Personalized Recipe Recommendation Using Heterogeneous Graphs

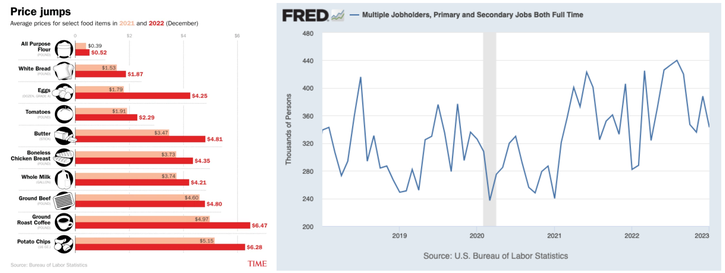
Nicholas B. DeGroot[1], Jonathan Herke[2], Jay Yu[2]

1: HDSI, University of California San Diego, USA 2: TigerGraph, USA

Abstract

• TODO

Motivation



TA Note: I plan on pulling the data and creating graphs more stylistically in-line with the project.

This project was born from some recent insights in the economy:

- Roughly 388,000 Americans are currently working more than two **full time** jobs (nearly 5% of the workforce).
- The average cost of groceries for a household has risen by 13.5% over the last year.

In essence, people can't afford to eat out due to budget constraints, but can't afford to eat at home due to time constraints. To this end, we propose a system that can automate the process of meal planning and grocery shopping. By automatically generating meal plans custom tailored to the users taste preferences, users save the time previously spent planning meals and the money previously spent eating out.

Collaborative Filtering

The obvious solution to this problem is with *collaborative filtering*. Collaborative filtering works by finding users with similar tastes and recommending items that those users have liked.
We chose to define similarity as the pearson correlation between user's reviews, with the rating being equivalent to an edge weight.

$$Sim(u,o) = \frac{\sum_{i \in I_u \cup I_o}(R_{u,i} - \bar{R}_u)(R_{o,i} - \bar{R}_o)}{\sqrt{\sum_{i \in I_u \cup I_o}(R_{u,i} - \bar{R}_u)^2 \sum_{i \in I_u \cup I_o}(R_{o,i} - \bar{R}_o)^2}}$$

This approach has a number of advantages:

- Cheap calculation (particularly in a Graph DB)
- Explainable / understandable

But doesn't come without its drawbacks:

- Ignores extra recipe information
- Ignores higher-order relationships (i.e. ingredients)

Expanding with LightGCN

We can fix a number of these drawbacks by drawing on the power of graph neural networks. By extending the work of He u. a. [1], we can use a graph neural network to learn latent features for both users and recipes, and use them to make recommendations.

The model can be summarized as such:

- Goal: Learn latent features for users and recipes. Users with a high affinity for a recipe should have a high latent feature similarity (defined by cosine similarity).
- Each iteration, the model updates the latent features of each user and recipe by taking a weighted average of the latent features of their neighbors.

$$\mathbf{x}_i = \sum_{j \in \mathcal{N}(i)} \frac{1}{\sqrt{\deg(i)\deg(j)}} \mathbf{x}_j^{(l)}$$

- Loss: Bayesian Personalized Ranking loss (BPR). Given a positive ($+$) and negative ($-$) user-recipe interaction:

$$\mathcal{L} = -\frac{1}{N} \sum_{(u,+,-) \in \mathcal{D}} \log \sigma(\mathbf{x}_+ - \mathbf{x}_-)$$

We can further improve the performance in two ways:

- Concatenating known recipe information (e.g. nutritional information, cooking time, etc.) with the learned latent features.
- Weighing the edges of each recipe with the review score.

Higher-Order Relationships

While an improvement over baseline collaborative filtering, LightGCN still ignores higher-order relations such as shared ingredients and user-generated tags.
TODO.

Comparison of Methods

TODO

Conclusion

TODO

References

[1] He, Xiangnan ; Deng, Kuan ; Wang, Xiang ; Li, Yan ; Zhang, Yongdong ; Wang, Meng: *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*. 2020. – URL https://arxiv.org/abs/2002.02126

Code at github.com/nickthegroot/recipe-recommendation
Paper available by scanning the QR code