© The Author(s), under exclusive license to APress Media, LLC, part of Springer Nature 2023
P. Mishra, *Explainable AI Recipes*
https://doi.org/10.1007/978-1-4842-9029-3_5

5. Explainability for Natural Language Processing

Pradeepta Mishra[1]
(1) Bangalore, Karnataka, India

Natural language processing tasks such as text classification and sentiment analysis can be explained using explainable AI libraries such as SHAP and ELI5. The objective of explaining the text classification tasks or sentiment analysis tasks is to let the user know how a decision was made. The predictions are generated using a supervised learning model for unstructured text data. The input is a text sentence or many sentences or phrases, and we train a machine learning model to perform text classification such as customer review classification, feedback classification, newsgroup classification, etc. In this chapter, we will be using explainable libraries to explain the predictions or classifications.

There are three common problems where explainability is required in natural language processing.

- Document classification, where the input is a series of sentences extracted from a document, and the output is the label attached to the document. If a document is misclassified or someone wants to know why a document is being classified by the algorithm in a certain way, we need to explain why.
- For named entity recognition tasks, we need to predict the entity to which a name belongs. If it is assigned to another entity, we need to explain why.
- For sentiment analysis, if a sentiment category is wrongly assigned to another category, then we need to explain why.

Recipe 5-1. Explain Sentiment Analysis Text Classification Using SHAP

Problem

You want to explain sentiment analysis prediction using SHAP.

Solution

The solution takes into account the most common dataset available, which is the IMDB sentiment classification dataset from the SHAP library. It can be accessed using the SHAP dataset. We will be using the SHAP library to explain the predictions.

How It Works

Let’s take a look at the following example (see Figure 5-1 and Figure 5-2):

```
!pip install shap
import warnings
warnings.filterwarnings("ignore")
import sklearn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import numpy as np
import shap
import pandas as pd
from keras.datasets import imdb
corpus,y = shap.datasets.imdb()
corpus_train, corpus_test, y_train, y_test = train_test_split(corpus, y, test_size=0.2, random_
vectorizer = TfidfVectorizer(min_df=10)
X_train = vectorizer.fit_transform(corpus_train).toarray() # sparse also works but Explanation
X_test = vectorizer.transform(corpus_test).toarray()
corpus_train[20]
Well how was I suppose to know this was...............................
y
array([False, False, False, ..., True, True, True])
model = sklearn.linear_model.LogisticRegression(penalty="l2", C=0.1)
model.fit(X_train, y_train)
explainer = shap.Explainer(model, X_train, feature_names=vectorizer.get_feature_names())
shap_values = explainer(X_test)
shap.summary_plot(shap_values, X_test)
```

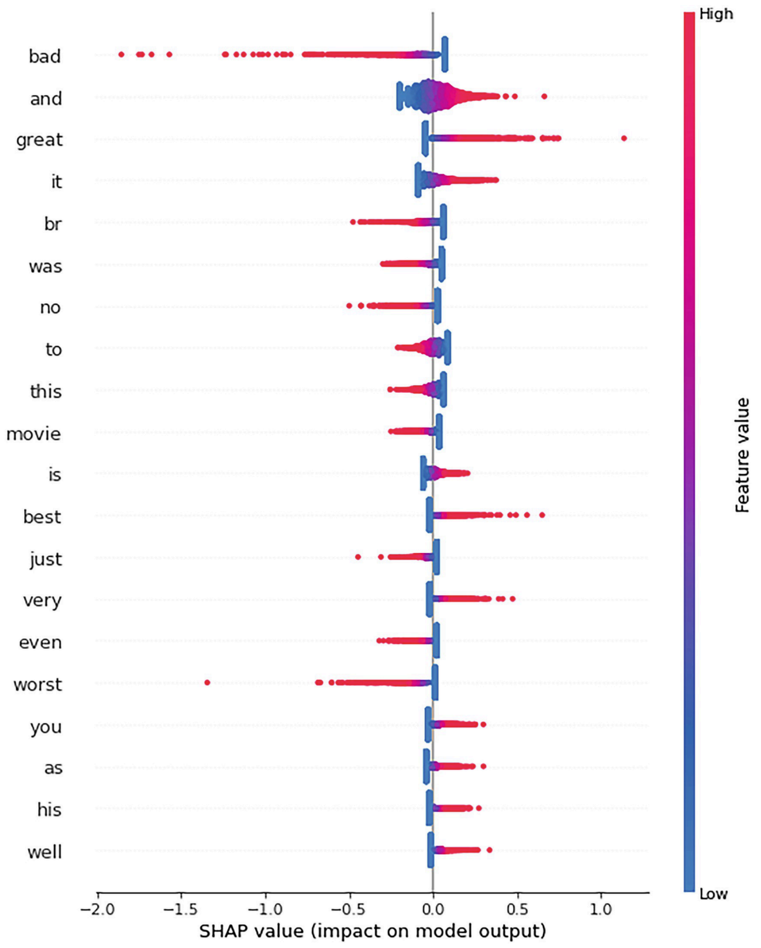

Figure 5-1 Summary plot from sentiment classification

```
shap.plots.beeswarm(shap_values)
```

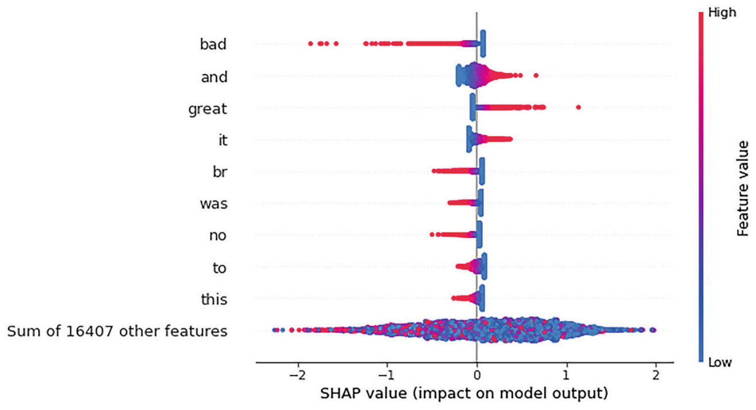

Figure 5-2 SHAP values showing very sparse features

```
names = vectorizer.get_feature_names()
names[0:20]
pd.DataFrame(X_train,columns=names)
```

	00	000	007	01	02	05	06	10	100	1000	...	zombi	zom-bie	zom-bies
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.078969	0.0	0.0	...	0.0	0.0	

```
ind = 10
shap.plots.force(shap_values[ind])
print("Positive" if y_test[ind] else "Negative", "Review:")
print(corpus_test[ind])
Positive Review:
"Twelve Monkeys" is odd and disturbing, .......................................
```

Recipe 5-2. Explain Sentiment Analysis Text Classification Using ELI5

Problem

You want to explain sentiment analysis prediction using ELI5.

Solution

The solution takes into account the most common dataset available, which is the IMDB sentiment classification. We will be using the ELI5 library to explain the predictions.

How It Works

Let's take a look at the following example:

```
!pip install eli5
import eli5
eli5.show_weights(model, top=10) #this result is not meaningful, as weight and feature names are
```

y=True top features

Weight?	Feature
+3.069	x6530
+2.195	x748
+1.838	x1575
+1.788	x5270
+1.743	x8807
… 8173 more positive …	
… 8234 more negative …	
-1.907	x15924
-1.911	x1239
-2.027	x9976
-2.798	x16255
-3.643	x1283

The ELI5 results are not meaningful as they provide only the weights and features, and the feature names are not meaningful. To make the results interpretable, we need to pass the feature names.

```
eli5.show_weights(model,feature_names=vectorizer.get_feature_names(),target_names=['Negative','P
#make sense
```

y=Positive top features

Weight?	Feature
+3.069	great
+2.195	and
+1.838	best
+1.788	excellent
+1.743	love
+1.501	well
+1.477	wonderful

Weight?	Feature
+1.394	very
… 8170 more positive …	
… 8227 more negative …	
-1.391	just
-1.407	plot
-1.481	poor
-1.570	even
-1.589	terrible
-1.612	nothing
-1.723	boring
-1.907	waste
-1.911	awful
-2.027	no
-2.798	worst
-3.643	bad

Recipe 5-3. Local Explanation Using ELI5

Problem

You want to explain an individual review in the sentiment analysis prediction using ELI5.

Solution

The solution is takes into account the most common dataset available, which is the IMDB sentiment classification dataset. We will be using the ELI5 library to explain the predictions.

How It Works

Let’s take a look at the following example. Here we are taking into account three reviews, record numbers 1, 20, and 100, to explain the predicted class and relative importance of each word contributing positively and negatively to the predicted class.

```
Eli5.show_prediction(model, corpus_train[3], vec=vectorizer,
                     target_names=['Negative','Positive'])
# explain local prediction
```

y=Positive (probability **0.739**, score **1.042**) top features

Contribution?	Feature
+0.869	Highlighted in text (sum)
+0.174	<BIAS>

as a matter of fact, this is one of those movies you would have to give 7.5 to. The fact is; as already stated, it's a great deal of fun. Wonderfully atmospheric. Askey does indeed come across as over the top, but it's a great vehicle for him, just as oh, mr porter is for will hay. If you like old dark house movies and trains, then this is definitely for you.<br /><br />strangely enough it's the kind of film that you'll want to see again and a

```
........................
eli5.show_prediction(model, corpus_train[4], vec=vectorizer,
                     target_names=['Negative','Positive'])
# explain local prediction
```

y=Negative (probability **0.682**, score **-0.761**) top features

Contribution?	Feature
+0.935	Highlighted in text (sum)
-0.174	<BIAS>

how could 4 out of 16 prior voters give this movie a 10? How could more than half the prior voters give it a 7 or higher? Who is voting here? I can only assume it is primarily kids -- very young kids. the fact is that this is a bad movie in every way. the story is stupid; the acting is hard to even think of ....

```
eli5.show_prediction(model, corpus_train[100], vec=vectorizer,
                     target_names=''Nagativ''''Positiv'']) # explain local prediction
```

y=Negative (probability **0.757**, score **-1.139**) top features

Contribution?	Feature
+1.313	Highlighted in text (sum)
-0.174	<BIAS>

this movie was so poorly written and directed i fell asleep 30 minutes through the movie..................

The green patches show positive features for the target class `positive`, and the red parts are negative features that correspond to the `negative` class. The feature value and the weight value indicate the relative importance of words as features in classifying sentiments. It is observed that many stop words or unwanted words are present in the tokenization process; hence, they are appearing as features in the feature importance. The way to clean it up is to use preprocessing steps such as applying stemming, removing stop words, performing lemmatization, removing numbers, etc. Once the text cleanup is completed, then the previous recipes can be used again to create a better model to predict the sentiments.

Conclusion

In this chapter, we covered how to interpret the text classification use cases such as sentiment analysis. However, for all such kinds of use cases, the process will remain same, and the same recipes can be used. The modeling technique selection may change as the features increase, and we can use complex models such as ensemble modeling techniques like random forest, gradient boosting techniques, and catboost techniques. Also, the preprocessing methods can change. For example, the count vectorizer, TF-IDF vectorizer, hashing vectorizer, etc., can be applied with stop word removal to clean the text to get better features. The recipes to run different variants of ensemble models were covered in the previous chapter. In the next chapter, we are going to cover times-series model explainability.