REPORT

BING LIU

Abstract. In this report, I will talk about my work achievement. After systematically learning Probability theory and Pattern recognition.I test the learning results by completing a classification kaggle question. The following is a detailed introduction about the kaggle competition of what's cooking.

Contents

Date: 2021-04-26.
2020 *Mathematics Subject Classification.* Artificial Intelligence.
Key words and phrases. Pattern recognition, NLP, python.

1. Introduction

1.1. **Background.** If you're in Northern California, you'll be walking past the inevitable bushels of leafy greens, spiked with dark purple kale and the bright pinks and yellows of chard. Across the world in South Korea, mounds of bright red kimchi greet you, while the smell of the sea draws your attention to squids squirming nearby. India's market is perhaps the most colorful, awash in the rich hues and aromas of dozens of spices: turmeric, star anise, poppy seeds, and garam masala as far as the eye can see.

1.2. **Target.** Some of our strongest geographic and cultural associations are tied to a region's local foods. This playground competitions asks to predict the category of a dish's cuisine given a list of its ingredients.

2. Data

2.1. **Data Description.** In the dataset, it include the recipe id, the type of cuisine, and the list of ingredients of each recipe (of variable length). The data is stored in JSON format.

In the test file test.json, the format of a recipe is the same as train.json, only the cuisine type is removed, as it is the target variable you are going to predict.

- **train.json** – the training set containing recipes id, type of cuisine, and list of ingredients. It contains 39774 entries.
- **test.json** – the test set containing recipes id, and list of ingredients. It contains 9944 entries.
- **sampleSubmission.csv** – a sample submission file in the correct format. It just contains 2 columns that is "id" and "cuisine".

2.2. **Missing Values Analysis.** I statistic missing value in the dataset, Luckily the dataset do not has any missing value.

```
(df.isnull().sum() / len(df)) * 100
(test_df.isnull().sum() / len(test_df)) * 100
```

Figure 1. Missing values analysis

2.3. **Statistic Cuisine.** There are a total of 20 types of cuisines, and the percentage of each type of cuisine is as follows.

2.4. **Word Frequency Statistics.** Count the number of occurrences of each word in the entire dataset 'ingredients', In order to analyze the importance of related vocabulary.

Count the number of 'ingredients' per cuisine. There are twenty cuisine in the train data, just show the 'Greek' word frequency statistics. Just Visualize the 25 most commonly used ingredients.

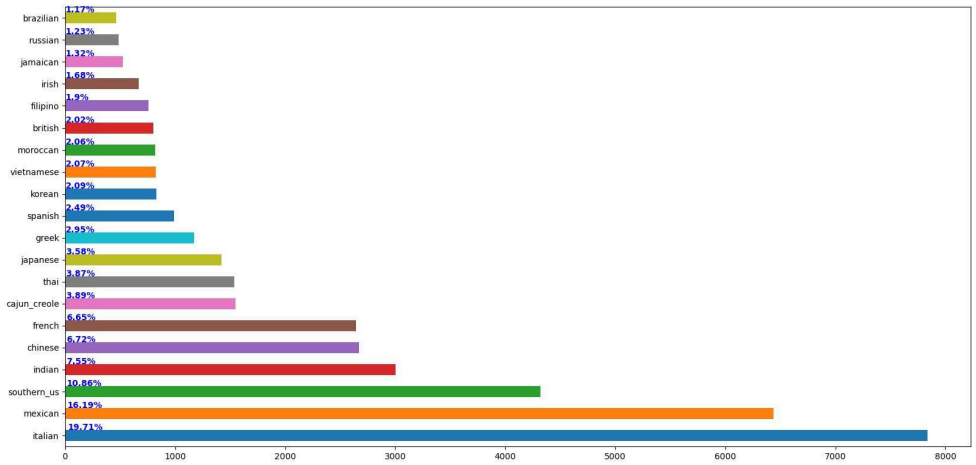

FIGURE 2. Statistic Cuisine

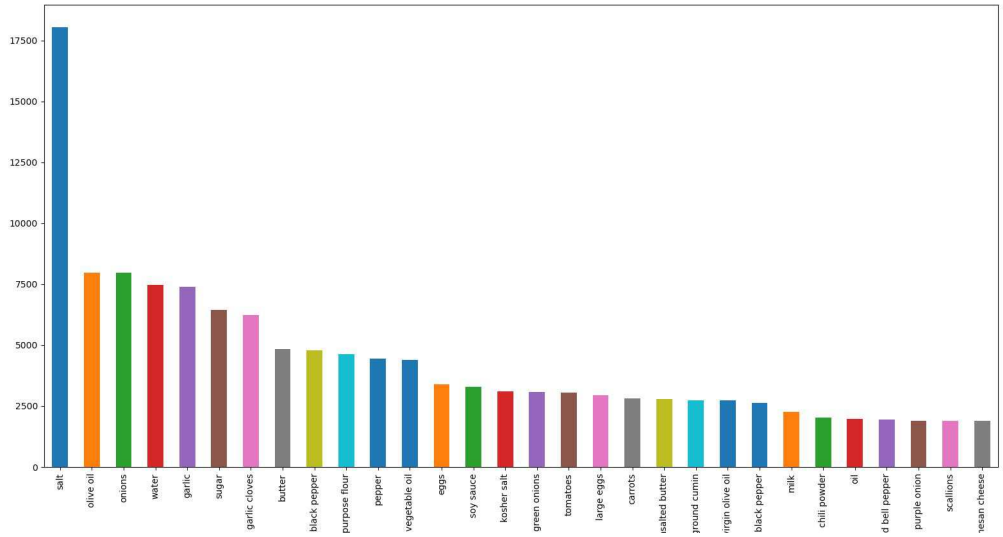

FIGURE 3. Word frequency statistics of entire dataset

2.5. **String Preprocess.** 1:Use the WordNetLemmatizer().lemmatize() method to restore the part of speech
2:remove the useless suffix of the word
3:remove the non-letter symbols
4:change the uppercase letters to lowercase

3. FEATURE ENGINEERING

3.1. **Count Vectorizer.** Convert a document into a vector by counting to complete feature extraction, which get a word frequency matrix.

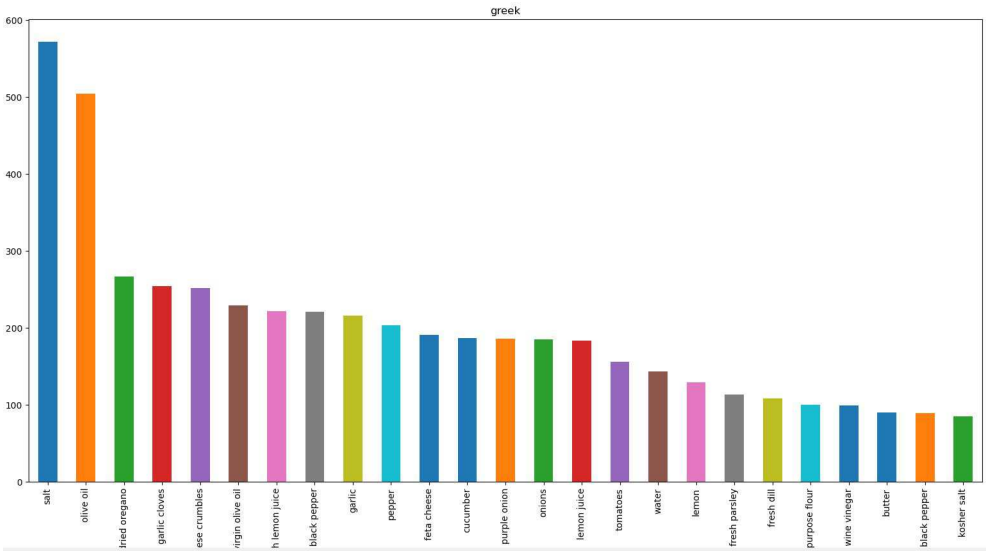

FIGURE 4. Greek word frequency statistics

```
# 处理字符串
def process_string(x):
    x = [" ".join([WordNetLemmatizer().lemmatize(q) for q in p.split()]) for p in x]  # Lemmatization
    x = list(map(lambda x: re.sub(r'\(.*oz.\)|crushed|crumbles|ground|minced|powder|chopped|sliced', '', x), x))
    x = list(map(lambda x: re.sub("[^a-zA-Z]", " ", x), x))  # To remove everything except a-z and A-Z
    x = " ".join(x)  # To make list element a string element
    x = x.lower()  # 所有大写字母转换为小写字母
    return x
```

FIGURE 5. String preprocess

```
# Count Vectorizer
def count_vectorizer(train, test=None):
    cv = CountVectorizer()
    train = cv.fit_transform(train)
    if test is not None:
        test = cv.transform(test)
        return train, test, cv
    else:
        return train, cv
```

FIGURE 6. code of count vectorizer

3.2. **TFiDF Vectorizer.** Input the word frequency matrix to get the TF-IDF weight matrix.

3.3. **Cluster as Parameter.** There are 20 different types of cuisine to classify. Certain groups of cuisine may have much more similarity than others. So we use

```python
# TFiDF Vectorizer
def tfidf_vectorizer(train, test=None):
    tfidf = TfidfVectorizer(stop_words='english', ngram_range=(1, 1), analyzer="word", max_df=.57,
                            binary=False, token_pattern=r'\w+', sublinear_tf=False)
    train = tfidf.fit_transform(train)
    if test is not None:
        test = tfidf.transform(test)
        return train, test, tfidf
    else:
        return train, tfidf
```

FIGURE 7. code of TFiDF vectorizer

the clustering information as part of the feature.

The "cuisine_df" is also used to generate the weight matrix through the TFiDF Vectorizer. Use PCA to reduce dimensionality.

Predict clusters in the test data, encoded as Onehot vectors.

Combine the TFIDF vector and the cluster vector as a feature vector.

4. Model And Conclusion

I have choose the SVC as classification model. The percentage of the number of

```python
from sklearn.svm import LinearSVC, SVC

C = 604.5300203551828
gamma = 0.9656489284085462

clf = SVC(C=float(C), gamma=float(gamma), kernel='rbf')
clf.fit(train, target)
y_pred = clf.predict(test)
```

FIGURE 8. SVC

correctly classified cuisines in the total number is used as the accuracy rate.
My accuracy rate is 81.06%

(A. 1) School of Computer Science,, Jilin University, ChangChun 130012, China
Email address, A. 1: bliu@tulip.academy