The AAAI-17 Workshop on
Crowdsourcing, Deep Learning, and Artificial Intelligence Agents
WS-17-07

Deep Style Match for Complementary Recommendation

Kui Zhao†, Xia Hu‡, Jiajun Bu†, Can Wang†
†College of Computer Science, Zhejiang University
Hangzhou, China
{zhaokui, bjj, wcan}@zju.edu.cn
‡Hangzhou Science & Technology Information Research Institute
Hangzhou, China
hx@hznet.com.cn

Abstract

Humans develop a common sense of style compatibility between items based on their attributes. We seek to automatically answer questions like "Does this shirt go well with that pair of jeans?" In order to answer these kinds of questions, we attempt to model human sense of style compatibility in this paper. The basic assumption of our approach is that most of the important attributes for a product in an online store are included in its title description. Therefore it is feasible to learn style compatibility from these descriptions. We design a Siamese Convolutional Neural Network architecture and feed it with title pairs of items, which are either compatible or incompatible. Those pairs will be mapped from the original space of symbolic words into some embedded style space. Our approach takes only words as the input with few preprocessing and there is no laborious and expensive feature engineering.

Introduction

We have a common sense of style compatibility between items and can naturally answer questions like "Does this shirt go well with that pair of jeans?" These kind of style compatibility information can be exploited in many commercial applications, such as recommending items to users based on what they have already bought; or generating the whole purchase outfits (see Figure 1 for an example of clothes) to users querying certain items, if sufficient compatibility relationships between items are provided.

To identify these compatibility relationships, existing methods such as frequent itemset mining (Han, Pei, and Yin 2000) attempt to generate match items automatically by analyzing historical purchasing patterns. However, frequent itemset mining relies on historical purchasing records to find items frequently purchased together and new items will inevitably suffer from the "cold start" problem (Schein et al. 2002).

Recently, McAuley et al. (McAuley et al. 2015) and Veit et al. (Veit et al. 2015) intend to discover the style match relationships between items using visual information presented in the images of items. However, besides being computationally expensive, image-based matching methods are frequently plagued by the plentiful contents presented in images. For instance, Figure 2 shows a part of the image for leggings from Taobao, which is the largest e-commerce platform in China. Besides leggings, the image also contains a coat, a pair of shoes and a very complex background etc. These contents will confuse the learning machines if only leggings are expected.

Figure 1: An example for clothing outfits: on the left are query items and on the right are corresponding complementary items.

Figure 2: A typical image from Taobao, it actually is an image to show leggings.

To overcome the limitations of existing methods, we propose in this paper a novel style match approach using the title descriptions of items in online stores. The basic assumption of our work is that online sellers will place most of the important attributes of a product in its title description, so that the product can be easily found by a keyword-based query. Therefore, the title description is a highly condensed collection of attribute descriptions for a product. So it is fea-

sible to model compatibility between two items better if we are capable of mapping the title pairs from the original space of symbolic words into some embedded style space.

We here design a Siamese Convolutional Neural Network architecture for matching title sentences. It will map two title sentences from an item pair into low-dimensional vectors respectively in parallel, which are then used to learn the compatibility between these two items in the style space. When designing the amalgamation part of Siamese CNN for computing compatibility, we have considered the ability of our model to be extended to big data scenarios, which is critical for real-world recommendation applications. We test our approach on two large datasets: a Chinese dataset from Taobao provided by Alibaba Group and an English dataset from Amazon provided by (McAuley et al. 2015). Our approach demonstrates strong performance on both datasets, which indicates its ability of learning human sense of style compatibility between items.

Related Work

Finding complementary items has been studied for a long time. The early works can be traced back to frequent itemset mining (Han, Pei, and Yin 2000), which generates match items automatically by analyzing history purchasing patterns. Frequent itemsets such as "beer and diaper" sometimes have nothing to do with compatibility. What's more, they are challenged by the "cold-start" problem, which means new products with no historical records are invisible to the algorithm (Schein et al. 2002).

Many approaches such as content-based recommendation or social recommendation are proposed to address this problem (see (Pazzani and Billsus 2007) for a survey). Closely related to our work are (McAuley et al. 2015) and (Veit et al. 2015), in which McAuley et al. and Veit et al. attempt to learn clothing style similarity based on their appearance in images. However, our work differs from (McAuley et al. 2015) and (Veit et al. 2015) in the following two aspects: (1) we use title descriptions, which contain rich attribute information instead of item images; (2) the objective of our method is to find matching items from their attribute description instead of learning visual similarities from item images.

Problem formulation

We here describe the problem in a formal way: given a query item set $Q = \{q_1, \cdots, q_m\}$ and a candidate item set $C = \{c_1, \cdots, c_n\}$, where each query item $q_i \in Q$ comes together with the compatibility judgements $\{y_{i_1}, \cdots, y_{i_n}\}$. The complementary item $c_j \in C$ is labeled with $y_{i_j} = 1$ and $y_{i_j} = 0$ otherwise. Our goal is to build a model to compute the compatibility probability between q_i and c_j:

$$P(y = 1|q_i, c_j) = f(\phi(q_i, \theta_1), \phi(c_j, \theta_1), \theta_2), \qquad (1)$$

where function $\phi(\cdot)$ is the sentence model mapping a title sentence into a low-dimensional representation vector and function $f(\cdot)$ computes the compatibility probability between two items in the style space. The parameter vectors θ_1 and θ_2 are learned in the training process.

Style Match

The main building block of our approach is a sentence model based on CNN. This sentence model will map two title sentences from an item pair into low-dimensional vectors respectively in parallel, which are then used to learn the compatibility between two items in the style space.

Sentence model

We model sentences with function $\phi(\cdot)$, which is a convolutional architecture as shown in Figure 3.

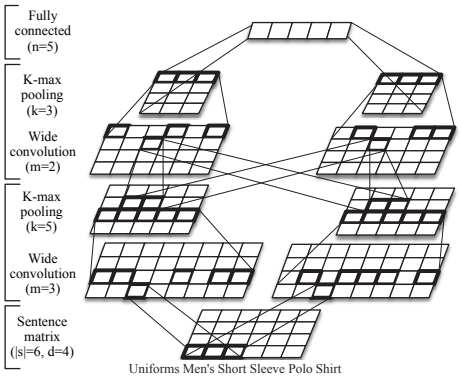

Figure 3: The architecture of our sentence model.

In the following, we give a brief explanation of the main components in our Convolutional Neural Network.

Sentence matrix. Our sentence model takes a sentence $\mathbf{s}$ as the input, where it is treated as a sequence of raw words: $[s_1, \cdots, s_{|\mathbf{s}|}]$ and each word s_i is from a vocabulary V.

Firstly, each word s_i is represented by a distributional representation vector $\mathbf{w}_i \in \mathbb{R}^d$, looked up from the word-level embedding matrix $\mathbf{W} \in \mathbb{R}^{d \times |V|}$. Then a sentence matrix $\mathbf{S} \in \mathbb{R}^{d \times |\mathbf{s}|}$ can be built for the input sentence $\mathbf{s}$:

$$\mathbf{S} = \begin{bmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_{|\mathbf{s}|} \\ | & & | \end{bmatrix}, \qquad (2)$$

where the i-th column is the distributional representation vector $\mathbf{w}_i$ for the i-th word in $\mathbf{s}$. The values in the embedding matrix $\mathbf{W}$ are parameters initialized with an unsupervised neural language model (Mikolov et al. 2013) and sequentially optimized during training. The embedding dimension d is a hyper-parameter of the model.

As shown in Figure 3, for the sentence $\mathbf{s}$=[Uniforms, Men's, Short, Sleeve, Polo, Shirt], when the embedding dimension d is 4, the sentence matrix is a matrix in $\mathbb{R}^{4 \times 6}$.

Convolutional feature maps. Convolution can be seen as a special kind of linear operation and aimed to extract local patterns. We use the *one-dimensional convolution* to recognize discriminative word sequences from the input sentences. The one-dimensional convolution is an operation between two vectors $\mathbf{f} \in \mathbb{R}^m$ and $\mathbf{s} \in \mathbb{R}^{|\mathbf{s}|}$. The vector $\mathbf{f}$ is called as a *filter* of size m and the vector $\mathbf{s}$ is a *sequence* of size $|\mathbf{s}|$. The specific operation is to take the dot product of

the vector $\mathbf{f}$ with each m-gram sliding along the sequence $\mathbf{s}$ and obtain a new sequence $\mathbf{c}$ where:

$$\mathbf{c}_j = \mathbf{f}^{\mathrm{T}}\mathbf{s}_{j-m+1:j}. \quad (3)$$

In practice, we usually add a bias b to the dot product result:

$$\mathbf{c}_j = \mathbf{f}^{\mathrm{T}}\mathbf{s}_{j-m+1:j} + b. \quad (4)$$

There are two types of convolution depending on the allowed range of index j: *narrow* and *wide*. The narrow type restricts j in the range $[m, |\mathbf{s}|]$ and the wide type restricts j in the range $[1, |\mathbf{s}| + m - 1]$. The benefits of wide type over the narrow type in text processing are discussed in detail in (Blunsom et al. 2014). Briefly speaking, unlike the narrow convolution where words close to margins are seen fewer times, wide convolution gives equal attention to all words in the sentence and so is better at handling words at margins. More importantly, a wide convolution always produces a valid non-empty result $\mathbf{c}$ even when $|\mathbf{s}| < m$. For these reasons, we use wide convolution in our model.

The sentence matrix $\mathbf{S}$ is not just a sequence of single values but a sequence of vectors, where the dimension of each vector is d. So when we apply the one-dimensional convolution on the sentence matrix $\mathbf{S}$, we need a filter bank $\mathbf{F} \in \mathbb{R}^{d\times m}$ consisting of d filters of size m and a bias bank $\mathbf{B} \in \mathbb{R}^d$ consisting of d baises. Each row of $\mathbf{S}$ is convoluted with the corresponding row of $\mathbf{F}$ and then the corresponding row of $\mathbf{B}$ is added to the convolution result. After that, we obtain a matrix $\mathbf{C} \in \mathbb{R}^{d\times(|\mathbf{s}|+m-1)}$:

$$\mathrm{conv}(\mathbf{S}, \mathbf{F}, \mathbf{B}) : \mathbb{R}^{d\times|\mathbf{s}|} \to \mathbb{R}^{d\times(|\mathbf{s}|+m-1)}. \quad (5)$$

The values in filter bank $\mathbf{F}$ and bias bank $\mathbf{B}$ are parameters optimized during training. The filter size m is a hyper-parameter of the model.

In Figure 3, after applying a 4×3 filter bank and a bias bank of size 4 on the 4×6 sentence matrix, we obtain an intermediate matrix of size 4×8.

Activation function. To make the network capable of learning non-linear functions, a non-linear activation $\alpha(\cdot)$ need to be applied in an element-wise way to the output of the preceding layer and a matrix $\mathbf{A} \in \mathbb{R}^{d\times(|\mathbf{s}|+m-1)}$ is then obtained:

$$\alpha(\mathbf{C}) : \mathbb{R}^{d\times(|\mathbf{s}|+m-1)} \to \mathbb{R}^{d\times(|\mathbf{s}|+m-1)}. \quad (6)$$

Popular choices of $\alpha(\cdot)$ include: *sigmod*, *tanh* and *relu* (rectified linear defined as $\max(0, x)$). In practice, our experimental results are not very sensitive to the choice of activation, so we choose *relu* due to its simplicity and computing efficiency. In addition, we can see the bias b in (4) plays the role of setting an appropriate threshold for controlling units to be activated.

Pooling. Pooling layer will aggregate the information in the output of preceding layer. This operation aims to make the representation more robust and invariant to small translations in the input. More importantly, pooling helps to handle inputs with varying size, e.g. processing sentences with uncertain length.

For a given vector $\mathbf{a} \in \mathbb{R}^{|\mathbf{a}|}$, traditional pooling aggregates it into a single value:

$$\mathrm{pooling}(\mathbf{a}) : \mathbb{R}^{|\mathbf{a}|} \to \mathbb{R}. \quad (7)$$

The way of aggregating the information defines two types of pooling operations: *average* and *max*. Max pooling is used more widely in practice. Recently, max pooling has been generalized to *k-max pooling* (Blunsom et al. 2014), in which k max values are selected from the vector $\mathbf{a}$ and arranged in their original order:

$$\text{k-pooling}(\mathbf{a}) : \mathbb{R}^{|\mathbf{a}|} \to \mathbb{R}^k, \quad (8)$$

where k is a hyper-parameter of the model.

When we apply k-max pooling on the matrix $\mathbf{A}$, each row of $\mathbf{A}$ is pooled respectively and we obtain a matrix $\mathbf{P} \in \mathbb{R}^{d\times k}$:

$$\text{k-pooling}(\mathbf{A}) : \mathbb{R}^{d\times|\mathbf{a}|} \to \mathbb{R}^{d\times k}. \quad (9)$$

In Figure 3, after applying k-max pooling (with $k = 5$) on the intermediate matrix of size 4×8, we obtain a new intermediate matrix of size 4×5.

Multiple feature maps. After a group of above operations, we obtain the first order *representation* learning to recognize the specific m-grams in the input sentence. To obtain higher order representations, we can use a deeper network by repeating these operations. The higher order representations can capture patterns of the sentence in much longer range.

Meanwhile, we can also extend network to learning multi-aspect representations. Let $\mathbf{P}^i$ denote the i-th order representation. We can compute K_i representations $\mathbf{P}^i_1, \cdots, \mathbf{P}^i_{K_i}$ in parallel at the same i-th order. Each representation $\mathbf{P}^i_j$ is computed by two steps. First, we compute convolution on each representation $\mathbf{P}^{i-1}_k$ at the lower order $i-1$ with the distinct filter bank $\mathbf{F}^i_{j,k}$ and bias bank $\mathbf{B}^i_{j,k}$ and then sum up the results. Second, non-linear activation and k-max pooling are applied to the summation result:

$$\mathbf{P}^i_j = \text{k-pooling}(\alpha(\sum_{k=1}^{K_{i-1}} \mathrm{conv}(\mathbf{P}^{i-1}_k, \mathbf{F}^i_{j,k}, \mathbf{B}^i_{j,k}))). \quad (10)$$

In Figure 3, there are two representations at the first order and two representations at the second order: $\mathbf{P}^1_1 \in \mathbb{R}^{4\times5}$, $\mathbf{P}^1_2 \in \mathbb{R}^{4\times5}$ and $\mathbf{P}^2_1 \in \mathbb{R}^{4\times3}$, $\mathbf{P}^2_2 \in \mathbb{R}^{4\times3}$.

Full connection. Full connection is a linear operation to combine all representations at the highest order into a single vector. More specifically, for the highest order representations $\mathbf{P}^h_1, \cdots, \mathbf{P}^h_{K_h}$ (assume $\mathbf{P}^h_k \in \mathbb{R}^{d\times l}$), we first flat them into a vector $\mathbf{p} \in \mathbb{R}^{K_h\times d\times l}$. Then we transform it with a dense matrix $\mathbf{H} \in \mathbb{R}^{(K_h\times d\times l)\times n}$:

$$\mathbf{x} = \mathbf{p}^{\mathrm{T}}\mathbf{H}, \quad (11)$$

where $\mathbf{x} \in \mathbb{R}^n$ is the final representation vector. The values in matrix $\mathbf{H}$ are parameters optimized during training. The representation size n is a hyper-parameter of the model.

In Figure 3, we finally represent the input sentence with a vector of size $n = 5$.

Matching items

We compute the compatibility probability between two items with function $f(\cdot)$, which is a Siamese Convolutional Neural Network as shown in Figure 4.

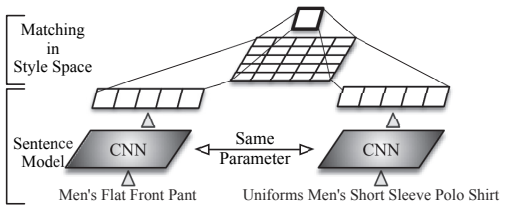

Figure 4: The whole architecture of our model.

Siamese setup is introduced by Hadsell et al. (Hadsell, Chopra, and LeCun 2006) and used widely in learning distance metrics. When designing the amalgamation part of our model, we have considered its scalability for big data scenarios, which is critical for real-world applications.

Style space. For two given items q and c, after generating the representation vectors $\mathbf{x}_q \in \mathbb{R}^n$ and $\mathbf{x}_c \in \mathbb{R}^n$ of their title sentences respectively, we compute the compatibility probability between them as follow:

$$\begin{aligned} P(y=1|q,c) &= \sigma(\mathbf{x}_q^{\mathrm{T}}\mathbf{M}\mathbf{x}_c + b) \\ &= \frac{1}{1+e^{-(\mathbf{x}_q^{\mathrm{T}}\mathbf{M}\mathbf{x}_c+b)}}, \end{aligned} \tag{12}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ is a matrix and b is a scalar. We call $\mathbf{M}$ as the *compatibility matrix* and the space spanned by $\mathbf{M}$ as the *style space*. After transformation $\mathbf{x}'_q = \mathbf{x}_q^{\mathrm{T}}\mathbf{M}$, $\mathbf{x}'_q$ represents the item which is most style compatible to q. We seek items whose representations are close to $\mathbf{x}'_\mathbf{q}$ under linear kernel distance. The values in compatibility matrix $\mathbf{M}$ and the bias b are parameters optimized during the training.

On the other hand, $\mathbf{x}_q^{\mathrm{T}}\mathbf{M}\mathbf{x}_c$ in (12) can be viewed as a noisy-channel model, which has been widely used in the information retrieval and QA system (Echihabi and Marcu 2003) (Bordes, Weston, and Usunier 2014).

Recommendation

In recommendation applications, we are usually given a query item set $Q = \{q_1, q_2, \cdots, q_m\}$ and a candidate item set $C = \{c_1, c_2, \cdots, c_n\}$, where the query item set is relatively small and the candidate item set is usually very large. For each query item q_i, we intend to query its K most complementary items from the candidate set C and rank them from high compatibility to low compatibility. When the item candidate set is very large, it is inefficient and even unacceptable to compute the compatibility for all item pairs (q_i, c_j) and then sort them.

Our approach can be easily extended to handle these big data scenarios. Given two items q and c, we first generate their representation vectors $\mathbf{x}_q$ and $\mathbf{x}_c$ respectively. Then their compatibility probability is computed according to (12). We notice that the function $\sigma(\cdot)$ in (12) is a monotonic increasing function and b is a learned constant. Thus for a query item q and two candidate items c_1, c_2, we have:

$$\begin{aligned} &P(y=1|q,c_1) \le P(y=1|q,c_2) \\ &\Leftrightarrow \mathbf{x}_q^{\mathrm{T}}\mathbf{M}\mathbf{x}_{c_1} \le \mathbf{x}_q^{\mathrm{T}}\mathbf{M}\mathbf{x}_{c_2}. \end{aligned} \tag{13}$$

Based on this property, we transform the original problem of querying the K most complementary items of q from the item candidate set C into another problem, namely searching K *nearest neighbors* of $\mathbf{x}'$ ($\mathbf{x}'_q = \mathbf{x}_q^{\mathrm{T}}\mathbf{M}$) from $\{\mathbf{x}_{c_1}, \cdots, \mathbf{x}_{c_n}\}$ under the linear kernel distance. It is well known as Maximum Inner Product Search (MIPS). There are many methods solving MIPS efficiently on the large scale data, such as tree techniques (Ram and Gray 2012) and hashing techniques (Shrivastava and Li 2014) (Shen et al. 2015) etc.

Training

We train the model to maximize the likelihood of a observed relationship training set $\mathcal{R}$, where $r_{ij} \in \mathcal{R}$:

$$r_{ij} = \begin{cases} 1 & \text{, if items } i \text{ and } j \text{ are compatible;} \\ 0 & \text{, otherwise.} \end{cases} \tag{14}$$

Maximizing the likelihood is equal to minimizing the binary-cross entropy loss function:

$$L = -\sum_{r_{ij}\in\mathcal{R}} \left[r_{ij}\log(p) + (1-r_{ij})\log(1-p) \right], \tag{15}$$

where $p = P(y=1|i,j)$.

The parameters to be optimized in our network are θ_1, θ_2, which have been mentioned above:

$$\theta_1 = \{\mathbf{W}, \mathbf{F}, \mathbf{B}, \mathbf{H}\} \text{ and } \theta_2 = \{\mathbf{M}, b\}, \tag{16}$$

namely the word embeddings matrix $\mathbf{W}$, filter bank $\mathbf{F}$, bias bank $\mathbf{B}$, dense matrix $\mathbf{H}$, compatibility matrix $\mathbf{M}$ and compatibility bias b. Note that there are multiple filter banks and bias banks to be learned.

In the following sections, we present several crucial details for training our deep learning model.

Regularization

To alleviate the overfitting issue, we use a popular and efficient regularization technique named *dropout* (Srivastava et al. 2014). Dropout is applied to the flatted vector $\mathbf{p}$ (presented in (11)) before transforming it with the dense matrix $\mathbf{H}$. A portion of units in $\mathbf{p}$ are randomly dropped out by setting them to zero during the forward phase, which is helpful for preventing the feature co-adaptation. The dropout rate is a hyper-parameters of the model.

Hyper-parameters

The hyper-parameters in our deep learning model are set as follows: the embedding dimension is $d = 100$; the size of filters at the first order representation is $m = 3$; the number of max values selected by k-max pooling at the first order representation is $k = 5$; the size of filters at the second order representation is $m = 2$; the number of max values selected by k-max pooling at the second order representation is

$k = 3$; the dimension of the vector used to represent the sentence is $n = 100$; the dropout rate is $p = 0.2$. What's more, there are $K_1 = 100$ representations computed in parallel at the first order representation and $K_2 = 100$ representations computed in parallel at the second order representation.

Optimization

To optimize our network, we use the Stochastic Gradient Descent (SGD) algorithm with shuffled mini-batches. The parameters are updated through the back propagation framework with Adagrad rule (Duchi, Hazan, and Singer 2011). The batch size is set to 256 and the network is trained for 20 epochs. The training progress will be early stopped if there is no more update to the best loss on the validation set for the last 5 epochs.

We train our network on a GPU for speeding up. A Python implementation using Keras[1] powered by Theano (Bastien et al. 2012) can process 428k text pairs per minute on a single NVIDIA K2200 GPU.

Experiments

We evaluate our method on two large datasets: a Chinese dataset from Taobao and an English dataset from Amazon.

Datasets

Taobao. This dataset is collected from Taobao.com and provide by Alibaba Group[2]. It includes a Clothing category and there are about 406k compatibility relationships covering 61k items. The compatibility relationships in this dataset are labelled manually by clothes collocation experts.

Amazon. This dataset is collected from Amazon.com and provided by (McAuley et al. 2015). Though it includes multiple categories, in order to investigate the performance of our approach on both datasets, we mainly focus on the Clothing category. In this category, there are about 12 million compatibility relationships covering 662k items. Unlike the Taobao dataset, the compatibility relationships in Amazon dataset are not labelled manually. They are the co-purchase data from Amazon's recommendations (Linden, Smith, and York 2003).

Setup

Our goal is to differentiate compatibility relationships from non-compatibility ones. We consider all positive relationships (compatibility) and generate random non-relationship distractors of the equal size. That is to say the ratio between positive and negative samples in the dataset is 50:50. Then we separate the whole dataset into training, validation and testing sets according to the ratios 80:10:10. Although we do not expect overfitting to be a serious issue in our experiment with the large training set, we still carefully tune our model on the validation set to avoid overfitting on testing set. We compare our approach against baselines from two aspects: visual one and non-visual ones.

[1]http://keras.io

[2]http://tianchi.aliyun.com/datalab/index.htm

Visual baseline. We take the method in (Veit et al. 2015) as the visual comparison since it is also in the end-to-end fashion. In particular, we consider the specific setting configured with GoogLeNet and naive sampling for two considerations. First, in all situations of their experiments, GoogLeNet (Szegedy et al. 2015) outperforms AlexNet (Krizhevsky, Sutskever, and Hinton 2012). Second, naive sampling means sampling randomly from the dataset, which is consistent with the setup in our experiments. We experiment their method on the Taobao dataset and take the results on the Amazon dataset directly from (Veit et al. 2015).

Non-visual baselines. We take three methods as the non-visual comparison:

1) *Naive Bayes on Bag of Words (NBBW)*. We treat the title sentences from an item pair as the bag-of-words and feed Naive Bayes classifier with it as the feature vector;

2) *Random Forest on Bag of Words (RFBW)*. Random Forest is capable of modeling extremely complex classification surface. We apply Random Forest classifier on the bag-of-words representation of the title sentences from an item pair;

3) *Random Forest on Topic Model (RFTM)*. For the given item pair $\{q, c\}$, we first generate the topic representations $\mathbf{x}_q, \mathbf{x}_c$ of items q, c by LDA model (Blei, Ng, and Jordan 2003) respectively, where the topic number is set as 100. Then we concatenate them into a single feature vector $\mathbf{x}_{q,c}$ and process Random Forest classifier on it.

The implementation of Naive Bayes and Random Forest is taken from scikit-learn (Pedregosa et al. 2011). We turned their parameters to obtaine the best loss on the validation set.

There is no preprocess on images and texts. All results reported in the following section is on the testing set.

Results

Comparison to baselines. Tabel 1 shows the corresponding areas under the ROC curves of compatibility prediction on the testing set . The results show clearly that our approach outperforms all other baselines.

Methods	Taobao	Amazon
Visual	0.579	0.770
NBBW	0.712	0.820
RFBW	0.807	0.931
RFTM	0.796	0.893
Ours	**0.891**	**0.983**

Table 1: AUC scores for all methods.

The visual method collapses on the Taobao dataset because unlike Amazon, Taobao is a Consumer to Consumer (C2C) platform and has few strict requirement about quality of item images uploaded by users. A majority of images are like Figure 2, where the information is mixed up and confusing to learning machines. In contrast, the title description is a highly condensed collection of more attributes besides appearances with few noises. When using title descriptions, a simple method like Naive Bayes on Bag of Words can achieve an acceptable performance and a more sophisticated method like Random Forest on Bag of Words can generate

Category	AUC	Category	AUC	Category	AUC
Automotive	0.922	Electronics	0.948	Patio Lawn & Garden	0.966
Baby	0.917	Grocery & Gourmet Food	0.959	Pet Supplies	0.972
Beauty	0.935	Health & Personal Care	0.929	Sports & Outdoors	0.912
Books	0.897	Home & Kitchen	0.949	Tools & Home Improvement	0.952
CDs & Vinyl	0.815	Movies & TV	0.878	Toys & Games	0.985
Cell Phones & Accessories	0.969	Musical Instruments	0.983	Video Games	0.890
Digital Music	0.818	Office Products	0.974		

Table 2: AUC scores for compatibility prediction on twenty top-level categories from Amazon dataset.

competitive results. However, using topic models on title descriptions is not a good idea since most title descriptions are short texts. One important reason why our approach achieves better performance is that our approach can recognize specific m-grams and more complicated patterns not captured by bag-of-words models. For instance, the complementary styles of the item titled with "white shirt with blue stripes" and "blue shirt with white stripes" are very different, but they have the same bag-of-words representation.

The performance upper bound of our approach on Taobao dataset is limited by the segmentation quality of Chinese. This is one of the reasons that all AUC scores on the Taobao dataset are lower than that on the Amazon dataset.

Tuning sentence model. There are several crucial setups in the sentence model: 1) the word embedding dimension d; 2) the sentence representation dimension n; 3) whether the word embedding matrix $\mathbf{W}$ is initialized or not.

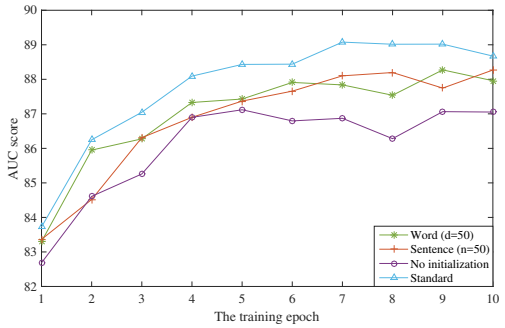

Figure 5: Convergence processes of our approach with sentence models under different setups.

We show the first ten epochs of training processes of our model on the Taobao dataset with sentence models under different setups in Figure 5, where the standard setup means that we set $d = 100, n = 100$ and initialize the word embedding matrix with an unsupervised neural language model (Mikolov et al. 2013). We can see that the setup with larger d or n claims better performance. Furthermore, the performance can get better when we continue to increase the value of d or n. In practice, there is a tradeoff between the performance and resources requirement according to specific situations. What's more, initializing the word embedding matrix $\mathbf{W}$ with an unsupervised neural language model is indeed benefit to the convergence rate and the final performance.

Discussion

Toward general match. While the previous section mainly focuses on clothes matching, we also train classifiers on the other twenty top-level categories from the Amazon dataset and present the results in Table 2. As can be seen, we obtain good accuracy in predicting compatibility relationships in a variety of categories. What's more, we have also tried to train a single model to predict compatibility relationships for all categories. There appears to be no "silver bullet" and the result is dissatisfactory: the AUC score of that single model is only 0.694.

The comparison across categories is particularly interesting. Our approach performs relatively poor on the categories "CDs & Vinyl" and "Digital Music" since the content of music is too rich to be described very clearly in a short title description. In contrast, the title description is long enough to describe an item from the category 'Musical Instruments' clearly and thus our approach performs very well on that. In a word, the better titles can describe the attributes of items in a category, the higher performance can be achieved on that category by our approach.

Conclusions

In this paper, we present a novel approach to model the human sense of style compatibility between items. The basic assumption of our approach is that most of the important attributes for a product in an online store are included in its title description. We design a Siamese Convolutional Neural Network architecture to map the title descriptions of an item pair from the original space of symbolic words into some embedded style space. The compatibility probability between items can be then computed in the style space. Our approach takes only words as the input with few preprocessing and requires no laborious and expensive feature engineering. Moreover, it can be easily extended to big data scenarios with KNN searching techniques. The experiments on two large datasets confirm our assumption and show the possibility of modeling the human sense of style compatibility.

There are several interesting problems to be investigated in our future work: (1) we would like to use more sophisticated sentence models without injuring the simplicity of our approach; (2) we are wondering whether it is possible to use the text and image information simultaneously, e.g. hybrid model or mapping the texts and images of items into the same embedded space for mutual retrieval and matching.

Acknowledgments

We would like to thank Alibaba Group and Julian McAuley for providing the valuable datasets. This work is supported by Zhejiang Provincial Natural Science Foundation of China (Grant no. LZ13F020001), Zhejiang Provincial Soft Science Project (Grant no. 2015C25053), National Science Foundation of China (Grant nos. 61173185, 61173186).

References

Bastien, F.; Lamblin, P.; Pascanu, R.; Bergstra, J.; Goodfellow, I. J.; Bergeron, A.; Bouchard, N.; and Bengio, Y. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3:993–1022.

Blunsom, P.; Grefenstette, E.; Kalchbrenner, N.; et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

Bordes, A.; Weston, J.; and Usunier, N. 2014. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases*. Springer. 165–180.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12:2121–2159.

Echihabi, A., and Marcu, D. 2003. A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, 16–23. Association for Computational Linguistics.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, 1735–1742. IEEE.

Han, J.; Pei, J.; and Yin, Y. 2000. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, 1–12. ACM.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Linden, G.; Smith, B.; and York, J. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE* 7(1):76–80.

McAuley, J.; Targett, C.; Shi, Q.; and van den Hengel, A. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 43–52. ACM.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Pazzani, M. J., and Billsus, D. 2007. Content-based recommendation systems. In *The adaptive web*. Springer. 325–341.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Ram, P., and Gray, A. G. 2012. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 931–939. ACM.

Schein, A. I.; Popescul, A.; Ungar, L. H.; and Pennock, D. M. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 253–260. ACM.

Shen, F.; Liu, W.; Zhang, S.; Yang, Y.; and Tao Shen, H. 2015. Learning binary codes for maximum inner product search. In *Proceedings of the IEEE International Conference on Computer Vision*, 4148–4156.

Shrivastava, A., and Li, P. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Advances in Neural Information Processing Systems*, 2321–2329.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

Veit, A.; Kovacs, B.; Bell, S.; McAuley, J.; Bala, K.; and Belongie, S. 2015. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *Proceedings of the IEEE International Conference on Computer Vision*, 4642–4650.