capstoneProposal

March 7, 2018

1 Machine Learning Engineer Nanodegree

1.1 Capstone Proposal

Claudia Cassidy March 3rd, 2018

1.2 Proposal: Recipe Recommender

1.2.1 Domain Background

"What's for dinner?". That is the dreaded question at the end of a work day when families get together after school and work and it is time to prepare dinner. Unless you are well organized, plan meals a week in advance, shop for all the ingredients required, and have readily available recipes, it can be very challenging to prepare a family dinner that is both delicious and nutritious.

You can search through cookbooks and thousands of recipes online, but it takes time to find just the right recipe that can be made quickly, that everyone will like and that you have all the ingredients for. Many cooks repeat the same tried and true recipes each week. It can get boring after a while.

Alternatively, you can order pizza, buy fast food, prepare frozen dinners or make mixes from a box. Food delivery services like Blue Apron have tried to solve this problem by sending a box of ingredients and directions for cooking them. They can be expensive and limited in portion sizes.

When shopping at the grocery store, most people I see do not carry a shopping list. They buy main ingredients, such as meats and vegetables, and stock up on staples like onions, garlic and spices. Sometimes, ingredients go unused and have to be discarded.

Supermarkets often feature a meal of the day and showcase all the ingredients needed to make that meal along with a demo and recipe card. The meals are general, not personalized. From what I've seen, not many people buy the featured meal items.

This proposal is to build a Recommender System using Natural Language Processing to parse the ingredients, title, description and reviews of recipes in order to make accurate predictions for which recipes to recommend.

Deep learning methods have become a powerful tool to process Recommender System tasks such as news, music, and movies. When there are thousands of choices, Recommender Systems can help people deal with choice overload. Choice overload is a cognitive process in which people have a difficult time making a decision when faced with many options. With thousands of recipes to choose from online, it can be overwhelming to find the best one for you.

In recent years, Recommender Systems have been developed which give people personalized recommendations. For example, movies on Netflix and products on Amazon are recommended based on algorithms which predict what a person will probably like based on user preferences, item features and items a user has looked at in the past. Recommender Systems guide users in a personalized way to discover products or services they might be interested in from a large space of possible options.

1.2.2 Problem Statement

There are two goals for this project:

1 - Make it easier for home cooks to find recipes at the end of the day, spur of the moment, which use the ingredients already at home.

2 - If an item is on sale at the store, for example, if there's a sale on lamb, then recommend recipes for that person which contain that item, so that the person can buy the featured item and any extra ingredients, if needed, to make the recipe at home.

The goal is to build a tool which will make it easier and less expensive for home cooks to find recipes that they like. This project will use Deep Natural Language Processing to build a Recommender System for recipes. Once trained, the algorithm should be able to recommend relevant, personalized recipes for each user. The front end will be an iOS app.

1.2.3 Datasets and Inputs

The data will consist of a csv file of recipes scraped from popular cooking websites such as AllRecipes.com. Using BeautifulSoup, we can scrape a sample of at least 1,000 publicly listed recipes.

User Preference Training Data: We will rely on the user having an account on the iOS app and a saved list of favorite recipes.

If the user does not have any saved favorites then we will use the features "Title", "Description", "Ratings", "Number of Reviews" and "Reviews" fields more heavily until the user saves recipes.

The iOS app can also have an onboarding processes to ask users what their preferences are, for example "vegetarian", "low calorie", "5 ingredients or less".

The app will have search functionality so we can also save the user's search queries to enhance the recommendations.

When training the data, we will rely on the fields "Ratings" (1-5), "Number of Reviews", and "Review". Number of Reviews tells us how many times people reported making the recipe. Ratings tell us how much people liked the recipe, with 5 being the highest and 1 the lowest rating. Ratings and Number of Reviews indicate general popularity. There are up to six Reviews for each recipe, which consist of a paragraph of feedback from people who felt strongly enough about a recipe to post their thoughts. "Title" is the name of the recipe and "Description" is a sentence describing the recipe as entered by the recipe's author.

Item-Item distance will be calculated to see which ingredients most often go together, for example meatballs and spaghetti, peanut butter and jelly, chocolate and cake.

Recipe structured data The major recipe websites follow Google's schema for recipes: https://developers.google.com/search/docs/data-types/recipe. This consistent structure will make it easier to import the data in a pre-structured format.

Shown below is a sample of a recipe formatted with structured recipe data <script type="application/ld+json"> "@context": "http://schema.org/", "@type": "Recipe", "name": "Grandma's Holiday Apple Pie", "author": "Elaine Smith", "image": "http://images.edge-generalmills.com/56459281-6fe6-4d9d-984f-385c9488d824.jpg", "description": "A classic apple pie.", "aggregateRating": "@type": "AggregateRating", "ratingValue": "4", "reviewCount": "276", "bestRating": "5", "worstRating": "1" , "prepTime": "PT30M", "totalTime": "PT1H", "recipeYield": "8", "nutrition": "@type": "NutritionInformation", "servingSize": "1 medium slice", "calories": "230 calories", "fatContent": "1 g", "carbohydrateContent": "43 g", , "recipeIngredient": ["1 box refrigerated pie crusts, softened as directed on box", "6 cups thinly sliced, peeled apples (6 medium)", "..."], "recipeInstructions": ["1...", "2..."] </script> I collected about 1,000 recipes via BeautifulSoup from AllRecipes.com. Here is a link to a subset of the training data containing 284 recipes: https://github.com/cassfinn/Deep-NLP-Recipe-Recommender/blob/master/dataAReveryDay.csv

1.2.4 Solution Statement

As the app learns the user's preferences it will predict more accurate "Recommended for You" recipes.

Elementary algorithms that harness content data are TF-IDF, Word2Vec and LDA (Latent Dirichlet Allocation). The best known user-based recommender systems are built using collaborative filter algorithms.

There are at least three kinds of data: content, user and interaction data. Different algorithms are suitable for each. We can start by using TF-IDF vectorization to assign a signature to each document. This will map every recipe to a multidimensional mathematical space, or vector. On the basis of those vectors we recommended events by selecting the k-nearest neighbors to a source recipe. There are numerous variations and enhancements that can be applied to this favoured approach, such as incorporating rating data and number of reviews.

There are also approaches that use neural networks such as Google's Word2Vec. The Word2Vec algorithm focuses on the meaning of words. It is a neural network implementation that leanrs distributed representations for words. It does not need labels in order to create meaningful representations. Given enough training data, words with similar meanings appear in clusters. Distributed word vectors can be used for word prediction.

Another approach to consider is algorithms based on topic modelling and building a probabilistic language model such as Latent Dirichlet allocation (LDA).

Collaborative Ăfiltering in Recommender Systems assumes that people who share an interest in certain things will probably have similar tastes in other things as well. Item-item collaborative filtering draws inferences about the relationship between different items based on which items go together. The more often two items (say, peanut butter and jelly) appear in the same recipe, the closer they are to one another. So, when someone searches for "peanut butter", the algorithm should find other ingredients that are most often in recipes with peanut butter, like "jelly" or "white bread", over things that aren't, like bologna.

User-item filtering takes a different approach. Instead of calculating the distance between items, we calculate the distance between users based on their ratings (or likes). When coming up with recommendations for a particular user, we look at the users that are closest to them and then suggest items those users also liked but that our user hasn't interacted with yet. So, if you've saved a certain number of recipes the app can look at other users who saved those same recipes and recommend one that they also saved but which you might not have seen yet.

The front end of the project will be an iOS app. The Deep Natural Language Processing algorithm will be Word2Vec. The model will be trained on a dataset of about 1,000 recipes in particular the recipe's title, description and ingredients. There are a number of parameter choices that can affect the quality of the final model. One example is architecture: the options are skip-gram or continuous bag of words. The training algorithm options include hierarchical softmax or negative sampling. Minimum word count will help to limit the size of the vocabulary to meaningful words.

As the user uses the app, s/he will have the ability to save recipes.

The app will prompt the user to enter at least one ingredient. The model will use the user's list of saved recipes and the ingredient(s) submitted to return 5-10 recommended recipes.

The user will see a result list of recommended recipe titles. Selecting a recipe will open the recipe details on the host recipe's website (e.g. AllRecipes.com). If the user chooses to Save that recipe, then the recipe data is added to the user's saved recipe list.

For example, if a user saves recipes for "beef stew", "lasagne", "macaroni and cheese", and "chili", then the app can recommend recipes that are similar to those based on cuisine and ingredients in those recipes. If a different user saves recipes for "vegetarian chili", "banana smoothie", "gluten free pizza", then different recipes might be recommended for that user. Given that there are hundreds of recipes for something like "lasagne", the app should find the top 10 that are most similar to recipes the individual user has previously saved, for example vegetable lasagne vs. three meat lasagne.

1.2.5 Benchmark Model

A benchmark would be to see if the app predicts recipes for the user that resemble ingredients and/or cuisines in the user's saved recipe box. The recommended recipes should make sense given the main ingredient that the user requests. If the user submits "chicken", then the first ingredient in the ingredients list of the recipe should contain "chicken" as should the recipe title.

The app should be smart enough to detect if a user prefers to cook more recipes where the ingredients are vegetables vs. meats or if the user prefers recipes that have Italian cuisine. If a user has saved mostly "stir-fry" recipes, then the model should predict "Chicken Stir-Fry" with a higher probability than "Chicken Stew".

The assumption is that the user saves recipes (s)/he likes.

kNN is a machine learning algorithm to find clusters of similar users based on common book ratings, and make predictions using the average rating of top-k nearest neighbors.

1.2.6 Evaluation Metrics

Given that a user's saved recipes list contains a sufficient amount of saved recipes and that there are hundreds of recipes to choose from, if a user searches for an ingredient, for example, "ground beef", then the algorithm should find recipes where ground beef is the primary ingredient and the other ingredients and cuisine are similar to recipes the user has previously saved.

The app should let the user enter more than one ingredient, for example "ground beef" and "tomatoes" (assuming the user is at home and that's what's in the pantry). The app should recommend 5 to 10 recipes which use ground beef and tomatoes. Those recommended recipes should be similar to the user's already saved recipes. For example, if the user has 15 recipes saved and 8 of those are casseroles, then the recommender should predict that casserole recipes have a higher preference (weight) for this user then hamburger recipes.

1.2.7 Project Design

In a nutshell: Kid to Mom: "What's for dinner?" Mom: "I don't know yet, give me 15 seconds to find a recipe".

Mom opens the iOS app and types (because it's faster to type than to take a picture with the phone and wait for image recognition):

```
Primary ingredient:   _ground beef__
Secondary ingredient: _tomatoes__
```

NLP algorithm submits Mom's previously saved recipes along with search ingredient terms. The database of recipes has been trained using the word2vec model to learn vector representations of the words in the ingredients lists.

Of the hundreds of recipes that contain both ground beef and tomatoes, the model predicts which of those recipes have a high probability that Mom will like them. The recipes with the highest probability of Mom liking them are returned to the app and presented to Mom.

When analyzing recipes, the model will have to learn the nature of each word. Is it a measurement (tablespoon), an ingredient (chicken), an action word (stir) or a descriptive word (slowly).

There are two main challenges: 1 - polysemy: words that have several meanings 2 - synonymy: different words that have similar meanings

Given the words "red pepper" in an ingredient, what are the nearby words? When we see the word teaspoon, we can determine the kind of pepper is a spice, when we see "chopped" as part of the ingredient, it is a vegetable. The co-occurrence of words can be used to remove ambiguity.

Training the NLP model: The model will have to learn which words are measurements and which are food items in the ingredients list. For example: Recipe for Meat Loaf: 1/2 cup packed brown sugar 1/2 cup ketchup 1 1/2 pounds lean ground beef 3/4 cup milk 2 eggs 1 1/2 teaspoons salt 1/4 teaspoon ground black pepper 1 small onion chopped 1/4 teaspoon ground ginger 3/4 cup finely crushed saltine cracker crumbs

The model will have to learn that black "pepper" is a spice, while red "pepper" can be a vegetable or a spice. Measurement terms should not be considered as ingredients but they can affect the meaning of the ingredient. Spices, such as salt, pepper and ginger, are considered staples and can also be ignored.

1.2.8 References:

1 - Bag of Words Meets Bags of Popcorn: https://www.kaggle.com/c/word2vec-nlp-tutorial

2 - Vector Representations of Words: https://www.tensorflow.org/tutorials/word2vec