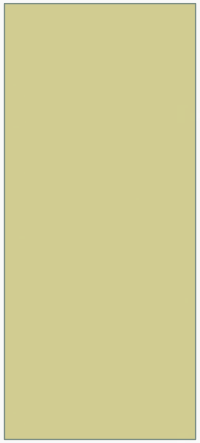
# NATURAL LANGUAGE PROCESSING

GENERAL ASSEMBLY – DATA SCIENCE LA



# WHAT IS NATURAL LANGUAGE PROCESSING (NLP)?

- Computers are really bad at understanding text.
- Natural Language Processing (NLP) is the process of transforming text into **features** that can be used by various algorithms.

# TOKENIZATION AND NORMALIZATION

- These should be the same:
  - “Data Science is AWESOME!!! I love it”
  - “data science is awesome. I LOVE IT!”
- Solution:

```
def normalize(text):
    return text.lower()

def tokenize(text):
    alphabetic_only = ''.join(c for c in text
                              if c == ' ' or c.isalpha())
    return alphabetic_only.split()

>>> normalize(“Data Science is AWESOME!!! I love it”)
“data science is awesome!!! i love it”
>>> tokenize(normalize(“Data Science is AWESOME!!! I love it”))
[“data”,”science”,”is”,”awesome”,”i”,”love”,”it”]
```

# STOPWORDS

- Some words don't contain very much information about a document.

```
Living in LA is great. The weather is better than Boston.
```

- How can we remove these words from a text?

```
from nltk.corpus import stopwords
stopwords = stopwords.words('english')

def remove_stopwords(text):
    return [word for word in tokenize(normalize(text))
            if word not in stopwords]


>>> remove_stopwords("Living in LA is great. The weather is better than
Boston.")
["living","la","great","weather","better","boston"]
```

# STEMMING

- Which of the following sentences is different from the others?
  - I like to teach students.
  - I like teaching students.
  - I like eating students.
- Problem: computers don't know that "teach", "teacher", "taught", and "teaching" are similar contents.

# STEMMING

- Solution:

```
from nltk.stem.porter import PorterStemmer

stemmer = PorterStemmer()
def stemmed(text):
    return [stemmer.stem(word) for word in remove_stopwords(text)]

>>> stemmed('I like to teach students’)
['like', 'teach', 'student']
>>> stemmed(‘I like teaching students’)
['like', 'teach', 'student']
>>> stemmed('I like eating students’)
['like', 'eat', 'student']
```

# NGRAMS

- Consider this sentence:

```
This is the steering committee meeting of the New Hampshire Green Party.
```

- Does this accurately represent its meaning?

```
["hampshire", "committee", "new", "this", "party", "green", …]
```

- Better:

```
["steering committee", "new hampshire", "green party", …]
```

# NGRAMS

- Solution:

```
>>> from textblob import TextBlob
>>> tb = TextBlob('This is the steering committee meeting of the New
Hampshire Green Party.').lower()
>>> ngrams = tb.ngrams(n=2)
>>> for n in ngrams:
...     print ' '.join(n)
...
this is
is the
the steering
steering committee
committee meeting
meeting of
of the
the new
new hampshire
hampshire green
green party
```

# SENTIMENT ANALYSIS

```
from textblob import TextBlob

>>> happy = TextBlob('I love GA! Learning is so much fun!')
>>> happy.sentiment
(0.5, 0.4)
>>> sad = TextBlob('I hate GA. My teachers are so boring.')
>>> sad.sentiment
(-0.9, 0.95)
```

# WORDS -> VECTORS

- Problem: Classify tweets into 10 categories given a training set using Naïve Bayes.
  - But scipy.naive_bayes only works with numerical data!
  - Solution: **turn text documents into lists of numbers (vectors)**

# WORDS -> VECTORS

## Input

```
That movie was awesome. I felt like watching an action movie.
```

## Dictionary

| | |
|---|---|
| that | 0 |
| movie | 1 |
| was | 2 |
| … | |
| action | 9 |

## Output

| | |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 1 |
| … | |
| 9 | 1 |

# WORDS -> VECTORS

- Implementation

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(min_df=1)
corpus = ['This is the first document.',
          'This is the second second document.',
          'And the third one.',
          'Is this the first document?']

X = vectorizer.fit_transform(corpus)

>>> vectorizer.get_feature_names()
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third',
'this']
>>> X.toarray()
array([[0, 1, 1, 1, 0, 0, 1, 0, 1],
       [0, 1, 0, 1, 0, 2, 1, 0, 1],
       [1, 0, 0, 0, 1, 0, 1, 1, 0],
       [0, 1, 1, 1, 0, 0, 1, 0, 1]]...)
```