Interpretable Relational Representations for Food Ingredient Recommendation Systems

Kana Maruyama, Michael Spranger
SonyAI
kana.maruyama@sony.com
michael.spranger@sony.com

Abstract

Supporting chefs with ingredient recommender systems to create new recipes is challenging, as good ingredient combinations depend on many factors like taste, smell, cuisine style, texture, chef's preference and many more. Useful machine learning models do need to be accurate but importantly– especially for food professionals – interpretable and customizable for ideation. To address these issues, we propose the Interpretable Relational Representation Model (IRRM). The main component of the model is a key-value memory network to represent the relationships of ingredients. The IRRM can learn relational representations over a memory network that integrates an external knowledge base- this allow chefs to inspect why certain ingredient pairings are suggested. Our training procedure can integrate ideas from chefs as scoring rules into the IRRM. We analyze the trained model by comparing rule-base pairing algorithms. The results demonstrate IRRM's potential for supporting creative new recipe ideation.

Introduction

Data mining and machine learning methods play an increasingly prominent role in food preference modeling, *food ingredient pairing discovery*, and *new recipe generation*. Solving these tasks is non-trivial, since the goodness of ingredient combinations depends on many factors like taste, smell, cuisine, texture, culture, and human creative preferences. Although efforts have been made to detect good ingredient combinations using Machine Learning and build models that help in the creation of recipes or discover novel food ingredient pairs - there is no current machine learning method in this field that 1) allows embedding chef specific ideas to be incorporated in the creation process and 2) offer interpretations why a suggested ingredient pair is good.

Our work is aimed at interpretable and customizable food ingredient recommendation systems that inspire chefs to find new recipe ideas. In this paper, we propose the Interpretable Relational Representations Model (IRRM) an interpretable and customizable neural network score function (see Fig. 1). Given a set of pre-selected ingredients (cardinality 1 or more) by a user, the IRRM suggests top-N ingredients from a set of candidates. For example, suppose a user selects *apple* and *chocolate* as the pre-selected ingredients, IRRM suggests compatible ingredients (e.g. *cinnamon*), and also identifies reasons (e.g. *cinnamon* is good for *apple* and *chocolate* in terms of their flavor affinity).

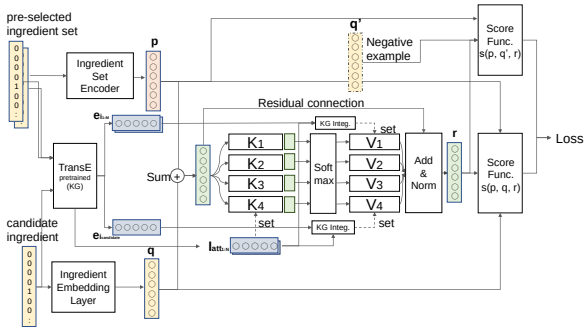

Figure 1: IRRM architecture

Professional chefs already have a lot of their own favorite recipes and are inspired by everything around them to develop new recipes. That is, in the process of creating new recipes they might want to constrain or input prior knowledge into the system. For example a list of existing recipes either by the chef or a list of recipes that the chef finds inspiring even if not by him or herself. Therefore, we allow recipes (i.e. ingredient lists of a particular chef) as input to IRRM.

Our contributions are as follows:

1. We present an extensible framework for scoring ingredient-ingredient combinations incorporating prior ideas from chefs via recipes.
2. We introduce the Interpretable Relational Representations Model (IRRM), inspired by session-based recommendation systems with implicit feedback. Leveraging a pre-trained ingredient knowledge graph, our model can learn pair-specific relational representations for one-to-one (i.e. ingredient to ingredient) and many-to-one (i.e. ingredient-set to ingredient) food ingredient pairing tasks from recipes (i.e. a list recipes that are apriori available constraints). The trained relational vectors are also interpretable.
3. We propose a training procedure to integrate chef's ideas as scoring rules via positive sampling strategies.

Problem Definition

We model food ingredient pairing as a session-based recommendation scenario with implicit feedback (Huang et al. 2018; Tay, Tuan, and Hui 2018).

Let $\mathcal{I}$ denote a set of ingredients and $\mathcal{I}_{target} = \{i_1, \ldots, i_M\}$ denote a pre-selected ingredient set, where $i \in \mathcal{I}$ is the ingredient, M is the number of ingredients, and $\mathcal{I}_{target} \subset \mathcal{I}$. We call $\mathcal{I}_{target}$ a pre-selected ingredient set in this paper. Next, let $\mathcal{I}_{candidate}$ denote a set of candidate ingredients. $\mathcal{I}_{candidate}$ depends on each pre-selected ingredient set, that is, $\mathcal{I}_{candidate} = \mathcal{I} - \{i_1, \ldots, i_M\}$.

In addition, we use an ingredient knowledge base (KB). The KB helps to estimate good ingredient pairs in terms of contextual information on ingredients.

Based on these preliminaries, we define the food ingredient recommendation task. Given a pre-selected ingredient set $\mathcal{I}_{target}$ and candidate ingredients $\mathcal{I}_{candidate}$, we would like to infer the top-N ingredients from $\mathcal{I}_{candidate}$.

Recommendations with Key-Value Memory Networks

Ingredients are represented as one-hot encoding vectors (corresponding to a unique index key belonging to each ingredient). At the ingredient embedding layer, this one-hot encoded vector is converted into a low-dimensional real-valued dense vector representation which is multiplied with the embedding matrices $\boldsymbol{Q} \in \mathbb{R}^{d \times |\mathcal{I}|}$. d is the dimensionality of the ingredient embeddings while $|\mathcal{I}|$ is the total number of ingredients. $i_{candidate} \in \mathcal{I}_{candidate}$ is converted to $\boldsymbol{q}$ using this embedding layer. On the other hand, a pre-selected ingredient set $\mathcal{I}_{target} = \{i_1, \ldots, i_j, \ldots, i_M\}$ is encoded by the Ingredient Set Encoder. At first, each ingredient i_j is converted to a vector using the ingredient embedding layer (same as $i_{candidate}$). As a result, $\{\boldsymbol{i}_j \in \mathbb{R}^d | j = 1, \ldots, M\}$ vectors are generated. The sum of these vectors is normalized and converted to the ingredient set vector $\boldsymbol{p}$ using a feed-forward network with a single hidden layer, followed by Layer Normalization. Given a pair of a pre-selected ingredient set vector and a candidate ingredient vector, $\langle \boldsymbol{p}, \boldsymbol{q} \rangle$, the Relation Encoder first applies $\boldsymbol{s} = \boldsymbol{p} + \boldsymbol{q}$ to generate the joint embedding of $\boldsymbol{p}$ and $\boldsymbol{q}$. The generated vector $\boldsymbol{s} \in \mathbb{R}^d$ is of the same dimension of $\boldsymbol{p}$ and $\boldsymbol{q}$. This joint embedding $\boldsymbol{s}$ is used as the input to the key-value memory network. The attention vector $\boldsymbol{a} \in \mathbb{R}^d$ is a vector of importance weights over keys which are represented as the key matrix $\boldsymbol{K} = [\boldsymbol{l}_{att_1}, \ldots, \boldsymbol{l}_{att_N}]^T \in \mathbb{R}^{N \times d}$, where N is the number of key-value pairs in the memory network and $\boldsymbol{l}_{att_j} \in \mathbb{R}^d$ is a key vector. Each element of the attention vector $\boldsymbol{a}$ can be defined as $a_j = \boldsymbol{s}^T \boldsymbol{l}_{att_j}$, where $a_j \in \mathbb{R}$. In order to normalize the attention vector $\boldsymbol{a}$ to a probability distribution, we use the Softmax function:$\mathrm{Softmax}(a_j) = \frac{\exp(a_j)}{\sum_{n=1}^{N} \exp(a_n)}$. We generate the vector $\boldsymbol{m} = \sum_{n=1}^{N} \mathrm{Softmax}(a_n) \boldsymbol{v}_{att_n}$ as the summation of weighted value vectors which are represented as the value matrix $\boldsymbol{V} = [\boldsymbol{v}_{att_1}, \ldots, \boldsymbol{v}_{att_N}]^T \in \mathbb{R}^{N \times d}$. Finally, in order to generate the relational vector $\boldsymbol{r}$, $\boldsymbol{m}$ is added with the joint embedding $\boldsymbol{s}$ (residual connection) and Layer Normalization is applied as follows $\boldsymbol{r} = \mathrm{LayerNorm}(\boldsymbol{s} + \boldsymbol{m})$.

We use pre-trained knowledge graph embeddings over a given KB for the key matrix $\boldsymbol{K}$ and the value matrix $\boldsymbol{V}$, where N depends on the number of attribute types which you want to integrate and $\boldsymbol{K}$ is constant through training. Given a pair of a pre-selected ingredient set $\mathcal{I}_{target} = \{i_1, \ldots, i_M\}$ and a candidate ingredient $i_{candidate}$, $\{i_1, \ldots, i_M, i_{candidate}\}$ is converted into the entity vectors using knowledge graph embeddings which provide the entity vectors $\boldsymbol{e} \in \mathbb{R}^{d^{KB}}$ and the relationship vectors $\boldsymbol{l} \in \mathbb{R}^{d^{KB}}$. We use the TransE (Bordes et al. 2013) for the knowledge graph embeddings. The reason for this choice is that given triplet $\langle e_i, l_{att}, e^i_{att} \rangle$, TransE can learn entity vectors and relationship vectors to follow $\boldsymbol{e}^i_{att} = \boldsymbol{e}_i + \boldsymbol{l}_{att}$. Using it, we define a value vector as $\boldsymbol{v}_{att_j} = LayerNorm(\sum_{i \in \{i_1, \ldots, i_M, i_{candidate}\}} FF(e^i_{att}))$. FF is a feed-forward network with a single hidden layer.

Finally, we define our score function as the relationship between the pre-selected ingredient set vector $\boldsymbol{p}$, the candidate ingredient vector $\boldsymbol{q}$, and the relational vector $\boldsymbol{r}$:

$$s(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}) = \mathrm{CosSim}(\boldsymbol{p}, \boldsymbol{q}) + \mathrm{CosSim}(\boldsymbol{p} + \boldsymbol{q}, \boldsymbol{r}) \quad (1)$$

where CosSim is the cosine similarity. This function scores the affinity for the relationships. Note that some studies use distance functions instead of score functions for the same purpose. We suggest a new loss function for our problem settings. Softmax-based triplet loss with cosine similarity score function was introduced by Wang et al. (2018). Here, we extend it by integrating the concept of multiple positive sampling (Hermans, Beyer, and Leibe 2017). Note that while the hinge-based triplet loss is also possible, we found that using softmax instead of hinge has better performance and is more stable. Our loss function is defineded as:

$$L = \sum_{x=1}^{Batch} \sum_{y=1}^{Pos} -log[\frac{\exp(\frac{s(\boldsymbol{p}_x, \boldsymbol{q}_y, \boldsymbol{r}_{xy}) - \lambda}{\tau})}{\exp(\frac{s(\boldsymbol{p}_x, \boldsymbol{q}_y, \boldsymbol{r}_{xy}) - \lambda}{\tau}) + \sum_{z=1}^{Neg} \sum_{w=1}^{Pos} \exp(\frac{s(\boldsymbol{p}_x, \boldsymbol{q}_z, \boldsymbol{r}_{xw})}{\tau})}] \quad (2)$$

where λ is the margin that separates the golden pairs and corrupted pairs, τ is a temperature parameter, $Batch$ is the mini-batch size, Pos is the number of positive examples, Neg is the number of negative examples. Note that the score function for negative examples takes the same relational vectors as the positive examples.

Training

Using pre-processed recipes, we train our models in the following steps (Fig. 2): At first, we randomize the order of recipes and their ingredients (Fig. 2 (1)). We then generate sequences of ingredients from recipes (Fig. 2 (2)). After that, we generate pairs of an ingredient set and a candidate ingredient. Pre-selected ingredient sets are selected based on the sequence (see Fig. 2 (3)) – *unordered session data feeding*. We also sample candidate ingredients based on heuristic rules for ingredient pairings – *customizable positive sampling*.

A specified function – a heuristic rule – is used to weight all possible ingredients, and the probability of each ingredient to be sampled is determined by its relative weight. In our experiments here we use two sampling heuristics:

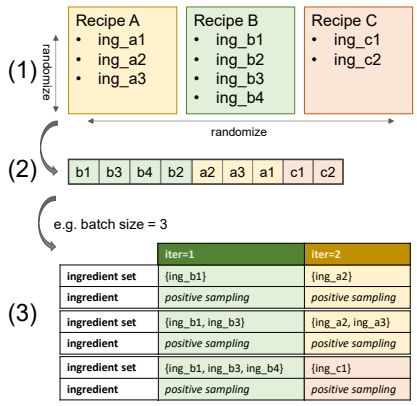

Figure 2: How to generate mini-batches for unordered session data feeding.

Recipe Fit Rule which uses co-occurences of ingredients in recipes to bias sampling. This rule samples positive examples by weighting ingredient pairs higher that frequently occur together in recipes.

Flavor Fit Rule which uses shared flavor compounds between ingredients to bias sampling. This rule samples positive examples by weighting ingredient pairs higher that have a large overlap in flavor compounds.

Finally, we sample negative examples randomly. The negative sampling is biased by the frequency of ingredient occurrence on training recipes.

Results

Evaluating whether ingredient pairs are correct from the perspective of creativity is not trivial since evaluations can change over time with experience and with context. Classic crowdsourcing approaches often used in evaluating recommender systems do not work in the case of ingredient pairing tasks. In prior experiments - we found that while ingredient pairing recommendation systems do stimulate professional chefs, amateur chefs do not find pure ingredient-ingredient suggestions useful as they do not include cooking instruction. In this paper we therefore focus on assessing whether the model can learn to approximate a ground truth score. We use CulinaryDB (Bagler 2017) for this experiment. The dataset consists of 45,772 recipes: lists of ingredients and attributes for 658 ingredients: flavor compounds, cuisines, and ingredient categories. Before training models, recipes are divided into a train, a validation and a test set. Additionally, we generate 172,207 triplets from all ingredients in order to construct a knowledge graph.

We trained two variations of IRRM to evaluate our positive sampling approach proposed to customize the IRRM in the heuristics. The first uses the Recipe Fit Rule as a positive sampling strategy and the second uses the Flavor Fit Rule. Table 1 shows the comparison of the top-10 ingredients with the highest score for all possible ingredients on the CulinaryDB by changing IRRM positive sampling strategies

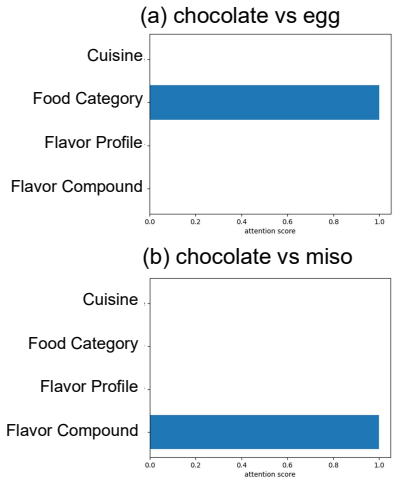

Figure 3: Visualizations of attention weights over ingredient attributes on CulidnaryDB.

for the $\mathcal{I}_{target} = \{orange\}$ as a example. And the results of Flavor Fit Rule is shown as a reference.

The results of the IRRM with the Flavor Fit Rule are intermediate between the IRRM with the Recipe Fit Rule and the pure Flavor Fit Rule. For example, while *welsh onion* and *tomato* come from the recipe rule, *lemon* comes from the flavor rule. Moreover, the correlation coefficient between IRRM with Flavor Fit Rule and pure Flavor Fit Rule was $0.611 (p < 0.001)$ and between IRRM with Recipe Fit Rule and pure Flavor Fit Rule was $0.280 (p < 0.001)$. *orange* is one of flavor effective ingredients. So, we calculated the correlation coefficient for all one-to-one pairs, too. The result between IRRM with Flavor Fit Rule and pure Flavor Fit Rule was $0.298 (p < 0.001)$ and between IRRM with Recipe Fit Rule and pure Flavor Fit Rule was $0.078 (p < 0.001)$. We found, even for all ingredient pairs, the specified rule biases the scores from this result. Consequently, we found our positive sampling approach can effectively customize the IRRM based on specified rules. Even if we use a rule, feeding recipes also affect the results. This means that both the chef's recipes and the specified rules can contribute to the score estimated by the model.

We also analyzed attention weights for confirming interpretability in the trained IRRM for some specific food pairs around chocolate (see Fig. 3). The data shows that *egg* is paired with *chocolate* because of correlations in food category. Whereas, *miso* has considerable flavor compound related affinity to chocolate. This interpretation for *eggs* is consistent with the results reported by De Clercq et al. (2016).

Related Work

Ahn et al. (2011) firstly introduced the flavor network to uncover fundamental principles of food pairing. Using this idea, Garg et al. (2017) developed a rule-based food pairing system. Recently, Park et al. (2019) introduced a Siamese Neural Networks based model trained on a large-scale dataset for food ingredient pairing.

Rank	IRRM Pos. sampling: Recipe Fit Rule		IRRM Pos. sampling: Flavor Fit Rule		Flavor Fit Rule	
	Ingredient	Score	Ingredient	Score	Ingredient	Score
1	butter	1.215	mint	1.234	tea	170
2	water	1.198	welsh onion	1.204	mandarin orange	165
3	sugar	1.198	tomato	1.188	lemon	163
4	welsh onion	1.185	sesame	1.181	apple	153
5	tomato	1.178	parsley	1.180	ginger	151
6	apple cider vinegar	1.173	lemon	1.179	guava	149
7	vinegar	1.168	canola oil	1.170	pepper	148
8	garlic	1.167	poppy seed	1.163	mango	147
9	mustard	1.163	mustard	1.162	black currant	146
10	mint	1.162	rosemary	1.160	laurel	145

Table 1: IRRM comparison based on positive sampling strategies. Top-10 ingredients with the highest score from all ingredient candidates $\mathcal{I}_{candidate}$ on the CulinaryDB for the $\mathcal{I}_{target} = \{orange\}$ are shown. Pos. sampling: Positive sampling strategy.

On the other hand, Morris et al. (2012) firstly suggested Computational Creative System in the culinary domain. They used a model trained by user rating scores on the recipe websites to evaluate generated recipes. And, Pinel and Varshney (2014; 2015) proposed creativity metrics based on Bayesian Surprise and a human flavor perception model. França et al. (2017) suggested the Regent-Dependent Creativity metric that combines novelty and value. They used Bayesian surprise as a novelty metric and Synergy as a value metric. Pini et al. (2019) presented a graph based surprise as a creative metrics using knowledge graph. In this research, we assume there are many possible different reasons for good ingredient combinations via many potential relationships between ingredients and suggest a model to learn creative metrics that are interpretable and customizable.

Conclusion

We have presented a framework for interpretable and customizable food ingredient recommender systems for both one-to-one and many-to-one settings based on recipes. The main feature that distinguishes our work from previous is that ingredient pairing is modeled as a session-based recommendation task with implicit feedback and suggests a training procedure to integrate chef's ideas.

We demonstrated that qualitatively our model can learn interpretable relational representations and detect interesting correlations between ingredients and factors such as flavor compounds. And also, it can be customized by chef's recipes and heuristics. Future work will carry out user studies comparing trained score functions and also assessing the plausibility of visualized attributes for interpretability.

Author Contributions

Author 1 was in charge of writing the manuscript and planning the study, conducted the analysis and developed the a significant part of the tool. Author 2 contributed to the planning of the study and the writing of the manuscript.

Acknowledgments

The authors would like to thank all anonymous reviewers for their helpful comments.

References

Ahn, Y.; Ahnert, S.; Bagrow, J.; and Barabási, A. 2011. Flavor network and the principles of food pairing. *Sci Rep 1, 196*.

Bagler, G. 2017. Culinarydb.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc. 2787–2795.

De Clercq, M.; Stock, M.; De Baets, B.; and Waegeman, W. 2016. Data-driven recipe completion using machine learning methods. *Trends in Food Science & Technology* 49:1–13.

França, C.; Góes, L.; Amorim, A.; and Silva, A. 2017. Creative flavor pairing: Using rdc metric to generate and assess ingredients combinations. 1–8.

Garg, N.; Sethupathy, A.; Tuwani, R.; NK, R.; Dokania, S.; Iyer, A.; Gupta, A.; Agrawal, S.; Singh, N.; Shukla, S.; Kathuria, K.; Badhwar, R.; Kanji, R.; Jain, A.; Kaur, A.; Nagpal, R.; and Bagler, G. 2017. FlavorDB: a database of flavor molecules. *Nucleic Acids Research* 46(D1):D1210–D1216.

Hermans, A.; Beyer, L.; and Leibe, B. 2017. In defense of the triplet loss for person re-identification. *ArXiv* abs/1703.07737.

Huang, J.; Zhao, W. X.; Dou, H.; Wen, J.-R.; and Chang, E. Y. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. *In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* 505–514.ACM.

Morris, R. G.; Burton, S. H.; Bodily, P.; and Ventura, D. 2012. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *ICCC*.

Park, D.; Kim, K.; Park, Y.; Shin, J.; and Kang, J. 2019. Kitchenette: Predicting and ranking food ingredient pairings using siamese neural network. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 5930–5936. International Joint Conferences on Artificial Intelligence Organization.

Pinel, F., and Varshney, L. R. 2014. Computational creativity for culinary recipes. CHI EA '14, 439–442. New York, NY, USA: Association for Computing Machinery.

Pinel, F.; Varshney, L. R.; and Bhattacharjya, D. 2015. *A Culinary Computational Creativity System*. Paris: Atlantis Press. 327–346.

Pini, A.; Hayes, J.; Upton, C.; and Corcoran, M. 2019. Ai inspired recipes: Designing computationally creative food combos. 1–6.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. *In Proceedings of the 2018 World Wide Web Conference (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland* 729–739.

Wang, F.; Cheng, J.; Liu, W.; and Liu, H. 2018. Additive margin softmax for face verification. *IEEE Signal Processing Letters* 25(7):926–930.