**Kaggle**

Introduction

Sberbank, Russia’s oldest and largest bank, The goal of the Sberbank Russian Housing Market Kaggle competition

Workflow

Exploratory Data Analysis

Feature Engineering and Selection

Multiple Linear Regression

Random Forest

XGBoost

Conclusions

Future Work

**Capstone**

Introduction/Motivation

Workflow

EDA  
Sentiment Analysis

Content-Based Filtering

Collaborative Filtering

Social Network

Base Recommendation

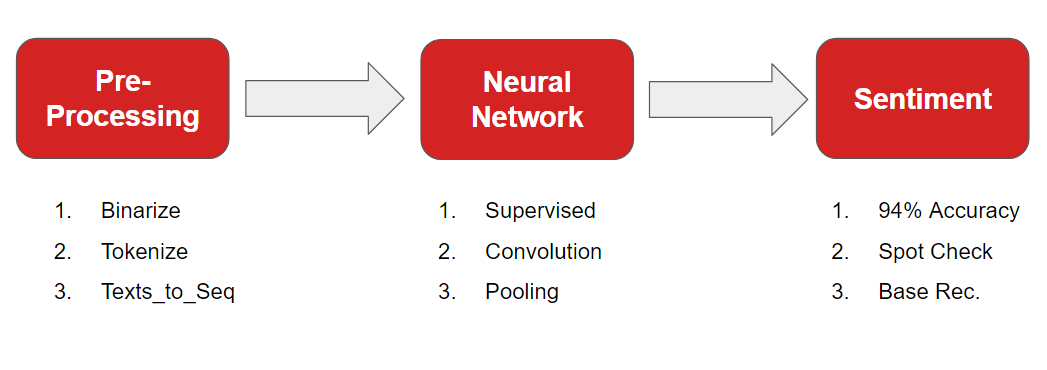
Data Pipeline

Conclusions & Future Work

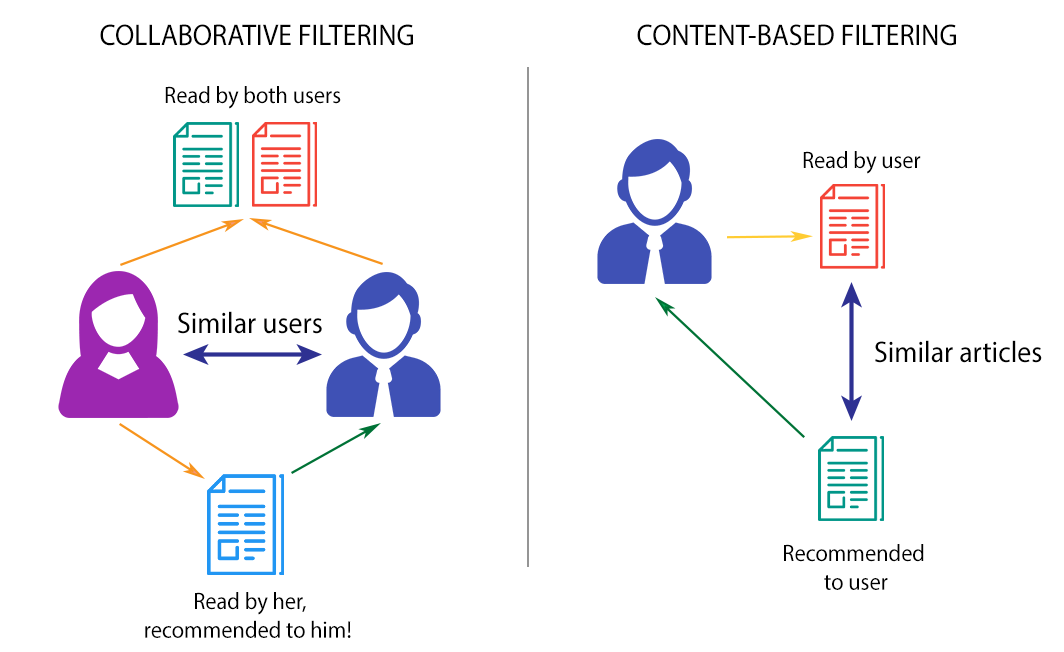
NLP

First, we decided to analyze the Las Vegas subset using Natural Language Processing (NLP), a machine learning technique that aims to understand human language with all its intricacies and nuances. The 800,000 reviews were divided into low ratings of 1 and 2 stars and high ratings of 5 stars. Our goal was to create a supervised learning neural network that could predict the sentiment of a review as positive or negative based on the language used. Restaurant reviews with ratings of 3 or 4 stars were thrown out due to the lack of consistency between reviewers (i.e., one reviewer’s 3 star review could be another’s 5 star review).

The low rated reviews were given a rank of zero and the high rated ones were given a rank of one. The text of each review was then tokenized and converted to a sequence using the keras package. A neural network was set up in keras as well using a convolutional filter, pooling filter, and both ReLU and sigmoid activation function to predict the sentiment of each review as either 0 or 1. The final model was 94% accurate and was spot-checked on newly-written reviews and on the remaining 3 and 4 star reviews. The model did quite well for our purposes and so we fit all 800,000 reviews using the sentiment analysis neural network and then averaged the reviews by restaurant and used the results as a new feature in the location-based recommendation (link to location based section). The overall process for the sentiment analysis is outlined below.

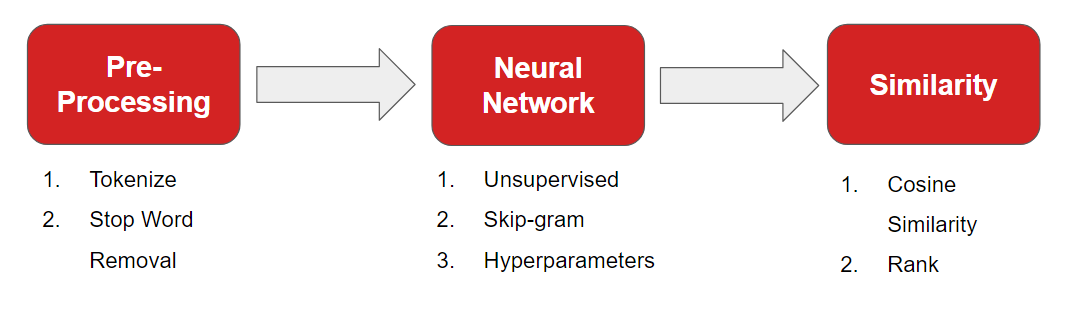


We also wanted to use NLP to make content-based recommendations for our app. Item-to-item similarity is calculated to make these types of recommendations. Think of when you read an article online: there is usually a side menu or a section at the bottom of the page that recommends new articles based on your interest in the current article. Often, these new articles are written on the same or a similar topic. That’s a content-based recommendation!



Content-based recommenders can use a user’s profile and past ratings to make new recommendations to the user. This, however, presents the cold-start problem, an issue that arises for a brand-new user who has no rating history. To combat this, our NLP recommendation is based on a keyword search rather than the user profile. So, typing “tacos” into our app should bring up a ranking of Mexican joints.

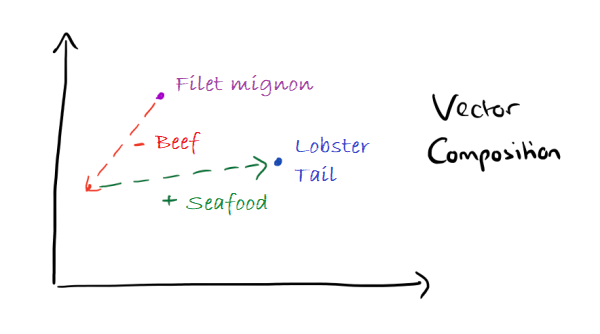
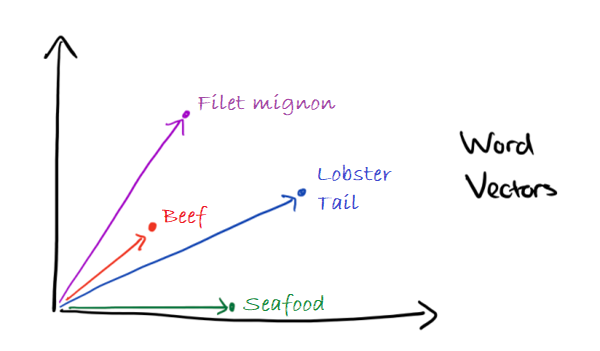
To make this happen, we once again decided to implement a neural network to understand the language used in the reviews. We used the Word2Vec function from Spark MLlib to create the model. Our process is outlined in the figure below.



To pre-process the data, we concatenated the reviews for each business together for a total of 6,199 restaurants. Then, we tokenized the reviews and removed all stop words. Stop words are common words in the English languages that provide function within a sentence but no context, such as “of”, “the”, and “has”.

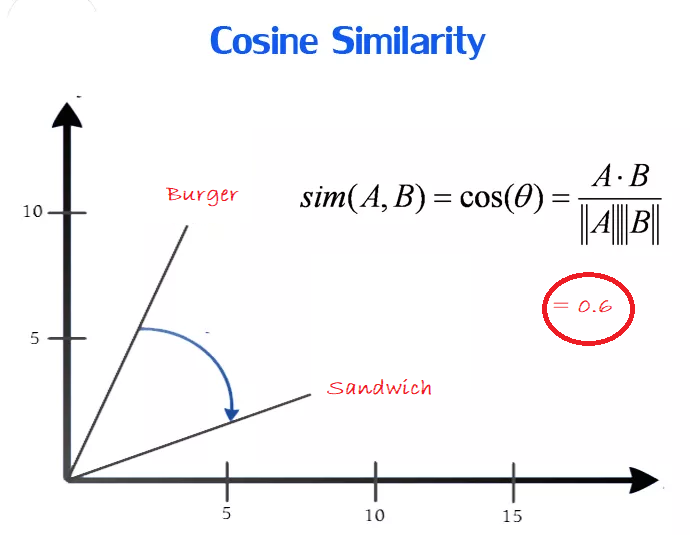
Word2Vec then translates each word into a vector of 100 features. These vectors are located in a feature space of 100-dimensions, with similar words closer together and unrelated words farther apart. For example, the vectors for “ice cream” and “frozen yogurt” should be pointing in nearly the same direction, but the vectors for “delicious” and “disgusting” would be far apart.

Once the words are translated into vectors, we can perform some word algebra! We can add or subtract the vectors from one another to find a new word.



The word vectors above have been flattened into a 2D feature space. Beef and filet mignon are both foods from a cow, while seafood and lobster tail both come from the sea. Lobster tail and filet mignon are exquisite and expensive types of beef and seafood. If we take filet mignon, take away the fact that it is beef and add in a new category of seafood, we end up with lobster tail!

In addition to word algebra, we can also determine how similar two words or documents are based on cosine similarity. Mathematically, cosine similarity is the dot product of the two vectors divided by the product of the magnitudes of those two vectors. This calculates the angle between two vectors. The smaller the angle, the more similar the words, the closer to 1 the value becomes. Larger angles are more dissimilar and are closer to negative one. Vectors perpendicular to one another will have a cosine similarity of zero.



In the figure above, “burger” and “sandwich” point in somewhat similar directions and have a similarity of about 0.6. Below, we can see the results of a similarity search for the word “Chinese”.



Since the business reviews are more than a single word and a user may want to search using multiple words as well, Word2Vec averages the vectors of all the words together and then calculates the similarity between the user’s keywords and each of the available restaurants’ reviews. The results are then ranked from most similar to least and returned to the user on the map.

**Conclusion**

Yelper Helper is a user-friendly interface powered by a robust and varied recommender and supported by an efficient data pipeline making it a quick and easy way to find nearby restaurants the customer will love!

Having recommendations available for all levels of interaction with the app provides quick suggestions for the casual digital passerby and yet promotes more consistent user engagement with the benefit of more personalized results. The location-based recommendations aim to provide a quick and dirty service for passing users. The NLP neural networks of the content-based recommendations allow users to filter restaurants by cuisine or dish. Collaborative filtering and social network recommendations provide individualized recommendations based on personalized taste and friends’ opinions.

Building the machine learning models using Apache Spark and setting up a Flask-Kafka-RDS-Databricks pipeline creates a powerful and scalable system robust to working with big data and a continuous stream of user requests.

With more time, we would improve Yelper Helper with the following ideas.

1. Scale the recommender to include all cities, restaurants, and businesses.
2. Convert it to a mobile app to automatically input the user’s location for quicker recommendations
3. Use A/B testing to design an optimal user-interface, possibly including pop-ups of helpful tips or friends’ reviews of recommended restaurants.
4. Increase the speed of the engine (by … low latency …?)
5. Collect data on user’s final decisions and subsequent ratings to potentially provide local restaurants with insights on how to improve their business and attract customers.
6. Build a Latent Dirichlet Allocation (LDA) model or Doc2Vec model to improve and vary the NLP recommendation method.
7. Allow users to personalize their experience with favorite lists or the option of planning meals for a vacation.

In a short two-week period, we learned a great deal about working effectively as a team by using an agile approach to divide up the labor and regularly meet up to discuss progress and problems. We gained experience in SQL queries, Amazon Web Services, PySpark programming language, and Kafka streaming. We implemented several machine learning techniques and built an entire data pipeline to create a useful and professional product for the modern consumer. Enjoy the fruits of our labor *here*! <<add link to demo>>