PlateClick: Bootstrapping Food Preferences Through an Adaptive Visual Interface

Longqi Yang[†,‡], Yin Cui[†,‡], Fan Zhang[†,‡], John P. Pollak[‡], Serge Belongie[†,‡], Deborah Estrin[†,‡]
[†]Department of Computer Science, Cornell University; [‡]Cornell Tech
{ylongqi, ycui, fanz}@cs.cornell.edu; {jpp9, sjb344, de226}@cornell.edu

ABSTRACT

Food preference learning is an important component of wellness applications and restaurant recommender systems as it provides personalized information for effective food targeting and suggestions. However, existing systems require some form of food journaling to create a historical record of an individual's meal selections. In addition, current interfaces for food or restaurant preference elicitation rely extensively on text-based descriptions and rating methods, which can impose high cognitive load, thereby hampering wide adoption.

In this paper, we propose *PlateClick*, a novel system that bootstraps food preference using a simple, visual quiz-based user interface. We leverage a pairwise comparison approach with only visual content. Using over 10,028 recipes collected from Yummly, we design a deep convolutional neural network (CNN) to learn the similarity distance metric between food images. Our model is shown to outperform state-of-the-art CNN by 4 times in terms of mean Average Precision. We explore a novel online learning framework that is suitable for learning users' preferences across a large scale dataset based on a small number of interactions (≤ 15). Our online learning approach balances exploitation-exploration and takes advantage of food similarities using preference-propagation in locally connected graphs.

We evaluated our system in a field study of *227* anonymous users. The results demonstrate that our method outperforms other baselines by a significant margin, and the learning process can be completed in less than one minute. In summary, *PlateClick* provides a light-weight, immersive user experience for efficient food preference elicitation.

Categories and Subject Descriptors

H.4.m [**Information Systems Applications**]: Miscellaneous; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Food preference elicitation; visual interface; online learning

CIKM '15 19-23 October, 2015 Melbourne, Australia

DOI: http://dx.doi.org/10.1145/2806416.2806544.

1. INTRODUCTION

The problem of capturing and understanding people's food preferences has attracted substantial attention from industry (e.g., Yelp and Foursquare) and academia [12, 13, 17, 38]. Food preferences guide our diet choices [32] which in turn have a strong effect on our personal health and social lives [32]. A recipe advice system [13, 27] could more effectively coach users to prepare healthier meals at home if alternative food suggestions provided were appealing to them. This is important because healthy diet recommendations are of no benefit if users don't adopt them. Another application area that could leverage user preferences is commercial restaurant recommender systems like Yelp and Foursquare. The recommendations will be more accurate and personalized if the system output is tuned to the user's diet profile. However, food preference is notoriously difficult to learn because of its dependence on context (e.g., evolving personal goals). Existing systems and algorithms suffer from several limitations that interfere with efficient learning and wide user adoption:

Data sparsity. Current food preference learning systems require longitudinal records of the meals that users have eaten [12, 13, 17, 38]. These historical data traces [11] typically come from location sensing, which is not always related to food preference, or burdensome food journaling, which is often abandoned after short periods of adoption [9]. As a result, data points are too sparse to provide enough food preference information.

High cognitive load. Preference elicitation [29], in which users are asked to rate different food items explicitly (on a scale from 1 to 5) [12, 17], is an approach that is complementary to the above longitudinal methods. While text-based instruments and rating methods [5, 10, 35, 43] effectively address the *cold-start* problem [23, 34, 42] in movie recommender systems, the counterpart for food preference is especially hard for users [9] as it is time consuming [28] and presents a high cognitive load [8].

Insufficient Visual Understanding. Traditional methods mainly use food tags and other metadata for learning tasks [12,17]. However, as people's diet decisions are greatly influenced by visual appearances of meals [9], analysis of image features can provide a valuable signal for diet profiling and food preference elicitation.

In this paper, we propose ***PlateClick*, a novel system for efficient food preference elicitation using a simple, visual quiz-based user interface**. *PlateClick* alleviates the limitations mentioned above with deep understanding of food images and a user-friendly visual interface. To

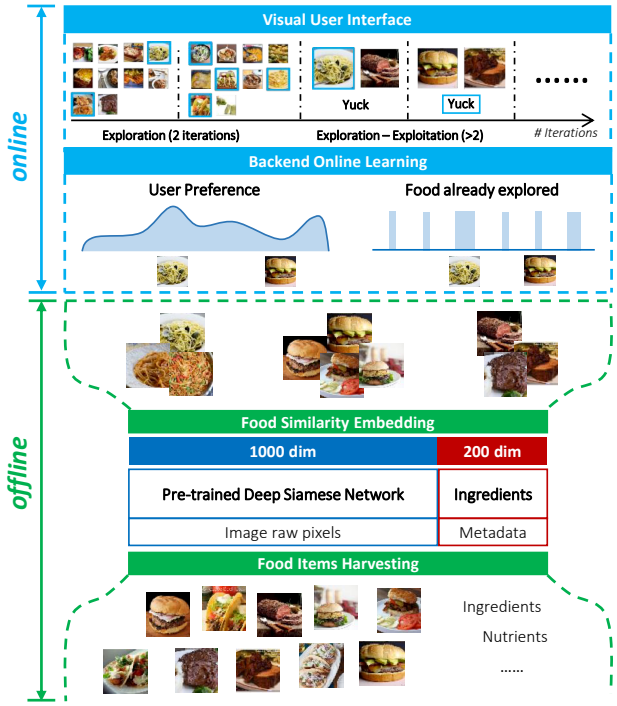


Figure 1: *PlateClick* system pipeline. The system is divided into two major parts: *Offline preprocessing*, which is surrounded by *green dotted line* and *Online preference learning*, which is surrounded by *blue dotted line*.

the best of our knowledge, this is the first system and algorithm that learns users' food preferences through real-time interactions without any requirements of diet history. We developed this system as a lightweight, easily accessible web service that can be completed within 60 seconds. Through a field study with *227* anonymous users in the wild, we show that our system is able to predict the food items that a user likes/dislikes with high accuracy. The system pipeline of *PlateClick* is shown in Fig. 1, which consists of offline and online stages with several components, as follows.

Food Items Harvesting. We pulled 12,000 main dish recipes with their images and metadata (ingredients, nutrients, tastes, etc.) via the Yummly API [1] and filter out image outliers. The final dataset has 10,028 food items across various cuisines (American, Asian, Mexican, Italian, etc.)

Food Similarity Embedding. In order to tame a large number of food items and facilitate image understanding, we learn the image similarity distance metric based on labels from the Food-101 dataset [4] using a deep Siamese network [7]. With the trained network, we extract *1,000 dim* visual features for each food image. The method we propose improves the performance of other state-of-the-art visual feature extraction approaches. We append *200 dim* ingredients features to visual features, resulting in an embedding of the food items into a *1,200 dim* space in which similar items are nearby one another and dissimilar ones are farther away.

Visual User Interface. The process of our food preference elicitation consists of several iterations. We explore the advantages of image picking and pairwise comparison in interface design, both of which offer a potentially improved user experience [8,28]. In each of the first two iterations, we present ten images and users are asked to tap on *all the ones they like*. In each subsequent iteration, we present a pair of food images and ask users either to tap on *whichever they prefer* or click *yuck*, indicating a preference for neither.

Backend Online Learning. The backend of our system consists of a novel online learning algorithm that explores the similarity between food items. Our algorithm is inspired by label propagation [44] in locally connected graphs and the Exponentiated Gradient Algorithm for bandit settings (EXP3) [2]. We demonstrate that this algorithm is more effective in our proposed workflow than other baselines.

Compared to traditional food preference learning systems, our work offers 3 major contributions and points of novelty.

- We get rid of the requirements for users' historical diet records and completely bootstrap food preference from scratch. This design enables *context aware preference learning* that adapts preference information in various conditions.
- We propose a novel image similarity measure that significantly outperforms state-of-the-art algorithms. By leveraging an improved embedding of food items, we simplify the UI by making it completely image based. This offers opportunities for personal interpretation [28] and can provide an immersive experience with personalized, adaptive information [9].
- We design a novel online learning algorithm that can support real-time elicitation of food preference from modest number (≤ 15) of pairwise comparisons on a large scale food image database ($> 10,000$ instances). The pairwise comparison method is known to have lower cognitive load [15], and our system is thus more user-friendly.

We envision that *PlateClick*, a light-weight and efficient food preference elicitation system, will provide a personalized user experience capable of fueling a wide range of applications in domains including health care, diet planning, and restaurant recommendation.

2. RELATED WORK

Collaborative Filtering. As one of the most popular algorithms adopted in current recommender systems, collaborative filtering (CF) [18] has been widely studied in a variety of applications. The main idea of this method is to predict and learn a user's preferences based on similarity measures such as user-based CF [18] and item-based CF [33]. It has also been shown that latent factor models [21] and matrix factorization [31] are promising to predict users' ratings for previously unobserved items.

A major limitation of collaborative filtering is its requirement for historical user data. Although several techniques have been proposed to address the cold-start problem [23, 42], the performance of CF is still largely dependent on the number of active users, availability of contextual information [42] and observed ratings for different items [23]. In the case of food preference learning, it's typically difficult to get access to a user's diet history since meal journaling is burdensome [9]. Therefore, in the design of *PlateClick*, we don't assume any prior knowledge of the users.

Preference elicitation. To alleviate the cold-start problem mentioned above, several models of *preference elicitation* have been proposed in recent years. The most prevalent method of elicitation is to train decision trees to poll users

[1]http://developer.yummly.com

in a structured fashion [10,14,30,35,45]. These questions are either generated in advance and remain static [30] or change dynamically based on real-time user feedback [10, 14, 35, 45]. In addition, another previous work explores the possibility of eliciting item ratings directly from the user [5, 43]. This process can either be carried at item-level [43] or within-category (e.g., movies) [5].

The preference elicitation methods we mentioned above largely focus on the domain of movie recommendations [5, 30, 35, 43] and visual commerce [10] (e.g., cars, cameras) where items can be categorized based on readily available metadata. When it comes to real dishes, however, categorical data (e.g., cuisines) and other associated information (e.g., cooking time) possess a much weaker connection to a user's food preferences. Through the design of *PlateClick*, we leverage the visual representation of each meal so as to better capture the process through which people make diet decisions.

Food preference learning system. Most existing food preference learning approaches are hybrids of historical record mining and rating elicitation [12,13,17,38,40]. To avail oneself of these systems to promote healthy eating, one is often required to record daily meal intake and provide this information as a bootstrapping resource to a diet recommender system [12,13,38]. After that, several food items are selected for display based on matching scores between meal metadata and the user's previous choices. For each item provided, the user is prompted to enter a rating on a scale from 1 to 5.

To the best of our knowledge, no existing systems take visual features – arguably one of the most important factors in assisting people's daily food choices [9] – into consideration. Additionally, the most common methods adopted in food preference elicitation (i.e. text based interface and numerical rating scale) impose a high cognitive load on the user [8] and are susceptible to noisy and unreliable responses. Inspired by elicitation strategies in other domains (e.g. crowdsourcing [6], housing [15]), we propose a simplified, purely visual interface that presents users with simple pairwise comparisons. Through our field study with anonymous users, we show that this lightweight interface can promote efficient food preference learning.

3. METADATA PREPROCESSING

For 12,000 main dishes recipes pulled from Yummly API, we filter out entries with unrelated (or missing) image content, resulting in a final dataset $\mathcal{S}$ containing 10,028 food items, i.e., $\mathcal{S} = \{s_1, s_2, ..., s_{10028}\}$. As illustrated in Fig.1, apart from visual features, we append *200 dim* ingredient features as the representation of each food image. The ingredient feature vector is calculated according to the following pre-processing steps:

1. *Keyword extraction and lemmatization.* For each ingredient appearing in the metadata, we extract keywords from its description and apply lemmatization; see Table 1.

2. *Aggregation and Filtering.* We aggregate and count the occurrences of each ingredient appearing in our dataset. We select top 200 most frequent ingredients [2] as our list of ingredient features.

3. *Feature Vector Calculation.* For each food item $s_i \in \mathcal{S}$, its $d_{ingr} = 200\ dim$ normalized ingredient feature vector $\boldsymbol{g}^{s_i} = [g_1^{s_i}, ..., g_{d_{ingr}}^{s_i}]$ is finally calculated based on whether ingredient j appears in food item s_i's ingredient list $Ingr\{s_i\}$, as Equation (1) shows:

$$g_j^{s_i} = \begin{cases} 1 \ / \ |Ingr\{s_i\}| & : j \in Ingr\{s_i\} \\ 0 & : j \notin Ingr\{s_i\} \end{cases} \quad (1)$$

original ingredient	filtered ingredient
low moisture ***mozzarella***	***mozzarella***
fresh ***mozzarella***	***mozzarella***
less sodium ***beef broth***	***beef broth***
lower sodium ***beef broth***	***beef broth***
chicken ***egg***s	***egg***
soft-boiled ***egg***	***egg***

Table 1: Keyword extraction and lemmatization.

4. FOOD SIMILARITY EMBEDDING

4.1 Training the Deep Siamese Network

Recent advances in similarity metric learning with Deep Convolutional Neural Networks (CNNs) have resulted in breakthroughs in areas including Face Verification [37], Image Retrieval [41], Geo-localization [24] and Product Design [3]. The CNN architectures in these works are based on the Siamese Network [7], which is trained on a large number of paired inputs of similar and dissimilar examples. In light of the prior efforts mentioned above, we adopt this approach to generate a distance embedding for meals.

The proposed CNN architecture (Food-CNN) is illustrated in Fig. 2. The inputs of Food-CNN are a pair of color food images $x, y \in \mathcal{S}$, each of size $227 \times 227 \times 3$. Then, each image proceeds through an identical feature extraction CNN containing several layers from Convolution to Batch Normalization. Finally, the outputs of the last layer are used as their low-dimensional feature embeddings $f(x), f(y)$. The architecture from the first Convolution layer to the last Fully Connected layer (i.e. layers in dashed line bounding box in Fig.2) is the same as the architecture that achieved state-of-the-art image classification performance on ImageNet [22]. For each of the layers, the numbers at the top specify its window size and step size (Convolution and Max Pooling); the numbers at the bottom specify the size of its output feature map. For example, the first convolution layer takes a $227 \times 227 \times 3$ color image from image data layer as the input and convolves it with 96 filters. Each filter has a size of $11 \times 11 \times 3$ and convolves the image on a grid with step size 4×4. In this sense, the output of the first Convolution layer is a 55×55 feature map with 96 channels. We add a final Batch Normalization layer to normalize the *1000 dim* feature vector so that each dimension has zero mean and unit variance within a training batch. Batch Normalization provides a faster convergence rate and higher accuracy in practice [19].

Our goal of Food-CNN is to learn a low dimensional feature embedding that pulls similar food items together and pushes dissimilar food items far away. Specifically, we want $f(x)$ and $f(y)$ to have small distance (close to 0) if x and y are similar items; otherwise, they should have distance larger than a margin m. Therefore, we choose Contrastive

[2]We will incorporate more sophisticated methods such as *tf-idf* and homonyms/synonyms handling in the future.

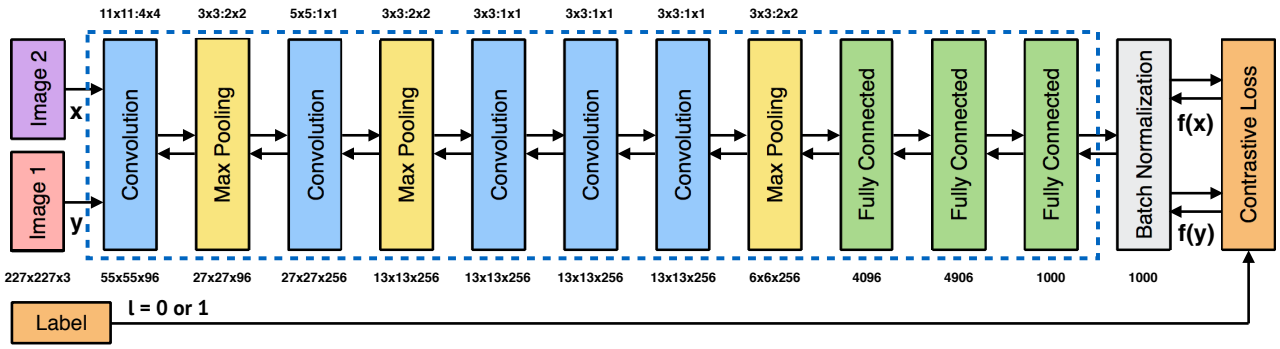


Figure 2: Food-CNN architecture for supervised food similarity distance metric learning.

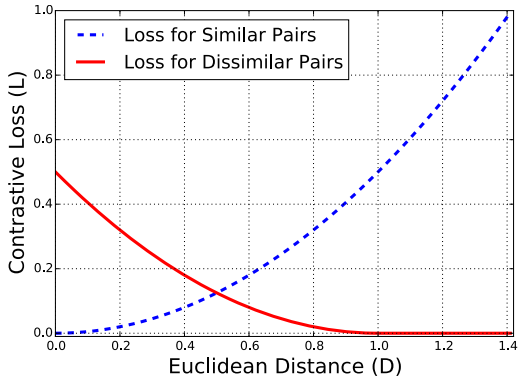


Figure 3: Contrastive Loss function with $m = 1$.

Loss proposed in [16] as the loss function to optimize Food-CNN, which can be expressed as:

$$\mathcal{L}(x, y, l) = \frac{1}{2} l D^2 + \frac{1}{2}(1 - l) \max(0, m - D)^2 \qquad (2)$$

where similarity label $l \in \{0, 1\}$ indicates whether the input pair of food items x, y are similar or not ($l = 1$ for similar, $l = 0$ for dissimilar), $m > 0$ is the margin for dissimilar items and $D = \|f(x) - f(y)\|_2$ is the Euclidean Distance between $f(x)$ and $f(y)$ in embedding space.

As illustrated in Fig. 3, Contrastive Loss function exactly matches our goal. Minimizing the loss in Eqn. (2) pulls similar food images closer and pushes dissimilar ones apart as it penalizes similar pairs by their distances quadratically and dissimilar pairs by their squared differences of the distances to the margin m if they are smaller than m.

Training a deep Siamese Network usually requires huge amount of training data that can't fit in memory. To address this problem, we adopt Nesterov's Accelerated Gradient Descent method [26] with Momentum algorithm [36]. We use back-propagation to compute the gradient of the loss with respect to the parameters of each layer. Suppose we have an n-layer CNN:

$$f(x) = g_n\left(g_{n-1}\left(\dots g_i(\dots g_1(x)\dots)\right)\right) \qquad (3)$$

where $g_i(.)$ represents the computation of i-th layer (e.g., convolution, pooling etc.), x, y and $f(x), f(y)$ represent the input and output pairs of the Siamese Network, respectively. We adopt function $\mathcal{L}(.)$ to calculate the loss of the input pairs x, y as $\mathcal{L}(x, y, l)$. To minimize loss by updating parameters W_i of i-th layer, we need the gradient of the loss with respect to W_i: $\frac{\partial \mathcal{L}(.)}{\partial W_i} = \frac{\partial \mathcal{L}(.)}{\partial g_i} \times \frac{\partial g_i}{\partial W_i}$. Using back-propagation, $\frac{\partial \mathcal{L}(.)}{\partial g_i}$ can be calculated with the chain rule: $\frac{\partial \mathcal{L}(.)}{\partial g_i} = \frac{\partial \mathcal{L}(.)}{\partial g_n} \times \frac{\partial g_n(.)}{\partial g_{n-1}} \times \dots \times \frac{\partial g_{i+1}(.)}{\partial g_i}$. Therefore, for k-th layer $g_k(.)$, we only need to calculate $\frac{\partial g_k(.)}{\partial g_{k-1}}$ and $\frac{\partial g_k(.)}{\partial W_k}$. We use the implementation of gradient descent and back-propagation in Caffe [20], an open source deep learning framework.

Since the food dataset $\mathcal{S}$ pulled from Yummly doesn't include categorical similarity annotations, we trained Food-CNN on the Food-101 dataset [4], which is the largest food image dataset so far and contains 101 food categories and 101,000 food images. We sampled 75,750 same pairs and 757,500 different pairs from the training set to train our Food-CNN.

4.2 Feature Extraction

After the training process, we use the pre-trained Food-CNN to extract visual features from images in our dataset. For each item $s_i \in \mathcal{S}$, we feed the food image to pre-trained feature extraction layers (i.e. layers in blue dashed line bounding box in Fig. 2) and get the feature vector $\tilde{\boldsymbol{v}}^{s_i}$. We normalize $\tilde{\boldsymbol{v}}^{s_i}$ so that it has unit ℓ_1 norm: $v_i^{s_i} = \tilde{v}_i^{s_i} / (\sum_{i=1}^{d_{vis}} |\tilde{v}_i^{s_i}|)$, where $\tilde{\boldsymbol{v}}^{s_i} = [\tilde{v}_1^{s_i}, \dots, \tilde{v}_{d_{vis}}^{s_i}]^T$ and $\boldsymbol{v}^{s_i} = [f_1^{s_i}, \dots, f_{d_{vis}}^{s_i}]^T$ are the feature vectors before and after normalization, respectively; $d_{vis} = 1,000$ is the dimension of visual features.

We then concatenate the visual features $\boldsymbol{v}^{s_i}$ with ingredient features $\boldsymbol{g}^{s_i}$ to create a *1,200 dim* feature embedding $\boldsymbol{f}^{s_i} = [\boldsymbol{v}^{s_i}, \boldsymbol{g}^{s_i}]$ for each food item $s_i \in \mathcal{S}$.

5. ONLINE PREFERENCE LEARNING

5.1 Online Settings and Framework

As discussed in previous sections, each food item $s_i \in \mathcal{S}$ has a *1,200 dim* feature vector $\boldsymbol{f}^{s_i}$ in the embedding space. Building upon the offline feature extraction results, we model the interaction between user and our backend system at iteration $t, (t \in \mathcal{R}^+, t = 1, 2, ..., T)$ as Fig. 4 shows. The symbols that will be used in our algorithms are defined as follows:

$\mathcal{K}_t$: Set of food items that are presented to user at iteration t ($\mathcal{K}_0 = \emptyset$). $\forall k \in \mathcal{K}_t$, $k \in \mathcal{S}$;

$\mathcal{L}_{t-1}$: Set of food items that user *prefer(select)* among $\{k | k \in \mathcal{K}_{t-1}\}$. $\mathcal{L}_{t-1} \subseteq \mathcal{K}_{t-1}$;

$\boldsymbol{p}^t = [p_1^t, ..., p_{|\mathcal{S}|}^t]$: User's preference distribution on all food items $s_i (i = 1, ..., |\mathcal{S}|)$, where $\|\boldsymbol{p}^t\|_1 = 1$. $\boldsymbol{p}^0$ is initialized as $p_i^0 = \frac{1}{|\mathcal{S}|}$;

$\mathcal{B}_t$: Set of food images that have been already explored until iteration t ($\mathcal{B}_0 = \emptyset$). $\mathcal{B}_i \subseteq \mathcal{B}_j (i < j)$;

Based on the workflow depicted in Fig. 4, for each iteration t, backend system updates vector $\boldsymbol{p}^{t-1}$ to $\boldsymbol{p}^t$ and set

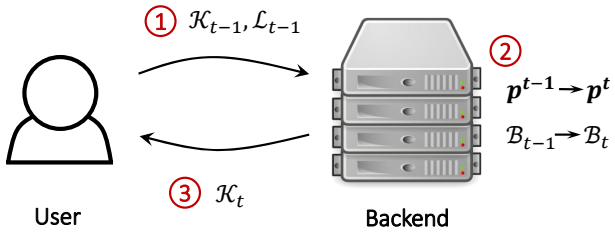


Figure 4: User-system interaction at iteration t.

$\mathcal{B}_{t-1}$ to $\mathcal{B}_t$ based on users' selections $\mathcal{L}_{t-1}$ and previous image set $\mathcal{K}_{t-1}$. After that, it decides the set of images that will be immediately presented to the user (i.e., $\mathcal{K}_t$). Our food preference elicitation framework can be formalized in Algorithm. 1. The core procedures are *update* and *select*, which will be described in the following subsections for more details.

Algorithm 1: Food Preference Elicitation Framework

Data: $\mathcal{S} = \{s_1, ..., s_{10028}\}$, $\mathcal{F} = \{\boldsymbol{f}^{s_1}, ..., \boldsymbol{f}^{s_{10028}}\}$
Result: $\boldsymbol{p}^T$

$\mathcal{B}_0 = \emptyset$, $\mathcal{K}_0 = \emptyset$, $\mathcal{L}_0 = \emptyset$, $\boldsymbol{p}^0 = [\frac{1}{|\mathcal{S}|}, ..., \frac{1}{|\mathcal{S}|}]$;
for $t \leftarrow 1$ **to** T **do**
  $[\mathcal{B}_t, \boldsymbol{p}^t] \leftarrow$ `update`$(\mathcal{K}_{t-1}, \mathcal{L}_{t-1}, \mathcal{B}_{t-1}, \boldsymbol{p}^{t-1})$;
  $\mathcal{K}_t \leftarrow$ `select`$(t, \mathcal{B}_t, \boldsymbol{p}_t)$;
  if t *equals* T **then**
    return $\boldsymbol{p}^T$
  else
    `ShowToUser`$(\mathcal{K}_t)$;
    $\mathcal{L}_t \leftarrow$ `WaitForSelection()` ;

5.2 User State Update

Based on user's selections $\mathcal{L}_{t-1}$ and the image set $\mathcal{K}_{t-1}$, the *update* module renews user's state from $\{\mathcal{B}_{t-1}, \boldsymbol{p}^{t-1}\}$ to $\{\mathcal{B}_t, \boldsymbol{p}^t\}$. Our intuition and assumption behind following algorithm design is that people tend to have close preferences for similar food items in *1,200 dim* space.

5.2.1 Preference vector $\boldsymbol{p}^t$

Our strategy of updating preference vector $\boldsymbol{p}^t$ is inspired by Exponentiated Gradient Algorithm in bandit settings (EXP3) [2]. Specifically, at iteration t, each p_i^t in vector $\boldsymbol{p}^t$ is updated by:

$$p_i^t \leftarrow p_i^{t-1} \times e^{\frac{\beta u_i^{t-1}}{p_i^{t-1}}} \tag{4}$$

where β is the exponentiated coefficient that controls update speed and $\boldsymbol{u}^{t-1} = \{u_1^{t-1}, ..., u_{|\mathcal{S}|}^{t-1}\}$ is the update vector used to adjust each preference value.

In order to calculate update vector $\boldsymbol{u}$, we formalize user's selection process as a data labeling problem [44] where for $s_i \in \mathcal{L}_{t-1}$, label $y_i^{t-1} = 1$ and for $s_j \in \mathcal{K}_{t-1} \backslash \mathcal{L}_{t-1}$, label $y_j^{t-1} = -1$. Thus, the label vector $\boldsymbol{y}^{t-1} = \{y_1^{t-1}, ..., y_{|\mathcal{S}|}^{t-1}\}$ provided by user is:

$$y_i^{t-1} = \begin{cases} 1 & : s_i \in \mathcal{L}_{t-1} \\ 0 & : s_i \notin \mathcal{K}_{t-1} \\ -1 & : s_i \in \mathcal{K}_{t-1} \backslash \mathcal{L}_{t-1} \end{cases} \tag{5}$$

For update vector $\boldsymbol{u}$, we expect that it is close to label vector $\boldsymbol{y}$ but with smooth propagation of label values

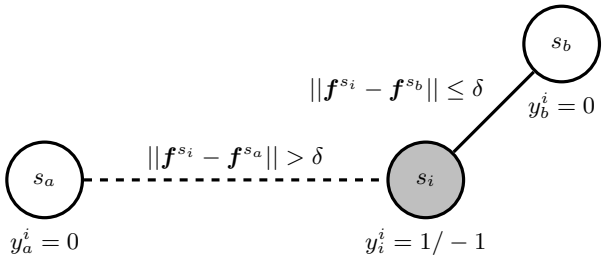


Figure 5: Locally connected graph with item s_i.

to nearby neighbors (For convenience, we omit superscript that denotes current iteration). The update vector $\boldsymbol{u}$ can be regarded as a soften label vector compared with $\boldsymbol{y}$. To make the solution more computationally tractable, for each item s_i with $y_i \neq 0$, we construct a locally connected undirected graph G^i as Fig. 5 shows: $\forall s_j \in \mathcal{S}$, add an edge (s_i, s_j) if $\|\boldsymbol{f}^{s_i} - \boldsymbol{f}^{s_j}\| \leq \delta$ ($\delta = 35$ in our implementation). The labels $\boldsymbol{y}^i$ for vertices s_j in graph G^i are calculated as $y_j^i = 0 (j = 1, \ldots, |\mathcal{S}| \setminus i), y_i^i = y_i$.

For each locally connected graph G^i, we fix u_i^i value as $u_i^i = y_i^i$ and propose the following regularized optimization method to compute other elements ($\forall u_j^i, j \neq i$) of update vector $\boldsymbol{u}^i$, which is inspired by the traditional label propagation method [44]. Consider the problem of minimizing following objective function $Q(\boldsymbol{u}^i)$:

$$\min_{\boldsymbol{u}^i} \sum_{j=1, j\neq i}^{|\mathcal{S}|} w_{ij}(y_i^i - u_j^i)^2 + \sum_{j=1, j\neq i}^{|\mathcal{S}|} (1 - w_{ij})(u_j^i - y_j^i)^2 \tag{6}$$

In Eqn. (6), w_{ij} represents the similarity measure between food item s_i and s_j:

$$w_{ij} = \begin{cases} e^{-\frac{1}{2\alpha^2}\|\boldsymbol{f}^{s_i} - \boldsymbol{f}^{s_j}\|^2} & : \|\boldsymbol{f}^{s_i} - \boldsymbol{f}^{s_j}\| \leq \delta \\ 0 & : \|\boldsymbol{f}^{s_i} - \boldsymbol{f}^{s_j}\| > \delta \end{cases} \tag{7}$$

where $\alpha^2 = \frac{1}{|\mathcal{S}|^2} \sum_{i,j \in \mathcal{S}} \|\boldsymbol{f}^{s_i} - \boldsymbol{f}^{s_j}\|^2$.

The first term of the objective function $Q(\boldsymbol{u}^i)$ is the *smoothness constraint* as the update value for similar food items should not change too much. The second term is the *fitting constraint*, which makes $\boldsymbol{u}^i$ close to the initial labeling assigned by user (i.e. $\boldsymbol{y}^i$). However, unlike [44], in our algorithm, the trade-off between these two constraints is dynamically adjusted by the similarity between item s_i and s_j where similar pairs are weighed more with smoothness and dissimilar pairs are forced to be close to initial labeling.

With Eqn. (6) being defined, we can take the partial derivative of $Q(\boldsymbol{u}^i)$ with respect to different u_j^i as follows:

$$\frac{\partial Q(\boldsymbol{u}^i)}{u_{j,j\neq i}^i} = 2w_{ij}(u_j^i - u_i^i) + 2(1 - w_{ij})(u_j^i - y_j^i) = 0 \tag{8}$$

As $u_i^i = y_i^i$, then:

$$u_j^i = w_{ij} u_i^i = w_{ij} y_i^i (j = 1, 2, ..., |\mathcal{S}|) \tag{9}$$

After all $\boldsymbol{u}^i$ are calculated, the original update vector $\boldsymbol{u}$ is then the sum of $\boldsymbol{u}^i$, i.e. $\boldsymbol{u} = \sum_i \boldsymbol{u}^i$. The pseudo code for the algorithm of updating preference vector is shown in Algorithm.2 for details.

5.2.2 Explored food image set $\mathcal{B}_t$

In order to balance the *exploitation* and *exploration* in image selection phase, we maintain a set $\mathcal{B}_t$ that keeps track of all similar food items that have already been visited by user and the updating rule for $\mathcal{B}_t$ is as follows:

$$\mathcal{B}_t \leftarrow \mathcal{B}_{t-1} \cup \{s_i \in \mathcal{S} | min_{s_j \in \mathcal{K}_{t-1}} \| \boldsymbol{f}^{s_i} - \boldsymbol{f}^{s_j} \| \leq \delta\} \quad (10)$$

With the algorithms designed for updating preference vector $\boldsymbol{p}^t$ and explored image set $\mathcal{B}_t$, the overall functionality of procedure *update* is shown in Algorithm.2.

Algorithm 2: User state update Algorithm - *update*

Function `update`$(\mathcal{K}_{t-1}, \mathcal{L}_{t-1}, \mathcal{B}_{t-1}, \boldsymbol{p}^{t-1})$
  input : $\mathcal{K}_{t-1}, \mathcal{L}_{t-1}, \mathcal{B}_{t-1}, \boldsymbol{p}^{t-1}$
  output: $\mathcal{B}_t, \boldsymbol{p}^t$
  $\boldsymbol{u} = [0, ..., 0], \mathcal{B}_t = \mathcal{B}_{t-1}, \boldsymbol{p}^t = \boldsymbol{p}^{t-1}$
  for $i \leftarrow 1$ **to** $|\mathcal{S}|$ **do**
    // preference update
    for s_j *in* $\mathcal{K}_{t-1}$ **do**
      $u_i \leftarrow u_i + (-1)^{(s_j \in \mathcal{L}_{t-1})-1} w_{ij}$
    $p_i^t = p_i^{t-1} e^{\frac{\beta u_i}{p_i^{t-1}}}$
    // explored image set update
    if `min`$(\|\boldsymbol{f}^{s_i} - \boldsymbol{f}^{s_j}\|, \forall s_j \in \mathcal{K}_{t-1}) \leq \delta$ **then**
      $\mathcal{B}_t \leftarrow \mathcal{B}_t \cup \{s_i\}$
  // normalize $\boldsymbol{p}^t$ s.t. $\|\boldsymbol{p}^t\|_1 = 1$
  `normalize`$(\boldsymbol{p}^t)$

Algorithm 3: Kmeans++ Algorithm for Exploration

Function `k-means-pp`$(\mathcal{S}, n)$
  input : $\mathcal{S}, n$
  output: $\mathcal{K}_t$
  $\mathcal{K}_t$=`random`$(\mathcal{S})$
  while $|\mathcal{K}_t| < n$ **do**
    prob $\leftarrow [0, ..., 0]_{|\mathcal{S}|}$
    for $i \leftarrow 1$ **to** $|\mathcal{S}|$ **do**
      $\text{prob}_i \leftarrow min(\|\boldsymbol{f}^{s_i} - \boldsymbol{f}^{s_j}\|^2 | \forall s_j \in \mathcal{K}_t)$
    sample $s_m \in \mathcal{S}$ with probability $\propto \text{prob}_m$
    $\mathcal{K}_t \leftarrow \mathcal{K}_t \cup \{s_m\}$

5.3 Images Selection

After updating user state, the *select* module then picks food images that will be presented in the next round. In order to trade-off between exploration and exploitation in our algorithm, we propose different images selection strategies based on current iteration t.

5.3.1 Food Exploration

For each of the first two iterations, we select ten different food images by using *K-means++* [1] algorithm, which is a seeding method used in *K-means clustering* and can guarantee that selected items are evenly distributed in the feature space. For our use case, *K-means++* algorithm is summarized in Algorithm.3.

5.3.2 Food Exploitation-Exploration

Starting from the third iteration, users are asked to make pairwise comparisons between food images. To balance the Exploitation and Exploration in our algorithm design, we always select one image from the area with higher preference value based on current $\boldsymbol{p}^t$ and another one from *unexplored* area, i.e. $\mathcal{S} \backslash \mathcal{B}_t$. (Both selections are **random** in a given subset of food items). With above explanations, the image selection method we propose in this application is shown in Algorithm 4.

Algorithm 4: Images Selection Algorithm - *select*

Function `select`$(t, \mathcal{B}_t, \boldsymbol{p}^t)$
  input : $t, \mathcal{B}_t, \boldsymbol{p}^t$
  output: $\mathcal{K}_t$
  $\mathcal{K}_t = \emptyset$
  if $t \leq 2$ **then**
    $\mathcal{K}_t \leftarrow$ `k-means-pp`$(\mathcal{S}, 10)$ // K-means++
  else
    // 99th percentile (top 1%)
    threshold $\leftarrow$ `percentile`$(\boldsymbol{p}^t, 99)$
    topSet $\leftarrow \{s_i \in \mathcal{S} | p_i^t \geq \text{threshold}\}$
    $\mathcal{K}_t \leftarrow$ [`random`$(topSet)$, `random`$(\mathcal{S} \backslash \mathcal{B}^t)$]

6. EXPERIMENTAL EVALUATION

6.1 Food Similarity Embedding

We examine and evaluate the clustering performance of Food-CNN model on Food-101 [4] dataset, where each image is tagged with a categorical label. We first extract *1,000 dim* visual feature for each food image in the test set. After that, we explore k, where $k = 1, 2, \ldots, N$ (N is the size of the test set), nearest neighbors of each food image and calculate the Precision and Recall values for each k:

Suppose $\mathcal{N}_k^i$ is the set of k nearest neighbors of item i under category $\mathcal{C}_i$, then the Precision and Recall values are:

$$Precision = \frac{|\{j | \forall j \in \mathcal{N}_k^i \wedge j \in \mathcal{C}_i\}|}{|\mathcal{N}_k^i|} \quad (11)$$

$$Recall = \frac{|\{j | \forall j \in \mathcal{N}_k^i \wedge j \in \mathcal{C}_i\}|}{|\mathcal{C}_i|} \quad (12)$$

In order to measure the overall performance of our embedding method on Food-101 test set, we average the Precision/Recall values over all food images for each method and plot Precision-Recall Curve (PR Curve) as Fig. 6 shows. We use mean Average Precision (mAP), which corresponds to the area under PR Curve, as the quantitative comparison metric. The mAP value of the ideal algorithm is equal to 1.

We compare our Food-CNN model with several state-of-the-art feature extraction methods: 1. *Pretrained AlexNet deep neural network*: This is the state-of-the-art feature extraction method using pretrained AlexNet [22]. We take *1,000 dim* feature representation from the output of the last fully-connected layer. 2. *SIFT+Bag of visual Words(BoW):* As the most popular method among hand-crafted feature representations, SIFT [25] has been shown to be effective in several applications. We extract visual features using 1000

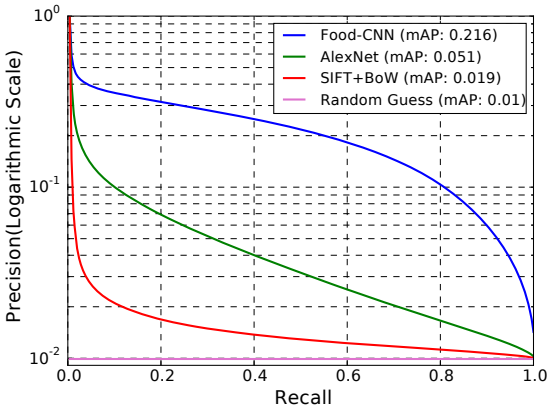


Figure 6: Precision-recall curve for food similarity embedding(*mAP:* mean Average Precision, which represents area under each curve).

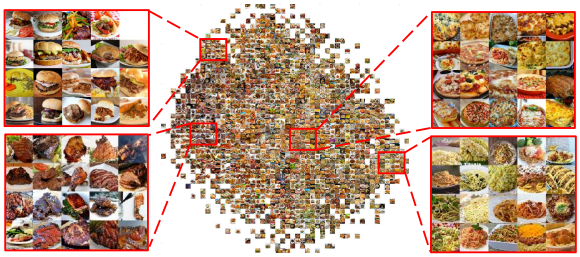


Figure 7: Food Embedding Visualization.

words so as to guarantee that it has the same feature dimension with our Food-CNN.

As can be seen in our results, although *SIFT+BOW* and *AlexNet* greatly outperform random guess baseline, they lack discriminative power for food images because these models are mainly designed for the general clustering purpose. With Food-CNN, we can achieve 4 times performance improvements over state-of-the-art models in terms of mAP value. Algorithm that with much better clustering power can help the whole system understand visual content and thus improve the efficiency of online preference learning.

To further verify and visualize the generalization power of our system, for each of the recipe images that collected from Yummly, we embed it into *1200 dim* feature space by first using Food-CNN to extract *1000 dim* visual representation and then concatenate it with *200 dim* ingredients feature. We project all image representations to 2-D plane by using t-Distributed Stochastic Neighbor Embedding(t-SNE) [39] method. As shown in Fig. 7, we divide the 2-D plane into several blocks and for each block, we sample a representative food image resides in that area. The final embedding results clearly show that our method can effectively group similar food items (recipes) together and push dissimilar items away based on their visual appearances and ingredients metadata. For example, in Fig. 7, we show that burgers, noodles, pizzas, and meat are grouped in different areas of the feature space.

6.2 Online user study

We conducted field study among 227 anonymous users that recruited from social networks and university mailing lists. The experiment was approved by Institutional Review Board (ID: 1411005129) at Cornell University. All participants were required to use this system independently for three times. Each time the study consisted of following two phases:

Training Phase. Users played with *PlateClick* for the first T iterations and the system learnt and elicited preference vector $\boldsymbol{p}^T$ based on the algorithms proposed in this paper or baseline methods, which will be discussed later. We randomly picked T from set $\{5, 10, 15\}$ at the beginning but made sure that each user experienced different values of T only once.

Testing Phase. After T iterations of training, users entered the testing phase, which consisted of 10 rounds of pairwise comparisons. We picked testing images based on preference vector $\boldsymbol{p}^T$ that learnt from online interactions: One of them was selected from food area that user liked (i.e. item with top 1% preference value) and the other one from the area that user disliked (i.e. item with bottom 1% preference value) Both of the images were picked **randomly** among **unexplored** food items.

6.2.1 *Prediction accuracy*

In order to show the learning performance of our algorithm, we compare it with several combinatorial baselines that mentioned next. Users encountered these online learning algorithms **randomly** when they logged into the system:

LE+EE: This is the online learning algorithm proposed in this paper that combines the ideas of **L**abel propagation, **E**xponentiated Gradient algorithm for user state update and **E**xploitation-**E**xploration strategy for images selection.

LE+RS: This algorithm retains our method for user state update (**LE**) but **R**andom **S**elect images to present to user without any exploitation or exploration.

OP+EE: As each item is represented by *1200 dim* feature vector, we can adopt the idea of regression to tackle this online learning problem (i.e. learning weight vector $\boldsymbol{w}$ such that $\boldsymbol{w}\boldsymbol{f}^{s_i}$ is higher for item s_i that user prefer). Hence, we compare our method with **O**nline **P**erceptron algorithm that updates $\boldsymbol{w}$ whenever it makes error, i.e. if $y_i \boldsymbol{w}\boldsymbol{f}^{s_i} \leq 0$, assign $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i \boldsymbol{w}\boldsymbol{f}^{s_i}$, where y_i is the label for item s_i (pairwise comparison is regarded as binary classification such that the food item that user select is labeled as +1 and otherwise -1). In this algorithm, we retain our strategy of images selection (i.e. **EE**).

OP+RS: The last algorithm is the **O**nline **P**erceptron that mentioned above but with **R**andom images **S**election strategy.

Among 227 participants in our study, 58 of them finally used algorithm **LE+EE**, 57 used **OP+RS**. For the rest of users (112), half of them (56) tested **OP+EE** and the other half (56) tested **LE+RS**. Overall, the participants for different algorithms are totally random so that the performances of different models are directly comparable.

After all users going through training and testing phases, we calculate the prediction accuracy of each **individual user** and aggregate them based on the context that they encountered (i.e. the number of training iterations T and the algorithm settings that mentioned above). The prediction accuracies and their cumulative distributions are shown in Fig. 8, 9 and 10 respectively.

Length effects of training iterations. As can be seen in Fig. 8 and Fig. 9, the prediction accuracies of our online learning algorithm are all significantly higher than the baselines.The algorithm performance is further improved with longer training period. As is clearly shown in Fig. 9, when

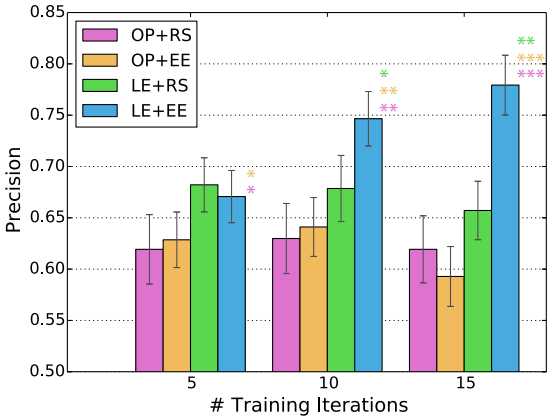


Figure 8: Prediction accuracy for different algorithms in various training settings (asterisks represent different levels of statistical significance: $***: p < 0.001$, $**: p < 0.01$, $*: p < 0.05$).

the number of training iterations reaches *15*, about half of the users will experience the prediction accuracy that exceeds 80%, which is fairly promising and decent considering small number of interactions that system elicited from scratch. The results above justify that *PlateClick*, as an online preference learning system, can adjust itself to explore users' preference area as more information is available from their choices. For the task of item-based food preference bootstrapping, our system can efficiently balance the exploration-exploitation while providing reasonably accurate predictions.

Comparison across different algorithms. As mentioned previously, we compared our algorithm with some obvious alternatives. Unfortunately, according to the results shown in Fig. 8 and Fig. 10, none of these algorithms works very well and the accuracy of prediction is actually decreasing as the user provides more information. Additionally, as is shown in Fig. 10, our algorithm has particular advantages when users are making progress (i.e. the number of training iterations reaches 15). The reasons why these techniques are not suited for our application is mainly due to the following limitations:

Random Selection. Within a limited number of interactions, random selection can not maintain the knowledge that it has already learned about the user (exploitation), nor explore unknown areas (exploration). In addition, it's more likely that the system will choose food items that are very similar to each other and thus hard for the user to make decisions. Therefore, after short periods of interactions, the system is messed up, and the performance degrades.

Underfitting. The algorithm that will possibly have the underfitting problem is the online perceptron (**OP**). For our application, each food item is represented by *1200 dim* feature vector and **OP** is trying to learn a separate hyperplane based on a limited number of training data. As each single feature is directly derived from deep neural network, the linearity assumptions made by perceptron might yield wrong predictions for the dishes that haven't been explored before.

6.2.2 System efficiency

As another two aspects of online preference elicitation system, computing efficiency and user experience are also very important metrics for system evaluation. Therefore, we recorded the program execution time and user response

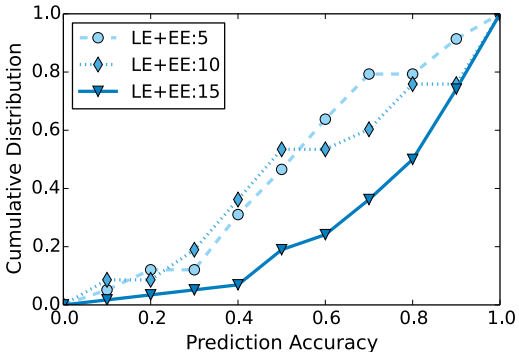


Figure 9: Cumulative distribution of prediction accuracy for *LE+EE* algorithm (Numbers in the legend represent the values of T through training phase).

time as a lens into the real-time performance of *PlateClick*. As shown in Fig. 11(b), the program execution time is about $0.35s$ for the first two iterations and less than $0.025s$ for the iterations afterwards[3]. Also, according to Fig. 11(a), the majority of users can make their decisions in less than $15s$ for the task of comparison among ten food images while the payload for the pairwise comparison is less than $2-3s$. As a final cumulative metric for the system overhead, it is shown in Table 2 that even for 15 iterations of training, users can typically complete the whole process within 53 seconds, which further justify that *PlateClick* is a light-weight user-friendly visual interface for efficient food preference elicitation.

# Iter: 5	# Iter: 10	# Iter: 15
28.75s	39.74s	53.22s

Table 2: Average time to complete training phase.

6.2.3 User Behavior

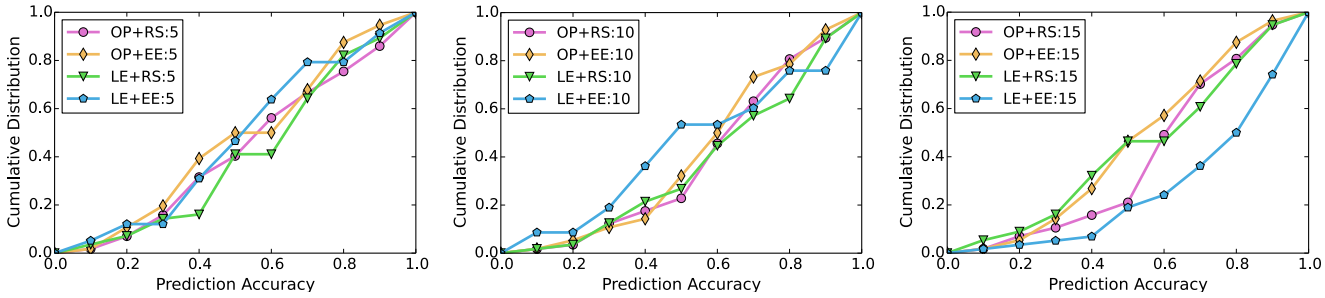
An interesting phenomenon that we observed from our field study is that there exists obvious correlation between *total user response time* and *Precision* of our model: Preference learning of *PlateClick* tends to be more accurate if the user made decisions within shorter period of time. As is shown in Fig. 12, we plot scatter diagram that contains all data points that users generated when they used *PlateClick* under **LE+EE** algorithm setting and with 15 number of training iterations ($T = 15$). Apparently, most of the points (blue ones) are above the dotted line in Fig. 12 and the expected total response time is higher for those users with lower prediction accuracy. The possible reason behind this result is that if food pairs are hard to be distinguished with each other, the responses from user will likely to have large noise and uncertainty,[4] which in turn affects the performance of online learning system.

7. DISCUSSION

In this section we discuss limitations and future directions for *PlateClick*.

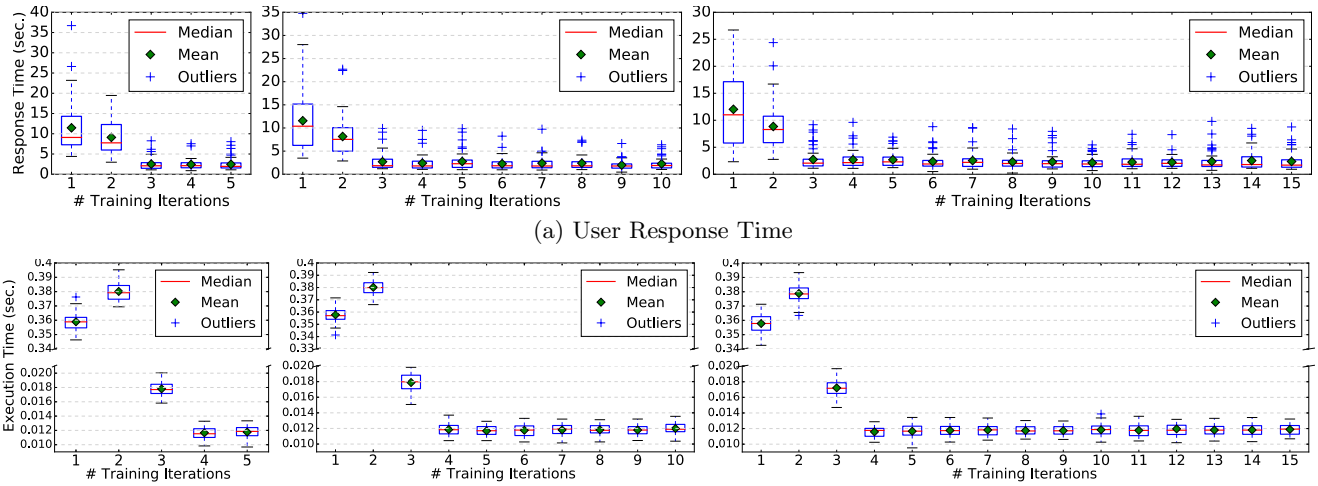
[3]Our web system implementation is based on Amazon EC2 t2-micro Linux 64-bit instance

[4]User engagement is a critical issue in our application. We will conduct studies/interviews on the effects of human behavior in the future.



(a) # training iterations: 5 (b) # training iterations: 10 (c) # training iterations: 15

Figure 10: Comparison of cumulative distribution of prediction accuracy across algorithms.



(a) User Response Time

(b) System Execution Time

Figure 11: Timestamp records for user response and system execution.

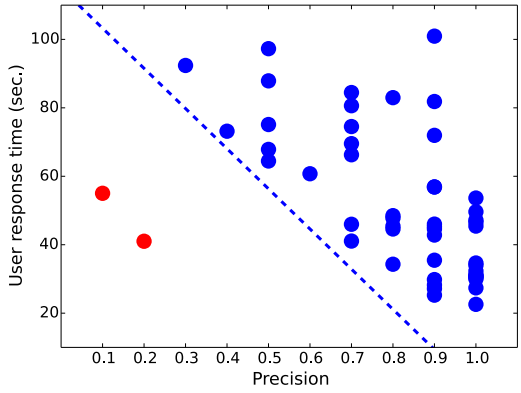


Figure 12: Scatter diagram of total user response time under different precision of model prediction (Points in the graph represent experimental results under LE+EE algorithm setting and with 15 number of training iterations).

Special diets. We noticed in the user study that our current system can not be efficiently used by people with special diets (e.g., vegetarian, vegan, kosher and halal) due to the limits of our image corpus. It's very unlikely that the system will select anything such a user would like from the current main-dishes dataset. It will degrade user experience to repeatedly show dishes that are not to their taste. In the future, we need to consider different diets and curate special food datasets accordingly.

Food courses and diversity. People's food preferences are highly dependent on context as determined by factors such as time of day. It is unnecessarily difficult and possibly confounding to compare dishes across different courses (e.g., between desserts and dinner). In our current system, we mainly focus on food items from main dishes, which conveniently constrains the space of choices available to the user. Future work will incorporate dishes across different courses (e.g., breakfast and lunch) and enrich the diversity of our dataset.

Healthy recommendation. One of the most important applications that could build on *PlateClick* is a healthy food recommendation system. Combining a user's diet profile and recipe nutrient metadata, we could recommend appealing but healthier food alternatives to users so that they are more likely to follow the system's advice in their daily lives and choices. Suggestions guided by people's preferences will be more effective and persuasive than a traditional strategy that lack awareness of what a given person actually likes. This was our initial motivation for developing *PlateClick* and we hope to pursue integration with existing nutritional behavior change apps.

8. CONCLUSION

In this work, we introduced *PlateClick*, a novel visual interface for real-time food preference elicitation from scratch. Compared with previous solutions and online learning systems, we don't assume any prior knowledge of the user. In addition, we greatly simplify the elicitation user interface

by replacing traditional text-based instruments with visual contents and leveraging a pairwise comparison method. We demonstrated that these design choices reduce user burden and cognitive load when using the elicitation system.

Although our algorithmic framework was originally designed for visual content based food preference learning, the techniques proposed in this paper could be used to enhance the interplay between human hedonic and content similarities in solving general *human-in-the-loop* problems. We envision that *PlateClick*, an efficient and user-friendly food preference learning system, could be used to capture personal diet profiles and fuel a wide range of applications in healthcare and commercial recommender systems.

9. ACKNOWLEDGMENTS

We would like to thank Dr. Curtis Cole for suggesting the development of food preference modeling and Dr. Thorsten Joachims for discussion of ML algorithms. Also, we appreciate the anonymous reviewers for insightful comments. This research is partly funded by AOL-Program for Connected Experiences, NSF grant IIS-1344587 and further supported by the small data lab at Cornell Tech which receives funding from UnitedHealth Group, Google, Pfizer, RWJF, NIH and NSF.

10. REFERENCES

[1] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *ACM-SIAM symposium on Discrete algorithms*, 2007.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 2002.

[3] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. on Graphics*, 2015.

[4] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101 – mining discriminative components with random forests. In *ECCV*, 2014.

[5] S. Chang, F. M. Harper, and L. Terveen. Using groups of items for preference elicitation in recommender systems. In *CSCW*, 2015.

[6] X. Chen, P. N. Bennett, K. Collins-Thompson, and E. Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *WSDM*, 2013.

[7] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[8] V. Conitzer. Eliciting single-peaked preferences using comparison queries. In *AAMAS*, 2007.

[9] F. Cordeiro, E. Bales, E. Cherry, and J. Fogarty. Rethinking the mobile food journal: Exploring opportunities for lightweight photo-based capture. In *CHI*, 2015.

[10] M. Das, G. De Francisci Morales, A. Gionis, and I. Weber. Learning to question: leveraging user preferences for shopping advice. In *KDD*, 2013.

[11] D. Estrin. Small data, where n = me. *Commun. ACM*, 57(4):32–34, Apr. 2014.

[12] J. Freyne and S. Berkovsky. Intelligent food planning: personalized recipe recommendation. In *ACM IUI*, 2010.

[13] G. Geleijnse, P. Nachtigall, P. van Kaam, and L. Wijgergangs. A personalized recipe advice system to promote healthful choices. In *ACM IUI*, 2011.

[14] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *WSDM*, 2011.

[15] S. Guo and S. Sanner. Real-time multiattribute bayesian preference elicitation with pairwise comparison queries. In *AISTATS*, 2010.

[16] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[17] M. Harvey, B. Ludwig, and D. Elsweiler. Learning user tastes: a first step to generating healthy meal plans? In *International Workshop on Recommendation Technologies for Lifestyle Change*, 2012.

[18] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, 1999.

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[21] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, 2008.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[23] J. Li and W. Zhang. Conditional restricted boltzmann machines for cold start recommendations. *arXiv preprint arXiv:1408.0096*, 2014.

[24] T.-Y. Lin, Y. Cui, S. Belongie, and J. Hays. Learning deep representations for ground-to-aerial geolocalization. In *CVPR*, 2015.

[25] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[26] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/sqrt(k)). In *Soviet Mathematics Doklady*, 1983.

[27] J. Noronha, E. Hysen, H. Zhang, and K. Z. Gajos. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011.

[28] J. P. Pollak, P. Adams, and G. Gay. Pam: a photographic affect meter for frequent, in situ measurement of affect. In *CHI*, 2011.

[29] P. Pu and L. Chen. User-involved preference elicitation for product search and recommender systems. *AI magazine*, 2009.

[30] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *ACM IUI*, 2002.

[31] J. D. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, 2005.

[32] P. Rozin, C. Fischler, S. Imada, A. Sarubin, and A. Wrzesniewski. Attitudes to food and the role of food in life in the usa, japan, flemish belgium and france: Possible implications for the diet–health debate. *Appetite*, 1999.

[33] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[34] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

[35] M. Sun, F. Li, J. Lee, K. Zhou, G. Lebanon, and H. Zha. Learning multiple-question decision trees for cold-start recommendation. In *WSDM*, 2013.

[36] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[37] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[38] C.-Y. Teng, Y.-R. Lin, and L. A. Adamic. Recipe recommendation using ingredient networks. In *Annual ACM Web Science Conference*, 2012.

[39] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[40] Y. van Pinxteren, G. Geleijnse, and P. Kamsteeg. Deriving a recipe similarity measure for recommending healthful meals. In *ACM IUI*, 2011.

[41] J. Wang, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, Y. Wu, et al. Learning fine-grained image similarity with deep ranking. *arXiv preprint arXiv:1404.4661*, 2014.

[42] M. Zhang, J. Tang, X. Zhang, and X. Xue. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In *SIGIR*, 2014.

[43] X. Zhang, J. Cheng, S. Qiu, G. Zhu, and H. Lu. Dualds: A dual discriminative rating elicitation framework for cold start recommendation. *Knowledge-Based Systems*, 2015.

[44] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *NIPS*, 2004.

[45] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *SIGIR*, 2011.