s,s^-)\in\mathcal{P}_{test}} \delta\left(x_s > x_{s^-}\right), \tag{18}$$

where $\delta(a)$ is an indicate function that returns one if the argument a is *true* and zero otherwise.

Table 1: The performance of models evaluated by Fill-in-the-blank accuracy and compatibility prediction AUC.

Method	Accuracy (FITB)	AUC (Compatibility)
Random	24.07%	0.5104
SiameseCNNs	48.09%	0.7087
LMT	50.91%	0.6782
Bi-LSTM	67.01%	0.8427
GGNN (visual)	72.78%	0.9516
GGNN (textual)	73.63%	0.9591
NGNN (visual)	77.01%	0.9600
NGNN (textual)	77.78%	0.9716
NGNN (multi-modal)	**78.13%**	**0.9722**

5.2 Experimental Settings

For optimization, we adopt the RMSProp optimizer. The optimal hyper-parameters (learning rate, batch size, trade-off parameter β, parameter for L2 regularization λ, hidden size d and the propagation steps T) are determined by the grid search strategy. We experimentally find that the model achieves optimal performance with the learning rate as 0.001, batch size as 16, β as 0.2, λ as 0.001, d as 12 and T as 3. We early stop the training process when the loss stabilizes, usually at 12 epoch. All the experiments were conducted over a sever equipped with 3 NVIDIA Titan X GPUs.

5.3 Model Comparison (RQ1)

To evaluate our proposed method, we compare it with the following state-of-the-art baselines. Note that SiameseCNNs and LMT focus on the compatibility between two items. In the task of outfit recommendation, we sum up the compatibility scores of all the pairs in the outfit as the score of the outfit.

Random: Randomly assign the compatibility scores of outfits.
SiameseCNNs: It's a pair-wise compatibility learning method [38], which minimizes the distance of image vectors between compatible items and maximums the distance between incompatible items. This method only focuses on the visual information.
LMT: Proposed by [23], it models compatibility between items in a style space via a single low-rank Mahalanobis embedding matrix. This method also focuses on visual modality.
Bi-LSTM: It's proposed by [9]. By viewing an outfit as a sequence, it uses the bidirectional LSTMs to predict the next item, taking multi-modal data as input, which can also predict the compatibility scores of outfits. This method focus on multi-modal information.
GGNN (visual / textual): We use GGNN to model the outfit compatibility with input of visual or textual features of items.
NGNN (visual / textual): We use our proposed NGNN to model the outfit compatibility with input of visual or textual features of items.
NGNN (multi-modal): We jointly models the outfit compatibility with NGNN from the visual and textual modalities.

We conduct comparison experiments on the tasks of fill-in-the-blank fashion recommendation and compatibility prediction. The results are shown in Table 1. Next we analyze the results of FITB accuracy and compatibility AUC respectively.

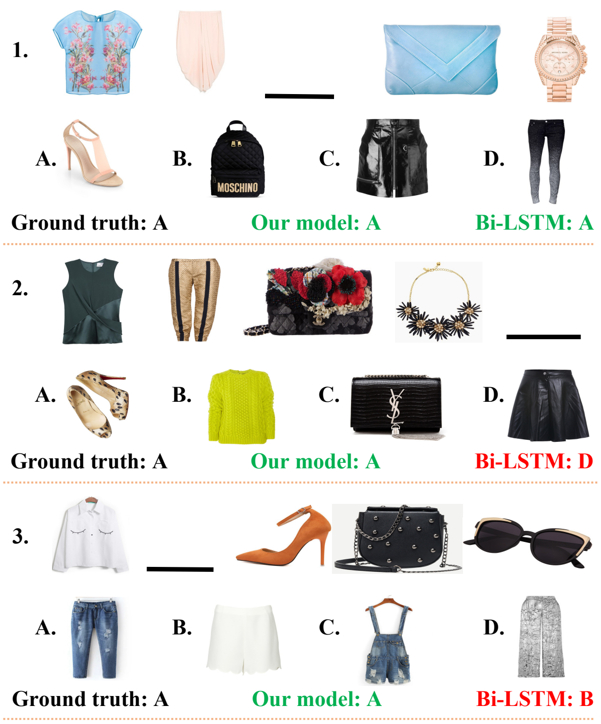


Figure 6: Comparison of our model and Bi-LSTM on several examples in fill-in-the-blank task. Green fonts represent the right chosen and red fonts represent wrong chosen.

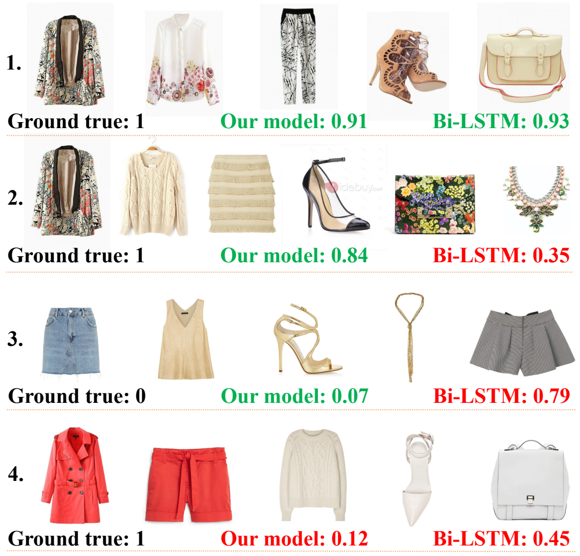


Figure 7: Comparison of our model and Bi-LSTM on several example of task compatibility prediction. Green fonts represent the right evaluation and red fonts represent wrong.

5.3.1 FITB Accuracy. The comparison of FITB accuracy is shown in the middle column of Table 1. We can obtain the following observations from the table: (1) SiameseCNNs and LMT achieve worse performance compared with the other methods, which demonstrates that the pair representation ignores the integrity of outfit. (2) GGNN outperforms Bi-LSTM, suggesting that our proposed graph representation can better reflect the complex relations among items in an outfit than sequence representation. (3) NGNN achieves a better performance than GGNN, which proves that our proposed NGNN can better infer compatibility information from the graph by better modeling node interactions. (4) Using textual features achieves better performance than visual features for both GGNN and NGNN, which suggests that the textual features can better represent fashion items than visual features in that the titles accurately summarize the key features of items. (5) NGNN with Multi-modal input achieves best performance, proving that the visual and textual information are good supplements to each other. In addition, our proposed method can jointly model the outfit compatibility from multiple modalities. We randomly select several example outfits in the FITB dataset to test on our model and Bi-LSTM. The results are shown in Figure 6. From example 1, we can see that Bi-LSTM and our model can both correctly select the item of complementary category as the outfit lacks of a pair of shoes. In example 2, our model chooses the right item while Bi-LSTM choose the skirt wrongly, which conflicts with the pants. This may be because that the position of pants is far away from the blank so that it has little influence on it. In example 3, our model selects the most compatible bottom jeans with the whole set from the four bottoms while Bi-LSTM selects the white short skirt which may be most compatible with the white shirt, but not such compatible with the other items. The wrong chosen of Bi-LSTM may be because that the nearby white shirt has strong influence on the blank. From these exemple, we can draw a conclusion: the items have strong influence on nearby items in the sequence while have less influence on items faraway. This makes the sequence represented method relies on a proper sequence order. Compared with the sequence-represented model Bi-LSTM, our graph-represented model can better model the complex relations among items in an outfit.

5.3.2 Compatibility AUC. The performance comparison of models evaluated by compatibility AUC is shown in the third column of Table 1. Similar with the comparison of performance evaluated by FITB accuracy, NGNN achieves best performance, GGNN is second best, and then is the sequence represented method Bi-LSTM, the pair represented methods (SiameseCNNs, LMT) are worst. Using textual features can achieve better performance than visual features. NGNN with multi-modal data achieves the best performance. These observations support our analysis in Section 5.3.1. In addition, it can be seen that GGNN and NGNN all achieve great performance and the gap is small, which proves the superiority of graph representation on compatibility prediction. GGNN is already good enough for compatibility prediction.

We randomly select several example outfits and show the compatibility scores predicted of Bi-LSTM and our model in Figure 7. In example 1, our model gives the outfit a high score correctly while Bi-LSTM gives it a relatively low score, which may be because that the sequence representation method fails to capture the high compatibility between the first top and the fifth handbag. While Bi-LSTM and our model both give the outfit a high score correctly in example 2. In example 3, the fifth shorts conflict with the first skirts. Our model gives it a low score correctly while Bi-LSTM gives it a relatively high score, which may be because Bi-LSTM fails to capture the conflict of the two bottoms across a long distance. Examples 2 and 3 suggest that Bi-LSTM relies on a proper order of sequence, which is flexible and hard to learn. The influence between two items is inversely proportional to their distance in the sequence. Our model can better reflect the relation among items compared with Bi-LSTM. In example 4, our model gives it a low score to the positive outfit. A plausible explanation is that there is little matching of coat, sweater and shorts in training dataset.

Table 2: The performance of our proposed method with different components.

Method	FITB Accuracy	Compatibility AUC
NGNN(-W-A)	75.17%	0.9541
NGNN(-W)	75.71%	0.9559
NGNN(-A)	76.12%	0.9578
NGNN	**77.01%**	**0.9600**

5.4 Component Comparison (RQ2)

To investigate the effect of different components on the performance of our proposed method, we compare our method with different components.

NGNN (-W): NGNN without weighted adjacency matrix. Specifically, we use 0-1 adjacency matrix instead of the weighted one as described in Section 4.2.

NGNN (-A): NGNN without attention mechanism. We only use a perception network to model the compatibility scores of nodes and sum them up them.

NGNN (-W-A): NGNN with neither the attention mechanism nor weighted adjacency matrix.

The performances are shown in Table 2. If not specifically stated, we use visual features as input of NGNN. We observe that NGNN outperforms all the ablative methods, which proves the necessity of the two components in our model. NGNN (-W) achieves worse performance than NGNN, suggesting that the weighted adjacency matrix we calculated is a supplement to the information of graph. NGNN outperforms NGNN (-A), which confirms the necessity of our attention mechanism.

To intuitively illustrate how the attention mechanism works, we show the attention weights of several example outfits in Figure 8. The depths of color are proportional to the attention weight of items. The higher attention weight item has, the deeper the color of bar below it is, the stronger influence the item has to the outfit compatibility. As can be seen in example 1, the shirt, short skirt and necklace have distinct and compatible styles and they have strong influence to the outfit compatibility. The black bag and black shoes have little influence to outfit compatibility while they have ordinary styles. Similarly, the sweater, shoes and hat in example 2 are all of distinct rustic style and they have strong influence to the outfit compatibility. The jeans and shoes in example 3 can match well with most styles while they have little influence to the outfit compatibility. The white outwear and jeans which can match with many styles in example 4 also have little influence to outfit compatibility. From these examples, we can find that more distinct style items have, more influential they are to the outfit compatibility. On the contrary, more ordinary items are (i.e., can match with many styles), less influential they are. This is reasonable in common sense.

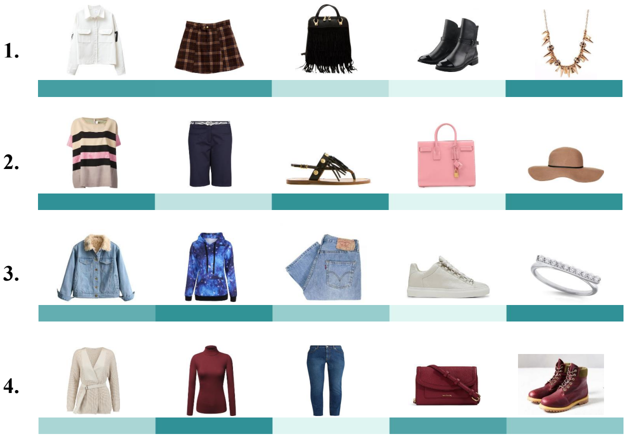


Figure 8: Illustration of attention mechanism. The depth of color are proportional to the attention weights of items.

5.5 Modality Comparison (RQ3)

To investigate the influence of different modalities on the performance, we compare the NGNNs (multi-modal) with different modality combinations. As described in Equation 13, β is the weight of visual compatibility score while that of textual compatibility score is $1 - \beta$. Note that when $\beta = 0$ the model is actually NGNN (visual), when $\beta = 1$ it's NGNN (textual). In Figure 10, we show the performance of NGNN (multi-modal) with β varying from 0 to 1. It can be seen when $\beta = 0.2$, our model achieves the best performance, which suggests that the visual and textual information are good supplement to each other and both contribute s to the inference of compatibility. We can also see that using textual features can achieve better performance than visual features. A plausible explanation is that the textual information is more concise to capture the key features of items.

To intuitively see the difference of NGNN with visual, textual and multi-modal input, we show the comparison of them on several testing examples in Figure 9. In the example 1, Visual NGNN chose the false answer due to its similar color and material with the jacket, which the visual information can reflect. However the similarity between the textual information (titles) of the true answer and the dress, jacket directly reveals their high compatibility. In example 2, it's hard to choose the answer with textual input since the titles of the two answers are similar, while with visual input NGNN can distinct the high compatibility between the styles of the true answer and the outfit from visual information. In summary, the visual and textual information are great supplements to each other. By integrating them we can achieve better performance.

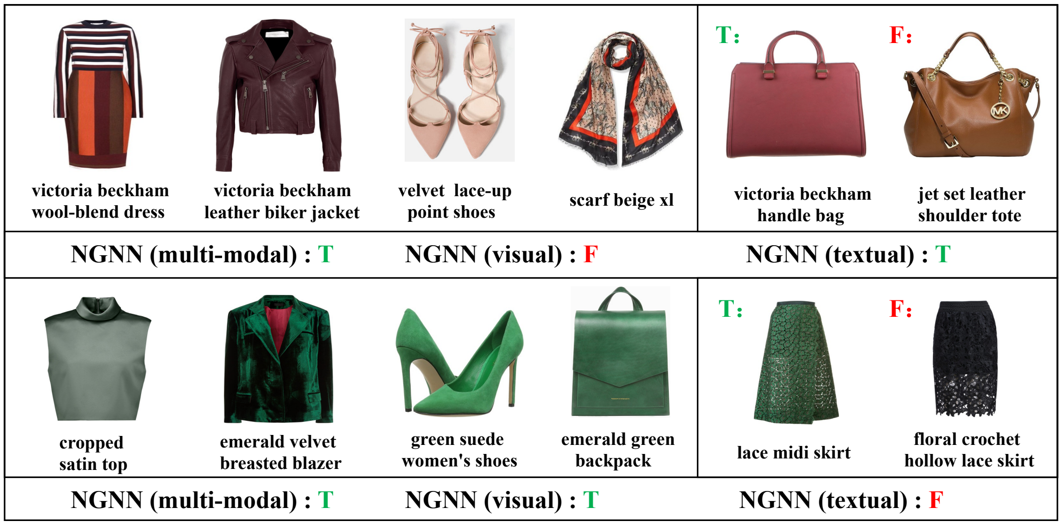

victoria beckham wool-blend dress
victoria beckham leather biker jacket
velvet lace-up point shoes
scarf beige xl
T:
F:
victoria beckham handle bag
jet set leather shoulder tote
NGNN (multi-modal) : T
NGNN (visual) : F
NGNN (textual) : T
cropped satin top
emerald velvet breasted blazer
green suede women's shoes
emerald green backpack
T:
F:
lace midi skirt
floral crochet hollow lace skirt
NGNN (multi-modal) : T
NGNN (visual) : T
NGNN (textual) : F

Figure 9: Comparison of NGNN(visual), NGNN(textual) and NGNN(multi-modal). For each outfit in the left, there is a true answer (T) and a false answer (F). Green fonts reprsent true chosen while red fonts represent false chosen.

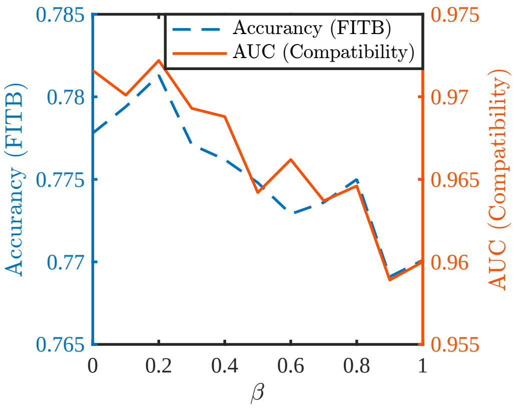
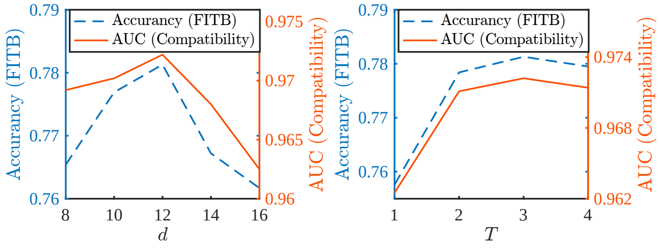


Figure 10: FITB accuracy and compatibility AUC of our model with different β.



(a) State dimensionality

(b) Propagation times

Figure 11: AUC and FITB accuracy performance with different dimensionality d (left) and propagation times T (right).

5.6 Hyper-parameter Discussion (RQ4)

In this section, we examine how the dimensionality of hidden vectors d and the propagation time d affect the performance. Comparison experiments are conducted on NGNN (multi-modal). Figure 11 shows the performance of our model with varying d and d. From the figure in the left, we can observe that the FITB accuracy and compatibility AUC reach the highest point when d is around 12. When d is large than 12, the performance drops sharply. This indicates that there is no need to represent the node state with too much parameters. From the figure in the right, it can be seen that the performance saturates when T is 3 and the nodes in NGNN have interacted with others sufficiently, suggesting that it's important to model the multi-hop propagation so that the nodes are better aware of other nodes' information. Moreover, when T is high, the superfluous information propagation will confuse the nodes.

6 CONCLUSION

In this paper, we first propose to model the outfit compatibility by representing an outfit as a graph since the graph structure can better capture the complex relations among items in outfits. To infer the compatibility information from the graph, we further propose a novel model NGNN, which can better model flexible node interactions and learn better node representations. Our proposed method can model outfit compatibility not only from visual or textual modality respectively, but also from multiple modalities jointly. Experimental results on a real-world dataset (Polyvore dataset) prove the great superiority of our method over existing works. In the future, we aim to model personalized compatibility since people have different notions of fashion compatibility.

ACKNOWLEDGMENTS

This work is jointly supported by National Natural Science Foundation of China (61772528, 61871378) and National Key Research and Development Program (2016YFB1001000).

REFERENCES

[1] Sean Bell and Kavita Bala. 2015. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 98.

[2] Long Chen and Yuhang He. 2018. Dress Fashionably: Learn Fashion Collocation With Deep Mixed-Category Metric Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2103–2110.

[3] Qiang Cui, Shu Wu, Qiang Liu, Wen Zhong, and Liang Wang. 2018. MV-RNN: A Multi-View Recurrent Neural Network for Sequential Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[4] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*. 647–655.

[5] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*. 2224–2232.

[6] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. (1999).

[7] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, Vol. 2. IEEE, 729–734.

[8] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

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

[10] Ruining He, Chunbin Lin, Jianguo Wang, and Julian McAuley. 2016. Sherlock: sparse hierarchical embeddings for visually-aware one-class collaborative filtering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 3740–3746.

[11] Ruining He, Charles Packer, and Julian McAuley. 2016. Learning compatibility across categories for heterogeneous item recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 937–942.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

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

[14] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907* (2016).

[15] Hanbit Lee, Jinseok Seol, and Sang-goo Lee. 2017. Style2Vec: Representation Learning for Fashion Items from Style Sets. *arXiv preprint arXiv:1708.04014* (2017).

[16] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. 2017. Situation recognition with graph neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 4173–4182.

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

[18] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).

[19] Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing Narrative Event Evolutionary Graph for Script Event Prediction. *arXiv preprint arXiv:1805.05081* (2018).

[20] Qiang Liu, Shu Wu, and Liang Wang. 2017. Deepstyle: Learning user preferences for visual recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 841–844.

[21] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: a recurrent model with spatial and temporal contexts. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 194–200.

[22] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. 2017. The More You Know: Using Knowledge Graphs for Image Classification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 20–28.

[23] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.

[24] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

[25] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5528–5531.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[27] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[29] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80.

[30] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR workshops*.

[31] Yong-Siang Shih, Kai-Yueh Chang, Hsuan-Tien Lin, and Min Sun. 2018. Compatibility family learning for item recommendation and generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*. 2403–2410.

[32] Xuemeng Song, Fuli Feng, Jinhuan Liu, Zekun Li, Liqiang Nie, and Jun Ma. 2017. Neurostylist: Neural compatibility modeling for clothing matching. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 753–761.

[33] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

[34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[35] Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. Deep Semantic Role Labeling with Self-Attention. *arXiv preprint arXiv:1712.01586* (2017).

[36] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[38] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. 2015. Learning visual clothing style with heterogeneous dyadic cooccurrences. In *Proceedings of the IEEE International Conference on Computer Vision*. 4642–4650.

[39] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Xing Xie, and Tieniu Tan. 2018. Session-based Recommendation with Graph Neural Networks. In *Thirty-Third AAAI Conference on Artificial Intelligence*.