



Saving Classifiers with NLTK



Training classifiers and machine learning algorithms can take a very long time, especially if you're training against a larger data set. Ours is actually pretty small. Can you imagine having to train the classifier every time you wanted to fire it up and use it? What horror! Instead, what we can do is use the Pickle module to go ahead and serialize our classifier object, so that all we need to do is load that file in real quick.

So, how do we do this? The first step is to save the object. To do this, first you need to import pickle at the top of your script, then, after you have trained with .train() the classifier, you can then call the following lines:

Ad closed by Google

```
save_classifier = open("naivebayes.pickle", "wb")
pickle.dump(classifier, save_classifier)
save_classifier.close()
```

This opens up a pickle file, preparing to write in bytes some data. Then, we use pickle.dump() to dump the data. The first parameter to pickle.dump() is what are you dumping, the second parameter is where are you dumping it.

After that, we close the file as we're supposed to, and that is that, we now have a pickled, or serialized, object saved in our script's directory!

Next, how would we go about opening and using this classifier? The .pickle file is a serialized object, all we need to do now is read it into memory, which will be about as quick as reading any other ordinary file. To do this:



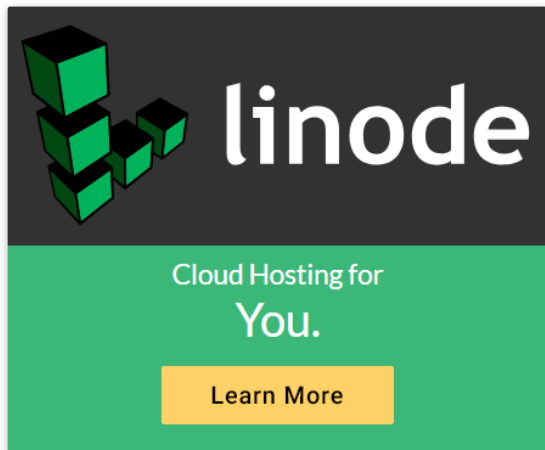
```
classifier_f = open("naivebayes.pickle", "rb")
classifier = pickle.load(classifier_f)
classifier_f.close()
```

Here, we do a very similar process. We open the file to read as bytes. Then, we use `pickle.load()` to load the file, and we save the data to the classifier variable. Then we close the file, and that is that. We now have the same classifier object as before!

Now, we can use this object, and we no longer need to train our classifier every time we wanted to use it to classify.

While this is all fine and dandy, we're probably not too content with the 60-75% accuracy we're getting. What about other classifiers? Turns out, there are many classifiers, but we need the scikit-learn (sklearn) module. Luckily for us, the people at NLTK recognized the value of incorporating the sklearn module into NLTK, and they have built us a little API to do it. That's what we'll be doing in the next tutorial.

The next tutorial: [Scikit-Learn Sklearn With NLTK](#)



Tokenizing Words and Sentences with NLTK

Stop words with NLTK

Stemming words with NLTK

Part of Speech Tagging with NLTK

Chunking with NLTK

Chinking with NLTK

Named Entity Recognition with NLTK

Lemmatizing with NLTK

The corpora with NLTK

Wordnet with NLTK



Converting words to Features with NLTK

Naive Bayes Classifier with NLTK

Saving Classifiers with NLTK

Scikit-Learn Sklearn with NLTK

Combining Algorithms with NLTK

Investigating bias with NLTK

Improving Training Data for sentiment analysis with NLTK

Creating a module for Sentiment Analysis with NLTK

Twitter Sentiment Analysis with NLTK

Graphing Live Twitter Sentiment Analysis with NLTK with NLTK

Named Entity Recognition with Stanford NER Tagger

Testing NLTK and Stanford NER Taggers for Accuracy

Testing NLTK and Stanford NER Taggers for Speed

Using BIO Tags to Create Readable Named Entity Lists

November 21, 2019

Trump Ends Another Obama Era Program

If you owe less than \$726,525 on your home and haven't missed a mortgage payment in 6 months, use Congress's mortgage stimulus program for the middle class. You'll be shocked when you see how much you can save.

Tap Your Age:

18-25	26-35	36-45	46-55	56-65	65+
-------	-------	-------	-------	-------	-----

Recalculate Your House Payment

©2019 NMLS ID 167283; 3306 nmlsconsumeraccess.org

You've reached the end!

Contact: Harrison@pythonprogramming.net.

Support this Website!

Consulting and Contracting

Facebook

Twitter


Instagram

Legal stuff:

[Terms and Conditions](#)

[Privacy Policy](#)



 search

[Home](#)

[+=1](#)

[Support the Content](#)

[Community](#)

[Log in](#)

[Sign up](#)