Neural Compatibility Ranking for Text-based Fashion Matching

Suthee Chaidaroon*
Yi Fang
{schaidaroon,yfang}@scu.edu
Santa Clara University
Santa Clara, CA, USA

Mix Xie
Alessandro Magnani
{mxie,amagnani}@walmartlabs.com
Walmart Labs
Sunnyvale, CA, USA

ABSTRACT

When shopping for fashion, customers often look for products which can complement their current outfit. For example, customers want to buy a jacket which can go well with their jeans and sneakers. To address the task of fashion matching, we propose a neural compatibility model for ranking fashion products based on the compatibility matching with the input outfit. The contribution of our work is twofold. First, we demonstrate that product descriptions contain rich information about product comparability which has not been fully utilized in the prior work. Secondly, we exploit such useful information from text data by taking advantages of semantic matching and lexical matching both of which are important for fashion matching. The proposed model is evaluated on a real-world fashion outfit dataset and achieves the state-of-the-art results by comparing to the competitive baselines. In the future work, we plan to extend the model by incorporating product images which are the major data source in the prior work on fashion matching.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**;

KEYWORDS

Fashion Compatibility; Neural Information Retrieval

ACM Reference Format:
Suthee Chaidaroon, Yi Fang, Mix Xie, and Alessandro Magnani. 2019. Neural Compatibility Ranking for Text-based Fashion Matching. In *42nd Int'l ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331365

1 INTRODUCTION

According to the recent study by eMarketer, fashion products have become the largest product category for online shopping, and its sales are expected to surpass $118 billion in 2019[1]. The increasing demand for buying fashion products through an online service initiates many e-commerce businesses to develop fashion product retrieval. One very common task when doing fashion shopping is to find products which are compatible with customers' existing outfit, which is usually named fashion matching. In this problem, the system asks an online shopper to enter a set of fashion products that she/he has or likes, and then the system will suggest the most compatible fashion products. The fashion matching task is very challenging as it requires the underlying algorithm to understand not only high-level common fashion concepts such as jackets going with jeans, but also compatibility relationships among the products such as styles which can be subtle and subjective.

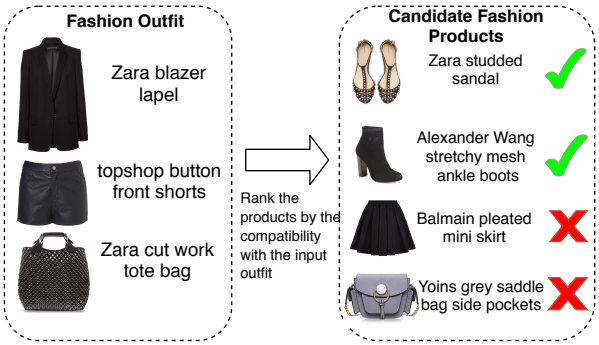

Figure 1: An example of a fashion outfit and candidate products. The model ranks the products in the candidate set by the compatibility with the outfit. It is worth noting that the proposed model only uses product descriptions. The check mark indicates a relevant item and the cross mark indicates an irrelevant item.

Most prior works infer a compatibility relationship from both product images and text descriptions [1–3] but they do not fully utilize product descriptions which contain rich information about product comparability. We posit that a product description consists of many informative keywords which are carefully chosen to describe the features and functionality of the product to increase sales. If the text data is fully utilized, we would be able to improve the state-of-the-art performance of the fashion matching task. This goal leads us to develop a sophisticated model to exploit such useful information from text data by resorting semantic matching and lexical matching techniques.

A latent semantic model is a good fit for this problem because an outfit and its compatible products may not share the same terms. This model groups similar terms that appear in a similar context into the same semantic cluster. It can map an outfit to its compatible products at the semantic level. For example, in Figure 1, the latent semantic model selects the candidate product 'Alexander Wang

*This work was done while the first author was doing internship at Walmart Labs.
[1]https://retail.emarketer.com/article/state-of-us-apparel-shopping-five-charts/5ae2106bebd4000b78fe1517

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331365

stretchy mesh ankle boots' as a relevant item. Although this product does not share any term with the outfit, it shares the same semantic meaning. However, conventional latent semantic models are not directly applicable to this task because an outfit has a variable number of items. We need a model that takes a variable-length input and is able to learn a compatibility relationship from the fashion outfit.

However, a latent semantic model has one drawback. The model tends to pay more attention to a high-frequency co-occurrence than a less common co-occurrence, which may turn out to be very important in our domain. For instance, when two products share the same rare term, it is possible that they share a unique aspect or functionality that most products do not have. From Figure 1, multiple products in the outfit have the term 'Zara.' which is a rare term. Without any prior knowledge, we could easily pick the first candidate item as the compatible item because it has the term 'Zara' in its description. Therefore, a lexical matching model such as a vector space model is suitable for capturing a rare pattern and could be used to complement a latent semantic model.

To overcome the aforementioned challenges, we propose a text-based neural compatibility ranking model that consists of two components: 1) a semantic matching, this component captures common fashion concepts by employing Gated Recurrent Units (GRU) to transform an outfit into a semantic vector while projecting fashion products to the same semantic space as the outfit; 2) a lexical matching, this component captures the rare fashion concepts by converting a fashion outfit and products into Term Frequency-Inverse Document Frequency (TFIDF) vectors and computes a cosine similarity between them. We summarize our contributions as follows:

- We propose a novel neural ranking architecture for fashion matching by capturing common fashion concepts via a semantic matching and rare fashion concepts via a lexical matching. A combination of these techniques is effective for learning a compatibility relationship as our model outperforms the competitive text-based retrieval models on a fashion matching task.
- We demonstrate that a product description contains rich information about product compatibility and has not been fully utilized by the prior works. Our work is able to exploit the text data and achieves the state-of-the-art results on this task by comparing to the competitive baselines.

2 RELATED WORK

We discuss recent works that are closely related to ours. Song et al. [2] use multiple autoencoders to learn a cloth matching function between tops and bottoms. Han et al. [1] model an outfit as a sequence of fashion products and train a bi-directional LSTM to predict the next fashion product. Li et al. [3] use like counts as a supervisory signal to train an RNN to classify an outfit's compatibility. These prior works exploit both product images and descriptions while our work exploits useful text information from product descriptions. Another approach is to learn a distance function between fashion items. McAuley et al. [4] learn a distance metric via low-rank embeddings while Veit et al. [5] further improve the performance by using a Siamese Network.

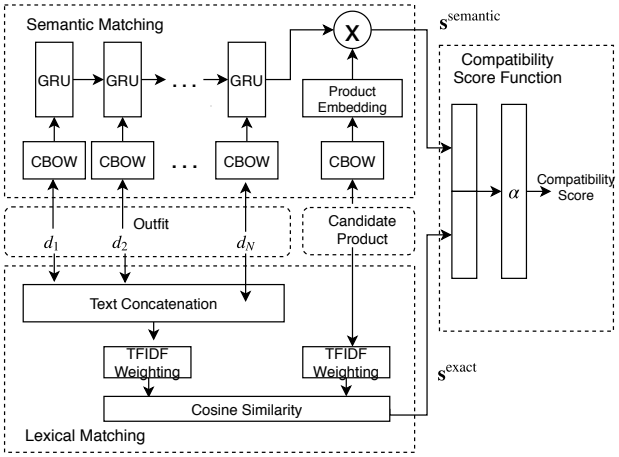

Figure 2: The model architecture of Neural Compatibility Ranking model (NCR). The top component is the semantic matching, and the bottom component is the lexical matching. The right component computes a compatibility score from the interaction vectors s^{semantic} and s^{exact} which are generated by semantic and lexical matching components. We denote d_i as a description of the i^{th} product in the outfit.

Our work is closely related to neural information retrieval models (NeuIR). Huang et al. [6] propose a latent semantic model using a deep learning architecture (DSSM). Xiong et al. [7] generalize a lexical matching model by proposing an end-to-end architecture to capture soft lexical matching. However, our model is different from these models because it is designed for a fashion matching task whose input query is an outfit that comprises multiple product descriptions. The recent tutorial by Mitra et al. [8] provides a comprehensive overview of NeuIR.

3 METHODOLOGY

3.1 Semantic Matching

This section describes the semantic matching component shown in Figure 2 for mapping an outfit and a candidate product into a latent semantic space. We describe each sub-component as follows:

3.1.1 Product Representation. We use a product description to represent a fashion product. We use a pre-trained GloVe word embedding[9] with 1.1 million unique word vectors of 300 dimensions[2] to convert each term in the description to a fixed length word vector. Then, we average all word vectors to obtain a product vector[3]. Although there are other sophisticated methods for a document embedding, we use CBOW for its simplicity and good performance. We do not fine-tune word embeddings because we want to separate the learning of word embedding and avoid overfitting.

3.1.2 Outfit latent vector. Once product vectors are obtained, each outfit becomes a set of product vectors. We randomly shuffle

[2]The pre-trained word embeddings are provided by Spacy library https://spacy.io/usage/vectors-similarity

[3]This is known as a continuous bag-of-words (CBOW)[10].

the order of the fashion products[4]. To handle a variable-length input, we use a Gated Recurrent Unit (GRU) [11] to aggregate the product vectors and use the last hidden state as an outfit semantic vector. We choose GRU because it is robust for a long input sequence while has fewer parameters than Long short-term memory unit (LSTM) [11].

3.1.3 Product latent vector. Since a fashion product has a different statistical property than an outfit, we use an additional non-linear function to map a product vector to the same semantic space as the outfit. We use a multi-layered feed forward neural network as a mapping function.

3.1.4 Interaction vector. The proposed model has two mapping functions: a GRU for an input outfit and a feed-forward neural network for an input product vector. We generate interaction vector $\boldsymbol{s}^{\text{semantic}}$ as a vector representation of a query outfit and a candidate product by computing a Hadamard product between outfit semantic vector and product semantic vector.

3.2 Lexical Matching

Rare terms are informative because they often correspond to brand, material, or designer. When multiple fashion products share the same rare term, it is likely that these products are strongly related. We found that a vector space model such as TFIDF is suitable for a lexical matching requirement. The weighting scheme of TFIDF places more importance on a rare term than a common term; hence, the products that share the same rare terms have a higher cosine similarity score. However, we cannot directly apply the vector space model because we need to convert an outfit into a single input query. Thus, we concatenate the terms from all product descriptions within the same outfit as the input text query while using the title of a candidate product as the text document. We transform both query and document to TFIDF vectors $\boldsymbol{o}^{\text{bow}}$ and $\boldsymbol{p}^{\text{bow}}$. We compute a cosine similarity as follows:

$$\boldsymbol{s}^{\text{exact}} = \text{cosine similarity}(\boldsymbol{o}^{\text{bow}}, \boldsymbol{p}^{\text{bow}}) = \frac{\boldsymbol{o}^{\text{bow}} \cdot \boldsymbol{p}^{\text{bow}}}{\|\boldsymbol{o}^{\text{bow}}\| \|\boldsymbol{p}^{\text{bow}}\|} \quad (1)$$

$\boldsymbol{s}^{\text{exact}}$ captures a lexical matching between outfit and product and is used as an additional feature when computing the compatibility score. We note that we can also replace the TFIDF model with other lexical matching models such as BM25 [12] or QL[13].

3.3 Compatibility Score Function

This component computes a compatibility score from the interaction vectors $\boldsymbol{s}^{\text{semantic}}$ and $\boldsymbol{s}^{\text{exact}}$. We concatenate both vectors and use a linear combination with a learnable weight vector $\boldsymbol{\alpha}$ to compute a compatibility score:

$$\text{compatibility score} = f(O, \boldsymbol{p}) = \sigma(\boldsymbol{\alpha}^T \left[\boldsymbol{s}^{\text{semantic}} \| \boldsymbol{s}^{\text{exact}}\right]) \quad (2)$$

The function f computes a compatibility score from outfit O and product $\boldsymbol{p}$. We denote $\|$ as a concatenation operation. A sigmoid function is applied to the final score to keep it within the range between 0 and 1. The main difference between our score function and a weight averaging is that we do not treat each semantic dimension equally. It is possible that some semantic dimensions are not useful and should be ignored. Moreover, due to the shared weights $\boldsymbol{\alpha}$, the interact vector $\boldsymbol{s}^{\text{exact}}$ influences the parameter learning in the semantic matching component which makes our model is different from an ensemble model. We refer the proposed model that combines both semantic and lexical matching as Neural Compatibility Ranking model (NCR). Figure 2 illustrates the architecture of NCR.

3.4 Parameter Learning

We train the proposed model as a point-wise ranking model. A training sample is a triplet of an outfit O, candidate product $\boldsymbol{p}$, and a relevant label r. The label is set to one when the candidate product is compatible with the outfit and is set to zero otherwise. We train the model on both positive and negative instances to minimize a binary cross entropy. A loss for a single training sample is:

$$\mathcal{L} = -r \cdot \log\left(f(O, \boldsymbol{p})\right) - (1 - r) \cdot \log\left(1 - f(O, \boldsymbol{p})\right) \quad (3)$$

4 EXPERIMENTAL SETTINGS

Due to the lack of negative labels, we evaluate the ranking performance on top-n recommendation tasks [14]. For each testing outfit, we randomly pick one product as a positive product and remove it from the outfit. Then, we sample 100 fashion products from other test outfits as negative products. The positive and negative products are ranked based on the compatibility scores to the testing outfit. The ranking position of a positive product determines the performance of the ranking model. We report the average of Mean Reciprocal Ranking (MRR) and HitRate (HR) [14] at 1, 5, and 10.

We use Polyvore fashion outfits dataset[5] for training and testing the proposed model. This dataset contains 21,889 fashion outfits in which 17,316 of them are used for training, 1,497 for validation, and 3,076 for testing. Each outfit is curated by a fashion blogger and comprises many fashion products that go well together. We remove the outfits with less than 3 products and the products whose descriptions have less than 3 words. There are no common fashion products between training and testing outfits.

We select two types of baseline methods for comparison, including lexical matching models and latent semantic models. The lexical matching models include: Query likelihood with Dirichlet smoothing (QL) [13] with μ is set to 5[6], BM25 [12], and Kernel-based neural ranking model (K-NRM)[7] [7] with the default settings. The latent semantic models include: DSSM[8] [6], SiameseNet [5], and BiLSTM+VSE [1]. Both SiameseNet and BiLSTM+VSE[9] are the prior works on a fashion matching task that utilize both product images and descriptions.

Our model uses the following setup: a CBOW vector of 300 dimensions as the input query. We use 1-layer GRU with a hidden dimension of 32 whose first hidden state is initialized with all zeros. A product embedding is a 3-layer feedforward network with 300, 128, and 32 neurons whose activation function is ReLU. The number

[4]We found that the order of the fashion products does not affect the performance of the model.

[5]https://github.com/xthan/polyvore-dataset

[6]The average length of a product title is around 5 words.

[7]https://github.com/AdeDZY/K-NRM

[8]We implement DSSM based on the original code:https://www.microsoft.com/en-us/research/project/dssm

[9]https://github.com/xthan/polyvore

of negative instances is 5 times more than the number of positive instances.

5 EXPERIMENTAL RESULTS

5.1 Baseline Comparison

Model	Data	MRR	HR@1	HR@5	HR@10
QL [13]	T	0.1762	0.1191	0.2073	0.2610
BM25 [12]	T	0.1649	0.1135	0.2032	0.2594
DSSM [6]	T	0.1743	0.0846	0.2285	0.3378
K-NRM [7]	T	0.1707	0.1150	0.1975	0.2584
SiameseNet [5]	T+I	0.2559	0.1320	**0.3667**	0.5167
BiLSTM+VSE [1]	T+I	0.2256	0.0975	0.3424	**0.5224**
NCR	T	**0.2678**†	**0.1547**†‡	0.3626†	0.5137†

Table 1: Ranking performances of our model and baseline methods. The bold font denotes the best result in that evaluation metric. Letter T and I stand for Text and Image respectively. †, ‡ indicate statistically significant improvements over the description-only baselines and prior works, respectively. The statistically significance is based on the paired t-test with p-value < 0.01.

Table 1 shows the results of different methods. We have several observations from the results. First, our model significantly outperforms the baselines that use only product descriptions. These baselines (QL, BM25, DSSM, and K-NRM) represent an outfit by concatenating all product descriptions and do not consider the complex product relationship. Secondly, the results demonstrate that the combination of latent semantic and lexical matching is effective and achieves comparable results with the baselines that utilize both image and text data.

Secondly, the lexical matching models, including QL, BM25, and K-NRM are unable to find a relevant product when there is no matching term, as demonstrated by their lower HR@5 and HR@10 when compared with the latent semantic models. Although K-NRM could get around the vocabulary gap problem by adding soft matching signals, the result shows that adding the soft matching degrades the ranking performance as K-NRM performs slightly worse than QL and BM25 do.

Lastly, the latent semantic models have the worst performance on HR@1, except the SiameseNet model. One explanation is that DSSM and BiLSTM+VSE use only one mapping function to project outfit and a candidate product into a semantic space. This choice of architectures could be inflexible because the mapping function needs to capture the statistical property of both outfit and products. Furthermore, due to the lack of lexical matching, it is possible that DSSM and BiLSTM+VSE cannot distinguish the difference between the top products because they are all semantically similar.

5.2 Model Components Analysis

We investigate the performance of each component in our model. As demonstrated in Table 2, the semantic matching component has the worst HR@1 while the lexical matching component has the worst HR@10. When we combine these two components as part of NCR's architecture, the MRR and HitRate performances are improved significantly. The results demonstrate that our model effectively leverages both semantic and lexical matching signals and achieves the best results.

Model	MRR	HR@1	HR@5	HR@10
Semantic Matching	0.1708	0.0603	0.2584	0.4296
Lexical Matching	0.1794	0.1212	0.2104	0.2610
NCR	**0.2678**	**0.1547**	**0.3626**	**0.5137**

Table 2: Ranking performances of all components in the proposed models. The bold font denotes the best result in that evaluation metric.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a neural compatibility ranking model for fashion matching. The model takes the advantages of semantic matching and lexical matching as it significantly outperforms the text-based ad-hoc retrieval models while achieves comparable results with the baselines that use both product images and descriptions. A combination of semantic and lexical matching is a crucial component for extracting product compatibility information from text data. In the future work, we plan to extend the proposed model by utilizing both text and image data.

ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers and Travis Ebesu for valuable comments. The work of Yi Fang is partially supported by the National Natural Science Foundation of China (41771426).

REFERENCES

[1] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. Learning fashion compatibility with bidirectional lstms. In *ACM on Multimedia Conference*, 2017.
[2] Xuemeng Song, Fuli Feng, Jinhuan Liu, Zekun Li, Liqiang Nie, and Jun Ma. Neurostylist: Neural compatibility modeling for clothing matching. In *ACM on Multimedia Conference*, 2017.
[3] Yuncheng Li, Liangliang Cao, Jiang Zhu, and Jiebo Luo. Mining fashion outfit composition using an end-to-end deep learning approach on set data. *IEEE Transactions on Multimedia*, 2017.
[4] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015.
[5] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *ICCV*, 2015.
[6] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, 2013.
[7] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *SIGIR*, 2017.
[8] Bhaskar Mitra, Nick Craswell, et al. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 2018.
[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.
[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *ICLR Workshop*, 2013.
[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. 2016.
[12] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.
[13] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *SIGIR*, 2001.
[14] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, 2010.