A Hybrid Approach to Recommending Recipes with Textual Information

Yinghao Sun
sunmo

Helena Huang
yhuang77

1 Introduction

The main problem in this project that we are exploring to solve is to predict rating by a user on a recipe based on two forms of data: 1) previous interactions consisting of users' ratings on recipes; 2) recipe attributes, mostly in textual form. There are two main approaches, where the first approach is collaborative filtering which mostly relies on the first form of data and the second approach is content-based approach, which mostly relies on item (recipe) attributes. In this project, we hope to explore ways to combine the two approaches in a (hopefully) generalized way: augment latent factors generated from matrix factorization approach with recipe features, which can then be used as augmented features (**input**) to feed into any reasonable supervised learning algorithms to predict ratings (**output**).

For matrix factorization, we use the well-known singular value decomposition (SVD) algorithm used by Simon Funk during the Netflix Prize. In extracting features from textual information of the recipe, we have explored tf-idf + truncated SVD, distributed word representations (GloVe), and latent dirichlet allocation. In predicting ratings, we view this as a regression problem and experimented with LASSO, Elastic Net, support vector regression, Random Forest, and XGBoosting. Note that we could alternatively view the problem of predicting ratings as a classification problem but in this project, we choose to view it as a regression problem, which is also meaningful, for example, one potential application is at the backend of the recommender system we could use continuous predictions to rate recipes for a user.

2 Related Work

In this section, we will briefly introduce collaborative filtering and content-based recommender systems, and hybrid approaches to potentially combine the two. The two broadly categorized approaches to recommender system can be summarized at a high level as below:

Collaborative Filtering In collaborative filtering, predictions of user's preference for an item (i.e., recipe in this case) is based on past preference of all users (thus collaborating). It can be further categorized into two approaches: *neighborhood-based* approach and *model-based* approach. The key idea of neighborhood-based approach is to first define a similarity measure and then predict the rating of the current user based on ratings on the same item from users that are most similar to the current user. One most commonly used similarity measure is Pearson's correlation [16].

The focus of the current project is on model-based approach which typically makes the assumption that users and items can be simultaneously represented as unknown latent factors, and the user-item rating matrix can be reproduced via these lower dimensional user and item latent factors. The process of decomposing a user-item rating matrix into lower dimensional user and item latent factors is called matrix factorization, and this class of methods include Singular Value Decomposition (SVD) Algorithm, Probabilistic Matrix Factorization [11], which makes further assumption about the underlying distribution of the data, maximum-margin matrix factorization [18] which uses a low-norm instead of a low-rank factorization, and non-negative matrix factorization where user and item factors are kept positive [8]. There are also explorations to combine factorization with the neighborhood method [7].

Content-based In content-based approach, predictions of user preference are generated based on the attributes of the user and of the items instead of based on user item rating matrix. For example, recommendations can be made based on user demographic information or tf-idf representation of website as documents [13]. Content-based approach sometimes could be viewed as information retrieval task or classification task [12].

Hybrid Approach Hybrid approach tries to leverage the strength from both approaches. One line of thought is ensemblling, where collaborative filtering and content-based recommenders are trained separately and combine their predictions as final outputs [4]. Other approaches include first applying content-based predictions to convert a user-item rating matrix into full rating matrix, and then apply collaborative filtering to decompose it into latent factors [10].

The main distinctive feature of the current project's approach is: we view matrix factorization as a way to generate precision-based user and item embeddings (latent factors), and these embeddings can be augmented with any other potentially useful features about the user or item attributes. The resulting augmented features can then be fed into any reasonable supervising learning algorithm to predict ratings. The potential advantage of this approach is that it is flexible and generalized, meanwhile this approach is less specialized than other approaches to recommender system and thus its performance may have less guarantees.

3 Dataset and Features

The dataset consists of over 180,000 recipes and over 700,000 ratings covering years of user interactions on Food.com (source: https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions) [9].

The dataset consists of multiple files. One main group of files are user-recipe interactions, which are already split into training (698901 rows; each row corresponds to one rating of a user on a recipe), validation (7023 rows), and test set (12455 rows). Another main file is recipe attributes, where each recipe has its id, name, tags, ingredients, cooking time, nutrition, cooking steps, descriptions, etc. Majority of these recipe attributes (e.g., tags, ingredients, cooking steps, descriptions) are in textual form. In total, there are 231637 unique recipes (note that in this project we directly deal with raw recipe dataset instead of preprocessed recipe datasets).

We examined the training user-recipe interactions (ratings) dataset, there are 25076 unique users, 160901 unique recipes (69.46% of all recipes), and the sparsity is 99.89% (i.e., among all possible 628805776 user-recipe pairs from the training set, only 698901 has ratings).

Ratings are integers from 0 to 5, and distributed as follows: 0.0: 2.43%; 1.0: 0.48%; 2.0: 0.98%; 3.0: 3.69%; 4.0: 18.23%; 5.0: 74.20%, which shows that ratings are distributed skewed to the left. Why is the rating skewed? This may highlight one constraint of the dataset: the dataset itself may be affected by existing recommendation algorithms, which tend to recommend recipes more likely to receive higher ratings.

There are **three groups of features**: 1) user and recipe **latent factors**; 2a) dense features from **textual** information of the recipe (tags, ingredients, and descriptions); 2b) features indicating whether a specific cooking technique appears in the recipe cooking steps (also **textual**); 3) **basic** recipe features including numerical values (cooking minutes, number of steps, number of ingredients, and numerical nutritions).

A considerable part of this project will be focusing on how to extract dense features from recipe textual information (we will be using the **raw recipe dataset with textual information**), with details covered in the Method section.

4 Methods

In this section, we will cover feature extraction, supervised learning algorithms, and evaluation metrics.

4.1 Feature Extraction

User and Recipe Latent Factors via Matrix Factorization To decompose user-recipe rating into lower rank user and recipe latent factors, we adopt the most basic and commonly used SVD matrix factorization algorithm where predicted rating $\hat{r}_{ui}$ by user u on recipe i is defined as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{1}$$

where b_u's, b_i's, q_i's and p_u's are estimated via stochastic gradient descent to minimize the following loss function:

$$\sum_{(u,i)\in I} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2) \tag{2}$$

where I is all the pairs of user and recipe interactions in the training set. We consider $[b_i, q_i]$ as latent factors for recipe i and $[b_u, p_u]$ as latent factors for user u. We tuned the algorithm via 3-fold cross-validation and the resulting weight of l2 penalty λ is 0.12, dimensionality of q_i and p_u is 50, learning rate is 0.005, and number of epochs is 20. We will use the averaged user/recipe latent factor for any unknown recipe or user during testing. Matrix factorization is implemented via the Surprise library [6].

Extracting Features from Recipe Textual Information We explored three ways to extract dense features from textual information in recipe tags, ingredients, and descriptions: bag-of-words (tf-idf) + Singular Value Decomposition (SVD), distributed word representations, and latent Dirichlet allocation. Before any extraction happens, **texts were first preprocessed**: punctuations removed, transformed into lower case, and tokenized (into 1-grams).

tf-idf + truncated SVD Term frequency+inverse document frequency (tf-idf) is a bag-of-words approach where each document d — $d \in D$, where D is the collection of all the documents — is represented as a vector $\mathbf{w}_d$ of dimension $|V|$ where V is the dictionary. Each entry $w_{t,d}$ ($t \in V$) is defined as $tf(t,d) \times idf(t,D)$ where $tf(t,d)$ is the count of token t in document d and $idf(t,D)$ is defined as

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{3}$$

Essentially, tf-idf represents the occurrence of each token in a document normalized by the token's occurrence in the corpus.

On top of tf-idf, we further apply SVD and only keep the first k largest singular values and corresponding singular vectors to find a reduced rank approximation of the tf-idf representations of all documents (thus called truncated SVD). This transforms the sparse features into dense features, making it easier to concatenate with other dense features. Number of components to keep in SVD was not extensively tuned in this project due to practical constraints and we picked 20 for recipe tags which accounts for 41.77% of variance, 30 for ingredients which accounts for 24.11% of variance, and 30 for descriptions which accounts for 11.16% of variance.

Distributed word representations GloVe [15] is a distributed word representation that utilizes co-occurrence statistics and shares the advantage of skip-gram type of approaches. In this project, we were using 50-dimension GloVe trained on wikipedia for recipe tags, ingredients, and descriptions. A document is represented as a 50-dimensional vector which equals the averaged GloVe word embeddings of tokens in this document.

Latent Dirichlet allocation (LDA) [2] LDA is a generative statistical model where given a corpus, it tries to find the underlying "topics" that generate the documents. It is a hierarchical Bayesian model where we assume that each token is generated via a multinomial distribution where the probabilities represent underlying topic of the document and are generated from a Dirichlet (multivariate beta) distribution. Number of latent topics is a tuning parameter, and we hand-tune that by examining the visualization from LDAvis [17] which projects each topic via multidimensional scaling to a 2-dimensional space and make sure that most latent topics are well separated. In the end, we pick 10 latent topics for tags, 10 for ingredients, and 20 for descriptions.

Other Recipe Features One textual recipe feature, i.e., cooking steps, were transformed into a 58-dimension vector of 0s and 1s. 58 common techniques are picked as in [9], and 1 indicates that the technique token appears in the cooking steps.

Other more "basic" recipe features are cooking minutes, number of steps, number of ingredients, and numerical nutritions.

4.2 Supervised Learning Algorithms

Matrix factorization-based collaborative filtering After extracting user and recipe latent factors, there are no further learning to be done, and predicted ratings are based on user and recipe latent factors as described above.

LASSO and Elastic Net [20] Elastic net method is a linear regression that minimizes the loss function: $||y - X\beta||_2^2 + \lambda((1-\sigma)||\beta||_2^2 + \sigma||\beta||_1)$ where l2 penalizes the norm of coefficients and l1 penalty imposes sparsity on the regression coefficients. σ is in $[0, 1]$ and controls the fraction of l1 penalty. When $\sigma = 1$, it is also called LASSO (least absolute shrinkage and selection operator). Tuning parameters of LASSO or Elastic Net can be computed efficiently via regularization path.

Approximate radial basis function (RBF) kernel regression [19] Typical kernel methods require computing the full kernel matrix, which has complexity $O(n^3)$ and can be infeasible for large n. One workaround is to rely on low-rank approximation of the rbf kernel matrix. Nystrom approximation provides such low rank approximation, and we can use it to transform the features first and then feed into a linear support vector regression, which minimizes the squared error epsilon insensitive loss function:

$$\psi(\hat{y}, y) = \max\{0, (\hat{y} - y)^2 - \epsilon\} \tag{4}$$

This loss function ignores squared error less than ϵ (similar to support vector machines) but penalizes on squared loss instead of absolute loss, which penalizes more on predictions further away from the true values. Note that we can also include l1 and l2 penalties.

k-Nearest Neighbors (k-NN) [1] k-NN regression is a non-parametric method where predictions are generated based on the averaged rating from k nearest neighbors of the training set. There is also option to predict based on weighted average, where weights are determined by Euclidean distance. In k-NN, all computation is deferred until prediction. When training set is huge, k-NN can be less efficient in making predictions and we apply KD-Tree for nearest neighbor search, with average time complexity $O(\log n)$.

Random Forest [5] This is an ensemble method that fits a number of decision trees based on a number of sub-samples of the training set, where the sub-samples are drawn with replacement from the original training set. The sub-sample may consists of a subset of the features or rows from the original training set. Predictions are made based on the predictions from each individual trees. Random forest tends to have lower variance without increasing bias too much compared with a single decision tree.

XGBoosting [3] XGBoost is an efficient implementation of gradient boosting framework. Gradient boosting combines weak learners (typically shallow decision trees) into a single strong learner in an iterative fashion, i.e., each state iteration, it trains on the residual from the model at the last iteration, and adds an estimator to the previous model for a better model. It is similar to random forest in the sense that it is also a collection of decision trees, with the difference that trees in random forest are independent while in gradient boosting they are dependent.

4.3 Evaluation Metrics

One commonly used metric is Root Mean Squared Error (RMSE), which weighs more heavily for larger absolute errors, and it is the main evaluation metric. We also consider mean absolute error as a secondary metrics. The two are given by:

$$RMSE = \sqrt{\frac{\sum_{u,i}(\hat{r}_{u,i} - r_{u,i})^2}{N}} \quad \text{and} \quad MAE = \sqrt{\frac{\sum_{u,i}|\hat{r}_{u,i} - r_{u,i}|}{N}} \tag{5}$$

where N is the total number of unrated user-item pairs in the test set that we need to predict. For the best model in terms of RMSE on the validation set, we will also provide more detailed evaluation of its performance on the final test set.

We also consider **accuracy/recall@k** metrics, however, we argue that it might not provide the most accurate evaluation in our case: 1) it is likely that every individual user has only seen a tiny subset of all the recipes, and hence there can be selection bias if we rank all recipes using our algorithm for a user and calculate the recall; 2) if we only rank recipes that a user has rated, there can still be bias due to the fact that every user could have rated very different number of recipes.

5 Experiments and Results

5.1 Experiment Design

The experiment design can be expressed as **feature combinations** $\times$ **sampling strategy** $\times$ **learning algorithm**. **Feature combinations** include: 1) basic recipe features only (cooking minutes, number of steps, number of ingredients, and numerical nutritions), 2) latent user and recipe factors only, 3) latent factors plus tf-idf + truncated SVD features from tags, ingredients, and descriptions, 4) latent factors plus GloVe features, 5) latent factors plus LDA features, and 6) latent factors plus LDA features plus techniques.

Sampling strategy includes 1) random sampling; 2) stratified sampling where in the sample we end up with equal number of ratings of 5, equal number of ratings of 4, and equal number of ratings of 1 or 2 or 3 combined. All learning algorithms were trained on 8% of training set with one thing worth mentioning and one exception: 1) **feature extractions** are always based on the **whole training set**; 2) due to 1), **matrix factorization algorithm does not involve any sub-sampling** otherwise it would be essentially using different (latent factor) features than other learning algorithms and thus incomparable. In general, sub-sampling allows us to do cross-validation more efficiently without hurting too much on prediction performance.

Learning algorithm includes the ones mentioned in the method section, and their tuning parameters are tuned via 3-fold cross-validation on the training set implemented mostly through Scikit-learn [14].

5.2 Latent Topics

We try to understand latent topics extracted as in Table 1 while results from averaged distributed word representation (GloVe) or tf-idf with truncated SVD can pose a challenge for intuitive understanding. Note that the number of topics were selected as described in Feature Extraction section.

Table 1: Example latent topics extracted through LDA

Example Topics	Salient Words	Interpretation of the Topic
tags topic 1	desserts, cookies, fruit, brownies, pies, chocolate, tarts, berries, sweet	recipes for desserts
tags topic 2	low, healthy, fat, sodium, cholesterol, calorie, saturated, protein, carb, dietary	health-aware recipes
ingredients topic 1	baking, flour, sugar, vanilla, soda, powder, butter, egg, salt, milk, brown	baking related recipes
ingredients topic 2	juice, lemon, orange, zest, yeast, water, pineapple	recipes for juice
descriptions topic 1	family, recipe, years, friend, loves, ago, favorite, mother, old, husband	recipes for families
descriptions topic 2	salad, green, fresh, fressing, tomatoes, beans, dip, salsa, serve, peppers, bread	recipes for salads

5.3 Comparison of Prediction Performance

Baseline model (not included in Table 2.) is LASSO with regularization multiplier 0.1 on basic recipe features only (number of ingredients, cooking minutes, nutritions) has RMSE of 1.3472 on validation set and 0.9588 on training set.

Comparison Various combinations of features, sampling strategies, and learning algorithms' (with best tuning parameters through cross-validation on the training set) performance on the validation set are listed in Table 2 and observations are summarized below.

Tuning parameters: Several parameters are tuned via **3-fold cross-validation on the training set**. The best tuning parameters may vary slightly for the same model on different features and we do not have space to list all of them but they are mostly consistent. For **elastic net**, best l1 regularization ratio is 1.0 and regularization multiplier is 0.0005, indicating that the linear model found most features not useful. For **approximate rbf kernel regression**, best gamma for rbf is 0.01 and Nystroem approximation with number of components as 250 and regularization multiplier is 1e-05, with l1 penalty ratio of 0.15. For **k-NN**, best number of nearest neighbors is 30. For **random forest**, the best maximum depth for a single tree is 6 and number of trees is 60. For **XGBoosting**, best learning rate is 0.1 while number of estimators is 500. Note that we did not tune all possible parameters extensively due to practical constraints.

Sampling: For several of the learning algorithms, sampling can greatly speed up the cross-validation and training, without hurting the performance too much. Balanced sampling appears to hurt performance except in k-NN where the pattern is reversed. This is expected as for other learning algorithms, as they are learning a distribution that is significantly different from the validation set. k-NN simply remembers the training data, and hence a balanced sampling allows it to have a more complete coverage of the feature space, hence producing better predictions. It is a little bit unexpected that xgboosting did not perform well on 8% sample, and we further experimented with a larger sample (43%), where the performance has notably improved (we did not do this for every other algorithm for practical constraints).

Note that matrix factorization algorithm does not involve any sampling since its predictions merely takes the inner product of user and recipe latent factors, and for all learning algorithms, all the features (including user and recipe latent factors) were extracted based on the whole training set.

Features: Interesting observations are: 1) with only latent factors, several model (elastic net, approximate rbf kernel regression, xgboosting) were able to slightly outperform matrix factorization; 2) additional dense features from recipe textual information

may help in certain cases, in particular, GloVe and LDA have demonstrated notable improvement in performance. Cooking techniques do not appear to add much value. Note that except in the baseline, basic recipe features were not included since they did not appear to be useful and including them would require extra normalization steps as they are on different scales.

Overfitting We have noticed that most models have notable gaps between training RMSE and validation RMSE as in Table 2, indicating potential overfitting.

Best model: Based on performance on the validation set, the best model is approximate RBF kernel regression with latent factors plus LDA extraction of recipe textual information (tags, ingredients, descriptions). XGBoosting is the next best model. This is expected as both models are able to recover complicated non-linear feature interactions. Random Forest did not perform too well but it can also be due to tuning parameters are not tuned extensively.

We took a closer look at this best model. Only this best model generates predictions on the test set. On test set, its RMSE is 1.3020, MAE is 0.8517, implying that on average it is within 0.85 of the actual rating. Table 3 shows predictions on different slices of ratings, and shows that: 1) there is a tendency to over-predict ratings; 2) predictions were worse on lower ratings; 3) fraction of ratings in test set appear to be different from the training set, implying potential distribution shifts which could make the prediction even harder.

Loss Examples/Analysis We went through a few loss examples in test set based on predictions from the best model, and had the observations that understanding user preference is hard due to: 1) lack of previous user response, e.g., for user 349752, there were only two ratings in the training set; 2) the distributions of user rating are drastically different between training and test, e.g., user 176615 had 2011 ratings in the training set and all ratings were 5.0, and there is only one rating in the test set and it is 0.0 (which was unsurprisingly over-predicted to be 4.6).

Table 2: RMSE on validation dataset with different features, sampling strategies, and learning algorithms

Algorithm/Sampling	Features [validation RMSE (training RMSE)]				
	latent factors only	latent/tf-idf+SVD	latent/GloVe	latent/LDA	latent/LDA/tech
ElasticNet/8% random	1.2618 (0.8111)	1.2619 (0.8167)	1.2625 (0.8123)	1.2626 (0.8105)	1.2637 (0.8082)
ElasticNet/8% balanced	1.3096 (1.2279)	1.3137 (1.2234)	1.3147 (1.2107)	1.3121 (1.2129)	1.3137 (1.2151)
Approx Kernel Reg/8% random	1.2677 (0.8016)	1.2626 (0.8015)	1.2582 (0.8064)	**1.2578 (0.8039)**	1.2588 (0.8072)
Approx Kernel Reg/8% balanced	1.3160 (1.2140)	1.3169 (1.2117)	1.3205 (1.2102)	1.3294 (1.2072)	1.3255 (1.2182)
k-NN/8% random	1.3469 (0.0)	1.3466 (0.0)	1.3553 (0.0)	1.3407 (0.0)	1.3505 (0.0)
k-NN/8% balanced	1.2927 (0.0)	1.2689 (0.0)	1.2923 (0.0)	1.2746 (0.0)	1.2752 (0.0)
RandomForest/8% random	1.2700 (0.7680)	1.2812 (0.7516)	1.2737 (0.7579)	1.2714 (0.7637)	1.2729 (0.7556)
RandomForest/8% balanced	1.3031 (1.2059)	1.3312 (1.2076)	1.3209 (1.2030)	1.3061 (1.2074)	1.3187 (1.2008)
XGBoosting/8% random	1.2707 (0.6519)	1.2804 (0.6581)	1.2773 (0.6486)	1.2781 (0.6490)	1.2680 (0.6577)
XGBoosting/8% balanced	1.9952 (1.7010)	2.0157 (1.6959)	1.9952 (1.7011)	2.0132 (1.7025)	1.9927 (1.6979)
XGBoosting/43% random	1.2641 (0.6835)	1.2668 (0.6993)	1.2591 (0.6996)	1.2660 (0.6920)	1.2634 (0.6856)
Matrix factorization/no sampling	1.2763 (0.7966)	-	-	-	-

latent/tf-idf+SVD: latent factors plus tf-idf truncated SVD features from tags, ingredients, descriptions; **latent/GloVe**:textual features were based on GloVe embeddings; **latent/LDA**: textual features from LDA; **latent/LDA/tech**: includes techniques from cooking steps; also note that we were able to increase sample size for xgboosting since it has an efficient implementation, but there is no need to increase for balanced sampling since it did not help with prediction.

Table 3: Confusion Matrix of predicted ratings ($\hat{r}$) and true ratings (r)

	$\hat{r} \geq 4.5$	$3.5 \leq \hat{r} < 4.5$	$\hat{r} < 3.5$	total
$r \geq 4.5$	36.59%	23.56%	0.76%	60.91%
$3.5 \leq r < 4.5$	9.46%	12.62%	0.46%	22.54%
$r < 3.5$	5.26%	10.22%	1.08%	16.56%
total	51.30%	46.40%	2.30%	100%

6 Conclusion and Future Work

We show that it is feasible to view user and item latent factors from matrix factorization as embeddings, which can then be augmented with other features to feed into any reasonable supervising learning algorithms. The predictive performance may exceed original matrix factorization collaborative filtering algorithm. Dense features extracted from recipe textual information also appears to slightly improve performance when combined with user and item latent factors. The benefit of this approach is that it is flexible and general. On the other hand, the problem is fundamentally challenging partially due to distribution shifts between training and test set. Future work would be to extend this explorations into developing more recommender system specialized algorithms that may result in more performance improvements or apply stronger regularization and tune parameters more extensively to better reduce overfitting.

7 Contributions

Each team member contribute equally to the work, with Helena focusing more on experiments and analysis and Yinghao focusing more on setting up infrastructure for feature extraction and modeling.

8 Source Code

Source code can be found at: https://www.dropbox.com/s/yay87eh1sbi8dgk/src.zip?dl=0

References

[1] ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician 46*, 3 (1992), 175–185.

[2] BLEI, D. M., NG, A. Y., AND JORDAN, M. I. Latent dirichlet allocation. *Journal of machine Learning research 3*, Jan (2003), 993–1022.

[3] CHEN, T., AND GUESTRIN, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016), KDD '16, ACM, pp. 785–794.

[4] COTTER, P., AND SMYTH, B. Ptv: Intelligent personalised tv guides. In *AAAI/IAAI* (2000), pp. 957–964.

[5] HO, T. K. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (1995), vol. 1, IEEE, pp. 278–282.

[6] HUG, N. Surprise, a Python library for recommender systems. `http://surpriselib.com`, 2017.

[7] KOREN, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (2008), ACM, pp. 426–434.

[8] LUO, X., ZHOU, M., XIA, Y., AND ZHU, Q. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics 10*, 2 (2014), 1273–1284.

[9] MAJUMDER, B. P., LI, S., NI, J., AND MCAULEY, J. Generating personalized recipes from historical user preferences. *arXiv preprint arXiv:1909.00105* (2019).

[10] MELVILLE, P., MOONEY, R. J., AND NAGARAJAN, R. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai 23* (2002), 187–192.

[11] MNIH, A., AND SALAKHUTDINOV, R. R. Probabilistic matrix factorization. In *Advances in neural information processing systems* (2008), pp. 1257–1264.

[12] PAZZANI, M., AND BILLSUS, D. Learning and revising user profiles: The identification of interesting web sites. *Machine learning 27*, 3 (1997), 313–331.

[13] PAZZANI, M. J. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review 13*, 5-6 (1999), 393–408.

[14] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., ET AL. Scikit-learn: Machine learning in python. *Journal of machine learning research 12*, Oct (2011), 2825–2830.

[15] PENNINGTON, J., SOCHER, R., AND MANNING, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[16] RESNICK, P., IACOVOU, N., SUCHAK, M., BERGSTROM, P., AND RIEDL, J. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (1994), ACM, pp. 175–186.

[17] SIEVERT, C., AND SHIRLEY, K. Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces* (2014), pp. 63–70.

[18] SREBRO, N., RENNIE, J., AND JAAKKOLA, T. S. Maximum-margin matrix factorization. In *Advances in neural information processing systems* (2005), pp. 1329–1336.

[19] WILLIAMS, C. K., AND SEEGER, M. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems* (2001), pp. 682–688.

[20] ZOU, H., AND HASTIE, T. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology) 67*, 2 (2005), 301–320.