A Neighbourhood Encoding Framework for Deep Mining Heterogeneous Texts in Recipe-image Retrieval

Changsheng Zhu1,2, Nan Ji1, Jin Yu1,3, Dazhi Jiang1,3 and Lin Zheng1,3,*
1Department of Computer Science, College of Engineering, Shantou University, China
2Research Office, Shantou University, China
3Key Laboratory of Intelligent Manufacturing Technology (Shantou University), Ministry of Education, China

*Corresponding author email: lzheng@stu.edu.cn

Abstract. Cross-modal retrieval usually fills the semantic gap between different modalities by sharing subspaces. However, existing methods rarely consider that the data in a certain modality may be heterogeneous when mapping multimodal data into a shared subspace. In addition, most existing methods focus on semantic associations between different modalities, while few approaches consider the semantic associations within a single modality. To address the above two deficiencies, we propose a Neighbourhood Encoding (NE) framework that mines the semantic association of data in the same modality, solves the problem of data heterogeneity by improving the semantic expression of a single modality. To verify the effectiveness of the proposed framework, we use two types of recurrent neural networks to instantiate the framework. Experiments show that the instantiated approaches outperform existing advanced methods in both text-to-image and image-to-text retrieval directions.

Keywords: Cross-modal retrieval; Heterogeneous text; Neighbor constraint; Neighbor contribution; Text encoder.

1. Introduction
The heterogeneous nature of different modal data and the semantic gap between the underlying feature information and high-level semantics make cross-modal retrieval a challenging research direction [1]. In recent years, advanced methods for solving cross-modal retrieval are mainly based on the idea of sharing subspace by different modalities [2,3]. This category of methods maps the information learned from different modalities into a shared subspace and calculates similarity metrics for the different modal data in the same subspace. The method based on a shared subspace by default assumes that the data of a single modality are consistent; however, in many scenarios, the data in a single modality is heterogeneous, such as texts [4]. For example, the existing work retrieves food images through joint learning features from both the corresponding recipe ingredient and cooking instruction [5,6]. However, the information provided by this recipe-image one-to-one relationship is limited. In other words, it is difficult to correlate dish images directly through the corresponding cooking instruction and recipe ingredient. Intuitively, only comparing different and similar cooking instructions can help us to better find the corresponding dishes. We call these similar cooking methods neighbors. Similarly, the recipe ingredients also have their corresponding neighbors. For long text (cooking instructions) and short text (recipe ingredients), if we can find their respective neighbors, learn the important features of the neighbors and fuse them, we can better express the semantics of the corresponding food image and improve the effect of cross-modal retrieval. To achieve this goal, the contributions of this paper are as follows:
We propose a Neighbourhood Encoding (NE) framework for heterogeneous text, which is used to search, constrain, and integrate neighbors of texts with different structures to achieve high-level semantic representation of the corresponding food images.

We propose a neighbor constraint mechanism to model the relation between current recipe and each neighbor. Then, we implement two different structured text encoders based on two types of heterogeneous texts (ingredient and instruction).

In the experiment, we instantiated the NE framework as NE-LSTM and NE-GRU. The experimental results verify that the two approaches have significant performance improvements over existing advanced methods in both text-to-image and image-to-text retrieval.

2. Methodology

2.1. Neighbourhood Encoding Framework for Heterogeneous Text

According to the two different structures of ingredients and cooking instructions, we illustrate a two-way (short text and long text) neighbourhood encoding framework to introduce our work, which can be easily extended to a N-way framework in other scenarios. Figure 1 shows our Neighbourhood Encoding (NE) framework for heterogeneous text. In the NE framework, each recipe consists of two parts, ingredients and cooking instructions. First, we define the concept of the top-N Neighbors. In particular, given a set of one category of heterogeneous text (ingredients or instructions) \( S = \{t_1, t_2, \ldots, t_n\} \) with its embedding set \( E \), let the function \( C(t_i, t_j, E) \) represent the embedding similarity between \( t_i \) and \( t_j \). The general definition of the top-N neighbors is formalized as follows.

**Figure 1.** The Neighbourhood Encoding (NE) framework for heterogeneous text.

**Definition 1.** (top-N Neighbors). For a heterogeneous text (ingredients or instructions) \( t \), suppose that \( t_j \in S \) is some text in the collection \( S \) except \( t \). Next, calculate \( C(t_i, t_j, E) \) and sort the similarity scores to obtain a descending ordered text set \( S_i = \{t_1, t_2, \ldots, t_N\} \). Then, the top-N neighbors of \( t \) are defined as the first \( N \) entities in \( S_i \). For a given recipe, the whole process of the neighborhood encoding scheme is completed in three steps: Step 1: top-N Neighbors, Step 2: Neighbor Constraint, and Step 3: Neighbor Integration.

2.1.1. Step 1: top-N Neighbors. We first searched the top-N neighbors through a similarity calculation on the embedding set \( E \). The similarity calculation or distance measurement is the main criteria for neighbor selection in many research work [7,8], in particular, the paper adopts the cosine similarity as the measurement method of text neighbors.

2.1.2. Step 2: Neighbor Constraint. After determining the top-N neighbors, we would like to constrain each neighbor to qualify its contribution. Specifically, we would like to apply a dynamic soft constraint on neighbors; and the definition of Neighbor Constraint is formalized as follows.
**Definition 2.** (Neighbor Constraint). For one type of heterogeneous text $I$, (e.g., recipe ingredient) and its top-$N$ neighbors $\{I_i, I_2, ..., I_N, I_a\}$, the neighbor constraint is defined in Equation 1.

$$
\alpha_i = \frac{\exp(K_iQ)}{\sum_j \exp(K_jQ)}
$$

(1)

Here, $\alpha_i$ is actually the attention weight for neighbor $I_i$ used in Equation 2. $Q$ converts a specific text representation $I_i$ using a Multi Layer Perceptron (MLP), and $K_j$ denotes its $j$-th neighbor representation $I_j$ using another MLP. The larger that $\alpha_i$ is, the greater the contribution of the neighbor. The details of neighbor constraint and encoding are described in section 3.2.

2.1.3. Step 3: Neighbor Integration. In terms of neighbor integration, the method we use is different from the existing ensemble learning method [8]. The most essential difference is that we design a dynamic mechanism to model neighbors instead of using a single weighting variable to control neighbors. In particular, we propose a simplified recurrent network to treat the $j$-th neighbor ($I_i$) of a text representation $I_i$ as the $j$-th input to the network as follows.

$$
h^j_{i} = \gamma(\alpha_i I_i + W_{\alpha} h^j_{i-1} + B_{\alpha}), f^j_i = \eta(W_{\eta} h^j_{i} + B_{\eta})
$$

(2)

where $j \in \{2,3,..,n\}$ and $(j-1) > 0$.

Here $\gamma$ and $\eta$ are some activation functions. $h^j_{i-1}$ and $h^j_i$ are the hidden states of the $(j-1)$-th and the $j$-th neighbors, respectively. $f^j_i$ is the integration result of the previous $j$ neighbors, therefore, the top-$N$ neighbor integration result is denoted by $f^j$. $W_{\alpha}$, $B_{\alpha}$, $W_{\eta}$, and $B_{\eta}$ are the affine transform parameters between states or results. This design helps to emphasize the contribution of neighbors (that is, $\alpha_i I_i$) while taking full advantage of RNNs.

Please note that $\alpha_i$ is the neighbor constraint in Equation 1 that represents the contribution of the $j$-th neighbor. After neighbor integration, we use concatenating operation to implement the neighbor encoding on two types of neighbor groups in preparation for the following recipe-image retrieval.

2.2. Implementation of the Neighbourhood Encoding Framework

2.2.1. Learning image feature. We adopt an abstract convolutional network to represent the image feature learning. In particular, the convolutional neural network is instantiated as Resnet-50 [9] and VGG-16 [10] to learn the image embedding vectors. We fine-tune the parameters, delete the last FC layer and mark the rest as the image feature expression, which is denoted as $imagef$.

2.2.2. Learning recipe representation. Each recipe includes ingredients and cooking instructions. The task of learning top-$N$ recipe neighbor representations is to learn the representations of top-$N$ ingredient neighbors $\{I_i, I_2, ..., I_N, I_a\}$ and top-$N$ instruction neighbors $\{S_j, S_2, ..., S_N\}$, respectively. Consequently, we propose an ingredient encoder and an instruction encoder to separately complete the two tasks. In ingredient encoder, we learn an ingredient representation via word2vec technology [11] for each word to obtain a sequential word2vec features $\{w_1, w_2, ..., w_r\}$. A Bi-RNN is composed of a forward $RNN$ that reads from $w_1$ to $w_r$ and a backward $RNN$ that reads from $w_r$ to $w_1$, which learns the $n$-th ingredient neighbor representation of the $i$-th recipe as follows:

$$
I_a = [RNN([w_1, ..., w_r]), RNN([w_r, ..., w_1])]
$$

(3)

where the $n$-th ingredient neighbor representation $I_a$ is formed by concatenating the forward output and the backward output together. A cooking instruction consists of various sentences of different
lengths \( \{s_1, \ldots, s_g\} \) that describe the detailed steps of preparing the food.

In instruction encoder, we employ a two-stage RNN (Ts-RNN) to obtain the representation of the cooking instructions. Specifically, in the first stage, each sentence of an instruction neighbor is represented as a Skip-Thought [12] (denoted as \( T \)) sentence vector. Then, in the second stage, a regular RNN (can be LSTM or GRU) is adopted for training over the sequence vectors to obtain the \( n \)-th instruction neighbor representation of the \( j \)-th recipe as follows.

\[
S_{nj} = \text{RNN}(T(s_1^j), \ldots, T(s_t^j))
\]

2.2.3. Neighbor constraint and encoding. According to Definition 2 and Equation 1, we take ingredient neighbors to illustrate the process, which is similar to obtaining the instruction neighbor constraints. Given an ingredient representation \( I_i \) and its top-N neighbors \( \{I_{i1}, I_{i2}, \ldots, I_{in}\} \), the self transformation representation \( Q_i \) and the neighbor relation \( K_j \) between \( I_i \) and \( I_{ij} \) are defined as:

\[
Q_i = \text{MLP}_{\text{base}}(W_0I_i + B_0)
\]

\[
K_j = \text{MLP}_{\text{base}}(W_k(I_i \odot I_j) + B_k)
\]

Here, we employ two different one-layer MLPs with ReLu as activation function to calculate \( Q_i \) and \( K_j \), where \( W_0, W_k, B_0, B_k \) are the transformation matrices and biases, respectively. In particular, the element-wise product \( I_i \odot I_j \) indicates the raw relation between \( I_i \) and \( I_j \), whereas the self transformation can better extract the abstract meaning of \( I_i \). Then, \( Q_i \) and \( K_j \) are used as inputs to Equation 1 to obtain the neighbor constraint \( \alpha_{ij} \). Finally, we integrate all the ingredient neighbors by using a simplified recurrent network as defined in Equation 2 to get the ingredient representation \( f_{\text{ingredient}} \). The instruction representation \( f_{\text{instruction}} \) can also be obtained in a similar way. For neighbor encoding, we simply concatenate \( f_{\text{ingredient}} \) and \( f_{\text{instruction}} \) to complete the neighbor encoding process for recipe representation as follows.

\[
 f_{\text{recipe}} = [f_{\text{ingredient}}, f_{\text{instruction}}]
\]

In experiments, the dimensions of \( f_{\text{ingredient}} \) is set as 600. Since instructions have dense descriptions; and \( f_{\text{instruction}} \) is set as a 1,024-dimensional vector.

2.2.4. Recipe-image joint learning. Both recipe and image representations can be mapped into a joint space using a linear transformation as follows.

\[
\phi_x = \tanh(W_x f_{\text{recipe}} + B_x)
\]

\[
\phi_v = \tanh(W_v f_{\text{image}} + B_v)
\]

where \( W_x, W_v \) and \( B_x, B_v \) are transformation matrices and biases to assist generating recipe embedding \( \phi_x \) and image embedding \( \phi_v \). Then, we train a cosine similarity loss function with margin, which is defined as follows:

\[
L_{\text{cos}}((\phi_x, \phi_v), y) = \begin{cases} 1 - \cos(\phi_x, \phi_v), & \text{if } y = 1 \\ \max(0, \cos(\phi_x, \phi_v) - \alpha), & \text{if } y = -1 \end{cases}
\]

where \( \alpha \) is the margin and \( \cos(.) \) is cosine similarity with normalization. The objective function calculates the similarity of positive recipe-image pairs \( (y = 1) \) in a cosine form and tries to maximize it; meanwhile, it minimizes the similarity between those non-matching pairs \( (y = -1) \), using \( \alpha \) as a specified margin range. The final objective function is formalized by adding a semantic regularization term to Equation 10.
3. Experiment
We use a well-known recipe dataset—Recipe1M [5,6]. To work efficiently, we randomly sample a subset of the Recipe1M dataset, which contains 52,432 pairs of images and recipes in its training set, 12,737 image-recipe matching pairs in its valid set and 12,655 matching pairs in the test set, respectively. Food-101 [13] is another food image dataset we used.

3.1. Experiment Setting

3.1.1. Implementation details. We instantiate the NE framework into two specific methods, NE-LSTM and NE-GRU. We use similar parameter settings for these two methods. Specifically, the learning rate is set to $10^{-5}$ with Adam optimizer, and batch size is set as 32 in all our experiments. The margin $\alpha$ is set as 0.1, whereas the regularization hyperparameter is set to be $\lambda = 0.2$. Meanwhile, to optimize the objective function, we randomly pick positive matching pairs with a 0.3 probability and negative recipe-image pairs with a 0.7 probability from the training set. We use the same evaluation metrics as [5,6], which are Median retrieval Rank ($MedR$) and Recall at top-K ($R@K$).

3.1.2. Testing. There are image-to-recipe (im2recipe) and recipe-to-image (recipe2im) two retrieval directions. To test the effectiveness of our NE-LSTM and NE-GRU models, we compare them with the following models: JNE (Joint neural embedding) [5], JNE-SR (JNE with semantic regularization), and JNE*-SR (JNE-SR with GRU as the recipe encoder).

3.2. Optimal Neighbor Number
Figure 2 shows our experimental results of the neighbor number influence conducted on both NE-LSTM and NE-GRU models. We take the top-$N$ ($N \in [0,9]$) nearest neighbors for a specific recipe. $N = 0$ means that the model has no neighbors, that is, the model considers only the recipe itself.

As seen from Figure 2, when the number of neighbors is 4, the NE-LSTM model reaches its optimal retrieval performance. In contrast, the optimal neighbor number on the retrieval performance of NE-GRU model is 6. We adopt the optimal neighbor number for the following experiments.

3.3. Performance Comparison

|        | im2recipe |          |          |          | recipe2im |          |          |          |
|--------|-----------|----------|----------|----------|-----------|----------|----------|----------|
|        | MedR      | $R@1$    | $R@5$    | $R@10$   | MedR      | $R@1$   | $R@5$    | $R@10$   |
| Random | 499       | 0.001    | 0.005    | 0.01     | 499       | 0.001   | 0.005    | 0.01     |
| JNE    | 38.4      | 0.068    | 0.202    | 0.291    | 37.9      | 0.069   | 0.205    | 0.283    |
| JNE-SR | 35.6      | 0.072    | 0.210    | 0.297    | 34.4      | 0.073   | 0.211    | 0.298    |
| JNE*-SR| 34.1      | 0.076    | 0.223    | 0.302    | 33.7      | 0.077   | 0.214    | 0.300    |
| NE-LSTM| 28.6      | 0.095    | 0.245    | 0.323    | 29.0      | 0.092   | 0.240    | 0.326    |
| NE-GRU | **26.3**  | **0.108**| **0.257**| **0.333**| **28.8**  | **0.107**| **0.259**| **0.338**|

Figure 2. The impact of neighbor numbers on the performance.
Table 1 shows the detailed performances of different models being compared. The NE-GRU model effectively improves the retrieval performance with respect to other baselines. By comparing the various models, it is confirmed that the instantiated NE framework for heterogeneous text has performance advantages in two retrieval tasks. In general, the experimental results show that the proposed NE framework is effective and applicable.

4. Conclusion
In the paper, we propose a Neighbourhood Encoding (NE) framework that mines the semantic association of heterogeneous data in the same modality. In future work, we will consider the heterogeneity of images and extend the NE framework to image data for better retrieval performance.

Acknowledgments
This research is funded by the Guangdong Basic and Applied Basic Research Foundation (No. 2020A1515010531), the National Natural Science Foundation of China (No. 61902231), and the Research Startup Fund Project of Shantou University (No. NTF18017).

References
[1] Wu, Y.; Wang, S.; Huang, Q. Multi-modal semantic autoencoder for cross-modal retrieval. Neurocomputing 2019, 331, 165 – 175.
[2] Zhang, L.; Ma, B.; Li, G.; Huang, Q.; Tian, Q. Cross-modal retrieval using multiordered discriminative structured subspace learning. IEEE Transactions on Multimedia 2016, 19, 1220–1233.
[3] Yan, J.; Zhang, H.; Sun, J.; Wang, Q.; Guo, P.; Meng, L.; Wan, W.; Dong, X. Joint graph regularization based modality-dependent cross-media retrieval. Multimedia Tools and Applications 2018, 77, 3009–3027.
[4] Vlachostergiou, A.; Caridakis, G.; Mylonas, P.; Stafylopatis, A. Learning representations of natural language texts with generative adversarial networks at document, sentence, and aspect level. Algorithms 2018, 11, 164.
[5] Salvador, A.; Hynes, N.; Aytar, Y.; Marin, J.; Ofli, F.; Weber, I.; Torralba, A. Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[6] Marin, J.; Biswas, A.; Ofli, F.; Hynes, N.; Salvador, A.; Aytar, Y.; Weber, I.; Torralba, A. Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. IEEE Trans. Pattern Anal. Mach. Intell. 2019.
[7] Spertus, E.; Sahami, M.; Buyukkokten, O. Evaluating similarity measures: a large-scale study in the orkut social network. Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. ACM, 2005, pp. 678–684.
[8] Zheng, L.; Zhu, F.; Huang, S.; Xie, J. Context Neighbor Recommender: Integrating contexts via neighbors for recommendations. Information Sciences 2017, 414, 1 – 18.
[9] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[10] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 2014.
[11] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 2013.
[12] Kiros, R.; Zhu, Y.; Salakhutdinov, R.R.; Zemel, R.; Urtasun, R.; Torralba, A.; Fidler, S. Skip-Thought Vectors. In Advances in Neural Information Processing Systems 28; Cortes, C.; Lawrence, N.D.; Lee, D.D.; Sugiyama, M.; Garnett, R., Eds.; Curran Associates, Inc., 2015; pp. 3294–3302.
[13] Bossard, L.; Guillaumin, M.; Van Gool, L. Food-101–mining discriminative components with random forests. European Conference on Computer Vision. Springer, 2014, pp. 446–461.