Textual-Visual GCNs with Cosine Loss for Multimodalities Recall

Jie Wu
wu.jie@westone.com.cn
Westone Information Industry INC.
Chengdu, Sichuan, China

Lei Zhu
zhu.lei@westone.com.cn
Westone Information Industry INC.
Chengdu, Sichuan, China

Ying Peng
peng.ying06059@westone.com.cn
Westone Information Industry INC.
Chengdu, Sichuan, China

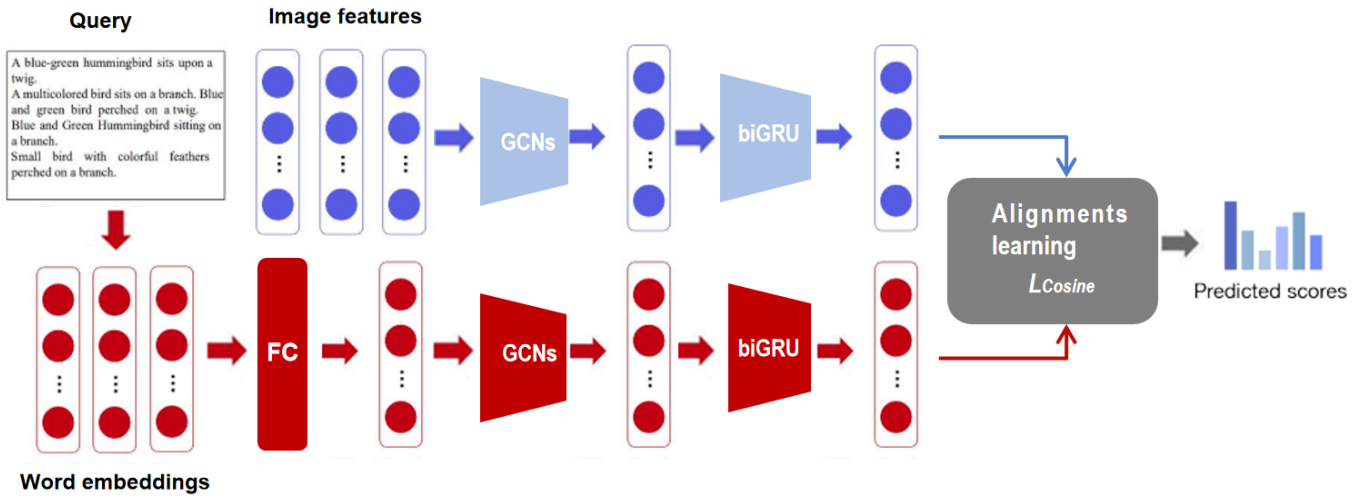

Figure 1: Network architecture. The model comprises an image encoder (in blue) and a text encoder (in red), taking image feature vectors and query text as input, respectively.

1 INTRODUCTION

The "KDD Cup 2020 Challenges for Modern E-Commerce Platform: Multimodalities Recall" requires searching for the most related products according to a particular search query. Given this task, our intuitive idea is to learn to align queries with their target images. We develop a model following the work of "Visual Semantic Reasoning for Image-Text Matching(VSRN)"[2]. According to the aim of the task and the characteristics of the provided data, We have made several improvements on the model architecture of VSRN, including 1) using multi-head region relationship reasoning module, 2) developing a text encoder based on multi-head region relationship reasoning module, 3) using a simplified image encoder, 4) adopting the global image features, and 5) proposing a cross-modal cosine loss. Besides re-designing the model architecture, in training the model, we use an undersampling technique to tackle the data imbalance problem, and use a random drop data augmentation method on queries to improve the generality of the model. After training the model on the whole training dataset, we fine-tune the model on a refined dataset (see section 4.5) to improve its ability in distinguishing between the unique attributes of the same product. We also use a re-ranking technique[5] in post processing. A single model reaches the score of 0.7955 on valid dataset, 0.7712 on testB. After model fusion, the ultimate score on the valid set is 0.8151, and 0.7872 on testB.

2 NETWORK ARCHITECTURE

The original work of VSRN is built for image-text matching, which operates by: 1)implementing the bottom-up attention with a Faster R-CNN model using ResNet-101 as the backbone to get region-level features of images, 2) applying Graph Convolutional Networks (GCN) to perform region relationship reasoning, 3) doing global semantic reasoning to select important information based on region relationship information and a GRU network, 4) encoding text with a GRU network, 5) learning image-text alignments by adopting a hinge-based triplet ranking loss and a text generation loss. Our model is developed based on, but different from VSRN in five ways as listed in the former section. To better introduce our design, Figure 1 shows the whole architecture, and this section gives a detailed explanation.

2.1 Multi-head Region Relationship Reasoning

Following VSRN, our model use GCN with residual connections to perform region relationship reasoning. As a modification, we use multi-head region relationship reasoning, formulated as:

$$V^* = W_r(cat_i^n(R_i V_i W_{gi})) + V \quad (1)$$

where W_g is the weight matrix of the GCN layer. $W_r$is the weight matrix of residual structure. R is the affinity matrix.

V^* is the relationship enhanced representation for image region nodes. n is the number of heads, set to 16 in our model. $cat(\cdot)$ denotes for a concatenate function.

2.2 Image Encoder

The image encoder consists of 4 multi-head region relationship reasoning layers and a bi-directional GRU. The original VSRN uses a full connection (FC) layer before the first region relationship reasoning layer, however, we find that FC layer lowers the performance of the model in our task, and we get rid of it.

2.3 Text Encoder

We find the text encoder in related works tends to be simple in structure, however, with the high complexity of queries in our task, the widely used LSTM or GRU based model have shown limits. To better extract text features, our text encoder inputs word embedding vectors into an FC layer, following with 4 multi-head region relationship layers and then a bi-directional GRU.

2.4 Global Image Feature

For an image, we get the global feature by calculating the mean feature of each detected region. And then concatenate it in the end. We use this new feature with global features as the new representation of the image. Experiments show that global features can improve model performance to a certain extent. The region-level visual features of an image is denoted as

$$V = [v_1, v_2, ..., v_R] \in \mathbb{R}^{d \times R} \tag{2}$$

, where R is the number of detected regions, d is the number of feature dimension. Global feature is obtained by the average of each region-level features, the formula is:

$$v_g = mean(v_1 + v_2 + ... + v_R) \tag{3}$$

, and the ultimate image features with global features are represented as

$$V = [v_1, v_2, ..., v_R, v_g] \tag{4}$$

2.5 Cross-modal Cosine Loss

We use a cross-modal cosine loss as our loss function, which is inspired by the previous work of Li,et al.[3], Wang,et al.[4], and Deng,et al.[1]. The function reduces intra-class distance while expanding inter-class distance. The function contains two hyper-parameters including margin m and temperature γ. m affects the distance between classes and γ affects the degree of compacting the instances of the same class. We use N to denotes the number of samples in a mini-batch. Let $\tilde{v_i} \in \mathbb{R}^d$ denote one image sample's feature, $T = [\tilde{t_1}, \tilde{t_2}, ...\tilde{t_N}] \in \mathbb{R}^{d \times N}$ denote textual samples' features in a mini-batch. $S_{ip} = cosine(\tilde{v_i}, \tilde{t_p})$is the cosine similarity of image feature $\tilde{v_i}$ with its paired positive textual features and $S_{in} = cosine(\tilde{v_i}, \tilde{t_n})$ is cosine similarity of image feature with its negative textual samples. The formula is as follows:

$$L_{cosine} = \frac{1}{N}\sum_{i}^{N} -log\frac{exp(\gamma(S_{ip} - m))}{exp(\gamma(S_{ip} - m)) + \sum_{j \neq p}^{N} exp(\gamma S_{ij})} \tag{5}$$

3 POST-PROCESSING

3.1 Re-ranking

Re-ranking is proposed to boost the performance of person re-identification, whose hypothesis is that if a gallery image is similar to the probe in the k-reciprocal nearest neighbors, it is more likely to be a true match. Specifically, given an image, a k-reciprocal feature is calculated by encoding its k-reciprocal nearest neighbors into a single vector, which is used for re-ranking under the Jaccard distance. The final distance is computed as the combination of the original distance and the Jaccard distance. We use this technique to compute the final distance of our encoded image and text features.

3.2 Text Concatenation

We notice that some queries contain same words but with a different order of words, for instance, "long-sleeve summer shirt" and "summer long-sleeve shirt", we hypothesize that the alignment of a query and its image is not strictly constrained by the order of attribute, but is loosely effected by it. After trying to randomly exchange words order in training and have not found obvious benefits to the model, we improve the problem by concatenate the original query with itself when validating and testing the model instead.

3.3 Model Fusion

We train the model under six different seeds, and then merge their score matrix in average to get the final result. The network under different seeds are initialized differently, resulting in slightly different models. Finally, through model fusion, the generalization ability of the network is improved.

3.4 Single Directional GRU Feature Testing

Although we train bi-directional GRU for text encoder, we only use its backwards features for validating and testing. Experiments show the setting very helpful in improving prediction results.

3.5 Consistency of The Translations

According to the observation to the dataset, we find the inconsistency of translation in training dataset and the test dataset of the expression of *forest style*(training) and *sen department*(testA and testB). To make the translation of the dataset consistent, we change the expression of *sen department* to *forest style*, which slightly improves the performance.

4 TRAINING DETAILS

4.1 Data Augmentation

In dealing with text, we use a random drop method to augment data, which randomly set one word of a query to a special mark of $< unk >$. Following the setting of VSRN, $< unk >$ stands for unknown words, which has its index in the vocabulary dictionary of the model, whose representation vector is updated during training.

4.2 Word Embedding

We compared the following word embedding method: a) randomly initialized word embedding, b) public pre-trained Glove word embedding, c) Glove word embedding that trained on KDD2020 train and valid query set, d) Word2vec word embedding that trained on KDD2020 train and valid query set, e) the concatenation of the previous. We find the method d) is the most useful one in our task. When training the Word2vec embedding model, we organize all the queries in training set and validation set into a corpus, use skip-gram, Softmax, set the window size to 5, and the dimension to 300. The threshold of rare word is set to 4 when initialing the vocabulary dictionary.

4.3 Stopwords

We maintain a stopwords list in our experiments. According to the observation to the dataset, the stopwords list contains numbers, articles, pronouns, punctuation, etc.

4.4 Sorted Box

For a specific detected box in an image, the box coordinates are denoted as $[x_1^j, x_1^j, x_2^j, y_2^j], j = 1, 2..., R$, where R is the number of detected regions. We sort the boxes in an image according to the first coordinate x_1. Box with smaller x_1 is in the front. Through this method, all detected boxes are arranged in a certain order, which is slightly beneficial to model training.

4.5 Fine-tune

After training the model on the whole training dataset, we fine-tune the model to improve its ability in distinguishing between the unique attributes of the same product. We noticed that in most conditions, the last one word of a query is a noun, which is the target product, for example, "long-sleeve summer shirt", "professional books", for convenience, we assume the finding is always true, so we can build a product-oriented dataset for fine-tuning. Our fine-tuning idea is to gather the queries of the same product to a mini-batch, so that the model is compelled to learn the differences between attributes of the same product. To balance the difficulty of learning and the fine-tuning effect, we set the mini-batch size to 128, with at most 16 queries per product, and several products per batch.

4.6 Undersampling

Since the data we used to train the retrieval model is unbalanced, the model may over fit to a few classes with a large number of samples. Therefore, we propose an undersampling method to balance the training data so as to regularize the model and improve model performance. In order to better mine the characteristics of the data itself, we designed a hierarchical undersampling strategy. Every time we read 20,000 samples into memory and apply undersampling algorithm on these samples. We use K_i to represent the number of samples in class i. For a specific sample in class i, if K_i is larger than a threshold, we drop this sample according to a certain ratio. The detail of our method is showed in Algorithm 1.

Algorithm 1 Undersampling method in our model.

```
if K_i <= 10 then
    Drop the sample with ratio 0%
else if 10 < K_i <= 50 then
    Drop the sample with ratio 40%
else if 50 < K_i = 100 then
    Drop the sample with ratio 50%
else if 100 < K_i <= 250 then
    Drop the sample with ratio 60%
else if 250 < K_i <= 500 then
    Drop the sample with ratio 80%
else
    Drop the sample with ratio 90%
end if
```

4.7 Implementation Details

The code is developed using python 3.6.9 based on LINUX operating system. NVIDIA GPUs are needed and the version of CUDA is 10.1 while the version of CUDNN is 7.6.4. The code is developed and tested using 1 server with 1 NVIDIA TITAN XP GPU card. For training, we use the Adam optimizer to train the model with 7 epochs. We start with learning rate 0.0002 for 4 epochs, and then lower the learning rate to 0.00002 for the rest 3 epochs. We set the hyper-parameter margin m and temperature in cross modal cosine loss function to 0.05 and 14 respectively. The batch size of the experiment is 128 and the best model and current model is saved every 500 steps. In addition, we set the word embedding size to 300 and the dimension of the joint embedding space d to 2048. For the fine-tuning process, we use Adam optimizer with an learning rate of 0.00002. We fine-tune the model for 6 epochs with a batch size of 128.

5 EXPERIMENT RESULTS

To test the efficiency of all the method we propose, we have done comprehensive experiments, finally, a single model reaches the score of 0.7955 on valid dataset, 0.7712 on testB. After model fusion, the ultimate score on the valid set is 0.8151, and 0.7872 on testB.

REFERENCES

[1] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. (2019), 4690–4699.

[2] Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. 2019. Visual semantic reasoning for image-text matching. In *ICCV*.

[3] Shuang Li, Tong Xiao, Hongsheng Li, Wei Yang, and Xiaogang Wang. [n.d.]. Identity-Aware Textual-Visual Matching With Latent Co-Attention. In *The IEEE International Conference on Computer Vision (ICCV)*.

[4] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. 2018. CosFace: Large Margin Cosine Loss for Deep Face Recognition. (2018), 5265–5274.

[5] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. 2017. Re-ranking Person Re-identification with k-reciprocal Encoding. (2017).