How I Built a Recommendation System for Fragrance

Kelly Peng Sep 16, 2017 · 8 min read

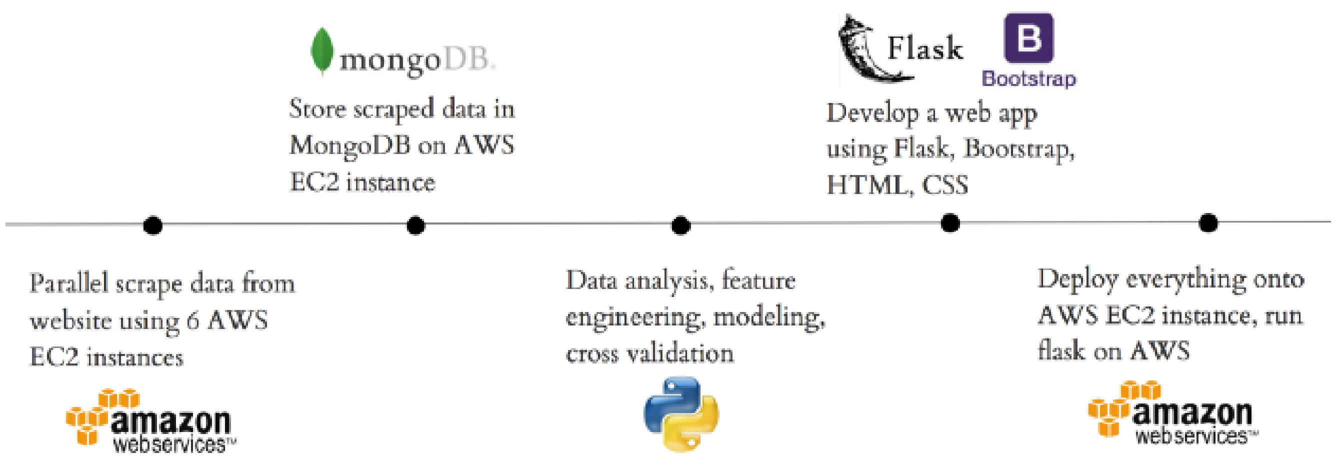
As Netflix CEO Reed Hastings says, *"You know, think about it, when you watch a show from Netflix and you get addicted to it, you stay up late at night. We're competing with sleep, on the margin."* From my point of view, the crucial part of the competition is about how to deliver the right content to the right people at the right time, it is about recommendation systems.

Recommendation systems have always attracted my attention. Not only because they are a big part of machine learning applications but also because they are related to understanding people. Whenever I'm listening to music on Pandora, buying things on Amazon, ordering clothes on Stitch Fix, watching movies on Netflix, reading news on StumbleUpon, sliding through photos on Instagram or Pinterest, I have a strong feeling that recommenders have filled in every single corner of our digital life, and they will continue to shape our lives in the future.

wear tells who you are. One's choice for perfume is affected by his/her gender, age, nationality, climate, mood, the season of the year, time of a day, etc. Since I've made several wrong decisions in the past, I was very cautious when deciding to buy a new one a few months ago. Then I thought, as a data scientist, I should be able to answer the question using my data science and machine learning knowledge. That's how I developed this idea — to help people find their signature perfumes.

1. Project Workflow

I scraped data from the largest fragrance forum in China using 6 AWS EC2 instances. Stored them in the 7th EC2 instance for MongoDB. After the data was ready, I did data analysis, feature engineering, modeling and cross-validation. Then I built a web app using Flask and Bootstrap. Finally hosted the web app on AWS.

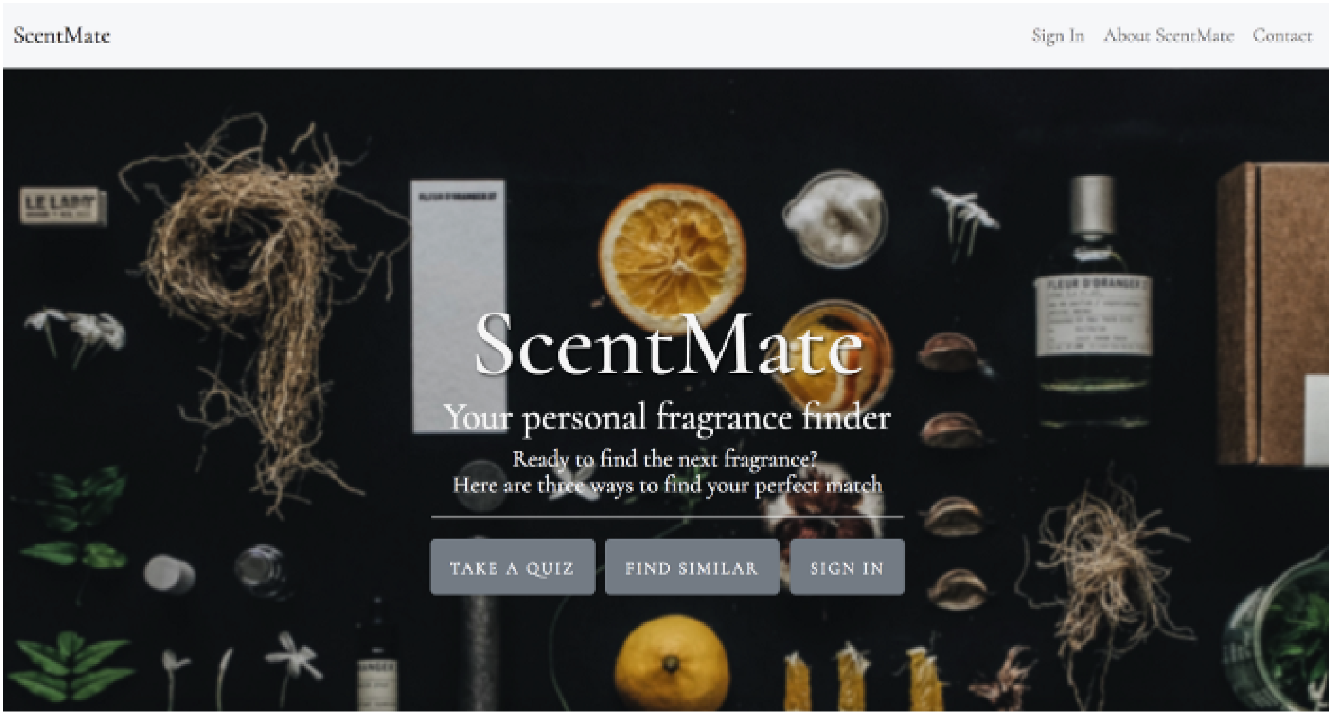


2. The Data

The data is scraped from the largest fragrance forum in China (Because the largest one in the US forbade me from scraping), it consists of two tables. One table is perfume data, with features such as brand, gender,

3. The Models

In this project, I implemented different approaches, targeting at different consumer groups. For cold start problem (new users without rating history), I will use content-based models. In the final web app, I will ask them to either take a quiz or enter a perfume you like to make recommendations based on perfume features. For users with rating histories, I will implement item-item similarity collaborative filtering model and matrix factorization model, then compare their performance. In the final web app, I will ask them to sign in their account, and personalized recommendations will be generated based on similar users.

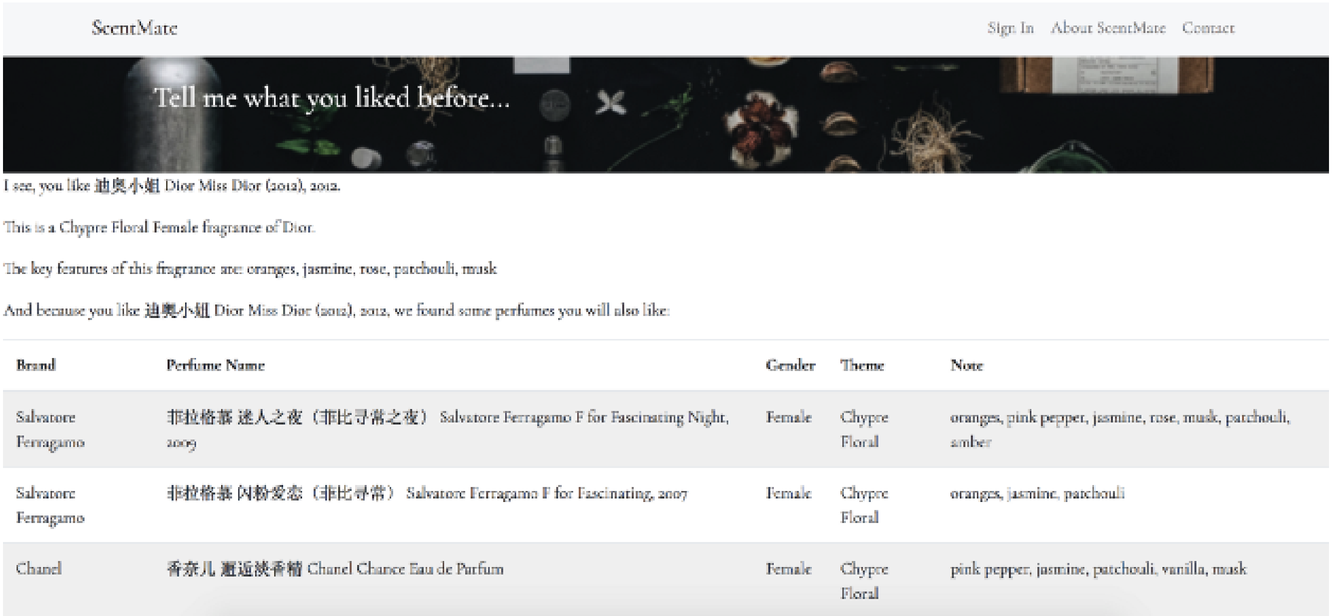


(Web app home page offers three options to find your next perfume)

Model 1: Content-Based Recommender

To build a content-based recommender, the only thing we care about is the perfume features. Since the data I collected has not had the feature

Allocation) to user comment texts for topic modeling. After comparing the topics generated from each method, I found that 12 topics generated by LDA are more interpretable. Thus I manually labeled the 12 LDA topics, combined them together with other perfume features such as gender, note, and theme, constructed perfume matrix, then built content-based similarity model based on Jaccard similarity.



Brand	Perfume Name	Gender	Theme	Note
Salvatore Ferragamo	菲拉格慕 迷人之夜（菲比寻常之夜） Salvatore Ferragamo F for Fascinating Night, 2009	Female	Chypre Floral	oranges, pink pepper, jasmine, rose, musk, patchouli, amber
Salvatore Ferragamo	菲拉格慕 闪粉爱恋（菲比寻常） Salvatore Ferragamo F for Fascinating, 2007	Female	Chypre Floral	oranges, jasmine, patchouli
Chanel	香奈儿 邂逅淡香精 Chanel Chance Eau de Parfum	Female	Chypre Floral	pink pepper, jasmine, patchouli, vanilla, musk

(A snapshot of the content-based recommender)

Model 2: Collaborative Filtering Recommender

The two primary areas of collaborative filtering are neighborhood methods and latent factor models. Neighborhood methods are centered on computing the relationships between items or, alternatively, between uses. The item-oriented approach evaluates a user's preference for an item based on ratings of "neighboring" items by the same user. A product's neighbors are other products that tend to get similar ratings when rated by the same user. Latent factor models, on the other hand, are an alternative approach that tries to explain the ratings by characterizing both items and users on latent factors inferred from the rating patterns. For perfumes, the discovered factors might measure the obvious or less well-defined dimensions such as floral or citrus theme, orientation to different sexes, or completely uninterpretable dimension. For users, each factor measures how much the user likes perfumes that score high on the corresponding movie vector.

In this project, for users with rating history, I applied item-item similarity model from the neighborhood methods and UV decomposition model from the matrix factorization method. The evaluation metric I used to evaluate my recommendations is RMSE (Root mean square error).

The baseline model is to randomly recommend perfumes for users, which means predicting the rating to be the average rating (7.25 out of 10) for everything, it gave me a RMSE of 2.20.

filled with value), and when the matrix is too sparse, neighborhood methods cannot perform well because little information going into each prediction.

Then I applied matrix factorization model, fitted via stochastic gradient descent. After tuning the number of factors, regularization on the linear term as well as the interaction term. Matrix factorization model gave me the best performance (RMSE: 1.95). Thus I used matrix factorization model as my final model for my collaborative filtering recommender.

$$\begin{bmatrix} 5 & 2 & 4 & 4 & 3 \\ 3 & 1 & 2 & 4 & 1 \\ 2 & & 3 & 1 & 4 \\ 2 & 5 & 4 & 3 & 5 \\ 4 & 4 & 5 & 4 & \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \\ u_{51} & u_{52} \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \end{bmatrix}$$

(Figure above: UV-decomposition of matrix M)

The performance of collaborative filtering models are shown in the picture below:

	Base Model	**Item-item Similarity**	**Matrix Factorization (UV decomposition)**
Methodology	Predict mean for everything	Recommend based on the most similar items found by user ratings	Decompose utility matrix into two matrices with latent factors
Performance on Test Set	RMSE: 2.20	RMSE: 7.32	RMSE: 1.95
Reason		Utility matrix too sparse lead to little information going into each prediction	Should perform better if more data available

came to my mind is: "How did you validate your recommender's performance?" Because it is the most challenging part of any recommendation systems. In my project, for content-based recommender, since it is unsupervised and recommended completely based on the perfume features, I validated it by letting my classmates try and see the result. The recommendations in general were good, especially the "Find Similar" function on the home page, but the biggest drawback is that perfume price data is not available, a customer who loves Chanel perfumes will not probably go to a mass-market brand even though they have similar ingredients. However for the "Take a Quiz" function on the home page, if a user only selected a few features, the result will be off because the model is building a vector based on all the features a user selected and calculate Jaccard distance between the user vector and every perfume.

For the collaborative filtering recommender, the first cross-validation method I tried was leave-one-out cross-validation, but it took me 18 hours to finish one cycle, which is too computationally expensive, thus I switched to manual K-fold validation. First, I manually removed users with less than 3 ratings, treat them as part of cold start problem and let them use the content-based recommender instead. Then, for users with 3 or more ratings, I made sure every user exists in my training set, validation set, and test set. Therefore I can use the user's rating in the training set to fit the model, predict the rating of the user's other ratings in the validation set, and tune the model based on RMSE on the validation set. And finally, I used the user's ratings in the test set to evaluate the final model.

5. Limitations and Future Work

not probably go to a mass-market brand even though they have similar ingredients.

- User demographic data not available: People's perception of each fragrance is different across cultures and geographies, my data comes from a Chinese fragrance forum, the feelings people feel about each perfume there does not reflect the feelings people feel about each perfume in other countries.

- Only 20% of all perfumes have been reviewed: one advantage of good recommendation systems is its serendipity. But in order to retrieve customers' feelings of each perfume, I can only look at perfumes that already have user comments. However, 80% perfumes do not, either because they are not popular or they are new. Therefore, in order to make my recommendation systems work better, more domain knowledge of perfume and feature engineering work is needed, so as I don't need to rely on user comments to gain information of how do people feel about each perfume.

2) Customers' tastes change over time

- The current models that I have are static. But in reality, customers' inclinations evolve, leading them to redefine their taste. Similarly, product perception and popularity constantly change as new perfumes emerge. Thus, the system should account for the temporal effects reflecting the dynamic, time-drifting nature of user-item interactions. Thus for future work, I should also collect user rating date, and my collaborative filtering model should take temporal effects into consideration.

6. Final thoughts

During the two-week project experience, I got to realize there's so much I can do with recommendation systems, and how powerful they can be. They are not only about perfumes or any consumer goods. They can be about the courses you may like on MOOC websites, or the people you may like on dating apps, needless to say, news articles, books, music… The main idea behind is about how do you know about your customers, how much domain knowledge you have, and how are you utilizing data science and machine learning to help you implement your understandings. I would more than happy to discuss with you if you happen to be interested in this field as well.

> The code can be found here: *github.com/kellypeng/scentmate_rec*
>
> Connect with author: https://www.linkedin.com/in/pengyuwei/

References:

1. Anand Rajaraman and Jeffrey Ullman, *Mining Massive Dataset, Chapter 9 Recommendation Systems*
2. Yehyda Koren, Robert Bell and Chris Volinsky, *Matrix Factorization Techiniques For Recommender Systems*
3. Yehuda Koren, *Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering model*
4. Stephen Gower, *Netflix Prize and SVD*
5. Jiunn Haur Lim, *Predicting User Ratings with Matrix Factorization*

Machine Learning Recommendation System Data Science Data

Data Driven