CS 5785: Applied Machine Learning Project

Jianni Hu
jh2585@cornell.edu

Renzhi Hu
rh662@cornell.edu

Snigdha Singhania
ss4224@cornell.edu

Project Title: Personalized Smart-Search for Recipe Recommendations

Category: Application of Machine Learning to a practical dataset

1 Introduction

With the increasing number of health issues among individuals, people are adopting more preventative as compared to remedying health approaches. An important aspect of keeping healthy is consuming a well-balanced diet daily. Identifying nourishing meals is particularly challenging for working individuals as they try to strike a balance between their professional and personal lives. Further, healthy meals seem unappetizing for many, which further lures people towards less health but more tasty and convenient meals.

In this regard, we wish to explore how recommendation systems could tackle this problem haunting the contemporary professionals, by learning about their food preferences and providing a plethora of healthy options to choose from.

2 Background

We use clickstream and item attributes from our dataset to experiment with Content-Based, Collaborative and Sequential Recommendation Models. Through this, we aim to recommend meals for users according to the time of day, their cuisine preferences, dietary restrictions and calorie target.

Personalised Search We receive user inputs for maximum calories and a free-form query text. Based on these user criteria, we rank the recipes in our database and prepare a list of candidates for our user.

Recommendations We explore implicit-feedback based matrix factorisation techniques, with latent vectors to represent the user and the recipes, to model the user's clickstream to provide personalised recommendations models [1]. Additionally, we explore neural network-based models sequence models [2] [3] [4] [5]. We also look at content-based recommender systems by using item attributes in an attempt to solve cold-start problems associated with new recipes.

3 Related Works

Work done in food recommendations Several related works on food recommendations are available but take a different approach. A real-time recommendation system application with RFID systems in grocery stores applying Bayesian networks for retailers to conduct real-time recommendations to customers based on the products placed in cart during a shopping event [6]. There was also a cloud-based food recommendation system named Diet-Right for dietary recommendations based on users' pathological reports using ant colony algorithm to generate optimal food list and recommends suitable foods [7]. A personalized nutrient-based meal recommender system named Yum-me shares some similarity with our project as it needs input by the user. However, Yum-me input does not included calories threshold and their approach was to implement a food image analysis model extracting features from food images [8].

Related works to personalized search Personalized search is widely explored and used in search engines and recommendation apps. Existing research works primarily fall into categories including (1) the integration of personalization features in the ranking algorithm itself [9], (2) personalized query expansion [10], and (3) re-ranking in a post-processing step [11]. Our work investigated the first category. There is relevant food personalization app, Personal Digital Assistants, where recommendations are generated by matching products to customers based on the expected appeal of the product and the previous spending of the customer [12].

Recommendations Recommender systems are utilized in a variety of areas and are most commonly recognized as playlist generators for media services, product recommenders for online stores, or content recommenders for social media platforms and open web content recommenders [13]. Common approaches include collaborative filtering and content-based filtering, which is one of our chosen experimental methods. Additionally, we found several recent studies have shown the utility of deep learning in the area of recommendation systems and information retrieval as well [14].

4 Experiments

4.1 Dataset

We use the Food.com [15] [16] datasets, from Kaggle, for our experiment. It comprises of 2 subsets - recipes and user interactions.

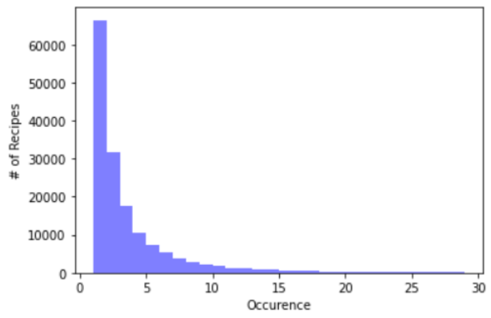
Table 1: Dataset

# Recipes	# Users	# Reviews	Sparsity
230,186	25,076	1,125,284	99.9805%

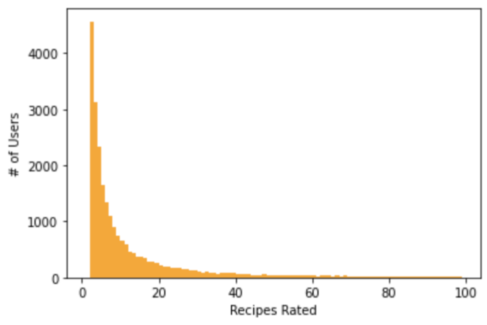
The raw recipes dataset has 12 columns, which includes information such as recipes, ingredients, descriptions, tags, minutes for preparation, and nutritional value. The user clickstream contains user ratings for different recipes over a period of time across 5 columns.

4.2 Pre-processing

For user interactions, we use a label encoder for both the *user id* and *recipe id* to limit the size of the sparse matrix created by Spotlight. We determine data pruning thresholds based on the graphs in Figure 1, which highlight the frequency of recipe and user interactions, respectively. In Figure 1(a), we see the slope become fairly constant after occurrences ≥ 4.Thus, we remove recipes that have been interacted with less than 4 times. Similarly, in Figure 1(b), the drop in frequency reduces significantly after 3, hence we remove users who have rated less than 4. Next, owing to the high sparsity in the dataset (99.98%), we also remove users who have interacted with more than 75 recipes.



(a) Recipe Occurrences in User Interactions



(b) No of Recipe Interactions Per User

Figure 1: Recipe and User Interaction Data Statistics

For the recipes dataset, we transform the *recipe id* using the label encoder mentioned above. Next, we combine 4 columns, namely *name*, *description*, *tags*, and *ingredients* into a new text column. This column is then pre-processed by removing punctuation, English stop words, and single character words after which all words are stemmed using the `PorterStemmer`. We store this pre-processed information in a new column and use it to create an inverted index. Implemented as a dictionary, this inverted index provides fast full-text search by mapping each word in the new column to the *recipe id*.

We group the dataset by `user_id`, using the last recipe in the sorted time sequence as our test, and the last-but-one recipe as the validation set.

Table 2: Dataset: Post Pruning

# Recipes	# Users	# Reviews	Sparsity
12,353	12,840	151,075	99.90%

4.3 Approach

We receive 3 inputs from the user, who is searching for an optimal recipe. These are user ID, maximum calories contained and a search criteria, which could contain ingredients, dietry restrictions, equipment required, time to cook, etc. We use the maximum calories and search criteria for candidate selection, and the user ID to find recommendations based on the user's past behaviour. We rank the recommendations based on personalised candidates and return the top-10 recipes for the user.

Candidate Selection We create 2 sets of candidate recipes, based on:

- *Calories* - We filter a list of all recipes which contain at most `k` calories, provided by the user
- *Search Criteria* - After pre-processing the search criteria using the steps defined in 3.2, we split the search criteria into tokens. Using the inverted index, we perform a search on recipes containing one or more of these tokens. Each recipe is ranked based on how many tokens from the search criteria it matches.

Recommendation We use the Spotlight library [17] for our preliminary experiments. We experiment with different loss functions for `ImplicitFactorizationModel` as well as different representations for the neural network-based `ImplicitSequenceModel` to model the user clickstream. We run a grid-search to find optimal hyper-parameters on our validation set. Based on the results obtained from validation, we use the best hyper-parameters to evaluate our test dataset. This results from the validation experiments are included in the Appendix (Table 7 & Table 8) and the final scores are covered in the next section.

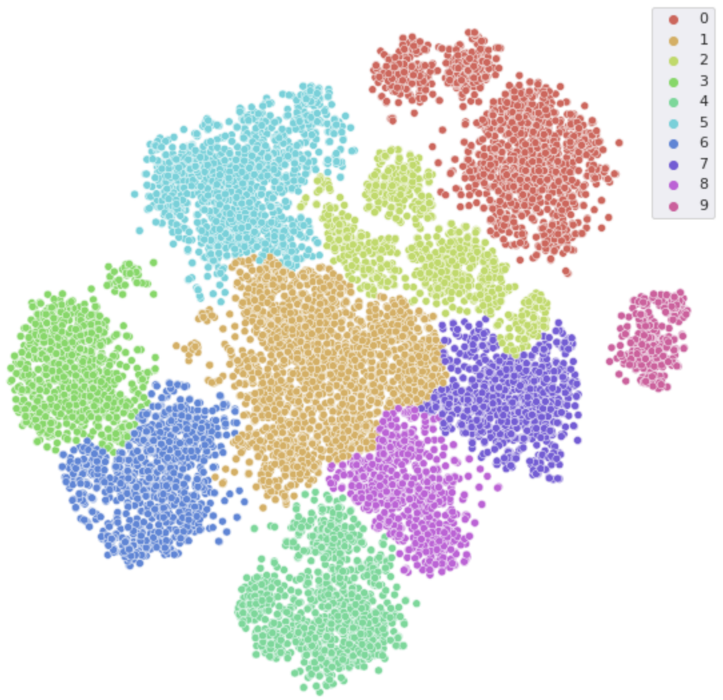


Figure 2: Recipes Clustered using t-SNE and Agglomerative Clustering

We use the tags available in our dataset to visualise and cluster the recipes. Due to the high dimensionality (453) of this feature set, we first use Principal Component Analysis to reduce them first to 100, followed by t-SNE to reduce and visualise these recipes in 2 dimensions [18]. We then use Agglomerative Clustering to group the recipes together, as we suspected unequal cluster sizes. Top tags from each cluster were identified and have been reported in the Appendix (Table 9). As clearly defined recipe clusters are visible in Figure 2, we explore Content-Based filtering by performing cosine similarity on this feature set.

Ranking The results are ranked such that the max calories filter is used for the primary sort order, followed by search criteria, and lastly by the recommendation scores.

4.4 Results & Evaluation

We create a reference performance by evaluating the *Most Popular* recommendations [19]. This recommends recipes with the highest number of interactions to all users. We empirically evaluate our models using 2 metrics.

Hit Rate When the users in the test set have rated one of the *top k* items the model predicts, we consider it a hit. Hit Rate is the ratio of hit versus the total number of item count. We picked the top 50 recommendations from our prediction and calculate the hit rate across all users.

Mean Reciprocal Rank MRR score represents the mean reciprocal rank of all their test items for a given user. As MRR is calculated based on the rank of the prediction, it is helpful in understanding the ranking efficacy of the model, particularly when the dataset is sequential.

Table 3: Implicit Factorization Results on Test Set

		Loss Function			
		Pointwise	BPR	Hinge	Adaptive Hinge
Best Hyper-Parameters	# Negative Samples	5	10	5	10
	L2 Regularizer	0.0005	0.0005	0.001	0.001
	Epochs	10	10	5	5
Metric	Hit Rate (%)	10.568	10.49	**10.83**	10.69
	MRR	0.0146	0.0149	0.0147	**0.0156**

Table 4: Sequential Model Results on Test Set (* A.Hinge = Adaptive Hinge)

		Parameters (Representation, Loss Function)			
		CNN, A.Hinge	Pooling, Pointwise	LSTM, A.Hinge	Mixture, A.Hinge
Best HyperParameter	# Neg Samples	10	5	10	10
	L2 Regularizer	0.001	0.001	0.001	0.0005
	Epochs	5	10	10	10
Metric	Hit Rate (%)	**10.67**	10.592	10.607	10.638
	MRR	**0.0076**	0.0066	0.0066	0.00751

Table 5: Different Model Comparison

Metric	**Model Type**			
	Most Popular	Implicit Factor(Hinge)	CNN(Adaptive Hinge)	Content-Based
Hit Rate (%)	10.60	**10.833**	10.67	2.219
MRR	0.0142	**0.0147**	0.0076	0.001

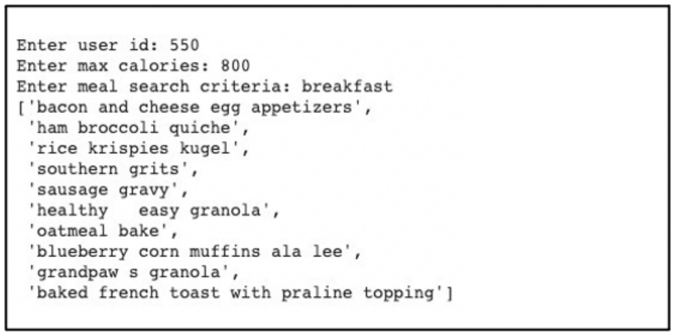
Discussion The results in Table 5 show that the implicit factorization method outperforms the baseline and sequential model, although not by a large margin. Not only is the sequential model the worst in terms of MRR, it reports a score

(0.0076) which is around 50% of that reported by the Most Popular Baseline and Factorization techniques. Further, data pruning (as performed in section 4.2) does not contribute significantly to performance improvement to this model, which was reporting $\sim 5\%$ hit rate without trimming. On the other hand, performance of the Implicit Factorization improves by ≥ 5 times for Hit Rate and ~ 10 times for MRR. The poor scores and improvements does reveal that the dataset does not follow a sequential pattern.

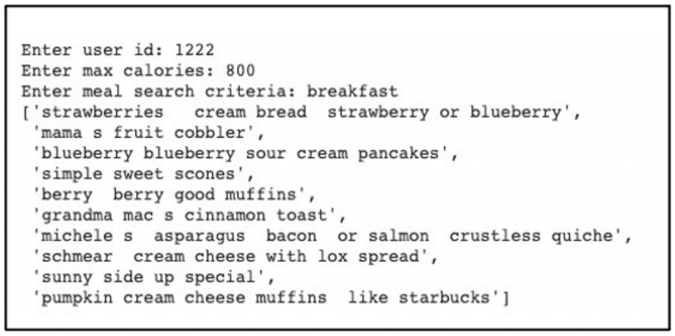
Content-based recommendations perform particularly poorly. As shown in Table 6, a user has on average interacted with recipes across 5 different clusters. This makes item-based similarity futile, as a user is not restricted or bound to one particular cluster, and each cluster contains several similar recipes.

Table 6: User Interactions Across Recipe Clusters

# Recipe Clusters	Mean	Median	Mode
10	5.36	5	4



```
Enter user id: 550
Enter max calories: 800
Enter meal search criteria: breakfast
['bacon and cheese egg appetizers',
 'ham broccoli quiche',
 'rice krispies kugel',
 'southern grits',
 'sausage gravy',
 'healthy   easy granola',
 'oatmeal bake',
 'blueberry corn muffins ala lee',
 'grandpaw s granola',
 'baked french toast with praline topping']
```



```
Enter user id: 1222
Enter max calories: 800
Enter meal search criteria: breakfast
['strawberries   cream bread  strawberry or blueberry',
 'mama s fruit cobbler',
 'blueberry blueberry sour cream pancakes',
 'simple sweet scones',
 'berry  berry good muffins',
 'grandma mac s cinnamon toast',
 'michele s  asparagus  bacon  or salmon  crustless quiche',
 'schmear  cream cheese with lox spread',
 'sunny side up special',
 'pumpkin cream cheese muffins  like starbucks']
```

User **A**

User **B**

Figure 3: User A: General User, User B: Pescatarian

Example A sample output from our recommender system can be seen in Figure 3. Based on our data exploration, we found **User 1222** has rated 75 recipes all of which are comprise either vegetarian or seafood. This is in accordance to our outputs, which suggests this user with similar choices, while **User 550** does not seem to have a restricted diet, and thus has a variety of different options.

5 Conclusion & Future Works

As compared to the preliminary experiments conducted during the checkpoint, our implicit feedback-based factorisation model and Most Popular Baseline report scores which are 8-10 times better as a result of data pruning. Data pruning discounts some noise in the data and reduces the collection of recipes significantly. However, the high sparsity in the dataset continues to be a hindrance to further enhance the results. The sparsity problem occurs when clickstream to identify similar users is insufficient, hence making predictions difficult.

We learn of two possible problems in our work which can be built upon to improve the recommendations. First, despite pre-processing and removing some noise from the recipe tags, the feature set continues to be quite large, many which are not particularly useful to identify user behaviour. We can look into weighing different tags in this data to better model user behaviour in another attempt to leverage attributes in prediction. If results do improve, one can also look into hybrid recommendation systems which combine collaborative and content-based filtering techniques [20]. Secondly, the highly sparse dataset is unable to model user preferences accurately. In spite of regularisation, the algorithms cannot optimise sufficiently and get stuck in local minimas. While user and item attributes are helpful in such situations, the large size and sparsity of this information is a limitation. We can look into additional data sources to add more attributes to the recipes.

It should be noted that this dataset has not been used in a similar context before, and hence, we do not have quantitative results to compare it with.

References

[1] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[2] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, “Neural machine translation in linear time,” *arXiv preprint arXiv:1610.10099*, 2016.

[3] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with recurrent neural networks,” *arXiv preprint arXiv:1511.06939*, 2015.

[4] M. Kula, “Mixture-of-tastes models for representing users with diverse interests,” *arXiv preprint arXiv:1711.08379*, 2017.

[5] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for youtube recommendations,” in *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, 2016.

[6] E. Nur Cinicioglu and P. P. Shenoy, “A new heuristic for learning bayesian networks from limited datasets: a real-time recommendation system application with rfid systems in grocery stores,” *Annals of Operations Research*, vol. 244, no. 2, p. 385–405, 2016.

[7] F. Rehman, O. Khalid, N. ul Haq, A. ur Rehman Khan, K. Bilal, and S. A. Madani, “Diet-right: A smart food recommendation system,” *Transactions on Internet and Information Systems*, vol. 11, no. 6, pp. 2910–2925, 2017.

[8] L. Yang, C.-K. Hsieh, H. Yang, N. Dell, S. Belongie, C. Cole, and D. Estrin, “Yum-me: A personalized nutrient-based meal recommender system,” 2017.

[9] T. Haveliwala, S. Kamvar, and G. Jeh, “An analytical comparison of approaches to personalizing pagerank,” 2003.

[10] P. A. Chirita, C. S. Firan, and W. Nejdl, “Personalized query expansion for the web,” Association for Computing Machinery, 2007.

[11] Z. Dou, R. Song, and J.-R. Wen, “A large-scale evaluation and analysis of personalized search strategies,” Association for Computing Machinery, 2007.

[12] R. D. Lawrence, G. S. Almasi, V. Kotlyar, M. Viveros, and S. S. Duri, *Personalization of Supermarket Product Recommendations*, pp. 11–32. Boston, MA: Springer US, 2001.

[13] R. Baran, A. Dziech, and A. Zeja, “A capable multimedia content discovery platform based on visual content analysis and intelligent data enrichment,” *Multimedia Tools and Applications*, 06 2018.

[14] A. Singhal, P. Sinha, and R. Pant, “Use of deep learning in modern recommendation system: A summary of recent works,” *CoRR*, vol. abs/1712.07525, 2017.

[15] B. P. Majumder, S. Li, J. Ni, and J. McAuley, “Generating personalized recipes from historical user preferences,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 5976–5982, Association for Computational Linguistics, Nov. 2019.

[16] “Food.com recipes and interactions.” `https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions`.

[17] M. Kula, “Spotlight.” `https://github.com/maciejkula/spotlight`, 2017.

[18] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[19] Y. Ji, A. Sun, J. Zhang, and C. Li, “A re-visit of the popularity baseline in recommender systems,” *arXiv preprint arXiv:2005.13829*, 2020.

[20] M. Kula, “Metadata embeddings for user and item cold-start recommendations,” in *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.* (T. Bogers and M. Koolen, eds.), vol. 1448 of *CEUR Workshop Proceedings*, pp. 14–21, CEUR-WS.org, 2015.

Appendix

Results on Validation Set

The table below summarizes the best hyper-parameters obtained for each loss function on the validation set. These hyper-parameters were used to evaluate the Test dataset.

Table 7: Implicit Factorization Results on Validation Set

		Loss Function			
		Pointwise	BPR	Hinge	Adaptive Hinge
Best Hyper-Parameters	# Negative Samples	5	10	5	10
	L2 Regularizer	0.0005	0.0005	0.001	0.001
	Epochs	10	10	2	2
Metric	Hit Rate (%)	10.537	10.623	10.429	10.561
	MRR	0.0143	0.0142	0.014	0.0137

Table 8: Sequential Model Results on Validation Set (* A.Hinge = Adaptive Hinge)

		Parameters (Representation, Loss Function)			
		CNN, A.Hinge	Pooling, Pointwise	LSTM, A.Hinge	Mixture, A.Hinge
Best HyperParameter	# Neg Samples	10	5	10	10
	L2 Regularizer	0.001	0.001	0.001	0.0005
	Epochs	5	10	10	10
Metric	Hit Rate (%)	9.62	**9.7**	9.56	**9.7**
	MRR	0.0074	0.0060	0.0064	**0.0082**

Most Common Tags in Recipe Clusters

Table 9: Common Tags describing Recipes clustered together

# Cluster	**Top Tags**
0	poultri, chicken, lowcarb, chickenbreast, highprotein
1	sidedish, vegatarian, american, 15minutesorless
2	pasta, seafood, stovetop, comfortfood, shellfish
3	dessert, cookiesandbrowni, chocol, kidfriendli, gift
4	bread, fruit, breakfast, brunch, muffin
5	lowcholestrol, lowcalori, lowfat, healthy, vegetarian
6	dessert, cake, fruit, lowprotein
7	beef, groundbeef, american, crockpotslowcook
8	eggsdairi, chees, lunch, egg, breakfast
9	seafood, fish, saltwaterfish, salmon, highprotein

These are the most commonly occurring tags for each cluster identified in Figure 2 above. Some clusters are clearly defined while there is some overlap. From the scatter plot, we see that Cluster 3 & 6 are very close, which is also explained through their tags which revolve around cakes and desserts.

While we are able to obtain some clearly defined clusters, this has not been very helpful in predicting user behaviour, as a user often consumes food across several clusters. In fact, 371 users have consumed recipes belonging to each of the 10 clusters.