Classifying Recipes into Cuisines Using Context Vectors

Nishant Mohan
Trinity College Dublin
mohanni@tcd.ie

Lujain Dweikat
Trinity College Dublin
dweikatl@tcd.ie

Rohan Bagwe
Trinity College Dublin
bagwer@tcd.ie

Aakash Kamble
Trinity College Dublin
kamblea@tcd.ie

Oommen Kuruvilla
Trinity College Dublin
kuruvilo@tcd.ie

April 14, 2020

Abstract

For classification problems, simple machine learning algorithms like Logistic Regression[1] have been used extensively with numerical data. However, the current challenge is to convert textual data that is not ordinal to numerical data in a meaningful way that preserves semantics in a way. In this essay, we present a method of converting ingredients in a recipe to context vectors using Gensim's[2] implementation of Word2Vec. The accuracy we got overall for this method was 65%, which is lower than our baseline method using Logistic Regression at 78%.

Keywords— text analytics, recipes, cuisines, context vectors, word2vec, classification

1 Introduction

Ever since the internet became popular, people have been increasingly sharing recipes on the internet. This has caused a lot of websites to be created to host those recipes and allow people to share, search, and review those recipes with ease. Being able to label a recipe with a cuisine could make it easier for users to look for recipes that they like manually, or for search engines to help find recipes from the same cuisine. Additionally, These types of websites would probably be able to be more beneficial to the user if they're able to recommend similar recipes from the same cuisine.

Classification isn't a new problem. However, simple classification algorithms usually work best with numerical data. For textual data, there are many ways to transform it into numerical data. For some of them, the semantics of the text is lost when converting to the numerical space. Therefore, a method that preserves the meaning of the words used, and creates consistently sized numerical sets that are appropriate for machine learning was needed.

Some similar work has been carried out previously. One paper[3] has attempted to predict cuisines using the same dataset we're using. They use XGBoost,[4] a gradient boosting library, and Random Forests[5] in order to achieve classification with 80% accuracy. In this paper, we explore whether using text analytics methods of representing text in a classification problem with a small corpus per entry yields better results than using classifiers without those aforementioned methods.

This paper first goes through the relevant literature. We cover concepts related to context vectors and word embedding, then we discuss related attempts at classifying recipes into cuisines. Next, the dataset, baseline, and our approach to solving this problem will be thoroughly explained in the methodology. We will be following that with our results, analyse them along with any errors. Finally, the paper will be ended with a conclusion and summary of the experiment, along with some suggestions for future work in the area.

1. Andriy Burkov, *The hundred-page machine learning book* (Andriy Burkov, 2019).
2. *Gensim: Topic Modelling for Humans*, https://radimrehurek.com/gensim/.
3. R. Kumar, M. Kumar, and Soman Kp, "Cuisine Prediction based on Ingredients using Tree Boosting Algorithms," *Indian Journal of Science and Technology* 9 (December 2016), doi:10.17485/ijst/2016/v9i45/106484.
4. *XGBoost Documentation*, https://xgboost.readthedocs.io/en/latest/.
5. *Random Forests Leo Breiman and Adele Cutler*, https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

2 Literature Review

2.1 Context Vectors and Word Embeddings

Machines do not understand natural language the way humans do. Although advanced computing technologies allow the end-user to carry out their work without any awareness of this fact, at the lowest levels of implementation, a computer program or an algorithm deals with numeric data.

A simple and straight-forward way of dealing with this problem using to be bag of words. When using bag-of-words, a dictionary of all the unique words in a sentence or a document is created. A corresponding vector representation of a word would be all zeroes, except for the index where the word appears, where the value will be one. This method usually yields a very sparse representation of the data. Although this method can be useful in some cases, for more complex language processing tasks, this representation loses the order of words, and does not preserve semantics.

Context Vectors,[6] a form of word embedding, are another method of representing textual data. The concept was first introduced by Stephen Gallant in 1998. The author suggests that these vectors should have four goals in mind. Firstly, the vectors need to convey similarity of use between two different words. For example, the more similar two terms are (like "car" and "auto") the bigger the dot product, and vice versa. Secondly, context vectors need to provide a quick way of searching and verifying that a word relates to the contents of a document. Thirdly, this method should enable the user to use text data in a machine learning algorithm. Finally, context vectors should be feasible to create for new unlabeled text.

Although Context Vectors achieve the goals above, they have a few shortcomings. The syntax is not represented in CVs at all, therefore no discourse or logic can be concluded using context vectors only. The author suggests that using phrasal CVs and adding a second run of CVs could possibly be used to represent syntax. However, for the purpose of this study, there doesn't seem to be a need for representing syntax, since a recipe is just a list of ingredients, not a coherent sentence.

More modern forms of word embedding usually fall into one of the following categories:[7]

- Frequentist methods: Count Vectors, TF-IDF Vectors and Co-occurrence matrix
- Predictive methods: Continuous Bag of words (CBOW) and Skip-Gram models

2.1.1 Word2Vec

Word2Vec[8] were first introduced by google in 2013. It falls into the predictive methods type of word embedding, using continuous bag of words, and Skip-Gram models, which are both shallow neural network models. It can be trained on any custom set of data, or, a pre-trained model trained some large corpus, like Wikipedia, could be used to retrieve vectors for words, then those words could be used to create vectors for the document.

2.1.2 GloVe

Another vector representation method for words is GloVe.[9] GloVe is an unsupervised learning algorithm created by researchers at Stanford University. It's trained on aggregated global word-word co-occurrence statistics from a corpus. Its main features include finding the nearest neighbor of a a word, i.e. frog's nearest neighbor could be the word toad, and finding linear substructures that allow arithmetic operations on those vectors to conclude concepts like:

king - man + woman = queen

The end result of both GloVe and Word2Vec is a vector representation of a word. However, GloVe is not based on a neural network, the way Word2Vec is. Another difference is the fact that Word2Vec produces context-predicting vectors, while GloVe produces context-counting vectors. It's been previously shown[10] that context-

6. Stephen I. Gallant, "Context Vectors: A Step Toward a "Grand Unified Representation"," in *Hybrid Neural Systems, Revised Papers from a Workshop* (Berlin, Heidelberg: Springer-Verlag, 1998), 204–210, ISBN: 3540673059.

7. Amit Mandelbaum and Adi Shalev, "Word Embeddings and Their Use In Sentence Classification Tasks," October 2016,

8. Tomas Mikolov et al., "Distributed Representations of Words and Phrases and their Compositionality," *CoRR* abs/1310.4546 (2013), arXiv: 1310.4546, http://arxiv.org/abs/1310.4546.

9. Jeffrey Pennington, Richard Socher, and Christopher D. Manning, "GloVe: Global Vectors for Word Representation," in *Empirical Methods in Natural Language Processing (EMNLP)* (2014), 1532–1543, http://www.aclweb.org/anthology/D14-1162.

10. Marco Baroni, Georgiana Dinu, and Germán Kruszewski, "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors," vol. 1 (June 2014), 238–247, doi:10.3115/v1/P14-1023.

predicting vectors are better than context-counting vectors. Based on this, we will be using Word2Vec for this experiment, intead of GloVe.

2.2 Related Work

Since the dataset was obtained from a Kaggle competition, many people from across the world have attempted to predict a dish's cuisine based on a recipe's ingredients as accuratly as possible. We found the paper by Kumar[11] et al. to be a well documented example of a solution. The authors first merged all ingredients into a single corpus. Then they converted all words to lowercase letters. They then removed punctuation, stop words, and white spaces, then performed stemming on the words.

This altered and cleaned set was then fed into two algorithms, XGBoost,[12] and Random Forests.[13] The first yielded an accuracy of 80.4% while the latter yielded an accuracy of 76.4%.

For this paper, we would like to see the effect of using word embeddings after cleaning up the data on the accuracy of the predicting model.

3 Methodology

3.1 Dataset

The dataset we've obtained from Kaggle,[14] in partnership with Yummly[15] was provided in the form of JSON files. Each recipe item has a unique ID, a single cuisine, and a list of ingredients.

When doing exploratory data analysis, we've found that the training dataset contains a total of 39774 recipes, each belonging to one of the following cuisines: French, Mexican, Korean, Spanish, Chinese, Indian, Cajun Creole, Filipino, Brazilian, Italian, Irish, Moroccan, Southern US, Japanese, Russian, Jamaican, Greek, Thai, British, or Vietnamese. Overall, the dataset contained 6714 unique ingredients.

Cuisines were not represented equally in the training data. Italian recipes were over-represented, comprising 19.7% of the dataset, while the training set had low composition Brazilian recipes, at around 1.17%.

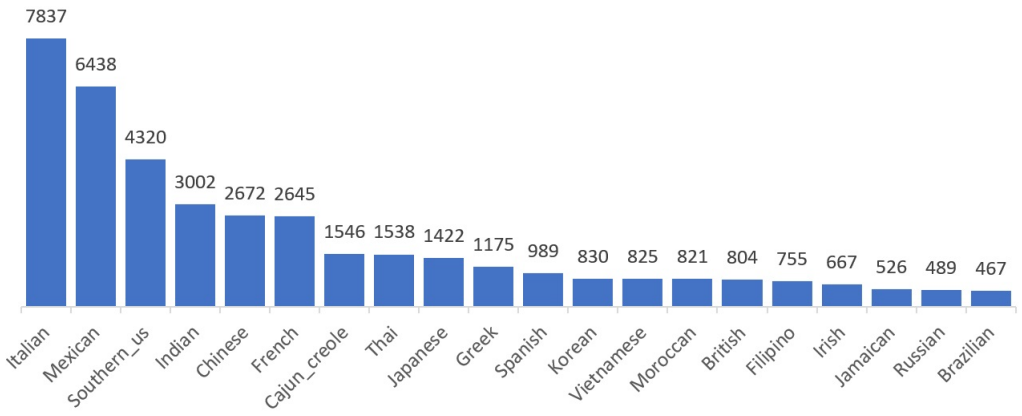

Figure 1: Recipe count per cuisine in dataset

3.2 Establishing A Baseline

In order for us to see whether or not using text analytics to pre-process the dataset would yield better results than using context vectors, we needed to establish a baseline. We chose simple Logistic Regression[16] provided by scikit-learn,[17] as it's a very simple way to do logistic regression, with not a lot of variables to control for our comparison.

11. Kumar, Kumar, and Kp, "Cuisine Prediction based on Ingredients using Tree Boosting Algorithms."
12. *XGBoost Documentation*.
13. *Random Forests Leo Breiman and Adele Cutler*.
14. *What's Cooking?*, 2016, `https://www.kaggle.com/c/whats-cooking`.
15. *Personalized Recipe Recommendations and Search*, `https://www.yummly.co.uk/`.
16. Burkov, *The hundred-page machine learning book*.
17. *scikit-learn: Machine Learning in Python*, `https://scikit-learn.org/stable/`.

We used the bag-of-words model to convert the ingredients into numerical data. We derived a dataset with 6714 columns that count the frequency of occurrence of each ingredient in a recipe. This dataset was then split into 85% for training and 15% for testing.

The prediction accuracy for this method was found to be 78%.

3.3 Cleaning Up The Data

As a first step to our approach, we first need to clean up the dataset. We parsed all the recipes one at a time. For each ingredient, we first remove all non-alphanumeric characters. Then, we removed all English stop words, removed the most common words in the corpus (i.e. large, salt, water, cold ... etc) and typographical errors..[18] Next, we used a NLTK[19] in order to get the singular form of nouns, then get the parts of speech of a word, and remove any words that are not nouns. This process reduced the total number of unique ingredients by 1729 from 6714 to 4985.

Using Gensim's[20] Word2Vec implementation, we created a vector for each recipe. We chose to get 300 features from the function, and we used a 10 unit wide window.

We then converted the dataset into a Pandas[21] dataframe to be used by the prediction model.

3.4 Model Training

Using scikit-learn's,[22] we randomly split the data frame into two parts. The training set makes 85% of the original set, while the testing set is comprised of 15% of the original data frame.

Next, scikit-learn's[23] Logistic Regression model is fitted using the training data. The resulting model is used to classify the training set.

4 Results

After training Word2Vec using our corpus, we tested to see if it truly gives high cosine similarity for similar ingredients. We used the ingredients "ginger root" and "ginger", which results in a 0.9526417 similarity, indicating that the model trained is behaving correctly.

Logistic regression yielded 65.35% accuracy on the test data. This is definitely lower than our baseline method of using logistic regression using bag of words only, which was at 78%. The following section will discuss some possible explanations for the drop in accuracy.

4.1 Error Analysis

The first hint we got for where the error must've been was seeing that the dataset was imbalanced, as seen in the previous section, with an over-representation of Italian and Mexican recipes.

These cuisines that were represented well, tended to have lower misclassification rates:

Mexican	11.53	Southern US	31.81	French	58.56	Irish	81.41
Indian	13.26	Cajun Creole	43.92	Filipino	64.64	Russian	84.87
Italian	15.69	Moroccan	45.31	Korean	66.63	Spanish	93.02
Chinese	20.77	Japanese	52.39	Jamaican	67.3	British	94.28
Thai	26.79	Greek	52.68	Vietnamese	72.24	Braziallian	94.86

Table 1: Misclassification rates per cuisine

We can see that for the three most represented cuisines, we got the lowest misclassification rates, while the least represented cuisine got the highest misrepresentation score.

18. nightowl21, *nightowl21/Extra-Projects*, `https://github.com/nightowl21/Extra-Projects/tree/master/Yummly-Kaggle`.
19. *Natural Language Toolkit*, `https://www.nltk.org/`.
20. *Gensim: Topic Modelling for Humans*.
21. *pandas*, `https://pandas.pydata.org/`.
22. *scikit-learn: Machine Learning in Python*.
23. Ibid.

This can be visually represented using the confusion matrix shown in figure 2, where cusines on both axis have been ordered by the percentage of representation in the dataset.

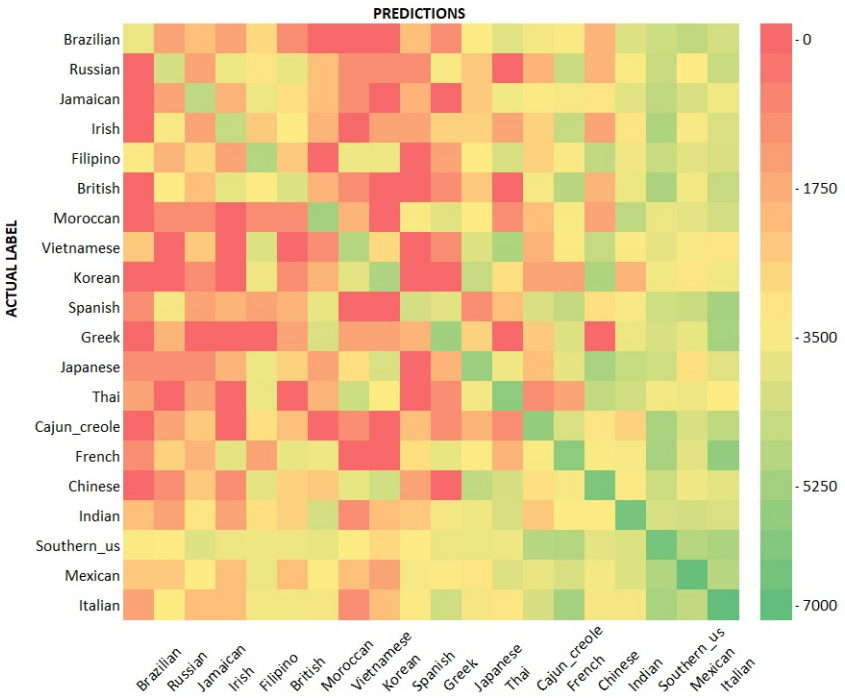

Figure 2: Confusion matrix for classification using Word2Vec

This can indicate that there is a fundamental problem with the lack of representation of other cuisines in the dataset. This problem could potentially be solved using a larger, more diverse dataset.

Starting from here, we've decided to take the product of the scaled misclassification rate and the scaled representation to see what cuisines got the best classification, despite its level of representation in the original dataset. This turned out to be French cuisine.

The next step was to see what were the misclassified French recipes being misclassified as. For the sake of comparison, we've decided to do the same for Italian, and Irish cuisine.

It turned out that French recipes were most frequently misclassified as Italian recipes, while Italian recipes were most frequently misclassified as french recipes. Irish recipes we most misclassified as Southern US recipes. We speculate that this is because Italian and French cuisines might be more closely related to each other than other cuisines.

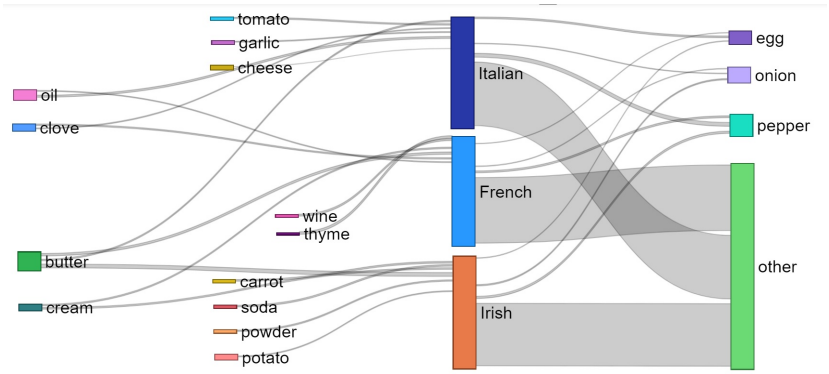

Figure 3: Sankey Diagram of Italian, French, and Irish cuisines, and their ingredients

Figure 3 shows the most common ingredients in the three cuisines under scrutiny. It can be observed that French cuisine shares two ingredients, oil and clove, with Italian cuisines. French cuisine also shares two main ingredients, eggs and butter with Irish cuisine. Interestingly, Italian cuisine and Irish cuisine do not share many main ingredients other than the ones shared by all three cuisines, which are pepper, onion and others. This further

strengthens our belief that French cuisine has similarities with Italian and Irish cuisine, however, Italian and Irish cuisine are not as similar. Therefore, misclassification among Italian and French cuisine is natural, since they more closely resemble each other based on the ingredients used in their recipes.

5 Conclusion

Although recipes and recipe books have existed for a very long time, ever since the widespread use of the internet started, more and more recipes have appeared on the internet, aggregated by recipe-book like websites, with a variety of cuisines. Knowing the cuisine a recipe belongs to could be helpful for recipe recommendations and browsing.

For this experiment, we established a baseline using Logistic Regression,[24] as it's the simplest form of classification algorithms with no other variable to be controlled. Using bag-of-words, the baseline resulted in 78% classification accuracy.

We've used context vector representations of the recipes created using Gensim's[25] implementation of Word2Vec[26] after carrying out some data cleaning tasks.

After classifying the data using Logistic Regression again, we got an accuracy rate of 65.35%, which is less than the baseline accuracy.

After analysing the dataset, we found out that the cuisines in the dataset do not have similar representation ratios. This has lead to having higher accuracy rates for the top four most represented cuisines than the overall accuracy rate.

We've also concluded that some cases of misclassification might be due to actual similarities between two cuisines, as we establish that Italian and French cuisines are similar, French and Irish cuisines are similar, however, Italian and Irish cuisines are as not similar, causing a lot of misclassifications between French and Italian cuisines.

6 Future Work

Since this project was as an essay submitted for the module CS7IS4 - Text Analytics at Trinity College Dublin, with limited time, we've decided to use one implementation of context vectors, that being Gensim's Word2Vec, based on a paper by Baroni et al..[27] Given more time, we would have attempted to use GloVe for the same experiment, as it could yeild different results for this case of ingredients in an ingredient list.

We also would have preferred to use another large dataset with better composition. If we've adjusted the dataset we've found to fix composition inequality, we would have ended up with a dataset that is too small to be used for training.

24. Burkov, *The hundred-page machine learning book*.

25. *Gensim: Topic Modelling for Humans*.

26. Mikolov et al., "Distributed Representations of Words and Phrases and their Compositionality."

27. Baroni, Dinu, and Kruszewski, "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors."

References

Baroni, Marco, Georgiana Dinu, and Germán Kruszewski. "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors," 1:238–247. June 2014. doi:10.3115/v1/P14-1023.

Burkov, Andriy. *The hundred-page machine learning book*. Andriy Burkov, 2019.

Gallant, Stephen I. "Context Vectors: A Step Toward a "Grand Unified Representation"." In *Hybrid Neural Systems, Revised Papers from a Workshop*, 204–210. Berlin, Heidelberg: Springer-Verlag, 1998. ISBN: 3540673059.

What's Cooking?, 2016. https://www.kaggle.com/c/whats-cooking.

Kumar, R., M. Kumar, and Soman Kp. "Cuisine Prediction based on Ingredients using Tree Boosting Algorithms." *Indian Journal of Science and Technology* 9 (December 2016). doi:10.17485/ijst/2016/v9i45/106484.

Mandelbaum, Amit, and Adi Shalev. "Word Embeddings and Their Use In Sentence Classification Tasks," October 2016.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality." *CoRR* abs/1310.4546 (2013). arXiv: 1310.4546. http://arxiv.org/abs/1310.4546.

Natural Language Toolkit. https://www.nltk.org/.

nightowl21. *nightowl21/Extra-Projects*. https://github.com/nightowl21/Extra-Projects/tree/master/Yummly-Kaggle.

pandas. https://pandas.pydata.org/.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation." In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. 2014. http://www.aclweb.org/anthology/D14-1162.

Gensim: Topic Modelling for Humans. https://radimrehurek.com/gensim/.

Random Forests Leo Breiman and Adele Cutler. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/.

XGBoost Documentation. https://xgboost.readthedocs.io/en/latest/.

Personalized Recipe Recommendations and Search. https://www.yummly.co.uk/.