What’s cooking. STAT 841 final report.

Sajin Sasy and Alexandra Vtyurina, University of Waterloo

I CAN’T WRITE THIS DAMN ABSTRACT!

1. INTRODUCTION

This report is describing the approach taken and results achieved in Kaggle[1] competition called “What's cooking”[2]. The evaluation was performed using a dataset provided by Yummly[3] – a popular culinary website. Two datasets - training and testing were given to the participants of the competition. Kaggle platform allowed participants submit their results 5 times every 24 hours.

While working on this project we have faced several challanges. First of all, the data provided could be interpreted in multiple different ways. We talk about data analysis in more details in the *Data munging* section. Second of all, we found that some cuisines have a very tightly overlapping list of ingredients in their recipes. Our assumption was that it was due to their close geographical proximity, as well as cultural influences. For example, we found that thai and vietnamese cuisines use similar sets of ingredients.

We have taken several classification approaches and compared their results. In particular, we used linear SVM, logictic regression and random forests. We found that linear SVM was performing the best for the given problem. We also used different tuning techniques to increase the accuracy. For example, we tried one versus one model, as well as one versus all. One versus all model yeilded high precision results, but very low recall, which was due to large overlap between classes. The detailed description of the approahces we have tried is in the *Classification approaches* section.

The *Experiments* section provides results we got for different data interpretation models and classification techniques. We also provide justification for the results obtained. Finally, we conclude with the general discussion and potential future work that could be done to enhance current results.

2. RELATED WORK

The problem of recipe classification is not the most popular, to the best of our knowledge. UbiComp people tackled a very similar problem, using a different dataset. The conclusions they found were also similar to ours. (But we didn’t rip off of them)

The research community is more active in the field of culinary recommendation systems. A BUNCH OF CITATIONS HERE.

3. DATA ANALYSIS

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

[1]https://www.kaggle.com/
[2]https://www.kaggle.com/c/whats-cooking
[3]http://www.yummly.com/

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. Figure **??** 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.

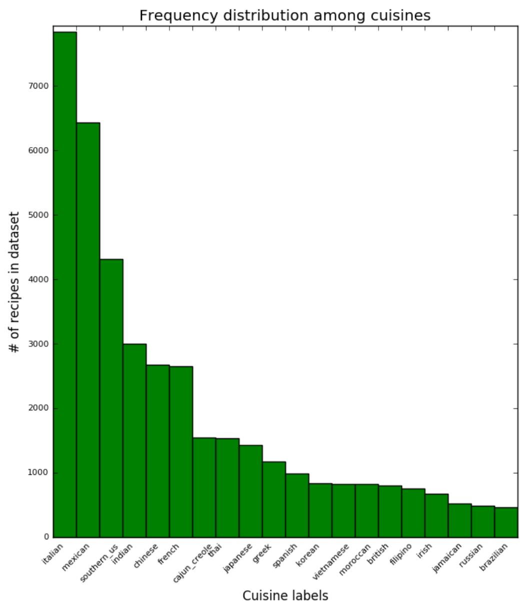

Fig. 1: Training data is unbalanced. Italian and mexican recipes dominate, whereas brazilian and russian are underrepresented.

3.1 Data munging

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[4] 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[5] after splitting would solve this problem by transforming "eggs" into "egg".

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

[4]We used NLTK built in tokenizer to tokenization
[5]We used Porter stemmer providd by NLTK

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.

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.

4. FEATURE SELECTION

While working with bigrams and trigrams, the number of features drastically increase as it would account for all potential two ingredient combinations over the feature space (of 6640 ingredients). We used 5-fold cross validation to pick a bound on the number of features (as shown in the graphs) to use, rather than arbitrarily limiting it to a fixed number of features, where in by increasing features and comparing the change in cross-validation accuracy, we limited the features at a point where the cross-validation results were not improving and rather stagnating and/or decreasing with increase of features.

4.1 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[6] 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[7]. 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 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 figure 5, the overlap between the two cuisines is quite large and the separation is more clear alomg the first principal component. We extracted the terms that contributed the most to the first PC. For that we considered the values with the largest magnitude. Table III shows the ingredients that contributed the most to the first principal

[6]Same as the number of presented cuisines
[7]Considering that we were looking at a single cuisine at a time, we could not submit to Kaggle to test the accuracy

Negative components	Positive components
chocol, unsalt, milk, larg, sugar, butter, bake, egg, yolk, almond, vanilla, water, whip, confection, heavi, allpurpos, powder, flour, cream, extract	oregano, garlic, black, basil, extravirgin, chees, parmesan, oil, oliv, leav, tomato, pepper, dri, clove, fresh, parsley, ground, red, onion, crush

Table II. : These ingredients contribute the most to the 1 principal component of the dataset comprised of italian and french recipes. The separation is not between cuisine, but between deserts and main dishes

Negative components	Positive components
Terms in italics have been noted by cooking enthusiasts as typical for	
thai cuisine	vietnamese cuisine
past, lime, breast, bell, curri, light, unsweeten, skinless, boneless, thai, red, green, *kaffir*, broth, stock, mushroom, *lemongrass*, chicken, *coconut*, milk	*mint*, garlic, white, vinegar, soy, pork, water, cucumb, black, *rice*, *lettuc*, *noodl*, sauc, peanut, sugar, beansprout, carrot, sesam, roast, egg

Table III. : These ingredients contribute the most to the 1 principal component of the dataset comprised of thai and taiwanese recipes.

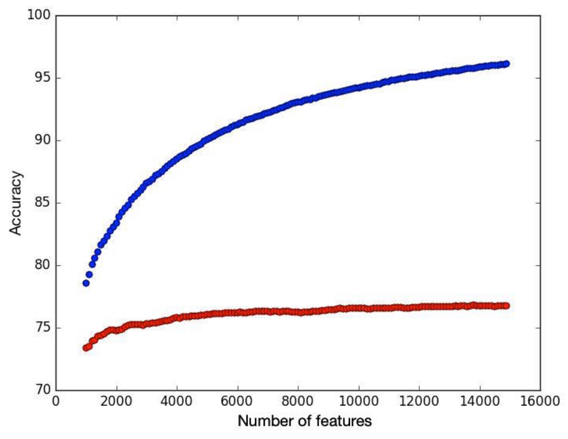

Fig. 2: A figure

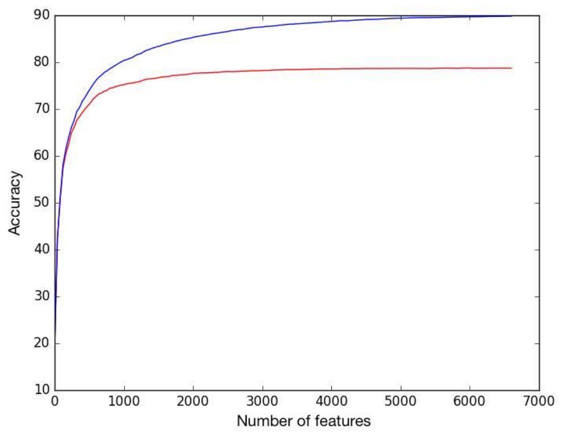

Fig. 3: Another figure

component. As we can see, it does not show the clear separation between the cuisies, however, having studied some typical features of Thai food, we found that some of the products from the table are typical from Thai (left) anf Vietnamese (right) cuisines (such products are in italics). However if we perform the same amalysis for French and Italian cuisines, we can see in Table II that the first principal component separates types of dishes (desert from main dishes) rather than cuisines themselves.

5. CLASSIFICATION APPROACHES

Since Whats cooking is a multi-class classification problem with an inherent input structure of the Bag-of-Words model, intuitively it would make sense to use a Naive Bayes which fits well into both the structure of the underlying problem as well as the input-format. However certainly Naive Bayes is unlikely to produce a strong classifier as opposed to Multiclass Logistic Regression or SVM.

Model	Score
Experimental Ensemble Technique	**0.79133**
SVM (linear kernel , using bigrams , one-vs-one model)	0.79053
SVM (linear kernel , one-vs-one model)	0.78831
SVM (linear kernel , using bigrams , one-vs-all model)	0.78761
Logistic Regression	0.78309
SVM (linear kernel , one-vs-all model)	0.77253
Boosting (xgboost)	0.76760
Random Forests (with 100 trees and Bootstrap)	0.74361
Naive Bayes	0.66814

Table IV. : The results of different approaches after submitting to Kaggle.

The classification method involved has to make judgements based on the presence of ingredients in a given recipe, this suggests a quite simple model of classification, intuitively the cuisine that a recipe belongs to would be based on just the presence or absence of an ingredient from the feature list of ingredients, hence suggesting a simple linear model of classification as opposed to a more complicated model like Neural Networks which would be more prone to overfitting in such an instance.

Support Vector Machine :

As opposed to other linear classifiers, the primary distinction that sets SVM aside is the fact that produces a max-margin hyperplane to separate the data points , in contrast other models would produce one among the infinite number of hyperplanes that could separate them (and this applies even in the case of data that is not linearly separable). Simply by concept itself, the notion of a classifier that produces the maximum margin between two classes would intuitively produce a better classifier as opposed to the others, and we were certain that this would be the ideal method for classification for our problem at hand.

The key advantage of using Support Vector Machine for us lie in its robustness to overfitting and its ability to produce a max-margin hyperplane. Just as we had expected our best results came from the SVM classifier itself. The only setback of SVM is that it is inherently a binary classifier, so we would have to use it in our problem either by using it as a one-vs-all or as a set of one-vs-one classifiers for every pair of cuisines.

For implementing it we made use of svm.SVC package of exposed in sklearn package for Python, which in itself is based on libsvm [1] . We tuned the SVM model to try both one-vs-one and one-vs-all to see that one-vs-one was indeed producing slightly better results (as seen in the Table ??) .

Random Forests :

As opposed to other linear models, we decided to use Random Forest, which is generally classified as an ensemble technique, since in effect it performs majority vote with a large number of decision trees used to estimate the value of the function. Since the underlying technique is actually of decision trees, one major advantage is that it is no longer bound to linearity, in the sense that it would not try to model based on linearity of features and would intuitively do well for binary features (like the bag-of-words). Another significant advantage is that random forests work in synchrony with bootstrap aggregating or bagging, as since each of the constituent decision trees can be trained with bootstrapped training sets, and this is a good mechanism for checking overfitting

6. EXPERIMENTS

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

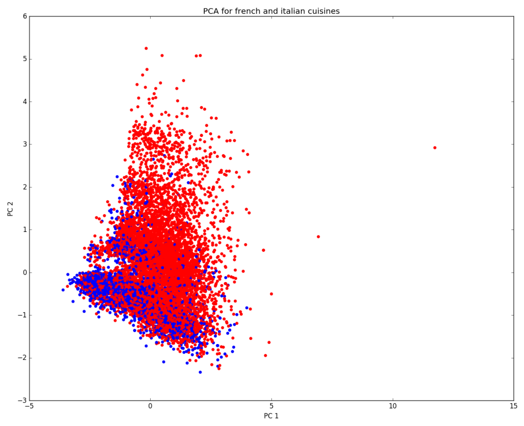

Fig. 4: A figure

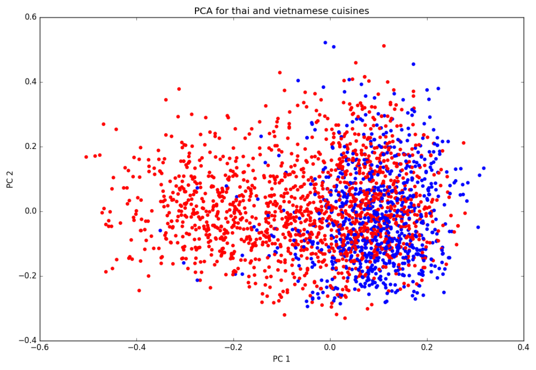

Fig. 5: Another figure

7. DISCUSSION

blah

8. CONCLUSION

In this report we tackled the problem of recipe classification based on their ingredients. We worked on the dataset provided by Yummly, and evaluated the results suing Kaggle platform. The problem has been shown to be challenging due to the variery of ingredients used in different cuisines and their overlapping. We tried tokenization and stemming of the multiwords ingredents, as well as using a list of attributive words (large, store-bought, etc). The latter has proven to yeild better results, as by dropping such words, we only considered the flavour of the dish. We tried several techniques for classification, out of which linear SVM was proven to be the best. The cuisine classification problem was shown to be quite challenging due to the large overlap in the ingredients.

REFERENCES