Natural Language Processing

Let’s learn something!

PIERIAN DATA

Python and Spark

- Let’s now learn about the basics of Natural Language Processing!
- This is the field of machine learning that focuses on creating models from a text data source (straight from articles of words).

Python and Spark

- The NLP section of the course will just contain a single custom code along example because the documentation doesn’t really have a full example and the custom code along is a larger multi-step process.

Python and Spark

- This is a very large field of machine learning with its own unique challenges and sets of algorithms and features, so what we cover here will be scratching just the surface!

Python and Spark

- Optional Reading Suggestions:
  - Wikipedia Article on NLP
  - NLTK Book (separate Python library)
  - Foundations of Statistical Natural Language Processing (Manning)

Python and Spark

- Examples of NLP
    - Clustering News Articles
    - Suggesting similar books
    - Grouping Legal Documents
    - Analyzing Consumer Feedback
    - Spam Email Detection

Python and Spark

- Our basic process for NLP:
  - Compile all documents (Corpus)
  - Featurize the words to numerics
  - Compare features of documents

Python and Spark

- A standard way of doing this is through the use of what is known as “TF-IDF” methods.
- TF-IDF stands for Term Frequency - Inverse Document Frequency
- Let’s explain how it works!

NLP

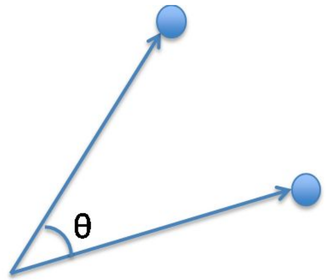
Simple Example:

- You have 2 documents:
  - “Blue House”
  - “Red House”
- Featurize based on word count:
  - “Blue House” -> (red,blue,house) -> (0,1,1)
  - “Red House” -> (red,blue,house) -> (1,0,1)

NLP

- A document represented as a vector of word counts is called a “Bag of Words”
  - “Blue House” -> (red,blue,house) -> (0,1,1)
  - “Red House” -> (red,blue,house) -> (1,0,1)
- These are now vectors in an N-dimensional space, we can compare vectors with cosine similarity:

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



NLP

- We can improve on Bag of Words by adjusting word counts based on their frequency in corpus (the group of all the documents)
- We can use TF-IDF (Term Frequency - Inverse Document Frequency)

NLP

- Term Frequency - Importance of the term within that document
  - TF(**x**,**y**) = Number of occurrences of term **x** in document **y**
- Inverse Document Frequency - Importance of the term in the corpus
  - IDF(**t**) = log(**N**/**dfx**) where
    - **N** = total number of documents
    - **dfx** = number of documents with the term

- Mathematically, TF-IDF is then expressed:

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Python and Spark

- Spark has a lot of pyspark.ml.feature tools to help out with this entire process and make it all easy for you!
- Let’s jump to a custom code along example!

Tools for NLP
Part One

Python and Spark

- Before we jump into the code along project, let's explore a few of the tools Spark has for dealing with text data.
- Then we'll be able to use them easily in our project!

Tools for NLP
Part Two

NLP Code Along

Python and Spark

- Let’s work through building a spam detection filter using Python and Spark!
- Our data set consists of volunteered text messages from a study in Singapore and some spam texts from a UK reporting site.
- Let’s get started