empythy Documentation

Release 0.9.1

Preston Parry

August 29, 2016

Contents

Contents

CHAPTER 1

Installation

```
pip install empythy
```

CHAPTER 2

Core Functionality

A quick demonstration of what you're really here for: easily getting sentiment predictions.

```
from empythy import EmpathyMachines
sentiment_classifier = EmpathyMachines()
sentiment_classifier.train()

text_string = "I love machine learning! And bikes. And wearing fun party pants."
sentiment_classifier.predict(text_string)
# returns ['positive']

text_list = [
  "I effing love pho.",
  "Ummm, can you add some exclamation points to the end of that previous one??",
  "I hate that soggy moment when you realize that bungee cording 8 pho containers \
  to the outside of your backpack is not, in fact, an effective way to carry noodle soup."
]

sentiment_classifier.predict(text_list)
# returns ['positive', 'neutral', 'negative']. Hopefully- neutral is tough :)
```

CHAPTER 3

Basic API Documentation

First, you must instantiate a new EmpathyMachines object. Convention is to save it into a variable called `sentiment_classifier`. The rest of these docs will assume that you have done exactly that (`sentiment_classifier = EmpathyMachines()`). If you're headstrong enough to do it differently, we'll assume you're also smart enough to adjust your reading of these docs appropriately :)

class `EmapthyMachines`

Param None. Literally, as in, don't pass in any arguments when creating a new EmpathyMachines instance.

`sentiment_classifier.`**`train`**(*corpus=Twitter*)

This method will train your nlp classifier. This must be done before trying to get predictions.

Return type None. This simply trains the classifier to prepare it to make predictions.

Parameters **`corpus`** (*`string`*) – Which corpus of documents you want to train this model on. Currently, empythy ships with two corpora (Twitter, MovieReviews), along with the ability to pass in your own corpus to train on! If you're interested in getting fancy, instructions on how to train on your own custom dataset are later in this doc.

The included corpora are:

- `Twitter`

  [CrowdFlower](http://www.crowdflower.com/data-for-everyone) hosts a number of Twitter corpora that have already been graded for sentiment by panels of humans. I aggregated together 6 of their corpora into a single, aggregated and cleaned corpus, with consistent scoring labels across the entire corpus. The cleaned corpus contains over 45,000 documents, with positive, negative, and neutral sentiments, along with a score of how confident they are in that assessment.

- `MovieReviews`

  The classic NLTK corpus of movie reviews.

`sentiment_classifier.`**`predict`**(*text*)

Pass in a text (or list of texts), and get a prediction back. Kind of like Christmas Eve: leave a cookie for Santa, and get presents back- and presumably magic happens in between.

Parameters **`text`** (*`string, or list of strings`*) – You can pass in either a single string, or a list (technically, nearly any iterable) of strings. If you pass in a list, you will get back a list of equal length. If you pass in a single string, you'll get back a *list* with a single string.

It's an imperfect design decision I made because I wanted to keep the return type consistent. If you don't like it, come help me build the next version- I love people who disagree with me!

Return type

List.

No matter whether you pass in a string or a list, you will *always* get a list back. It's like cooking with eggs & cheese: it doesn't really matter what you toss in, you'll always get something reliably tasty out of it.

Each item in the returned list will be one of three strings: `positive`, `negative`, or `neutral`.

CHAPTER 4

Training on your own corpus

Warning, bad pun ahead: This section also known as "flex those muscles".

This section is totally optional- you can train on the included Twitter or MovieReviews corpora totally fine and never have to go through the work of assembling your own custom corpus.

One of the best parts of this module is that you can train it on your own data! The corpora I included are somewhat useful, but what I'm most excited by is to see what y'all train it on. If you train it on something fun, please save your data as a .csv file and send it over! I might include it in the package for others to use.

Here's what the process looks like to train on your own data:

`sentiment_classifier.`**`train`**(*corpus='custom'*, *corpus_array=my_list_of_text_objects*, *file_name='path/to/data/file.csv'*)

There are two ways to pass in your own custom data to train on. In both cases, you *must* specify `corpus='custom'`.

1. `corpus_array`: Pass in a list of texts.
2. `file_name`: A path to a .csv file that holds your training data.

`corpus_array`

Each object in this list must have two properties: `text` and `sentiment`.

`file_name`

Simply point us to a .csv file with at least the following two columns: `text` and `sentiment`.

More info on each of these features/columns

1. `text`: The actual text of the message.
2. `sentiment`: The correct sentiment for this text.
3. `confidence`: OPTIONAL. If you have a confidence score for how confident you are this is the right sentiment for the message, you can pass that in here. Useful if you have, say, sentiment scored from an onsite team and sentiment scored using Amazon's Mechanical Turk. You would presumably give the onsite team a higher confidence (maybe 0.9) than the scores from Amazon's Mechanical Turk (maybe 0.5).

E

P

T