Recipe Completion Using Machine Learning Techniques

Lingzhen Chen
Department of Information Engineering And Computer Science
Universit degli Studi di Trento
lingzhen.chen@studenti.unitn.it

Abstract

In recent years, recipe related collaborative knowledge sharing platform have increasingly impacted the way individuals gain access to their cooking guidance. Moreover, there is a variety of available websites and applications providing recipe recommendation services, which are mainly achieved by content based recommender system that takes advantage of user preferences and profiles. The purpose of this paper is to develop an advanced recipe completion system in order to provide a solution in the scenario where a user is in possession of a numbers of ingredients, and needs a possible additional ingredient recommendation to complete a recipe. The process of representing ingredients by word embeddings are introduced, as well as how we incorporate their contextualized features from recipes. Several models are proposed and tested. Models are evaluated on their ability to retrieve a missing ingredient in the test recipe and their performance are compared. Results reveal that the proposed models with proper configurations are able to perform well on the task.

1 Introduction

Recipe sharing websites and applications have freed people from the cumbersome process of searching printed recipes in a cookbook. Especially in the past several decades, with a booming amount of recipe data being updated online through various platforms, and the development of sophisticated recommendation modelling approaches, it became ever-growingly feasible to incorporate abundant recipe data to provide a better recipe search or recommendation service. Not only did the recipe recommendation systems enriched the source which one obtains his or her cooking instructions, but also improved the efficiency of collaborative knowledge sharing. Tremendous endeavour has been put into design a more comprehensive computational model for feature representation of recipes or ingredients, as well as to develop a more robust recipe recommendation system.

Many existing recipe recommendation system are based on building user profiles, collecting information about their preferences and feedbacks in order to make prediction. It often requires user to know the name of a recipe in order to perform a search, or it would recommend a similar recipe to which a user had tried or liked. However, these recommendation systems are not solution in a very common scenario in life, that is, when a user already have some ingredients at hand, but does not know what he or her can do with them. Thus, one may wonder if there is a possibility to do perform recommendation not by user profiles as we discussed, but by discovering the connections and relations among ingredients in recipes. Assuming that a user has some ingredient available in his or her refrigerator without knowing what kind of dish these ingredients can form, an interesting solution to the problem would be a system that takes a list of ingredients as input, decide if there is a possibility that by adding one or more ingredients, they would make a reasonable recipe. To be able to perform such a prediction, the system needs to incorporate the information of an ingredient such as its food category, flavour components, or its occurrence time in the data set to generate a feature representation of the ingredient. It is also necessary to have a comprehensive template that represents a recipe by the ingredients that it contains. Moreover, an optimization function is the key to solve the recipe recommendation task. This paper demonstrates the procedure and experiments of developing such recipe recommendation sys-

tem. We discuss in detail the data set, the consideration and design of our model, the mathematical fundamentals, extensive experimental results and a throughout evaluation of the performance.

The rest of the paper is organized as follows. Section 2 explores the related work. Section 3 analyses the data set that is used and introduces the pre-processing steps that is carried out. Section 4 illustrates a detailed description of the model, followed by the presentation of implementation and experiment results in Section 5 and Section 6. Last but not the least, final conclusions are drawn in Section 7.

2 Related Work

Recipe recommendation has been a subject of may prior work. Typically, system developers have tried to perform recipe recommendation based on user profiles, such as their ratings on recipes in the past (Forbes et al., 2011) or their browsing history (Ueda et al., 2011). The algorithm is designed to find similar recipe based on the ingredients in common, either treating each ingredient equally (Freyne et al., 2010) or identifying major ingredient (Zhang et al., 2008). (Wang et al., 2008) tried to represent recipes as graphs that are based on ingredients and cooking instructions. (Teng et al., 2012) also tried to construct ingredient network incorporating the likelihood of ingredients to co-occur, as well as the potential of one ingredient to act as a substitute for another. (Clercq et al., 2016) employed machine learning methods, to be more specific, non-negative matrix factorization method and two-step regularized least squares method to perform recipe completion task. (Forbes et al., 2011) also implemented a content-boosted matrix factorization algorithm which incorporates content information directly as a linear constraint. Our solutions aim to solve a problem that is similar to which they intended to tackle by non-negative matrix factorization method in (Clercq et al., 2016).

3 Dataset

The dataset used in this report consists of two files, recipes.csv and ingredients.csv, which are both taken from (Ahn et al., 2011). The recipe file contains 56,498 recipes originating from 11 different cuisines. For each recipe, the geographical category and ingredient are listed. In the recipe file, there are in total 381 different ingredients, which forms the ingredient file together with their food category.

3.1 Data Preprocessing

Before using the dataset, some pre-precessing on the data are done in order to filter out uncommon examples in the dataset, as well as to obtain more unbiased results in later experiments. For starters, the geographical information, i.e. the first column in the recipe file is removed. And the recipes with less than 3 ingredients are deleted from the recipe file. The new file (*recipe_as_input.txt*), which will be the input for training embeddings of ingredients, consists of 55,001 entries. This modification does not change the fact that there are 381 different ingredients appear in the recipe file. However in order to increase the confidence of trained vector. Ingredients with occurrence less than 75 times are removed from the ingredient file. The new ingredient file (*ingredients_major.txt*), consists of 248 unique entries of ingredients. These ingredients still covers 52,375 recipes (more than 95% percent of the original recipe file). In later sections where the experiment settings are introduced, it would be clear that this occurrence restriction is also applied while calculating vector representations, by specifying the parameter $min - count$ (minimum occurrence of the ingredient in training file) as 75.

With these two files, we are able to generate matrices that represents the relationships between recipes and ingredients. A binary matrix M is calculated with shape $(m \times n)$, where m indicates the number of recipes and n indicates the number of ingredients. This matrix serves as an one-hot encoding of data to show which ingredients a recipe contains. The index in the matrix preserves the order recipes and ingredients appear in the data files. And its transpose, M^T is also useful when the information about which recipes that an ingredients appears in is needed. These two matrices are saved as *reci_ingr_matrix.txt* and *ingr_reci_matrix.txt*. Moreover, we obtain the co-occurrence matrix by calculating $R = M^T \times M$ and the common ingredient matrix by calculating $C = M \times M^T$. They are both integer matrix where R_{ij} indicates the number of times ingredient i and ingredient j occur together in one recipe, and C_{ij} indicates the number of common ingredients that recipe i and recipe j have. These matrices are saved as *co_occur_matrix.txt* and *common_ingr_marix.txt*.

3.2 Data Distribution

In our data file, 248 ingredients comes from 13 different food categories, among which the category 'vegetable' covers 51 (20.6%), while 'animal_product' and 'plant' has the lowest coverage, with both only 4 ingredients. The minimum number of ingredients in a a recipe is 3, while the maximum is 32. The average length of recipe is 8.

4 Methodology

In this experiment, we aim to develop an advanced recipe recommendation system that, according to an incomplete recipe input, can suggest one or more ingredient to complete it. The suggestion is computed by calculating the similarity scores between potential target ingredients and the incomplete input recipe (i.e. the recipe template).

In order to achieve this goal. There are couple issues that we need to tackle. Firstly, a suitable feature representation of ingredients and recipes data is needed. Secondly, a model for recovering the missing ingredient in the recipe needs to be developed. Last but not the least, a series of performance measures need to be decided for evaluating the feature representation quality and the model performance.

In this section, fundamentals about the model the mathematical principles behind the model are described in detail.

4.1 Vector Representation

For starters, in order to capture the characteristics of an ingredient, it is necessary to chose a suitable representation method for the ingredients in the data set. As discussed in previous sections, on the one hand, the characteristics of an ingredient are indicated by its own properties such as its total occurrence frequency in the data set and its similar ingredients. On the other hand, the characteristics are also manifested by its relationship with other ingredients, that is to say, whether an ingredient tends to co-occur often with some other ingredients or not. In order to predict if one ingredient is potentially a fit for the input recipe template, the similarity between the context of a recipe template and the set of ingredients that the target ingredient often co-occur with plays an important role.

Thus in order to capture these features of an ingredient, we exploit the ability of the tool Word2Vec to generate distributed representation of words and phrase. The vector representation generated by this tool is called word embeddings, which is shown to carry semantic meanings and are very useful in various natural language processing tasks. As documented in the paper (Mikolov et al., 2013), they introduced a task that given a word pair (a, b) and a word c, the task aims to find a word d, such that the relation between c and d is similar to that between a and b. Word2vec tool provides an efficient implementation of the continues bag-of-word(CBOW) and skip-gram architectures for computing vector representations of words. Intuitively, the vectors of similar ingredients would have shorter distance in the vector space, compared to the those that are not similar to each other. Hence the similar ingredient points would tend to cluster in the vector space.

There are a number of hyper-parameters that are needed to be decided for the training process. And for deciding which vector would be the best for our model, they are evaluated by maximizing average similarity metrics. These aspects are discussed in the later section of the paper.

4.2 Similarity Measures

The similarity between ingredients plays a crucial role in predicting the fitness of a target ingredient to a recipe template. While using Word2Vec, the similarity can be calculated by the cosine distance of vectors. In addition, the contextual similarity of an ingredient is also taken into consideration by calculating raw and normalized similarity metric in terms of recipes they are in.

Cosine Similarity and Angular Similarity

First and foremost, an intuitive way to compare the similarity between vectors are by comparing there distance in the vector space. There are a variety of distance metrics that we can choose from. In this paper, we use cosine distance due to its ability to capture the similarity in vectors regardless of the magnitude of the elements in the vectors.

For ingredient vector A and B, their cosine similarity is calculated by formula

$$cosine\_similarity(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

where n denotes the dimension of the vector. It is worth mentioning that if the vectors contain negative values, then the cosine similarity is bounded between $[-1, 1]$.

In order to obtain a similarity metric that is bounded between [0,1], transformation needs to be made from cosine similarity to angular similarity by following formula

$$angular\_similarity(A, B) = 1 - \frac{\cos^{-1}(cosine\_similarity(A, B))}{\pi} \quad (2)$$

where the $cosine\_similarity(A, B)$ is what defined by formula (1). In this case, the value 0 indicates absolute dissimilarity between vectors and 1 means two vectors are identical.

Raw Similarity and Normalized Similarity

This similarity measure is designed to capture the context information of an ingredient in terms of recipes that it is in. The binary recipe vector I_R (248 x 1) is represented in one-hot encoding indicating which ingredients it contains. For ingredient i and ingredient j, the sets of recipes contains them are represented by I and J. The similarity between ingredient i and ingredient j is given by formula

$$sim(i, j) = \frac{1}{n_I}\frac{1}{n_J}\sum_{I_{R_i}\in I}\sum_{I_{R_j}\in J} < I_{R_i}, I_{R_j} > \quad (3)$$

Intuitively, it is summing up the value of pairwise dot product of the recipe that contains ingredient i and the recipe that contains ingredient j between all the pairs in I and J. Then this value is divided by the scalar product of numbers of elements in I and J (i.e. the number of recipes contains ingredient i multiply that contains ingredient j). It is the average number of common ingredients between ingredient i's context (i.e. the recipes contain i) and ingredient j's context. The value of this similarity metric ranges from lower limit 0, which means that the contextual recipes share no common ingredient, to the upper limit– the maximum length of recipe, which means the contextual recipes are identical.

It is obvious that the upper limit varies according to the length of recipes, which could introduce some biased interpretation of our observation. Hence, in order to make it more evident and make the comparison between this similarity measure and other similarity measure easier, it is useful to normalize the result to obtain a score bounded in [0,1]. The normalized similarity measure is given by formula

$$sim(i, j) = \frac{\sum_{I_{R_i}\in I}\sum_{I_{R_j}\in J}\frac{2\cdot|I_{R_i}\cap I_{R_j}|}{|I_{R_i}|+|I_{R_j}|}}{|I|\cdot|J|} \quad (4)$$

where a recipe contains ingredient i is treated as a set of ingredients I_{R_i} while that contains ingredient j is represented by I_{R_j}. $|I_{R_i}|$ denotes the length of a recipe that contains ingredient i and $|I|$ represents the number of all recipes that contains ingredient i.

4.3 Optimization Model

The next step would be to design a model that can predict if a target ingredient is a fit for the input recipe template. It is achieved by calculating the profit score of an ingredient with regard to the recipe template, the scores of all the candidate ingredients are ranked and the top n ingredients with the highest scores are chosen. The profit between the ingredient and recipe template can be calculated by formula:

$$\begin{aligned}\text{profit}(T, i) = &\frac{\sum_{j\in T\setminus\{i\}}\text{cooccur}(i, j)}{|T\setminus\{i\}|} \\ &- \frac{\sum_{j,k\in T}\text{cooccur}(j, k)}{\binom{|T|}{2}} \\ &+ \frac{\max_{j\in T}\text{context}(i, j)}{2} \\ &+ \frac{\min_{j\in T}\text{context}(i, j)}{2}\end{aligned} \quad (5)$$

where T is the template and i is the target ingredient. The function cooccur(i, j) is the co-occurrence score between two ingredients in the dataset, which is given by the normalized PMI (pairwise mutual information) that is calculated by formula

$$\text{npmi}(x, y) = \frac{\text{pmi}(x, y)}{-\log p(x, y)} \quad (6)$$

where

$$\begin{aligned}\text{pmi}(x, y) &= \log\frac{p(x, y)}{p(x)p(y)} \\ &= \log p(x, y) - \log(p(x)p(y))\end{aligned} \quad (7)$$

in which

$$p(x) = \frac{N_x}{N} \quad (8)$$

where N_x is the number of recipes containing ingredient i and N is the total number of recipes. And

$$p(x, y) = \frac{N_{xy}}{N} \tag{9}$$

in which N_{xy} is the number of recipes containing both ingredient x and ingredient y.

The value of npmi(x, y) ranges in $[-1, 1]$. Similar to the processing of similarity metrics described in the previews section, we make this value bounded within $[0, 1]$ by adding one and dividing by 2, which gives us:

$$\text{npmi}_{0\text{–}1}(x, y) = \frac{\log(p(x)p(y))}{2 \log p(x, y)} \tag{10}$$

In formula (5), the function context(i, j) is the similarity function between ingredient that takes into account the context, which is calculated by formula (2) or formula (4), depending on which context information to be used.

4.4 Evaluation Metrics

The model produces a list of predicted ingredients sorted by the profit score as output. Thus for evaluating the performance of our model, following 5 metrics can be used as a reference:

1) Mean position of the eliminated ingredient in the ordered list of suggested ingredients.

2) Median position of the eliminated ingredient in the ordered list of suggested ingredients:

3) Percentage of test recipes for which the eliminated ingredient is located in the top 10 of suggested ingredients.

4) Percentage of the eliminated ingredient being predicted at the first rank.

5) The mean AUC. The AUC is defined as the area under the ROC curve. Since there is only one positive observation, the AUC is directly related to the rank of the eliminated ingredient and can be computed as:

$$AUC = 1 - (\frac{rank - 1}{N^-}) = \frac{N^- + 1 - rank}{N^-} \tag{11}$$

with N^- the number of negative observations and rank the position of the eliminated ingredient in the ordered list of suggested ingredients.

5 Implementation

The implementation of the experiments follows the step described in the previous chapter. Firstly, the vectors of ingredients are trained with different sets of hyper-parameters. After analysing the vectors according to the average similarity score they have in formed clusters, the best vector representation is chosen for calculating contextual information of ingredients with regard to recipe. The contextual information will also be calculated according to the normalized similarity discussed in the previous chapter. Apart from the Word2Vec tool, the rest of the experiments are implemented in Python.

5.1 Word2Vec Parameter Settings

First of all, experiments are done with fine-tuning the hyper-parameter of the vector training by Word2Vec. In order to find a optimal combination of hyper-parameters, coordinate descent method is used. It is a non-derivative optimization algorithm. Line search is done along one coordinate direction at current point in each iteration, and different coordinate directions are used cyclically through the procedure to find the local optimal of a function. In our experiment, it means optimizing one hyper-parameter at a time, until we find an optimal value. And then this hyper-parameter is set to its optimal value before we start to fine-tune the next one. Some critical parameters are $size$, $negative$, $window$, $cbow$, hs, $sample$, and $min - count$.

$-cbow$ indicates if continues bag of word model (-cbow 1) or skip gram model (-cbow 0) is used; from literature, skip-gram model works better with low frequency words, but overall these two model should generate similar results.

$-hs$ indicates if hierarchical softmax is used or not.

$-size$ indicates the dimension of vector.

$-negative$ indicates the number of negative samples that is used during training

$-window$ indicates the context window size surrounding a word (i.e. ingredient) that is considered during training. This value is set to maximum of total ingredient number in recipe –32.

$-sample$ mainly affects the speed of training but not much the result.

$-min - count$ the minimum times a word occur in the data set for it to be considered in the vocabulary.

In addition, since the vectors need to be clustered for deciding the one with best quality, the number of clusters for clustering algorithm also needs to be decided.

Hence the hyper-parameters for tuning come down to $size$, $negative$, $cbow$, hs and $n\_clusters$. The range of choices for the experiments are shown below in Table 5.1.

Parameter	Range of choices
size	{5,10,15,20,...,95,100}
negative	{1,2,3,...,20}
cbow	{0,1}
hs	{0,1}
n_clusters	{3,4,5,6...,13}

Table 5.1: Hyper-parameter Range

5.2 Vector Quality Analysis

Hierarchical clustering is performed on the vector files obtained from the previous step and average similarities in each cluster is calculated. Hierarchical clustering method is chosen due to the convenience it provides for understanding the clustering decision and visualizing clustered data. As an example dendrogram shown below in Figure 5.1, the distances between vectors and the threshold of clustering is evident, which helps the adjustment of parameter for clustering algorithm. $AgglomerativeClusting()$ implemented by $sklearn$ is used. It recursively merges the pair of clusters that minimally increases a given linkage distance. The parameter $n\_clusters$ is set to the best performing hyper-parameter obtained from the previous step while $affinity$ is set to $'cosine'$ for clustering according to vector's cosine distance. Theoretically, since the ingredient vectors with smaller distance are clustered together, the ingredients in one cluster should share larger similarity, thus higher average similarity score.

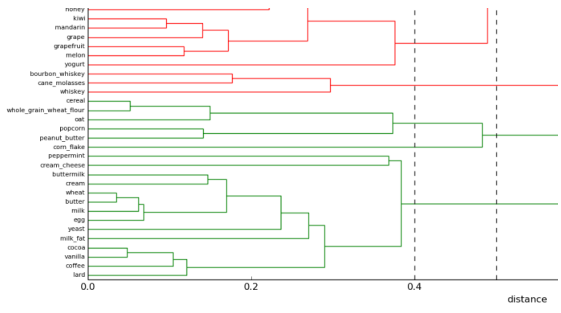


Figure 5.1: The Dendrogram of Hierarchical Clustering Result

Procedures described above would select the vector file which best represents the ingredient data according to our similarity metric. And in the later part of the experiments, the selected vector file is used to calculate the contextual information of the ingredient with regard to the recipe.

5.3 Optimization parameter settings

The realization of the optimization model exploits the contextual similarity metric that is discussed in **Section 4.2** and the cosine similarity of the ingredient vectors. For both similarity metrics, a context file is generated to describe the similarity scores between ingredients with the format on each line as:

ingredient_name_1, ingredient_name_2, similarity_score

According to the context file, a profit file is then generated to describe the profit each ingredient would bring to the recipe templates with format:

recipe_index, ingredient_name, profit_score

Furthermore, another profit file is generated to demonstrate the mean profit of each recipe with the format:

recipe_index, mean_profit_score

We choose to incorporate the contextual information with the co-occurrence information of ingredients for predicting the target ingredient. In the meantime, we also intend to discover the contribution of co-occurrence information in terms of making a correct prediction. Hence we design the configurations of experiments as listed below in Table 5.2.

Experiment No.	Configuration Settings
1	contexts with co-occurrence
2	contexts only
3	vector contexts with co-occurrence
4	vector contexts only

Table 5.2: Experiment settings for different models

The test data for this experiment, which contains 5,500 entries of recipes (10% of total data set), is randomly sampled from original data set. For each recipe entry, a random ingredient is removed and then the rest of the ingredients form a recipe template as input to our model. According to the recipe template, the model suggests a list of possible ingredients to complete the recipe. In the ideal case, the removed ingredient would ob-

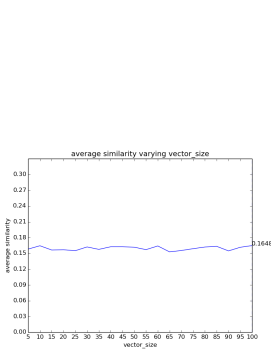
tain the highest profit score and hence the smallest rank, which is rank one. Then some metrics listed in **Section 4.4** will be calculated to evaluated the model performance of different configurations.

6 Results

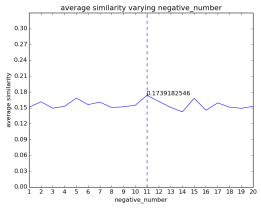
In this section, results of the experiments discussed above are presented and discussed.

6.1 Hyper-parameter Selection

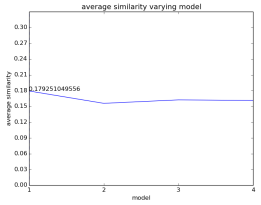
As illustrated in **Section 5.1**, a range of values of different hyper-parameters are tried before model developing. Some critical parameters that we experimented on are $size$, $negative$, $cbow$ and hs. The results of training ingredient vectors with different hyper-parameter set can be seen below in Figure 6.1. The horizontal coordinate represents the range of parameters, while the vertical coordination denotes the average similarity scores among clusters. In Figure 6.1c, the four configuration indicates the four combination of binary parameter $cbow$ and hs.



(a) Hyper-parameter : size



(b) Hyper-parameter : negative number



(c) Hyper-parameter : cbow & hs

Figure 6.1: Average Similarity variation according to different hyper-parameter settings

As can be seen from the graphs, the best hyper-parameter settings are $size = 100$, $negative\_number = 11$, $cbow = 0$ and $hs = 0$. From this setting, the vector file is obtained.

6.2 Similarity Metrics

The contextual similarity of ingredients and the cosine similarity of ingredient vectors are calculated. As plotted in Figure 6.2 the distribution of normalized contextual similarity among 30,504 ingredient pairs, it is evident that the distribution is dense, mainly ranging from 0 to 0.6.

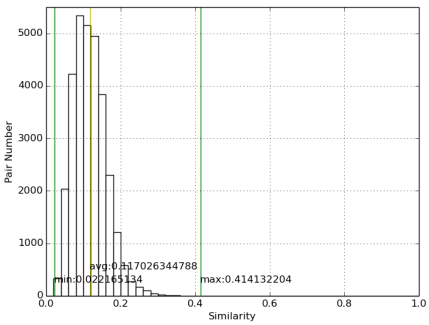


Figure 6.2: The distribution of normalized contextual similarity scores among all ingredient pairs

And on the other hand, the distribution of angular similarity among ingredient vector pairs is demonstrated in Figure 6.3. As can be seen from the graph, the cosine similarity is also dense but within a different range – $[0.4, 1.0]$.

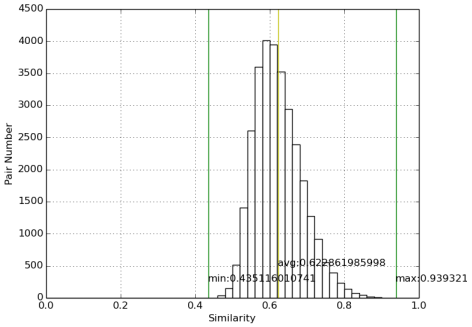


Figure 6.3: The distribution of angular similarity scores among all ingredient pairs

It is worth mentioning that the normalized contextual similarity distribution centralized in small number (with average at 0.117). It may be due to the fact that it is common for different recipes to contain various combinations of ingredients. Hence in terms of an ingredient pair, their context is usually different, hence the generally low similarity score. While the distribution of the cosine similarity of ingredient vectors are different from that of contextual similarity, they share a similar shape dense shape of distribution. Furthermore, as suggested in Figure 6.4, the average contextual similarity score in each food category is higher than the average among all ingredient pairs except for fruit and alcoholic beverage. It implies that the ingredients within the same food category may share similar context in a recipe.

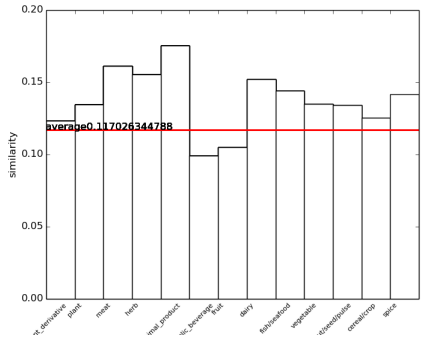


Figure 6.4: The average similarity in each food category

6.3 Model Performance

As introduced in **Section 5.3**, some configurations of optimization model is tested to see the amount of contribution of contextual information and co-occurrence information to our prediction function. And the result of experiment are shown below in Table 6.1 with the best performance of each metric being highlighted in bold font.

Experiment No.	1	2	3	4
Mean Rank	**24.6**	39.6	32.7	62.2
Median Rank	**10.0**	21.0	17.0	50.0
Rank<=10 (%)	**51.9%**	33.0%	40.0%	10.0%
Rank=1 (%)	**9.7%**	3.4%	6.3%	1.0%
Mean AUC	**0.904**	0.844	0.872	0.752

Table 6.1: Model performance with different configurations

From the results, it is clear that the best performance yields when using the contextual information with co-occurrence information. In general, using vector context information produces a worse result compared to using contextual information. Incorporating co-occurrence information improves the performance no matter which contexts file that we are using. Model with vector context information sees a even more drastic improvement in performance by taking the co-occurrence information into consideration. In terms of the mean and median rank in prediction of the missing ingredient, by taking co-occurrence information into consideration, the ranks are improved by more than 10 to the top. And the number of rank within 10 also increased nearly 20%. And the correct prediction of the model also increased 6.3%. In summary, both contextual information and co-occurrence information are important when making a prediction.

7 Conclusions

This paper proposed a model for suggesting ingredient to complete recipe template. Experiments are carried out for fine-tuning hyper-parameters while training ingredient vectors. Ingredients are compared with different similarity metrics. A carefully designed objective function is used for incorporating the contextual information and co-occurrence information of ingredients. Different model configurations are compared. Results indicates that both contextual information and co-occurrence information contributes to the prediction. A more comprehensive and flexible model for suggesting more than one ingredient for the recipe template or taking into consideration user's preferences will be part of the future work.

References

Forbes, P., and Zhu, M. 2011 Content-boosted matrix factorization for recommender systems: Experiments with recipe recommendation. *Proceedings of Recommender Systems* (2011).

Ueda, M., Takahata, M., and Nakajima, S. 2011 Users food preference extraction for personalized cooking recipe recommendation. *Proc. of the Second Workshop on Semantic Personalized Information Management: Retrieval and Recommendation* (2011)

Freyne, J., and Berkovsky, S. 2010 Intelligent food planning: personalized recipe recommendation. In *IUI,ACM* (2010), 321324.

Zhang, Q., Hu, R., Mac Namee, B., and Delany, S. 2008 Back to the future: Knowledge light case base cookery. In *Proc. of The 9th European Conference on Case-Based Reasoning Workshop* (2008), 15.

Wang, L., Li, Q., Li, N., Dong, G., and Yang, Y. 2008 Substructure similarity measurement in Chinese recipes. In *WWW, ACM* (2008), 979988.

Chun-Yuen Teng , Yu-Ru Lin , Lada A. Adamic. 2012 Recipe recommendation using ingredient networks. In *Proceedings of the 3rd Annual ACM Web Science Conference*, p.298-307, June 22-24, 2012

Marlies De Clercq, Michiel Stock, Bernard De Baets, Willem Waegeman. 2016 In *Data-driven recipe completion using machine learning methods. Trends in Food Science & Technology*, 49(2016) 1–13

Ahn, Y.Y., Ahnert, S. E., Bagrow, J. P., & Barabasi, A. L. 2011 Flavor network and the principles of food pairing. In *Scientific Reports* 1.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013 In *Distributed representations of words and phrases and their compositionality*. *Advances in Neural Information Processing Systems*, 26:3111–3119.