DATA 240 **Data Science Applications**

Project 2

Assigned: March 02 *Due: April 1/5*

Abstract

For this project we'll be working with text data in Python. You'll be required to load and preprocess some BBC news articles, train a machine learning classifier, and come up with some additional insights. You will again need to both write a report and give a final presentation.

The final presentation is due in-class on Thursday, April 1. The final report is due by midnight on Monday, April 5. **Note**: waiting until after the presentation to start on the report will probably result in a very bad grade. Having at least a draft of your report will make writing the presentation easier.

First Steps: Loading the Articles

You will first need to load the articles from the text file. This is not trivial – text data is often messy and inconsistent! I have made the data is relatively clean, but it is still difficult to work with clean text.

All of the news articles are in a single file. The first thing you'll need to do is read in the file. Then you'll have to split the text into articles. There are various ways to do this, but I recommend you learn about regular expressions and the `re` package in Python. That will work for this project, but more importantly, you are likely to come across regular expressions as a data scientist or anyone who ever writes code. At the *end* of every article is a line that starts with "Document ID: " and then a bunch of letters, numbers, and dashes. Use that to split into articles.

For each article, there are several pieces. First a topic number (there are 5, numbered 0–4), then a title and a subheading (summary sentence). Finally, there is a the article body, which occurs after "`-- --`" on a line by itself.

You can load this data into whatever format you find useful, but I recommend using a Pandas DataFrame. This way you will have topic number, title, subheading, and body columns, and a row for each article. Pandas DataFrames are very similar to R DataFrames, but may take some getting used to if you have only used R.

Classification Model

One of the major outputs you'll produce is a classification model. Specifically, we want to train a machine learning model so that it can take in information about an article and predict which topic number it came from.

I have placed you into groups such that every group has someone who took my machine learning class, in the hope that they can guide you through it. You will be using the `sklearn` package to produce your machine learning model.

The first step is to turn each text article into a series of numbers – machine learning models can only work with numbers. The `sklearn` documentation has some good information about "vectorizing" text in this manner. I normally use `TfidfVectorizer` for this sort of thing, though many of the other options will also work well..

The second step is to actually train the machine learning model on this "vectorized" data. The basic idea in (supervised) machine learning is that you'll choose some sort of statistical model, give it some data AND the answers for that data, in a similar way that a teacher gives you assignments and then grades them. There are many possible classification models to use. Don't worry – you don't need to know the statistical details behind any of them. `sklearn` makes them easy to use once you are comfortable reading the documentation.

The **checkpoint** presentation should come around this point. You should present for 10-12 minutes about what you have done, showing me how you loaded the data, processed it, and trained a classifier. It doesn't have to be a *good* classifier, but I would like to see you have something working.

Start with a very simple model. After the checkpoint you can spend a little time trying out different vectorizers, different models, and some of the numerous options available for each. Each model has lots of

options (called "hyperparameters") that change how they work. `sklearn` includes features for automatically finding the right versions of these options – see `GridSearchCV`.

I would like to see at least 85% accuracy by the end of the project. 90% would be even better, and I would love to see 95% or higher.

Descriptive Data Analysis

You should also spend a little time doing some descriptive data analysis of the data. Your goals are two-fold: (1) Tell me what the topics are, and (2) tell me something interesting about the data. For (1), you must give me some description of the topics. For (2) you have lots of options, some of which can also be used to help with (1):

- Visualize the articles with a 2D scatterplot. You will need to first "project" the data into 2 dimensions and then use `matplotlib` or `seaborn` to plot it. It's not as hard as it sounds. For projecting the data I recommend using `sklearn`'s PCA, although there are other ways to do it.
- Cluster the data and visualize the clusters. There are clustering algorithms which will take in your data and try to automatically group them into categories. You could do this for the entire dataset (in which case you may or may not get a clustering similar to the topic numbers), or for only a portion of the dataset (e.g., all articles in topic 0). You should visualize these clusters and produce some kind of description of each cluster. For example, you might show the most important 3 articles for each cluster, for some definition of important.
- Extract all the named entities (basically, proper nouns) and analyze them. For example, you might look at which entities were most common, and/or how the topics differed in use of entities. You should probably use the Python library `spacy` to do this. Another option is `nltk`, but `spacy` is easier to use.
- Analyze which words occur most frequently with other words. You might, for example, split the articles up into sentences (using `spacy` or `nltk`), and then figure out which words occur together most frequently. You could do this for all words, or you could start with a list of words you think are important, and see which *other* words occur with those words most often.
- Analyze which words appear most often, especially with respect to the different topics. This is easier than word co-occurrence, so I'll expect a deeper analysis. You might try producing some word clouds, which are always fun.
- Analyze the "sentiment" of sentences or paragraphs, especially with respect to the different topics or sentences that contain certain words (e.g., "Microsoft"). Sentiment analysis basically means figuring out whether a sentence is generally positive or negative. There are many sentiment analyzers, but the easiest way to get started is to use `nltk`.

Obviously you don't have time for all of these. Pick about 2 that interest you, or do 1 really well. It is particularly interesting to combine 2 of these types of analyses, e.g., co-occurrence of various entities.

Final Presentation and Report

As before, you need to give a final 20 minute presentation and create a report. This time, however, your report will be a Jupyter notebook file. Jupyter notebooks let you combine Python code with blocks of formatted text, similar to R Markdown.

Your presentation should again be pretty high-level: quickly but clearly present the key findings and insights in the data and your results. You don't necessarily need to do into detail about how your code works – I will see that in your `git` repo. Imagine I am a CEO who has little patience for coding details, but just wants enough details to make our company look good.

The final report should be pretty detailed, however. There is no specific length because it is hard to measure length with a Jupyter notebook file. Just make sure you are thorough. You should describe not only the main insights, but also create more visualizations and describe how you approached the project – what went well, what went horribly wrong, etc.

Resources

Like in project 1, I expect you'll spend a lot of time reading documentation and learning how to learn. Some resources that might be helpful:

- sklearn user guide
- Pandas user guide
- Spacy guide
- Online NLTK book
- NLTK + sklearn tutorial
- Matplotlib tutorial
- Seaborn tutorial
- word clouds in Python

Many of the user guides contain links to the documentation pages, which may sometimes be more useful than guides or tutorials.

If this all seems overwhemling, that's because it is. Pick a place to start, work as a team, and ask questions. Unlike project 1, I am very familiar with working with text, so I am available to help guide you in the right direction.