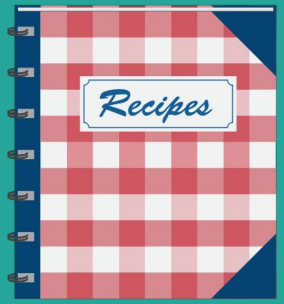


Recipe Recommender

Erdös Institute Data Science Bootcamp Spring 2024

Nadir Hajouji, Félix Almendra Hernandez, Nathan Schley, Ali Arslanhan, and Katherine Martin

https://www.youtube.com/watch?v=6edlBZ64TDk

Overview

Goal: Construct recommender system for recipes

Data: One million reviews of 500k recipes from food.com (Kaggle)

Research Question: Can we predict which recipes a user is likely to interact with using information about recipes we know they have already interacted with?

Industry use:

- Standalone product (à la NYT Cooking)
- Targeted advertising (Instacart, supermarkets)

Model

Want $f : \text{Users} \times \text{Recipes} \to \text{Reals}$;

One way of obtaining such a model:

- Find maps $\text{Users} \to V$ and $\text{Recipes} \to V$
- Set $f(\text{user}, \text{recipe}) = \text{user dot recipe}$

To find a good f, need a good way of embedding users and recipes in common vector space.

Data cleaning/processing

Recipes

Metadata: Name, User ID, Description, Recipe category, Keywords

Identify **main** Keyword groups

- Main ingredients (fruit, vegetables, dairy, non-meat protein, etc)
- Meal types (breakfast, holiday, kid friendly, weeknight, etc)
- Baking
- Regions (Mexican, French, etc)
- Appliances (pot, pan, ove, etc)

Continuous attributes: Cooking time, Calories, Fat Content, Fiber content etc.

Apply appropriate **transformations** and standardize

Data cleaning/processing

Recipes

Cleaned attributes

<u>Recipe ID</u>	Name	…	Dairy	Fruit	Dessert	Meat	…	log_calories	…
273649	Strawberry cake		1	1	1	0		2.97	

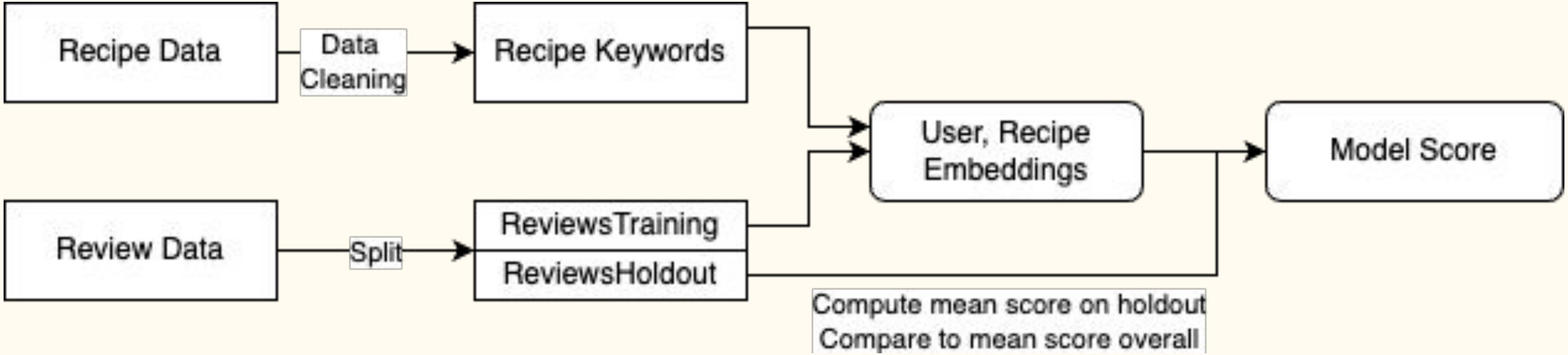
Reviews

Metadata: Recipe ID, User ID, Review, Rating

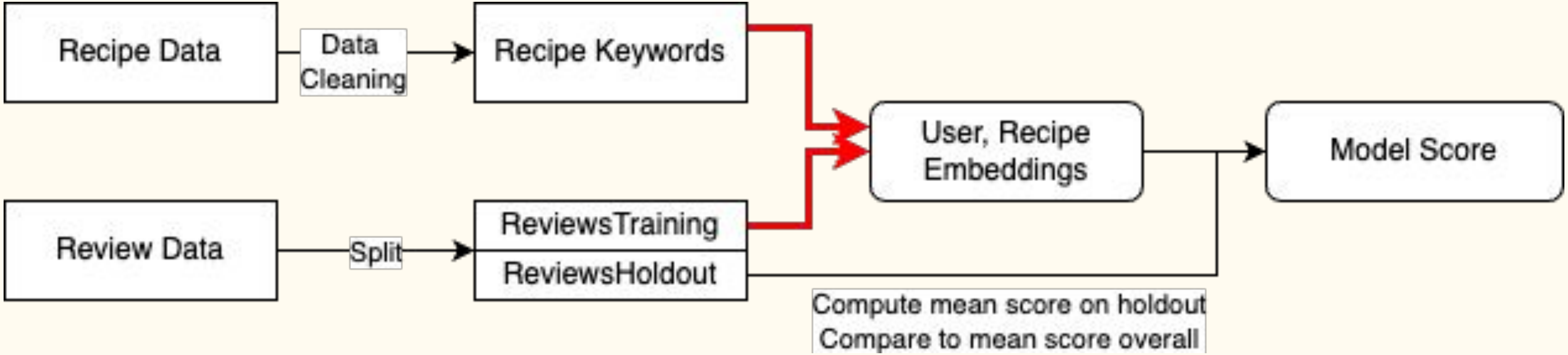
We made sure

1) Each recipe had <u>at least</u> 7 reviews
2) Each user had reviewed <u>at least</u> 5 recipes

Training and evaluating models: General Pipeline



Training and evaluating models: General Pipeline



What didn’t work: Recipe-based models

- Using “multi-hot” encoding and continuous attributes to obtain embeddings of recipes
- Define “user vectors” to be average of recipe vectors that the user has interacted with

What worked: User-based models

- Ignore recipe data

What worked: User-based models

- Ignore recipe data
- Encode (training portion) of review data into an “affinity matrix”
  (Affinity matrix = matrix where each row associated to user, each column associated to recipe, entry is 1 if user reviewed recipe, 0 otherwise)

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ & & & \vdots & \\ 0 & 0 & 1 & \dots & 0 \end{pmatrix}$$

n x m

$n =$ number users

$m =$ number recs

What worked: User-based models

- Ignore recipe data
- Encode (training portion) of review data into an “affinity matrix”
  (Affinity matrix = matrix where each row associated to user, each column associated to recipe, entry is 1 if user reviewed recipe, 0 otherwise)
- Use Singular Value Decomposition (SVD) to factor the affinity matrix

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ & & & \vdots & \\ 0 & 0 & 1 & \dots & 0 \end{pmatrix} = \begin{pmatrix} * & * \\ * & * \\ \vdots & \vdots \\ * & * \end{pmatrix} \begin{pmatrix} * & * & \dots & * \\ * & * & \dots & * \end{pmatrix}$$

n x k $\qquad$ k x m

$n =$ number users
$m =$ number recs
$k =$ hyperparam.

What worked: User-based models

- Ignore recipe data
- Encode (training portion) of review data into an “affinity matrix”
  (*Affinity matrix = matrix where each row associated to user, each column associated to recipe, entry is 1 if user reviewed recipe, 0 otherwise)*
- Use Singular Value Decomposition (SVD) to factor the affinity matrix
- **Interpret the factors as the user/recipe embeddings**

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ & & & \vdots & \\ 0 & 0 & 1 & \dots & 0 \end{pmatrix} = \begin{pmatrix} \dots & \text{user}_1 & \\ \dots & \text{user}_2 & \dots \\ & \vdots & \\ \dots & \text{user}_n & \dots \end{pmatrix} \begin{pmatrix} \vdots & \vdots & \vdots \\ \text{rec}_1 & \text{rec}_2 & \text{rec}_m \\ \vdots & \vdots & \vdots \end{pmatrix}$$

n x k $\qquad\qquad$ k x m

n = number users
m = number recs
k = hyperparam.

What worked: User-based models

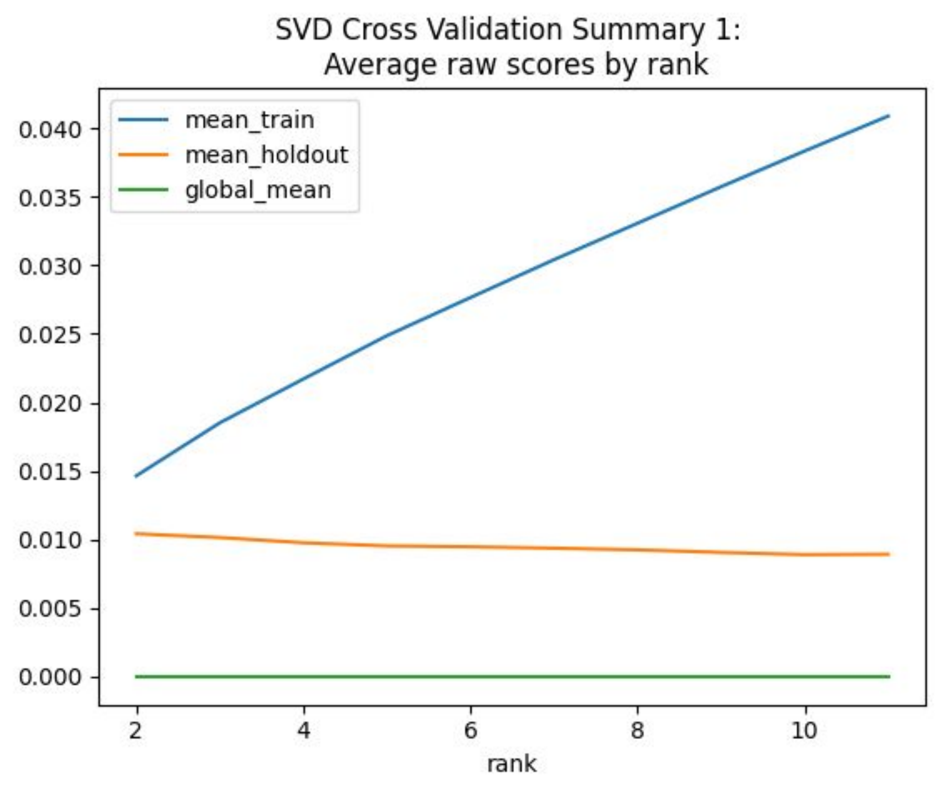
- Ignore recipe data
- Encode (training portion) of review data into an “affinity matrix”
  (Affinity matrix = matrix where each row associated to user, each column associated to recipe, entry is 1 if user reviewed recipe, 0 otherwise)
- Use Singular Value Decomposition (SVD) to factor the affinity matrix
- **Interpret the factors as the user/recipe embeddings**

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ & & & \vdots & \\ 0 & 0 & 1 & \dots & 0 \end{pmatrix} = \begin{pmatrix} \dots & \text{user}_1 & \\ \dots & \text{user}_2 & \dots \\ & \vdots & \\ \dots & \text{user}_n & \dots \end{pmatrix} \begin{pmatrix} \vdots & \vdots & & \vdots \\ \text{rec}_1 & \text{rec}_2 & & \text{rec}_m \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

n x $\boldsymbol{k}$ $\qquad$ $\boldsymbol{k}$ x m

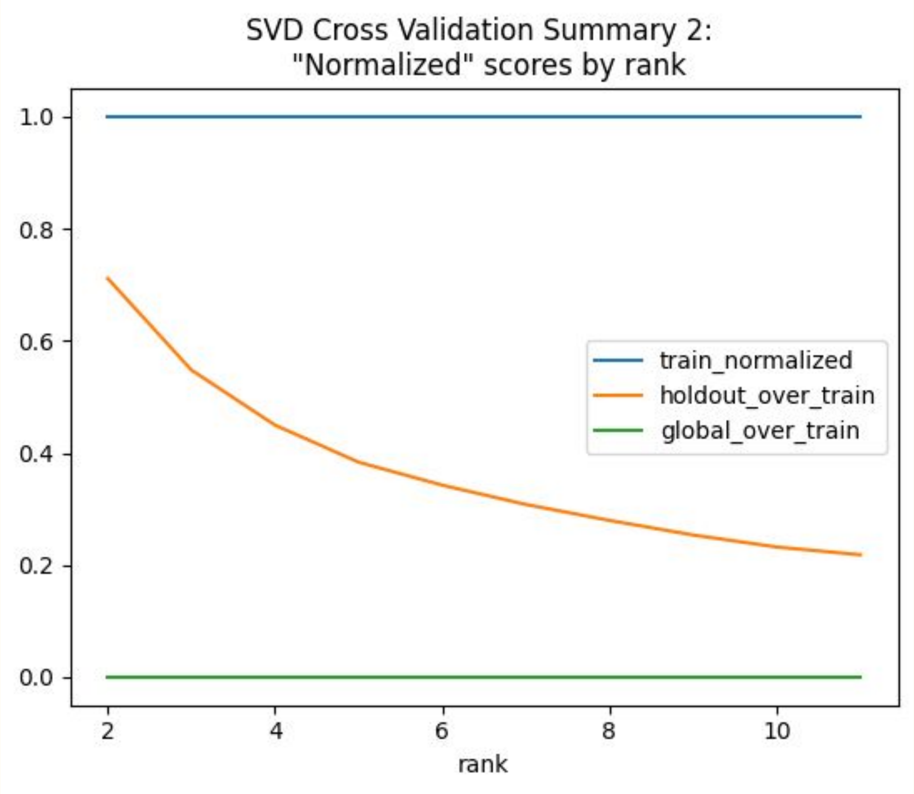
n = number users
m = number recs
$\boldsymbol{k}$ **= hyperparam.**
“The rank” = k

Results



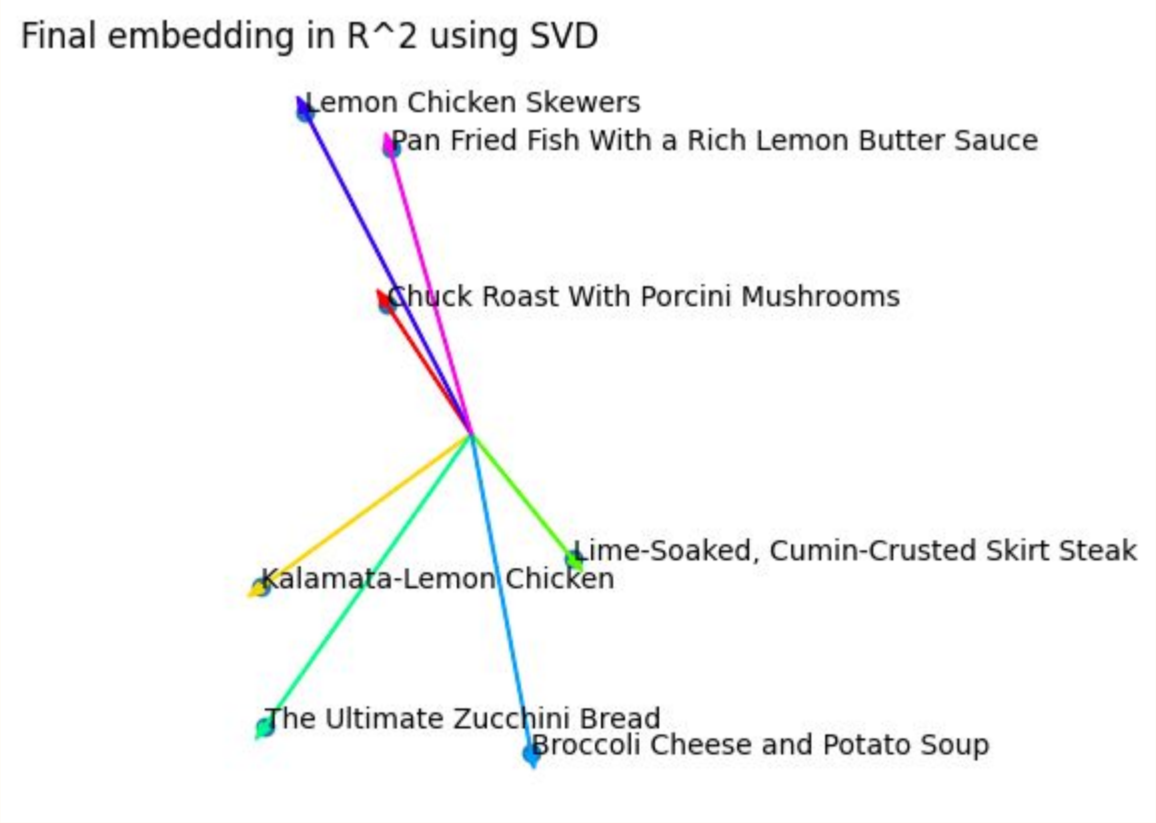
SVD Cross Validation Summary 1:
Average raw scores by rank
mean_train
mean_holdout
global_mean
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
2
4
6
8
10
rank

Results

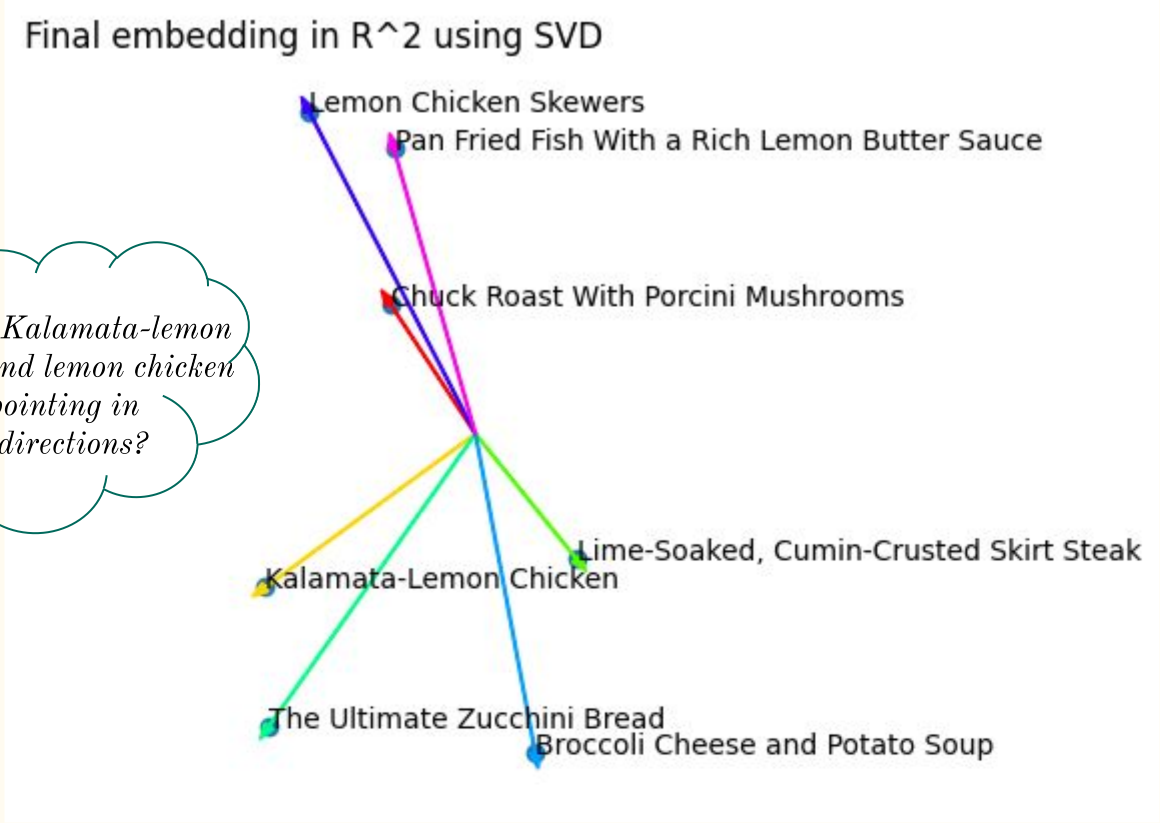


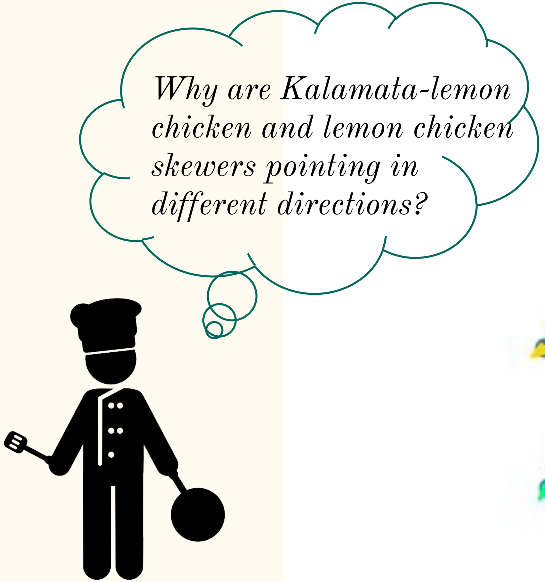
SVD Cross Validation Summary 2:
"Normalized" scores by rank
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10
rank
train_normalized
holdout_over_train
global_over_train

Visualizing the final model

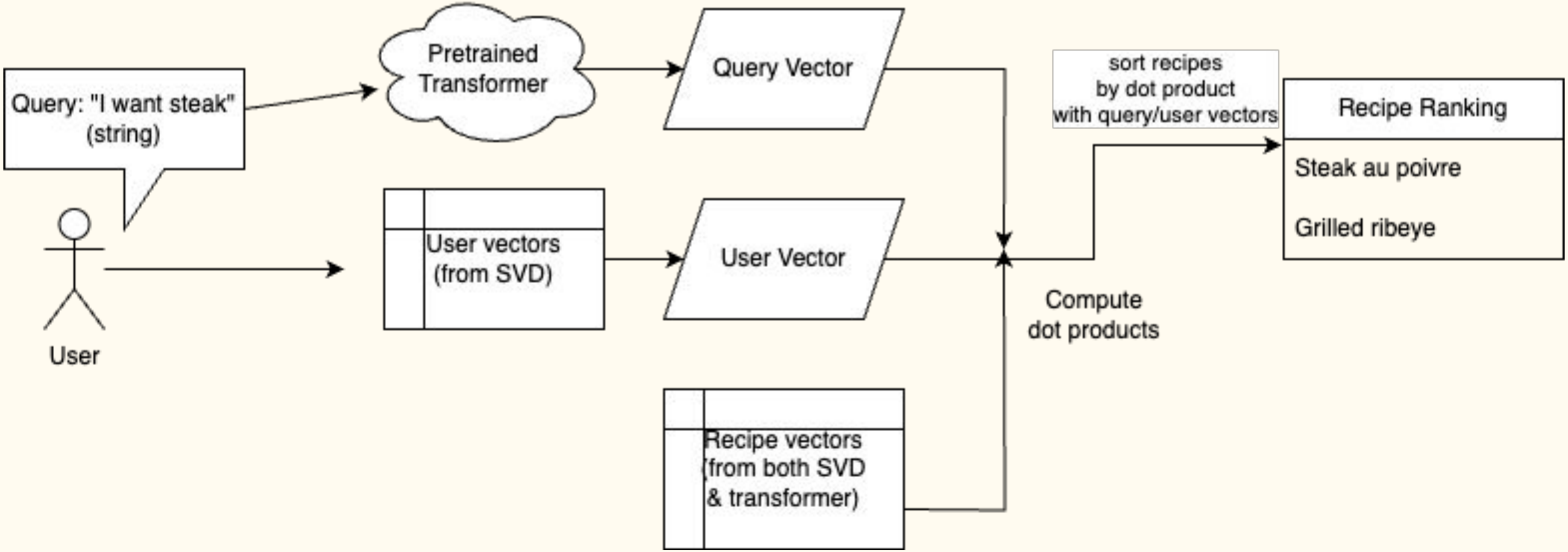


Visualizing the final model





Final product



Many thanks to...

Karthik Prabhu, Steven Gubkin, and Roman Holowinsky for your help and support for this project!