What's cooking. STAT 841 final report.

Sajin Sasy and Alexandra Vtyurina, University of Waterloo

I CAN'T WRITE THIS DAMN ABSTRACT!

1. INTRODUCTION

PROBLEM DESCRIPTON
INTRO
PAPER FLOW

2. DATA MUNGING

Kaggle provided all participants with two sets of data – training and testing. Training data included almost 40 thousand recipes encoded as a list of json objects. Every recipe consisted of the following fields: *id*, *cuisine*, *ingredients*. 20 different cuisines with 6714 unique ingredients were presented in the training dataset. The test data had the same format with the exception of the *cuisine* label, that should be predicted. The expected format of submission was a csv file with two columns - *recipe id* and its predicted *cuisine label*.

The training data provided was quite rich however unbalanced. 1 shows how many recipes for each type of cuisine are contained in the training dataset. Underrepresented classes include brazilian, russian, jamaican, etc and contain under 800 recipes, whereas italian recipes dominate.

It is worth mentioning the variability of the ingredients representation. Some only consisted of a single word (for ex. "tomatoes"), whereas others contained multiple words (for ex. "ground black pepper"). Moreover some ingredients also included attributes, such as "2 large eggs" and brand names "Sargento Traditional Cut Shredded Mozzarella Cheese". This variability introduced additional noise to the data, and prevented us from using each ingredient as a separate feature. We further elaborate on the topic of data cleaning.

2.1 Data cleaning

Given that ingredients may contain multiple words, using each ingredient as a separate feature will introduce a lot of noise, as egg and eggs will be considered two different features. Prior to classifying recipes, we cleaned the data in several different ways to see which one yields the best results and removes the most of noise. We chose different approaches, starting with a very mild one and building on top of one another to eventually reach the strictest cleaning strategy. We started off by splitting each ingredient into words[1] and using them as features. This approach could deal with ingredients like "large egg" and "egg", splitting "large egg" into "large" and "egg", after which we had two "egg" terms match. But we still had problems in cases of "three large eggs" and "egg", where after splitting "three large eggs" into "three", "large", "eggs", the terms "eggs" and "egg" were still considered to be different. Stemming terms[2] after splitting would solve this problem by transforming "eggs" into "egg".

[1] We used NLTK built in tokenizer to tokenization
[2] We used Porter stemmer providd by NLTK

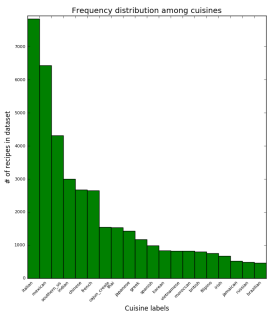

Fig. 1: Training data is unbalanced. Italian and mexican recipes dominate, whereas brazilian and russian are underrepresented.

After examining the data we found that there are now ingredients that dont add to the flavour of the dish, for example “three” and “large” (from “three large eggs”), or “ground” and “black” (from “ground black pepper”). Our assumption was that the main difference between cuisines comes from their flavour, hence only the ingredients themselves matter, rather than their modifiers. We wanted to remove modifiers based on the results of POS tagging (after manually examining the data we assumed that the majority of ingredients were nouns), but after performing POS tagging we found that the accuracy was extremely low due to the lack of context. E.g. we could only provide separate words and phrases to the tagger as opposed to entire sentences. As a result ingredients like “store bought low sodium chicken broth” the word “store” was considered a noun, even though it did not contribute to the flavour of the dish. Because of this issue we had to refuse from POS tagging and could not shrink initial data to bare ingredients.

3. FEATURE SELECTION

Mainly used bow

3.1 PCA story

After applying one-vs-one SVM to classify recipes we wanted to see how different the results would be for one-vs-all technique. For that we built 20[3] hyperplanes to separate every single cuisine from the rest of the recipes. In this section we discuss our findings.

First of all, to use one-vs-all approach we had to reduce the original problem to a binary classification problem. We did it by constructing a separate dataset for every given cuisine, where we changed labels to 1, in case the recipe belonged to the currently chosen cuisine, and 0 otherwise. For example, for separating italian recepies we created a dataset, where a label was 1, if the recipe was originally italian, and 0 otherwise.

Having done that for each of the 20 cuisines we used 5-fold cross validation to evaluate the accuracy of the new model[4]. We found that even though the reported accuracy was very high (it varied from 94% to 99% for different cuisines), some recipes did not have any label by the end of the classification, which means they were not classified as positive for any of the cuisines. From that we can conclude the recall was low, e.g. the proposed model was failing to identify all the recipes from a given cuisine.

We have also found that few of the recipes were classified as positive for multiple cuisines. Table I shows the list of cuisines that were confused most often and the number of recipes that were positively

[3]Same as the number of presented cuisines

[4]Considering that we were looking at a single cuisine at a time, we could not submit to Kaggle to test the accuracy

Confused cuisines	num of recipes	Confused cuisines	num of recipes
Thai and Vietnamese	31	Chinese and Korean	14
Southern US and Cajun creole	28	Italian and Greek	11
Italian and French	25	Indian and Moroccan	9

Table I. : Some of the recipes were classified as multiple cuisines. The table shows cuisines that were confused most often.

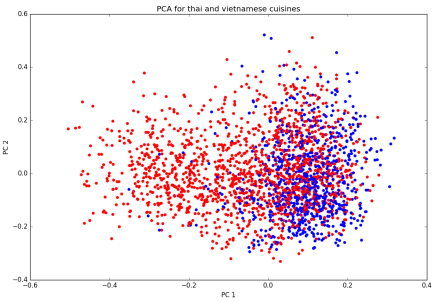

Fig. 2: The first and the second principal components for recipes of Thai and Taiwanese cuisines.

classified multiple times. As seen in the table the cuisines that get misclassified also share geographical location and/or cultural features.

After we found that some cuisines are hard for SVM to separate we looked at their recipes more closely and tried to find distinct features that could be used to differentiate the cuisines. We performed PCA on the recipes of Thai and Vietnamese cuisines. As showm in **??**, the overlap between the two cuisines is quite large and the separation is more clear alomg the first principal component.

4. CLASSIFICATION APPROACHES

This part should be done.

5. EXPERIMENTS

I don't really know what to write here. Probs different results we got from different datasets. And why

6. DISCUSSION

Future work and what not

7. CONCLUSION

We did that and it worked.

REFERENCES