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
lizekunlee@gmail.com

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

Xiaoyu Zhang
Institute of Information Engineering, Chinese Academy of Sciences
zhangxiao@ie.ac.cn

Liang Wang
Institute of Automation, 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.

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.

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW ’19, May 13–17, 2019, San Francisco, CA, USA
© 2019 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.
ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313444

CCS CONCEPTS
• Applied computing → Online shopping.

KEYWORDS
Compatibility learning, graph neural networks, multi-modal

ACM Reference Format:

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 [21] 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, 22, 23, 27, 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.
A compatible outfit should have a key property: the categories of vantage of two kinds of outfit representations: pair representation and sequence representation 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 and sequence representation. 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 a real-world dataset (Polyvore dataset) demonstrate the great superiority of our method over others. The code of our work is released at https://github.com/CRIPAC-DIG/NGNN. 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 measured the distance there [21]. Following that, Veit et al. proposed to learn the distance metric with an end-to-end SiameseCNNs.
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 [26]. In this work, we utilize the categorical information directly by putting items into its corresponding category nodes in the Fashion Graph.

### 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 ¹, 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].

There are totally 21889 outfits covering 380 categories in the Polyvore dataset, which was split into three non-overlapping sets.

---

¹http://www.polyvore.com/.
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.

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.

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 3 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.

4.1 Problem Formulation

In this paper, we aim to model the compatibility of outfits. We have a set of outfits $S = \{s_1, s_2, \ldots\}$ 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 $s_i$ 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 $v_i$ 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., 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.
According to the training dataset, we calculate the weights of directed edges as follow,
\[
    w(n_i, n_j) = \frac{\text{Count}_{c_i,c_j}}{\text{Count}_{c_i}} \cdot \sum_k \frac{\text{Count}_{c_i,c_k}}{\text{Count}_{c_k}},
\]
where \(w(n_i, n_j)\) denotes the weight of edge from \(n_i\) to \(n_j\). \(\text{Count}_{c_i,c_j}\) is the co-occurrence frequency of category \(c_i\) and \(c_j\), and \(\text{Count}_{c_i}\) is the occurrence frequency of category \(c_i\).

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 \(W^l_h\) for each category \(c_j\). Thus, we can obtain the representation of item \(v_i\) in the latent space as follows,
\[
r_i = W^l_h f_i.
\]
We then use the representation to initialize the hidden state \(h^0_i\) of its category corresponding node \(n_i\) as,
\[
h^0_i = \text{tanh}(r_i).
\]
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,
\[
a^t_i = \sum_{n_j \rightarrow n_i \in E} A[n_j, n_i] W_p h^{t-1}_j + b_p,
\]
where \(W_p\) and \(b_p\) are weights and biases of a shared linear transformation on all the edges. \(A\) is the adjacency matrix:
\[
A[n_i, n_j] = \begin{cases} 
    w(n_i, n_j), & \text{if } n_i \rightarrow n_j \in E, \\
    0, & \text{others}.
\end{cases}
\]
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 straightforward 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 \(W^l_{out}\) and an input matrix \(W^l_{in}\) for each node \(n_i\) in NGNN. The transformation function of edge \(n_i \rightarrow n_j\) from node \(n_i\) to node \(n_j\) could be written as
\[
W^l_{p(n_i \rightarrow n_j)} = W^l_{out} W^l_{in}.
\]
Thus, the Equation 4 could be rewritten as,
\[
a^t_i = \sum_{n_j \rightarrow n_i \in E} A[n_j, n_i] W^l_{out} W^l_{in} h^{t-1}_j + b_p.
\]
After receiving state information \(a^t_i\), the state of node \(n_i\) will be updated as following,
\[
z^t_i = \sigma(W_z a^t_i + W_h h^{t-1}_i + b_z),
\]
\[
r^t_i = \sigma(W_r a^t_i + W_{hr} h^{t-1}_i + b_r),
\]
\[
\tilde{h}^t_i = \tanh(W_h a^t_i + W_{hr} f^t_i \circ h^{t-1}_i + b_h),
\]
\[
\tilde{h}^t_i = \tilde{h}^t_i \circ z^t_i + h^{t-1}_i \circ (1 - z^t_i),
\]
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]. \(z^t_i\) and \(r^t_i\) 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,
\[
x_s = \sum_{i=1}^{|s|} \sigma(\theta(h^t_i)) \cdot \sigma(\delta(h^t_i)).
\]
\(\theta(\cdot)\) 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). \(\sigma(\cdot)\) and \(\sigma(\cdot)\) are leaky relu and sigmoid activate functions \(\sigma(x) = max(0.01x, x), \sigma(x) = 1/(1 + e^{-x})\).
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 \( x^v_s \) and a textual compatibility score \( x^t_s \). We calculate the final outfit compatibility score \( x_s \) as follow,

\[
x_s = \beta x^v_s + (1 - \beta) x^t_s
\]

where \( \beta \) is a non-negative trade-off parameter.

As described in Section 4.3.1, for each item \( v_i \), its visual feature \( f^{vis}_i \) and textual feature \( f^{txt}_i \) are mapped into a shared latent space respectively. We denote their representations in latent space as \( v^i \) and \( r^i \). 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,

\[
L_{con} = \sum_{i=1}^{n} \| v^i - r^i \|^2 \quad (14)
\]

4.5 Training Strategy

We denote outfits in the training dataset as positive outfits. For each outfit \( s = \{ v_1, v_2, ..., v_p, ... \} \in S \), we randomly select an item \( v_p \) and replace it with a random item \( v_q \in V \) to form a negative outfit \( s^- = \{ v_1, v_2, ..., v_q, ... \} \). Thus, we can obtain a dataset \( P_{train} \) 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,

\[
L_{bpr} = \sum_{(s,s^-) \in P_{train}} - \ln \sigma(x_s - x_{s^-}) \quad (15)
\]

The objective function of NGNN is:

\[
L_1 = L_{bpr} + \frac{\lambda}{2} \| \Theta \|^2 \quad (16)
\]

where \( \Theta \) refers to the set of parameters and \( \lambda \) 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 \( L_{con} \) as described in Section 4.4. Thus, we have the following objective function,

\[
L_2 = L_{bpr} + L_{con} + \frac{\lambda}{2} \| \Theta \|^2 \quad (17)
\]

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.

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 transformation 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 \( W^i_h \) to map the input features into the latent space, an input matrix \( W^i_{in} \) and an output matrix \( W^i_{out} \) to propagate state information. Totally we have \( 3|N| \) matrices, which are proportional to the number of nodes \( |N| \). In addition, we have two matrices of perception networks in the attention mechanism. Overall, we have \( 3|N| \) 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 affect the performance?
RQ3: How do different modalities influence the inference of compatibility?
RQ4: How do the hyper-parameters $d$ and $T$ affect 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 $V$ 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 $P_{test}$ 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 $V$. Thus, we can obtain a dataset $P_{test}$ 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:

$$\text{AUC} = \frac{1}{|P_{test}|} \sum_{(s, s^-) \in P_{test}} \delta(x_s > x_{s^-})$$  \hspace{1cm} (18)

where $\delta (a)$ is an indicator 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (FITB)</th>
<th>AUC (Compatibility)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>24.07%</td>
<td>0.5104</td>
</tr>
<tr>
<td>SiameseCNNs</td>
<td>48.09%</td>
<td>0.7087</td>
</tr>
<tr>
<td>LMT</td>
<td>50.91%</td>
<td>0.6782</td>
</tr>
<tr>
<td>Bi-LSTM</td>
<td>67.01%</td>
<td>0.8427</td>
</tr>
<tr>
<td>GGNN (visual)</td>
<td>72.78%</td>
<td>0.9516</td>
</tr>
<tr>
<td>GGNN (textual)</td>
<td>73.63%</td>
<td>0.9591</td>
</tr>
<tr>
<td>NGNN (visual)</td>
<td>77.01%</td>
<td>0.9600</td>
</tr>
<tr>
<td>NGNN (textual)</td>
<td>77.78%</td>
<td>0.9716</td>
</tr>
<tr>
<td>NGNN (multi-modal)</td>
<td>78.13%</td>
<td>0.9722</td>
</tr>
</tbody>
</table>

5.2 Experimental Settings

For optimization, we adopt the RMSProp optimizer. The optimal hyper-parameters (learning rate, batch size, trade-off parameter $\beta$, parameter for L2 regularization $\lambda$, 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, $\beta$ as 0.2, $\lambda$ 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 [21], 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.
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 suggests 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 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 example, 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>FITB Accuracy</th>
<th>Compatibility AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NGNN(W-A)</td>
<td>75.17%</td>
<td>0.9541</td>
</tr>
<tr>
<td>NGNN(W)</td>
<td>75.71%</td>
<td>0.9559</td>
</tr>
<tr>
<td>NGNN(-A)</td>
<td>76.12%</td>
<td>0.9578</td>
</tr>
<tr>
<td>NGNN</td>
<td>77.01%</td>
<td>0.9600</td>
</tr>
</tbody>
</table>

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 weights 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 outfit 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.

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, $\beta$ 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 $\beta$ 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 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.
5.6 Hyper-parameter Discussion (RQ4)

In this section, we examine how the dimensionality of hidden vectors $d$ and the propagation time $T$ affect the performance. Comparison experiments are conducted on NGNN (multi-modal). Figure 11 shows the performance of our model with varying $d$ and $T$. 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


[38] Andreas Veit, Balazs Kocsis, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. 2015. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *ICCV*.