Food χ: Building a Recommendation System for Chinese Dishes

Yogi Huang, Yiting Ji, Yu Zeng
{yhua6742, yj022011, zengyu} @stanford.edu

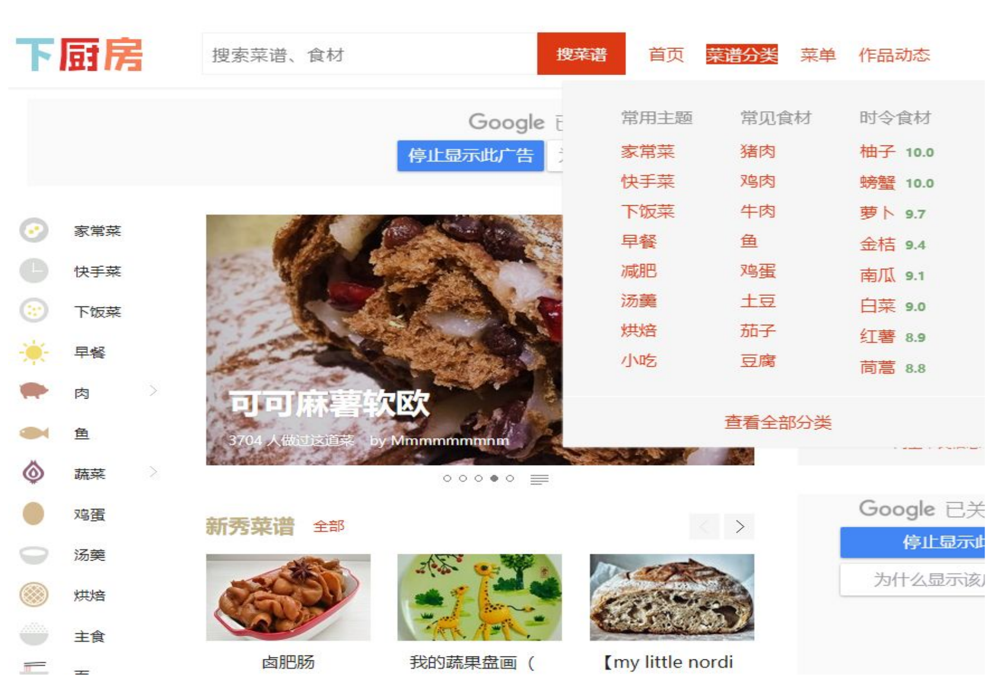
Objective

- Obtain user and dish data from the web, and provide English translation
- Build a recommendation system for dishes through rating predictions
- Examine the prediction performance of our selected algorithms
- Focus on Chinese food for specialization

Data Set and Features

Data

- Scrape a portion of the user ids (randomly) and their starred dishes from Xia Chu Fang, a community where users can publicly post recipes of dishes, and save dishes they are interested in
- Utilize parallel crawling and proxies to fetch data more efficiently, and ran a spider on Google Cloud
- Scraped ~230,000 valid users and 2.5 million dishes in their starred list (~12,000 unique dishes names)



Dish Name Mapping

- Map dishes to an online database of Chinese dish names with English translations as a dictionary
- Use Jaro-Winkler distance for mapping and reduced number of unique dishe names to 1,628.

Ratings

- User ratings are necessary to implement methods such as collaborative filtering
- Rating is defined as

$$R_k^{(i)} = \text{count of dish}_i \text{ in user}_k\text{'s starred list}$$

- Ratings from 5 to 10 is kept for simplification

	user_id	recipe_name	rating
246195	19775	484	5
2127989	166145	1579	5
1786193	139774	211	5
1551874	119794	1534	5
1195275	92602	502	5
184341	14998	1358	5
2163147	168693	150	5
149293	12219	355	5
1116379	86287	1202	5
982108	76384	2003	5

Figure 1: A Sample of Users' Ratings

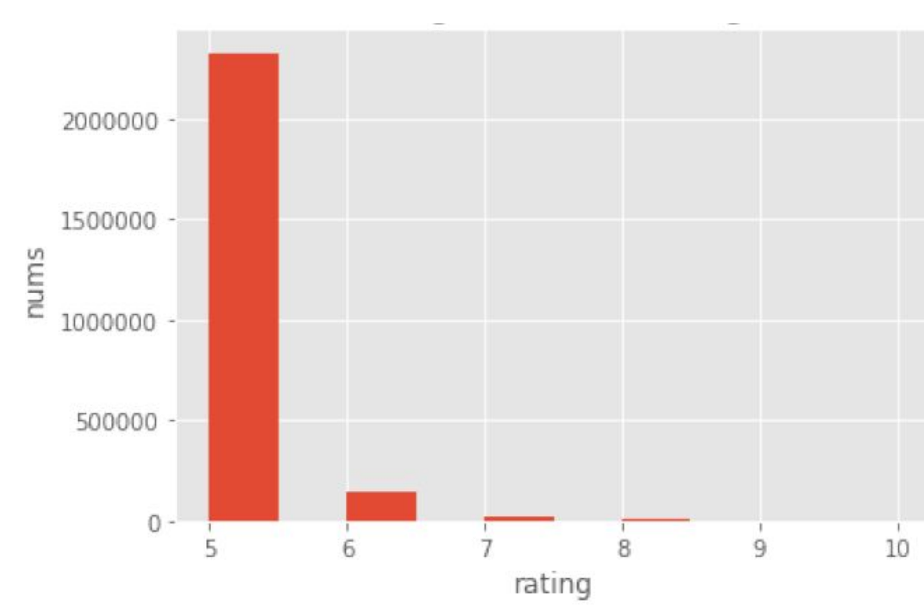


Figure 2: Histogram of User Ratings between 5 and 10

Methods

Word2Vec - Skip Gram

- Trains a neural network with a single hidden layer to perform, and outputs words most relevant to the input
- Minimizes the loss function E in each training iteration:

$$E = -\log p(w_{o,1}, w_{o,2}, \ldots, w_{o,c}|w)$$
$$= -\log \prod_{c=1}^{C} \frac{\exp(u_{c,j_c^*})}{\sum_{j'=1}^{V} exp(u_{j'})}$$
$$= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp(u_{j'})$$

- Returns a similarity score for each output word

```
Input: sour and swweet fish
#1 - Fried Boiled Pork with Sea Cucumber: 0.948
#2 - Braised Fish Filets in Taiwan Black Bean Sauce: 0.946
#3 - Boiled Seafood with Bamboo Fungus: 0.944
#4 - Pan-Fried Flatfish: 0.944
#5 - Braised Intestines in Brown Sauce: 0.941
#6 - Perch with Mushrooms: 0.940
#7 - Fresh Squid in Hot Chili Oil: 0.936
#8 - Braised Seafood with Japanese Tofu: 0.936
#9 - Prawns Kebab with Curry: 0.935
#10 - Dry-Braised Prawn: 0.935
```

Collaborative Filtering

- Matrix-factorization (MF)-based approaches prove to be highly accurate and scalable in addressing CF problems
- Implements non-negative matrix factorization (NMF) and singular value decomposition (SVD) for comparison
- **NMF**
  - Utilizes Python library 'Surprise'
  - Uses regularized stochastic gradient descent update rule
  - Uses $\lambda_i = 0.06$ and $\lambda_k = 0.06$

$$p_{kf} \leftarrow p_{kf} \cdot \frac{\sum_{i \in I_k} q_{if} \cdot R_k^{(i)}}{\sum_{i \in I_k} q_{if} \cdot \hat{R}_k^{(i)} + \lambda_k |I_k| p_{kf}}$$

$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{k \in K_i} p_{kf} \cdot R_k^{(i)}}{\sum_{k \in K_i} p_{kf} \cdot \hat{R}_k^{(i)} + \lambda_i |K_i| q_{if}}$$

- **SVD**
  - Minimizes by gradient descent

$$\sum_{R_k^{(i)} \in R_{train}} (R_k^{(i)} - \hat{R}_k^{(i)}) + \lambda(b_i^2 + b_k^2 + \| q_i \|^2 + \| p_k \|^2)$$

  - Uses learning rate $\gamma = 0.005$ and regularization factor $\lambda = 0.02$

$$b_k \leftarrow b_k + \gamma((e_k^{(i)}) - \lambda b_k),\ b_i \leftarrow b_i + \gamma((e_k^{(i)}) - \lambda b_i)$$

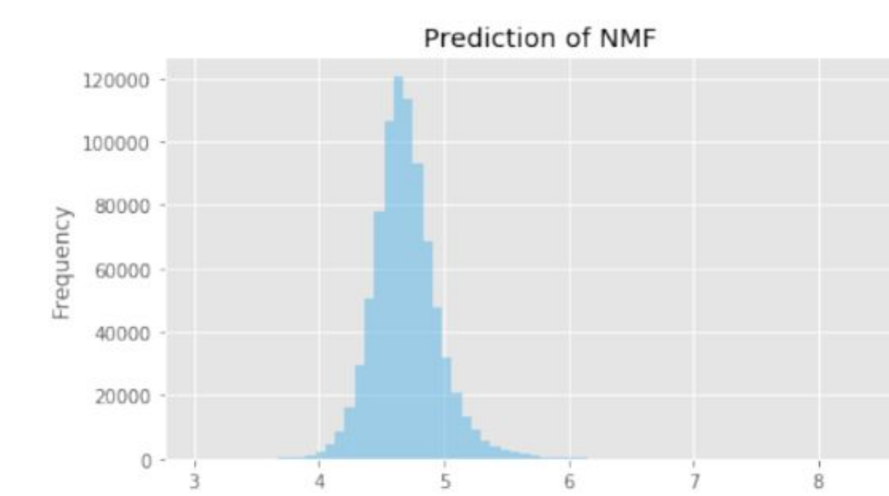
$$p_k \leftarrow p_k + \gamma((e_k^{(i)}) \cdot q_i - \lambda p_k),\ q_i \leftarrow q_i + \gamma((e_k^{(i)}) \cdot p_k - \lambda q_i)$$

Results

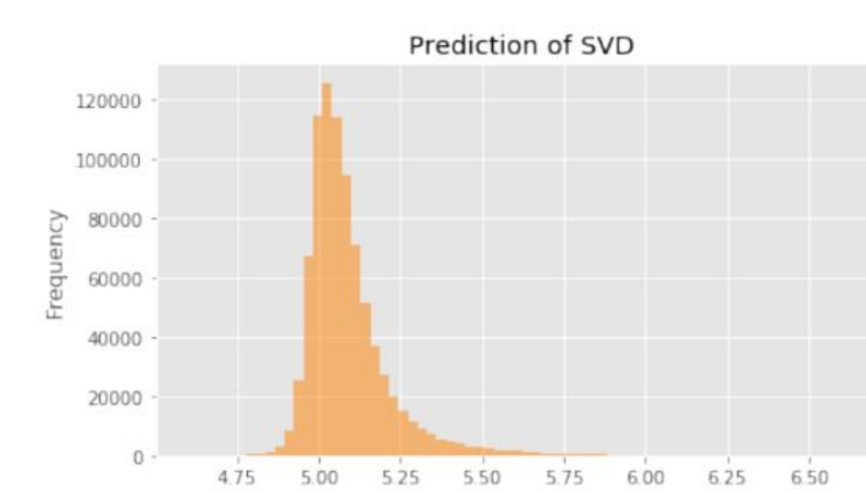
Skip-Gram

- Creates t-SNE graphs to represent similarity between dish names
- The closer the dishes are, the more similar they are

NMF Predictions



SVD Predictions



Error Analysis

- RMSE

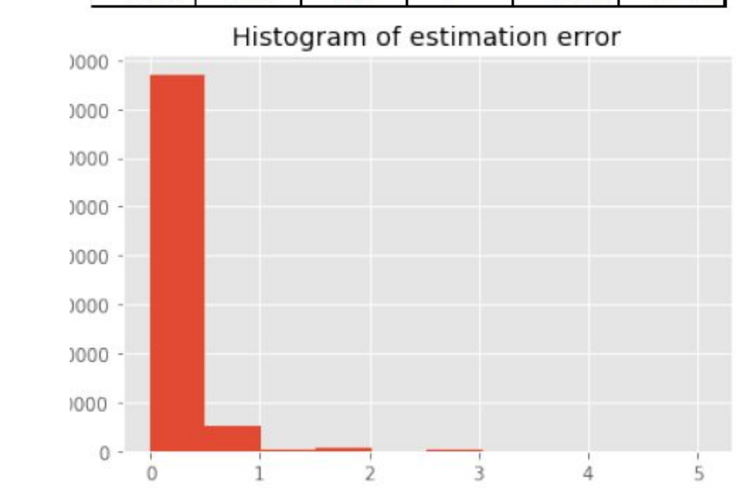
$$RMSE = [\sum \frac{{e_k^{(i)}}^2}{N}]^{0.5}$$

- Recall

$$\frac{TP}{TP + FN}$$

- SVD RMSE

	user_id	item_id	rating	estimation	err
251645	20794	1086	5	4.997827	0.002173
166778	56889	294	5	5.144476	0.144476
752427	158969	622	5	5.076316	0.076316
763904	105710	1226	5	4.9891	0.0109
162796	76755	621	5	5.010914	0.010914
752750	118278	313	5	5.085399	0.085399
292889	137962	1344	5	4.982749	0.017251
171279	122470	33	5	4.961425	0.038575
424391	137549	107	5	4.917739	0.082261
750003	20353	1888	5	4.953488	0.046512



- Error comparison between NMF and SVD

Model	Dev Set RMSE	Test Set RMSE	Dev Set Recall	Test Set Recall
NMF	0.4382	0.5679	0.5104	0.5446
SVD	0.3312	0.3670	0.9176	0.9298

- SVD performs better

Conclusions

- Word2Vec directly gives recommendations, but it is hard to conceptualize or quantify errors
- SVD model performs the best for CF as it has the lowest RMSE and highest Recall on dev set, and the test set error is close to the dev set error, which means it does not overfit and is fairly robust

Future Work

- try other recommendation systems (hybrid system, item-based CF, memory based algorithm, ...)
- obtain data from other dish websites to examine stability
- create user interface
- References available upon request