API Module - Project

November 11, 2018

At the end of the first module, we used the Yelp Fusion API to iterate through a bunch of Toronto-area restaurants and download their reviews and star ratings as training data for a very simple Naive Bayes classifier to do sentiment analysis into two 'positive' and 'negative' classes. If you need a refresher, consult the last sections of the Day 1 notebook on RAC.

However, it turns out that Yelp's Business Search API had a very significant limitation: it only displayed the first sentence or two of each review, truncating the rest with an ellipsis at the end. This was good enough for a simple proof-of-concept, but it obviously hinders a data-hungry model from reaching its peak potential.

The way Yelp intended the API to be used was for the consumer (i.e, the application developer using the API) to use the included `reviews[x].url` field to redirect the end-user to a web page containing the rest of the review (so that Yelp can get a few more ad impressions in the process). Luckily, in the second module, we learned how to use HTML parsers to extract relevant pieces of text from pages that we know the URL to!

The goal of this assignment, then, is to use any and all of the tools at our disposal (calling endpoints of the REST API, GraphQL, parsing HTML, crawling the rest of the Yelp website) to amass the best training data set for our task that we possibly can.

The scope of the project does not include the actual classification model itself; that is purely a litmus test for the quality of our data set. The input for this model, as implemented in the notebook, is extremely simple: just a massive list of tuples where the first element of each tuple is a natural-language review of a restaurant, and the second element is either the string `'positive'` or `'negative'` (which we infer by the star rating the API returned for that review). Our goal is simply to extract as large and as rich a training set for this simple model as we can.

A few notes and caveats:

- Your code should use no more than 1,000 Yelp Fusion API calls and/or 100 GraphQL calls, and it should take no longer than 2 hours to run (which places a practical limit on the number of HTML pages you can scrape).
- If your code does use a significant number of API calls, please include your Yelp API Client ID and API Key in your submission, since otherwise I risk exceeding my 5,000 API call daily limit when grading so many submissions. You can find this at `https://www.yelp.com/developers/v3/manage_app` once you've registered your Yelp developer account.
- You are not limited to the libraries discussed in this course - feel free to import any Python packages that are available through either PyPI or Anaconda (which basically amounts to every publicly-available Python package under the sun). I will figure out how to install it.
- If you're really inspired to augment (or even replace) the very simple sentiment classifier provided in the notebook, feel free to. One of the key principles of using APIs or scraping to assemble data sets for machine learning models is understanding how to tune the data you collect for the strengths and weaknesses of the models you're using, and vice versa.
- **Hint:** One of the biggest flaws in the sample implementation from the notebook is that our data skewed *very* heavily towards positive reviews. Having imbalanced sizes for the different classes in your training data is a notorious problem for classification problems. Is there a way we can overcome this bias in the data as we're gathering it?

Your full data set should be written to a file named `data.json` prior to feature extraction, in the same format as the notebook: a list of the form

```
[
  ["review text 1", "positive|negative"],
  ["review text 2", "positive|negative"],
  ...
]
```

Include as much data as you need to pull the F-score of the provided model as high as you can make it go.