Project 4 Summary
Recipe Recommender
Flora Xinru Cheng
November 17, 2019

For this project my goal was to build a working recommendation system for recipes, based on topics and cuisine type. I also wanted to make a working Flask app to get a usable product at the end. Although I did not get to finish the Flask app, I was happy with the recommendation results from my recommender. I also had to make some important design decisions in deciding which datasets and tools to use to best achieve my goals.I would like to continue working on this project in the near future.

Design Decisions

Because this was my first unsupervised learning project with a self-defined goal, I had to pause at several instances to reconsider whether the tools I was about to apply were really helpful in solving the problem and enhancing the data story. This proved beneficial, and I was able to avoid some issues brought up during other people's presentations (such as "where does Kmeans fit in in your analysis?").

I spent longer cleaning the RecipeBox data than expected. Since I wanted to test out the NLP tools we learned, as well as get a finished minimum viable product on time, I made the mvp using the smaller, cleaner recipe box dataset instead. This helped me get a good sense of what the topics look like after dimensionality reduction, and was helpful when comparing results from different models later applied to the larger dataset.

I was aware from the beginning that my goals for the project were ambitious. And even though I had almost 2 days allotted for learning to make my first Flask app as planned, I decided against it, and focused on cleaning the code and practicing my presentation instead. I am also less interested in web development compared to data analysis, and decided optimizing my model and interpreting the results should take priority over making a basic web app purely for demonstration purposes.

Issues, "Gotchas", Materials for Blog:

Dimensionality Reduction

- Before LSA and NMF, the dimension of the vectorized documents depends on how many unique word tokens there are
- After dimensionality reduction, the corpus is transformed into a sparse matrix, where the number of features is the number of topics
- A more rigorous way to determine the optimal number of topics is to use a method such as GridSearchCV

- The top 5 topics generated at k=10 are almost identical to the ones of k=5, so I decided not to spend time tuning that parameter when using LSA and NMF (this could be different for LDA which I plan to try later)

Cosine Similarity

- This sklearn function can take either ndarrays or sparse arrays as inputs.
- Initially it was running really slowly for me, and crashed. This was because I only used one input argument X, so it was calculating the pairwise cosine similarity between every single document inside the entire vectorized dataset. For the purpose of finding recipe recommendations based on cosine similarity, calculating one-vs-all was sufficient (the similarity between the one selected recipe and every other recipe in the dataset). This did not cause problems with running the code.
- For a larger dataset, if cosine similarity between one-vs-all pairs take too long again, one option would be to look at a “subset” of the data by performing clustering first, then find the cosine similarity between pairs of documents within only that cluster. A potential issue for this would be if the clusters are based on a certain ingredient or course of meal (such as “chocolate” or “beverages”), and the model might over fit for that topic and perform poorly for others.

Soft Cosine Similarity

- I found out there is an option in Gensim to calculate soft cosine similarity, where words that have related meanings are also considered “equal” — “Hi” and “Hello”, “President” and “Prime Minister” are vectors in the same direction, with cosine similarity equal to one.
- I was initially interested in testing this, but was short on time. Also I suspect that due to the nature of the dataset, using soft cosine similarity could lead to unique ingredients being lost in the results (considering distinct ingredients to be the same). So I decided not to implement it. But I would like to test out this hypothesis at a later time.

Kmeans Clustering

- I performed Kmeans clustering using both LSA and NMF, but decided to remove them from the submitted version of my code, as well as this presentation, because I was not able to interpret them in a way that I deemed would help with my narrative.
- There remains the option to cluster the topics, then only find similar recipes (recommendations) within the cluster of the target recipe. I did not follow this path during the allotted time because my initial motivation was to recommend recipes with similar ingredients but from cuisine types other than that of the target. Also clustering sacrifices interpretability when visualized in high-dimensional space. Although I learned about tools

for high-dimensional cluster visualization such as t-SNE and UMAP that could be useful for future projects.

Further Work

The next step of this project is to continue improving the recommendation results with the current model (TF-IDF + LSA) by doing more thorough preprocessing (trying different stemmers and lemmatizers, perhaps using SpaCy), and training it on more data. I also want to use Gensim and see if LDA gives better recommendations than LSA.

I would like to scrape another recipe website such as BBCfood. I can then use some of that data as the test data, and the rest to increase the training dataset.

Then the next step would be to develop a simple Flask app, with interactive elements (such as a search bar with potential recipes to start from based on keywords typed in by the user, a button to click on to get recommendations, and displaying recommendations in a neat format. Another step beyond that would be to cluster the topics to see if the clusters match cuisine types.

Building a cuisine type classifier using the smaller What’s Cooking dataset would also be useful. Then I can predict cuisine types for recipes in the large dataset, and compare that with how they cluster. It will also help to have metrics from supervised learning after generating recommendations in unsupervised learning, to know how good the results are, and when to stop optimizing.