Mitigating Hazards of Food Allergens

Mariam Hakobyan
mariam.hakobyan@epfl.ch

Devavrat Tomar
devavrat.tomar@epfl.ch

Harshdeep Singh
harshdeep.harshdeep@epfl.ch

Abstract

In this project we do analysis and modeling of food allergens present in OpenFood Facts dataset. The inspiration is to identify possible allergies that a given unlabeled food item may cause and recommend the food items that have specific nutrients, without these allergic risks. Word2Vec has been used to group different allergens together for their further predictions based on the food categories present in the dataset. We trained Random Forest Algorithm and achieved average test accuracy of about $\mathbf{89.88}\%$ for the task of predicting allergens. Moreover, for efficiently searching food items with specific nutrition and allergic limitations, we applied Binary Search Tree based Vector Quantization method and discussed the results.

1 Introduction

Working on Open Food Facts dataset by $5000+$ contributors with $600000+$ products from 150 countries is not only an interesting data science task for analyzing ingredients, allergens and nutrition facts of food, but also challenging in terms of its incompleteness and multilingual data representations. In this project, we explored the distribution of allergens and additives present in the food products, applied Word2Vec to cluster allergens into meaningful categories and predicted allergen category of unlabeled food based on ingredients and food categories. We also implemented binary search tree based fast vector query algorithm for recommending food items with allergic constraints. For text translation of allergens and food categories to English, Google Translate has been used.

2 Food Additives: Purposes and Effects

For keeping the safety, taste, freshness and appearance of food, additives are commonly added to food products (Food and (FDA), 2004), which are subject to confirmation by world food organizations. However, many people are allergic and show adverse effects on consuming food with certain of additives. The dataset itself proves the fact that there are some additives included in the allergen cautions of food items. Many studies have shown carcinogenic effect of color and flavour enhancers such as *Sodium Nitrate* and *Sodium Nitrite* which are added to processed meat.

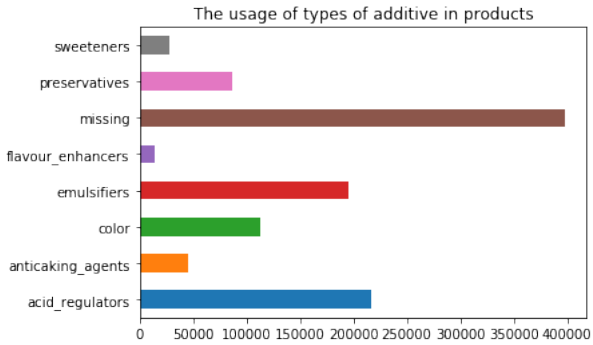

Figure 1: Main types of additives

To understand the purposes and effects of certain groups of additives we have clustered them based on the characterizing code level **Exxx**, into 8 main categories: *Color Enhancers, Preservatives, Flavour Enhancers, Acidity Regulators, Anti-Caking Agents, Antibiotics, Glazing Agents Gases, Sweeteners and Thickeners/Stabilizers/ Emulsifiers*. Figure [1] shows the distribution of each type of additives among products. We observe that the most common additives in the dataset are acidity regulators which control the acidity level of the food to prevent from spoilage.

It is interesting to know the brands which use additives more often for food production. We found that *Carrefour*, *Auchen* and *U-food* companies are the three top companies in terms of almost all additives' usage in their production.

Although the dataset has food products from all over the world, the distribution is not uniform, so we won't be able to conclude this statement for all companies around the world, but rather for the ones close to France, USA and Switzerland. In terms of flavour enhancers, the top 3 brands are *Maggi, Knorr and Spartan* which are mainly dehydrated soup and meal mixes, bouillon cubes and condiments with pleasurable flavours. Food and drug association reports that some flavour enhancers contain *Monosodium glutamate (MSG)* and the extensive usage of such products may result to headache, numbness and other body effects in special reactive groups of people (Leonard, 2018).

We have also observed the common products with special additives. For example, different types of ice-creams (gelatin) and candies mostly have coloring additives, and similar to brands dehydrated and con-

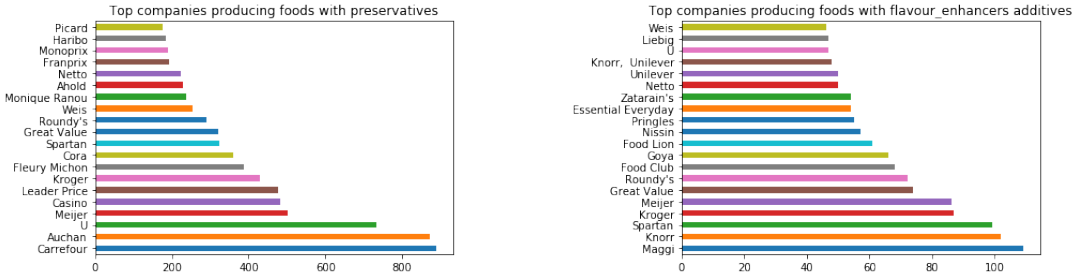

Figure 2: Top 20 brands using preservatives and flavour enhancer additives

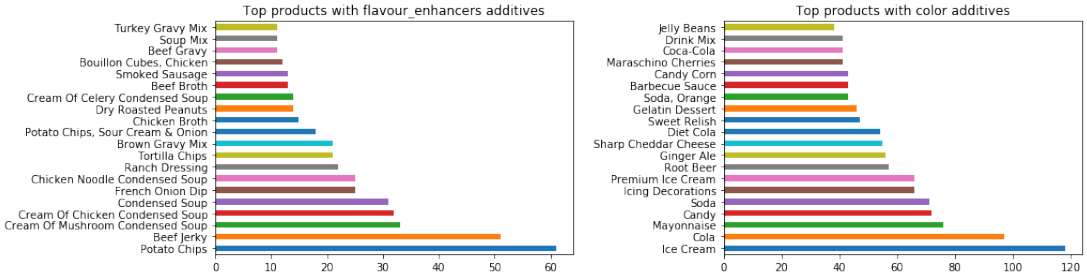

Figure 3: Top 20 products having flavour enhancers and color additives

densed soups, chips contains most of the flavour enhancers. The anti-caking agents are common in cookies and dough related products.

3 Efficient Vector Query for Nutrients

In order to query Open Food Fact data for food items having similar nutrition content, one trivial algorithm is to find n nearest neighbours of the given nutrition vector by comparing it will all the food items. The time complexity for this trivial algorithm is $\mathcal{O}(n)$, which is too slow for querying around $600,000$ food items. Thus, we used Binary Search Vector Quantization **(BSVQ)** (Lowry et al., 1987), (Katsavounidis et al., 1996) to store all food items into buckets as leaves of a binary search tree. The search algorithm for Binary Search Tree has complexity $\mathcal{O}(log(n))$. For more details please see appendix A.

3.1 Data Filtering

Open Food Facts dataset has total 85 nutrients. However, for the analysis we are considering top 35 nutrients that have number of food items greater than 1000. Also, for these 35 nutrients, we have many outlier food items that have been dropped for the analysis. For instance, some outliers have proteins value $31000g$ per $100g$ of food item.

3.2 Feature Normalization and Processing

Since nutrients are non-negative and have different scales in the dataset (protein is generally measured in grams, while vitamins are measured in milligrams) we normalize nutrients by dividing them with their means calculated over non zero elements.

We used cosine distance for comparing two nutrient vectors in place of euclidean distance so that the ratio of nutrients are compared for two food items (One can consume any amount of food to get same amount of nutrition). We also add small random noise $(\sim 10^{-8})$ to the nutrient vector so that during Vector Quantization, the buckets' size are uniform.

3.3 Model Parameters and Results

We used maximum bucket size (number of food items) of leaf node as 100 for constructing the Binary Search Tree. All food items belonging to a bucket have similar nutrition content. Also note that only leaf nodes contain the food items. When querying a given nutrient vector, we do tree traversal by eliminating almost half of the food items at every depth level till we reach the leaf node. We then return all food items in the leaf node bucket.

Table [1] shows an example query for the food item with ID: 21118244 that has allergens **wheat/flour** and **milk/fruit/nut**. We query the Binary Search Tree with its nutrient vector and return all food items that don't have these allergens but similar nutrition content.

4 Feature Engineering on Allergens

In this section, we describe the feature engineering on allergens for grouping them into categories.

4.1 Feature Preprocessing

Initially, there were more than 3000 unique allergens in the dataset. The allergens were also present in more than 15 languages. This was predicted with the help of the **langdetect** library. Since, there was no library to do

	Protein	Fat	Carbo-hydrate	Sugar	Salt	Sodium	Sat. Fat	Allergens
Query	9.30	8.20	20.70	0.60	0.85	0.33	3.10	wheat/flour, fruit/nut
Results	9.00	6.10	23.90	3.70	1.11	0.44	2.80	seafood
	8.50	4.00	19.40	1.40	0.92	0.36	1.20	seafood, powder/protein
	10.10	4.60	22.10	4.60	1.00	0.39	1.00	seafood
	8.50	4.00	19.40	1.40	0.92	0.36	1.20	seafood, powder/protein

Table 1: Example - BSVQ query for $100g$ of food item

translation, we translated the allergens in different languages manually using Google Translate. We did a bit of data cleaning by removing allergens which are just made up of numbers and removed unwanted characters. Since, a lot of the allergens had unwanted data, we tagged each allergen by its part of speech using **nltk** and only took nouns and adjectives and discarded all others. This is because most of the food items are nouns or have some sort of adjective before it which describes them. We also took out the additives with the format **'e'-followed by some number** using regular expression. All these steps were performed to make sure that the dataset is more consistent for Word2Vec.

4.2 Word2Vec

As there are about 1500 unique allergens after preprocessing in the data set, we needed a way to group these allergens together by their similarity. This is done with the help of the Word2Vec (Mikolov et al., 2013) algorithm which basically generates vectors for each word by taking into the neighbouring words. We basically took a pretrained model on the English Wikipedia; this pretrained model is called as GLoVE (Pennington et al., 2014). If the allergen is made up of multiple words such as **'milk solids'**, we find a vector for both **'milk'** and **'solids'** and add them together to get a resultant vector. As, all the word2vec vectors are in 300 dimensions we use the TSNE (Maaten and Hinton, 2008) algorithm for dimensionality reduction to show a graph in Figure [6] as to how the words are plotted against each other on the basis of similarity. For more details about how Word2Vec works see the appendix B.

4.3 KMeans Clustering

After we got the vectors for each of the allergens, we cluster them together by using the K Means algorithm and take the distance metric as cosine similarity. We divided these primarily into 5 clusters using the algorithm. More than 5 clusters were not possible since the algorithm made clusters with no allergens in it. The other clusters **additives_l** and **others** are made with the help of regular expressions and by detecting if the language is not English respectively as once can see in table [2].

Allergen Type	Total No.	Examples
wheat/flour	238	barley, wheat semolina
acids	179	bisulfite, anhydride
additives_l	15	e150d, e202, e223
milk/fruit/nut	498	oranges, eggs
others	185	grddpulver, congele
powder/protein	273	soy, vita protein
seafood	175	seafood, cod eggs

Table 2: Total number of allergens which are grouped together into 7 categories after translation

5 Predicting allergens from categories

In this section, we are going to describe about the process as to how we predicted allergens based on the categories for each of the product. We did not use ingredients even though there was more data because the data was inconsistent and it was hard to translate it in one language.

5.1 Data preprocessing

To make the categories more consistent we translated them using Google Translate; so that same categories in different languages can be grouped together. All the null categories were replaced with the keyword **missing** and then merged together with the allergens.

5.2 Algorithm

We used Random Forest algorithm from python's sklearn library to predict allergens based on categories.

Random Forest is one of the ensemble learning algorithm in which **weak learners** (decision trees) combine together to form a **strong learner**. The decision or labels is generated by taking the mean of the result of all the weak learners. The main parameter to train for Random forest is the **number of estimators** which basically tells the number of decision trees the random forest should be made up of.

The system is trained by taking N samples at random with replacement to create a subset of the data. Randomly **x** features are picked up and the features that provides the best split, according to some objective function (*gini* for sklearn), is used to do a binary split

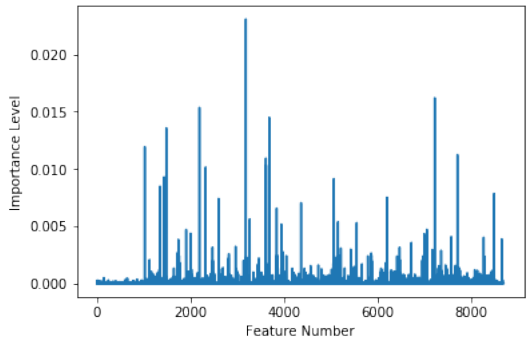

Figure 4: Feature importance of 8300+ translated categories

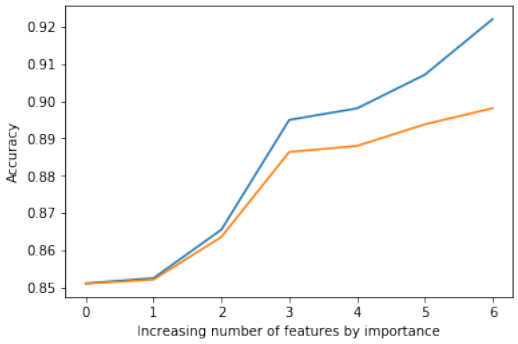

Figure 5: Increase in accuracy by increasing the number of features by importance level - starting with 0 having most important 11 features to 6 having 1398 most important features.

on that node. This process is repeated for all nodes and estimators.

We picked the Random Forest algorithm to train because their training times are faster as compared to other models like Neural Networks. They can also handle unbalanced data well and this was important since all the categories were not translated in one language.

5.3 Results

We trained and tested our Random Forest model on 54896 products. We achieved training accuracy of 93.7% and testing accuracy of 89.88%. The model achieves such a high accuracy because in Random forests the estimators should not be correlated and with a large number of features, the features' split for each node would be different and hence the trees are uncorrelated.

As shown in Figure [5], if we take the features on the basis of their importance levels, as show in Figure [4], we have an increase in accuracy until the model hits a saturation point. There are 11 features as described in the notebook which can give 85% accuracy and can be treated as the most important. **Fishes**, **sugary snacks** and **dairies** are some of them.

In Table [3], we have shown a confusion matrix as to how many allergen labels of food items are classified as **correct**, **incorrect** and **miss**. Instead of the usual true and false, we made these 3 categories since it is a **multiple multi-class** problem. All the labels which are correctly classified are termed **correct** (true positives and true negatives). If the ground truth of an allergen is 1 but classification is 0; then we treat it as **miss** (false negative), since the algorithm has failed to detect this allergen. All labels which are 0s but are classified as 1s are labeled as **incorrect** (false positive) since these allergens are not present in the food item but the algorithm predicted their existence.

Type	Correct (TP, TN)	Incorrect (FP)	Miss (FN)
Training	269543	8941	8971
Testing	87032	4989	4796

Table 3: Number of correct, incorrect and miss allergen labels predicted (Confusion Matrix)

6 Conclusion

In Open Food Facts data, only 10% of food items have allergens labels and in this project we trained a machine learning algorithm that can effectively predict possible allergens given the food categories. However, we think that ingredients of food items would give more suitable features for this type of classification problem. However, for the current dataset, the ingredient content of food items is very challenging to preprocess and extract useful features for training. The major challenges are non availability of ingredients text in one major language and sparse distribution of languages. As future work, we would like to explore various Natural Language Processing techniques to clean ingredients data and extract meaningful features.

We also tried to solve the problem of searching the Food Facts database for food items having similar nutritional value as the given search query. The results of this type of query can be filtered for various tags like allergens, additive, etc. and help in finding more apt food given the constraints.

References

International Food Information Council (IFIC) Foundation US Food and Drug Administration (FDA). 2004. Food ingredients and colors. *Journal*.

I. Katsavounidis, C. C.J. Kuo, and Zhen Zhang. 1996. Fast tree-structured nearest neighbor encoding for vector quantization. *Trans. Img. Proc.*, 5(2):398–404.

Jayne Leonard. 2018. Chinese restaurant syndrome: What you need to know. *Medical News Today*.

A Lowry, S Hossain, and W Millar. 1987. Binary search trees for vector quantisation. pages 2205 – 2208, 05.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

7 Appendix

A Binary Search Vector Quantization

Vector Quantization is a quantization technique that is commonly used to quantize vectors into K codebook symbols (or clusters). This is quite similar to K-means clustering algorithm and works in an iterative way. But unlike K-means algorithm, Vector Quantization can be used to construct a binary search tree for vectors as after every iteration, the number of clusters become twice and the algorithm ends in $\mathcal{O}(log(K))$ iterations.

A.1 Algorithm

We first provide the algorithm of constructing Binary Search Tree using Vector Quantization.

Let S denote the set of all vectors and $iter$ denote the current iteration step of vector quantization. Here we use breadth first traversal to traverse the tree. So, we denote the queue that holds the nodes of tree to be traversed as *Queue*.

Initialization of root node: Set the root node of Binary Tree as: **value** = mean of all vectors in S, **left-child** = $null$, **right-child** = $null$, **samples** = S.

For *iter* in $\{0, 1, ..., log(K)\}$:

Push the root node in *Queue*.

While *Queue* is not empty,

1. *current_node* = *Queue.pop()*. If *current_node* has children, push them to *Queue* and continue in current loop, else go to next step.

2. *current_node* does not have children, so split **value** into two init_mean by adding and subtracting small random number from it. Split the set **samples** into two parts where part1 is closer to init_mean 1 and part2 is closer to init_mean 2. Update the means of respective parts and assign them respectively to the children of *current_node*.

After $log(K)$ iterations, the binary search tree has approximately N/K vectors in each leaf node, where N is size of S.

Now, to query the binary search tree for a given vector, we compare it with the root node and choose the child node whose **value** is closer to query vector. Similarly, we choose the next nodes till we reach the leaf node. All vectors belonging to the leaf node are returned.

B Word2Vec

Word2Vec is a model involving the use of neural networks that are used to produce word embeddings i.e. words or phrases from the vocabulary are mapped to vectors of real numbers.

Normally, these are 2 layers networks that are used to reconstruct linguistic context of words and find patterns in text. It takes in a large corpus of data and produces vectors space models with hundreds of dimensions. There are certain terms to remember in the context of Word2Vec:

- Target word: The word for which the vector needs to be generated.
- Context words: The words neighbouring or close to the target word.
- Window: The size of the context window before and after a given word which would be included as context words. This helps in establishing a relation between the target word and the neighbouring words.
- Dimensionality: Quality of the word embedding increases with higher dimensionality since it is able to capture more information. However after a certain point, the gain would be marginal.
- Subsampling: With most of the NLP tasks, words with high frequency provide very little information or not important. Words with a certain frequency in the training data would be subsampled for faster training of the word embedding model.
- Workers: Number of cores to train the model on in parallel.

Word2Vec has 2 variants: Continuous Bag-of-Words model (CBOW) and the Skip- Gram model. These models are similar to each other except that CBOW predicts similar target words from input context words and the skip-gram does the inverse by predicting the similar input context words from the target words.

Consider the sentence with the skip-gram approach: the quick brown fox jumped over the lazy dog Lets define the context of these words from left to right with a window of size 1. Then we get,

([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...

The task is to predict the input context word(s) from the target word. For example, it needs to predict the words, the and brown from quick. This is fed to the skip-gram model and an update is performed to the embedding by calculating the gradient and by reducing the loss in the right direction. When this process is repeated over the entire training set, this has the effect of moving the embedding vectors around for each word until the model is successful at discriminating real words from noise words and infact captures some semantic information about words and their relationships to one another.

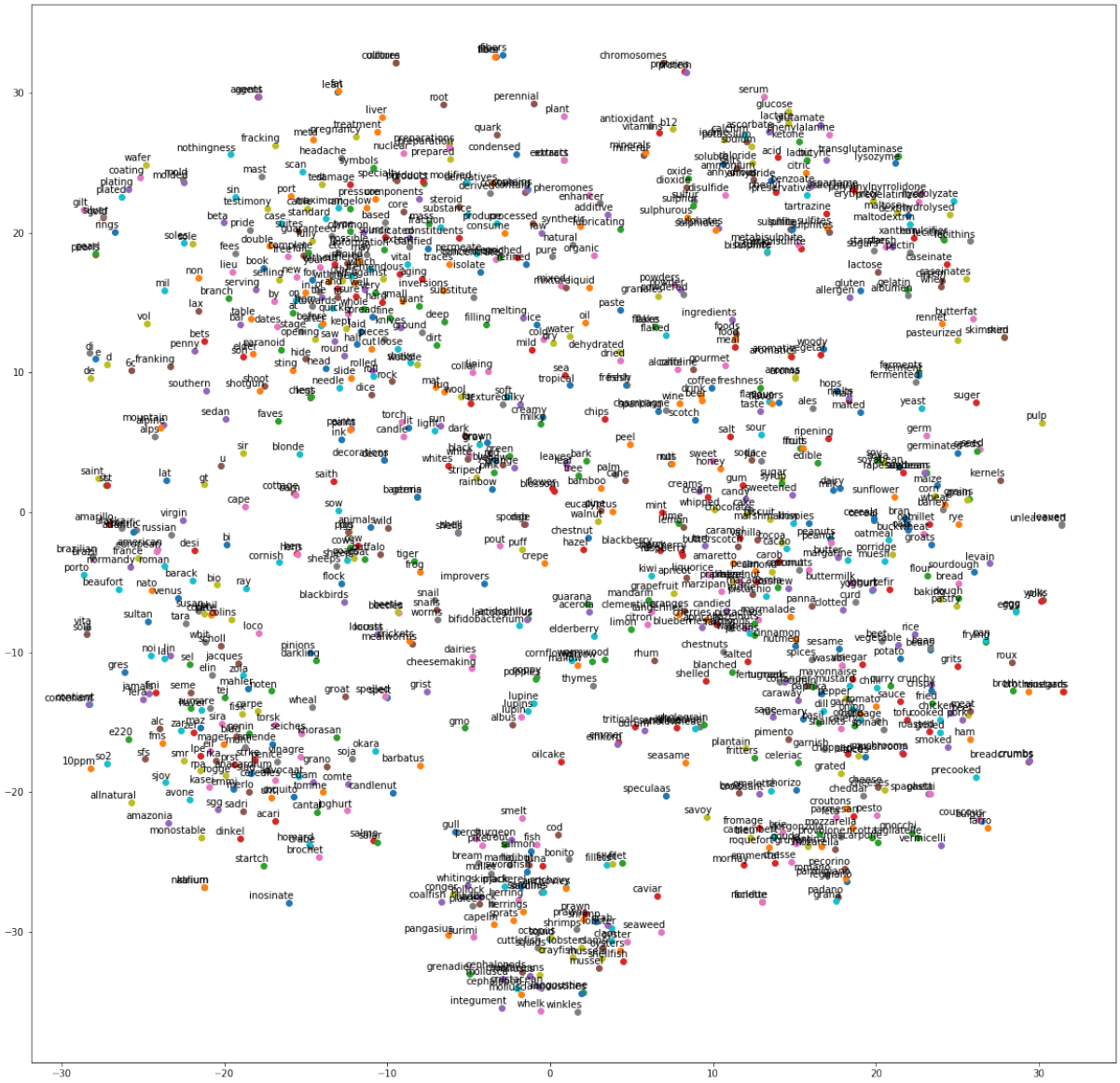

Figure 6: Word2Vec allergen words-separated and clustered together on similarity