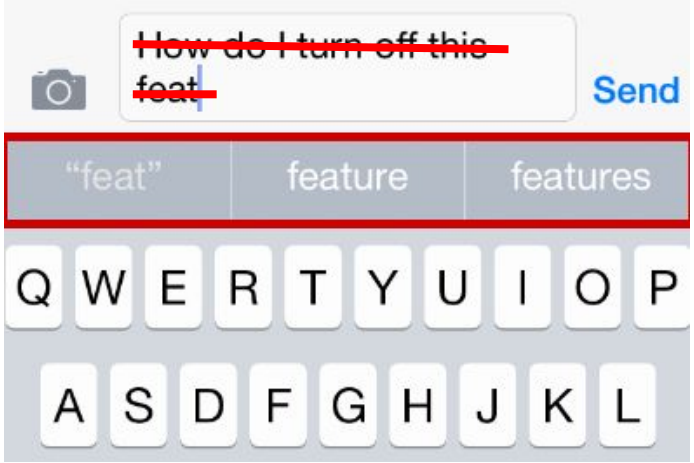
Natural Language Processing (NLP)

Joshua Hans
Jack Padalino

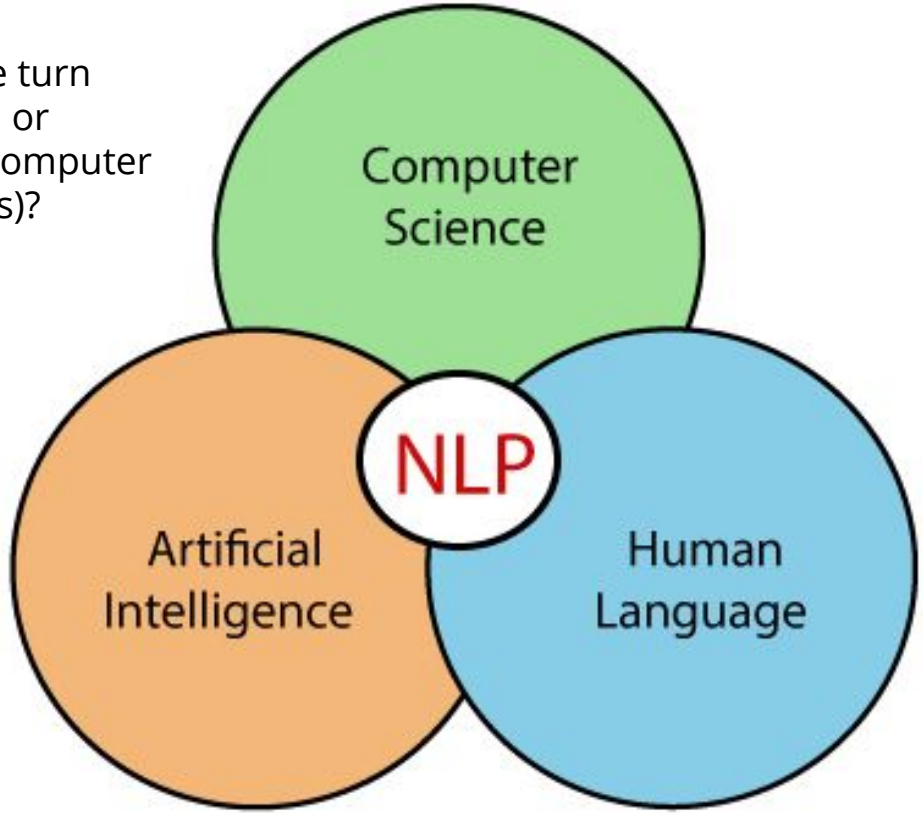
Take Out Your Phone



- Have predictive text enabled.
- Start a text message (private) or message in our Slack workspace if you dare.
- Introduce yourself: Start the message off with My name is...and let predictive text do the rest.
- Simply tap on the center word that your phone generates.

What is NLP?

Big question: How do we turn human language (spoken or written) into data that a computer can understand (numbers)?



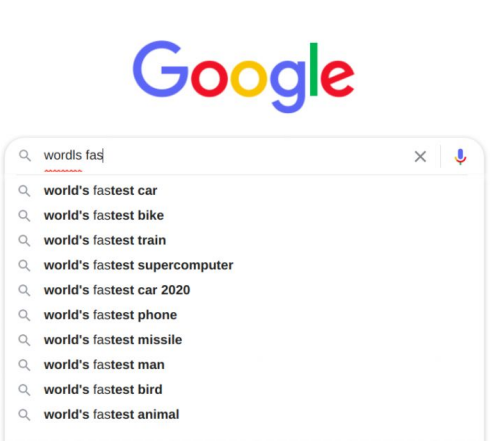
Where is this “NLP” used?

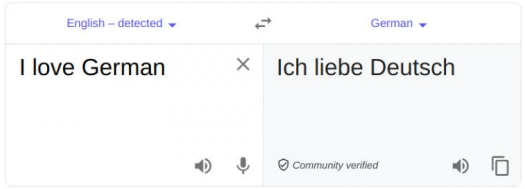
- Everywhere


That's all Folks!

Where is this “NLP” used?

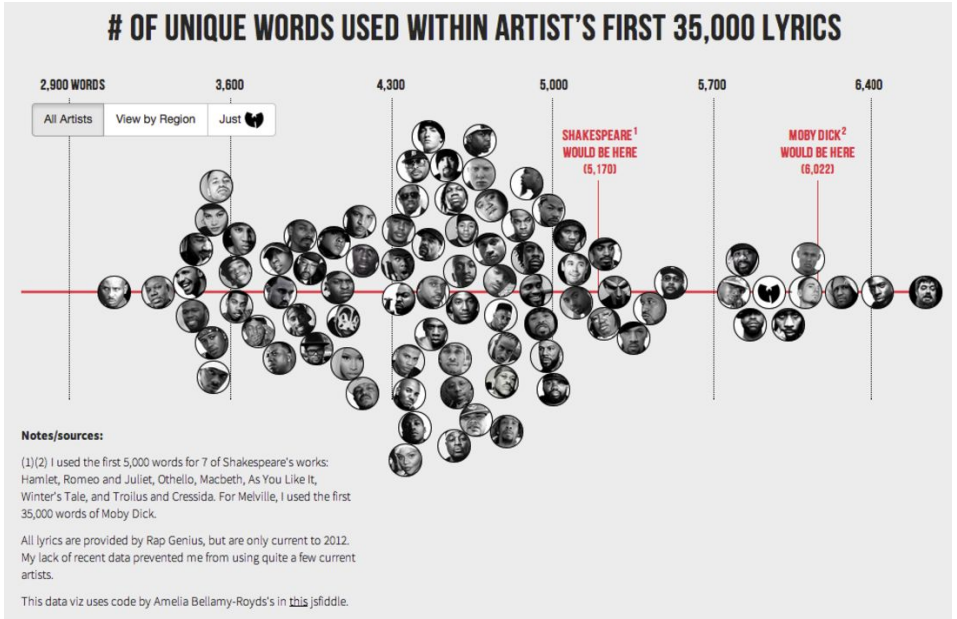
- Spell & grammar check
- Auto-complete
- Speech to text
- Language translation
- Amazon Alexa & Siri
- Email filtering
- Predictive analytics
- Hiring and recruitment





Motivating Exemplars

[Make Trump Tweet Again]



[Largest Vocabulary in Hip Hop]

The big, brown dog is waiting for food.

remove punctuation, make everything lower case

the big brown dog is waiting for food

convert words into root form or root meaning

the big brown dog be waiting for food

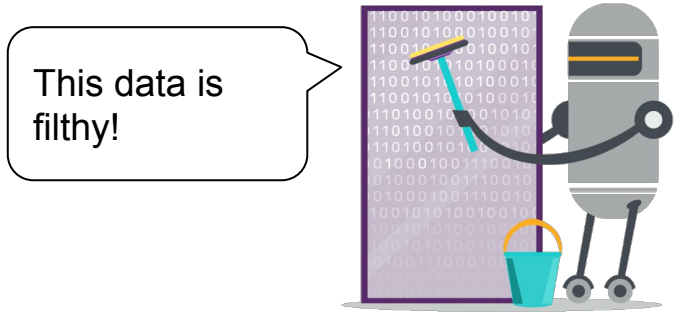
anotha one

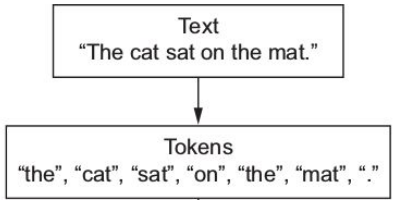
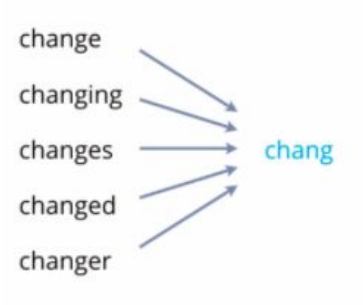
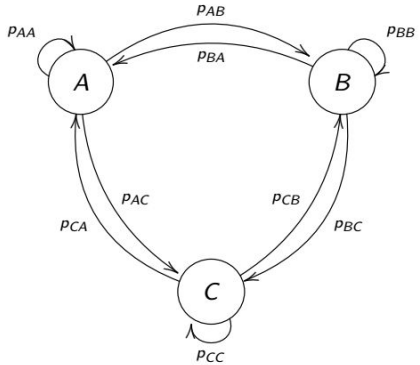
the big brown dog be wait for food

arrive at a final, simplified sentence

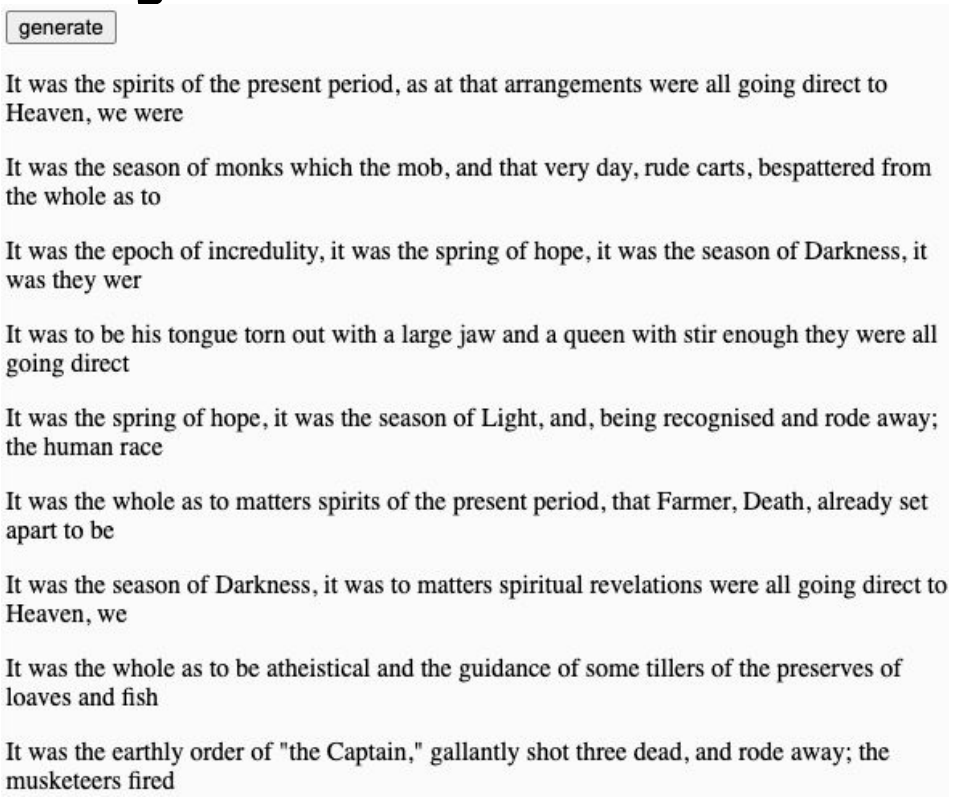
big brown dog be wait for food

Vocabulary / Key Steps



Tokenization	Stemming	Lemmatization	Markov Chain (Link 1 | Link 2)
			

Sample Code: N-gram Generator



Given the first chapter of A Tale of Two Cities by Charles Dickens, instances of randomly generated statements in the style of Dickens are shown with the starting stem, “It was the…”

Code & Important Snip

Subgoal: Create object/dictionary/HashMap of all n-grams generated from the text with array values that store all of the characters that follow.

Loop through the text, mindful of where the last n-gram will be generated.

Grab the next n-gram from the text

Update the data stored in the object:

- If the n-gram that was just created is not already an existing entry, create it.
- Then, in the array associated with that n-gram, add the next character that appears in the text (create the sample space of all possible next characters).

```
14  //loop through the length of the txt to make n-grams
15  for (var i = 0; i <= txt.length - order; i++) {
16
17    //generate n-gram substring
18    var gram = txt.substring(i , i + order);
19
20    //generate dictionary of arrays that will hold n-gram
21    //and array of the single character(s) that follows that n-gram
22
23    //if the n-gram wasn't already recorded, add it to dictionary
24    if (!ngrams[gram]) {
25      ngrams[gram] = [];
26    }//end if-statement
27
28    //add following character after n-gram to array
29    ngrams[gram].push(txt.charAt(i + order));
30
31  }//end for loop
```

Another Important Snip

Subgoal: Generate an output statement based on the data compiled, adding-on a randomly chosen character to follow the current n-gram.

Create the starting point of the output statement; in this script, the result begins with the first n-gram that exists in the supplied text.

This loop does all the magic:

- In this script, 100 characters will be added onto the original n-gram.
- The from the object created in the previous slide, its associated array is called and a random element it chosen to be the next character added onto the result string.

The length of the result string is updated so that we "shift" to the next n-gram in the result string so that when we loop around, the next character added on is based on the correct n-gram.

```
40 function markovIt() {
41   //starting string (the first n-gram)
42   var currentGram = txt.substring(0, order);
43
44   //copy currentGram to our result (final output)
45   var result = currentGram;
46
47   //loop to create a certain sized output statement
48   for(var i = 0; i < 100; i++){
49
50     //grab the array of possible next chararcters
51     //assosicated wit the current n-gram
52     var possibilities = ngrams[currentGram];
53
54     //randomly choose one of the characters to be
55     //added on the our output statement
56     var next = random(possibilities);
57     result += next;
58
59     var len = result.length;
60
61     //the next n-gram is the last "order" characters we currently have
62     currentGram = result.substring(len - order, len);
63   }//end for loop
```

Application: Text Summary Generator

https://www.nytimes.com/2020/07/24/sports/football/nfl-players-regular-season-start.html
'season', 'teams'

Earlier this week, the league said that fans would be required to wear masks at games and both the Giants and Jets be
came the first N.F.L. franchises to say that they would play regular-season games at MetLife Stadium without spectato
rs, heeding New Jersey's prohibitions on mass gatherings. But the cap will have a minimum of $175 million next season
. and the N.F.L.

https://en.wikipedia.org/wiki/Artificial_intelligence
'teams', 'computer', 'style'

[15] The traditional problems (or goals) of AI research include reasoning, knowledge representation, planning, learni
ng, natural language processing, perception and the ability to move and manipulate objects. [33] Attendees Allen Newe
ll (CMU), Herbert Simon (CMU), John McCarthy (MIT), Marvin Minsky (MIT) and Arthur Samuel (IBM) became the founders a
nd leaders of AI research. The AI field draws upon computer science, information engineering, mathematics, psychology
, linguistics, philosophy, and many other fields. [25] Currently, 50+ countries are researching battlefield robots, i
ncluding the United States, China, Russia, and the United Kingdom. [e][198][199] Default logics, non-monotonic logics
and circumscription[103] are forms of logic designed to help with default reasoning and the qualification problem. [2
6][14] Thought-capable artificial beings appeared as storytelling devices in antiquity,[27] and have been common in f
iction, as in Mary Shelley's Frankenstein or Karel Čapek's R.U.R. [109] These tools include models such as Markov dec
ision processes,[210] dynamic decision networks,[207] game theory and mechanism design. [8][50] In the 2017 Future of
Go Summit, AlphaGo won a three-game match with Ke Jie,[51] who at the time continuously held the world No.)[79][80][
81] This lack of "common knowledge" means that AI often makes different mistakes than humans make, in ways that can s
eem incomprehensible. This issue, now known as "robot rights", is currently being considered by, for example, Califor
nia's Institute for the Future, although many critics believe that the discussion is premature.

Given a list of URLs and a list of search words, a brief summary of each article is generated.

Sample Code: Text Summary Generator

Subgoal: Scrape a webpage for text, prep and clean the data

Scrape that data! Using the Python library **Beautiful Soup** the text from each article is stored in each object's text string. Things like images and ads are filtered out.

Clean that data! Using Python's **re** module, specified characters like brackets, extra spaces, and special characters are removed from each object's text string.

```
def web_scrape(self):
    scraped_data = urllib.request.urlopen(self.name)
    article = scraped_data.read()
    parsed_article = bs.BeautifulSoup(article,'lxml')
    paragraphs = parsed_article.find_all('p')
    text_string = ""
    for i in paragraphs:
        text_string += i.text
    self.raw_text_string = text_string

def clean_text_string(self):
    # Removing Square Brackets and Extra Spaces
    cleaned_text_string = re.sub(r'\[[0-9]*\]', ' ', self.raw_text_string)
    cleaned_text_string = re.sub(r'\s+', ' ', self.raw_text_string)
    self.cleaned_text_string = cleaned_text_string

def format_cleaned_text_string(self):
    # Removing special characters and digits
    formatted_text_string = re.sub('[^a-zA-Z]', ' ', self.raw_text_string)
    formatted_text_string = re.sub(r'\s+', ' ', self.raw_text_string)
    self.formatted_cleaned_text_string = formatted_text_string
```

Sample Code: Text Summary Generator

Subgoal: Tokenize words and sentences, count word frequency, and find most important sentences

Tokenize! We import the modules **word_tokenize** and **sent_tokenize** from the **NLTK** library to break the text up into lists of word and sentence tokens..

Count em up! Create a dictionary where each key,value pair is a word with its frequency in the text. This gives each word a "score" and is used to determine which words are most important in this text.

Keep score! Here's the real magic. Since we've tokenized each sentence, and we have a "score" for each word (its frequency), we are able to give each sentence a score by adding up all the word frequencies in that sentence. This helps us determine what are the most important sentences in the text.

```
def tokenize_words(self):
    tokenized_word_list = word_tokenize(self.formatted_cleaned_text_string)
    self.tokenized_word_list = tokenized_word_list

def tokenize_sentences(self):
    tokenized_sentence_list = sent_tokenize(self.cleaned_text_string)
    self.tokenized_sentence_list = tokenized_sentence_list

def word_freq(self):
    for word in self.tokenized_word_list:
        if word not in stopwords:
            if word not in self.word_freq_dict.keys():
                self.word_freq_dict[word] = 1
            else:
                self.word_freq_dict[word] += 1
    maximum_frequency = max(self.word_freq_dict.values())
    for word in self.word_freq_dict.keys():
        self.word_freq_dict[word] = (self.word_freq_dict[word]/maximum_frequency)

def sentence_score(self): #come back to this, may need to add a second parameter?
    for sent in self.tokenized_sentence_list:
        for word in nltk.word_tokenize(sent.lower()):
            if word in self.word_freq_dict.keys():
                if len(sent.split(' ')) < 30:
                    if sent not in self.sentence_score_dict.keys():
                        self.sentence_score_dict[sent] = self.word_freq_dict[word]
                    else:
                        self.sentence_score_dict[sent] += self.word_freq_dict[word]
```

Sample Code: Text Summary Generator

Subgoal: Produce a summary string made of the highest scoring sentences

Summarize! Using Python's **heapq** module a string of the 10 most important sentences is created.

```
def generate_summary(self):
    summary = heapq.nlargest(10, self.sentence_score_dict, key=self.sentence_score_dict.get)
    self.summary = ' '.join(summary)
```

Sample Summaries: Text Summary Generator

Some are better than others...

https://en.wikipedia.org/wiki/Artificial_intelligence
'intelligence', 'artificial', 'computer'

[15] The traditional problems (or goals) of AI research include reasoning, knowledge representation, planning, learni
ng, natural language processing, perception and the ability to move and manipulate objects. [33] Attendees Allen Newe
ll (CMU), Herbert Simon (CMU), John McCarthy (MIT), Marvin Minsky (MIT) and Arthur Samuel (IBM) became the founders a
nd leaders of AI research. The AI field draws upon computer science, information engineering, mathematics, psychology
, linguistics, philosophy, and many other fields. [25] Currently, 50+ countries are researching battlefield robots, i
ncluding the United States, China, Russia, and the United Kingdom. [e][198][199] Default logics, non-monotonic logics
and circumscription[103] are forms of logic designed to help with default reasoning and the qualification problem. [2
6][14] Thought-capable artificial beings appeared as storytelling devices in antiquity,[27] and have been common in f
iction, as in Mary Shelley's Frankenstein or Karel Čapek's R.U.R. [109] These tools include models such as Markov dec
ision processes,[210] dynamic decision networks,[207] game theory and mechanism design. [8][50] In the 2017 Future of
Go Summit, AlphaGo won a three-game match with Ke Jie,[51] who at the time continuously held the world No.)[79][80][
81] This lack of "common knowledge" means that AI often makes different mistakes than humans make, in ways that can s
eem incomprehensible. This issue, now known as "robot rights", is currently being considered by, for example, Califor
nia's Institute for the Future, although many critics believe that the discussion is premature.

https://nypost.com/2020/06/16/a-green-glow-has-appeared-around-mars/
'mars'

It’s a dry, dusty, windy place, but that wasn’t always the case. The planet was once very, very wet, with vast rivers
and lakes dotting its surface. Mars doesn’t have a strong magnetic field like Earth does, but it does have oxygen in
its atmosphere. The northern lights (or southern lights, depending on which end of the globe you’re on) are a well-kn
own and often-studied phenomenon. NASA and other space agencies have already made it clear that they plan on sending
humans to Mars in the not-so-distant future, so this information could come in handy. 386,340 This story has been sha
red 149,898 times. 149,898 This story has been shared 99,247 times. Thanks for contacting us. We've received your sub
mission. Not Now Yes Please

Sample Code: Word Cloud Generator

Given the full text of Alice's Adventures in Wonderland by Lewis Carroll,
a Word Cloud is generated showing the most common words used by the author.

Important Snips

Subgoal: Read the text file and begin the clean-up process by making the text fully lowercase, removing punctuation, etc...

Open the text file and create a usable version of the data.

```
alice = nltk.corpus.gutenberg.words('carroll-alice.txt')

aliceString = ""
for word in alice:
    aliceString += word
    aliceString += " "
```

The clean-up process begins.

Be sure to look at your data to see if you need to clean it up even more after this first round.

```
def cleanTextR1(text):
    '''Make text lowercase, remove punctuation and remove words containing numbers.'''
    text = text.lower()
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

round1 = cleanTextR1(aliceString)
```

Important Snips (cont.)

Subgoal: Continue the clean-up process by removing “stop words” and then catalog the frequencies of each word in the corpus in a dictionary.

A second round of clean-up, where “stop words” are removed from the data.
There are 127 English stop words in the NLTK library, such as “i”, “do”, “the”, “their”, etc…

```
stopWords = set(stopwords.words('english'))

round1Tokens = word_tokenize(round1)

filteredAlice = []

for w in round1Tokens:
    if w not in stopWords:
        filteredAlice.append(w)

#the more pythonic way of stating the above
filteredAlice = [w for w in round1Tokens if not w in stopWords]
```

Similar structure to the n-gram generator, but in this case, we build the data structure to store word frequency.

```
wordDictionary = {}

for i in range(len(filteredAlice)):
  if filteredAlice[i] in wordDictionary:
    wordDictionary[filteredAlice[i]] += 1
  else:
    wordDictionary[filteredAlice[i]] = 1
```

Important Snips (cont.)

Subgoal: Generate and display the WordCloud based on the accumulated data.

```
wc = WordCloud(background_color="white", colormap="Dark2",
               max_font_size=150, random_state=42).generate_from_frequencies(wordDictionary)

plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Creating a Word Cloud is pretty simple using after importing helpful tools.
Above the World Cloud library is used to generate and Matplotlib to display the figure.

The possibilities are endless; push your creativity!

Student Sample

Link

Load the data

Clean-up/prep the data

Build a dictionary to store frequencies

Display the data

```
1   import pprint
2   speech = open('speech.txt', 'r')
3   ignore = open('ignore.txt','r')
4
5   ignore = ignore.readlines()
6
7   speech = speech.read()
8   speech = speech.lower()
9
10  for i in range (len(ignore)):
11      ignore[i] = ignore[i].rstrip('\n')
12      speech = speech.replace(ignore[i],'')
13
14  words = speech.split()
15  wordDictionary = {}
16
17  for i in range(len(words)):
18      if words[i] in wordDictionary:
19          wordDictionary[words[i]] += 1
20      else:
21          wordDictionary[words[i]] = 1
22
23  pprint.pprint(wordDictionary)
```

Potential Rollout

This topic offers an engaging opportunity to summarize the uses of various data structures that may be taught in an intro course or even AP CS A.

As a summative project, students become data scientists:

- First, they devise an investigative question of their interest.
  - How is the Genie from Aladdin different when played by Robin Williams vs Will Smith?
  - What trends exist in a given dataset such as NYPD discipline records?
  - What should you study before being a contestant on Jeopardy!?
- Using subgoal labeling, students structure their code, justifying uses of specific data structures (lists/arrays, dictionary/HashMap, etc...)
- Students use their code to make conclusions, and with the aid of a visual, answer their original question.
- *Alternative:* Students produce a “Speak Like _____” generator where they feed quotes/lyrics/text from stories and use n-grams to create output in the style of their choice.

Work Leading Up to the Summative

- Dictionaries/HashMaps
  - Language/texting abbreviation translator:
    - Students preload some entries to their data structure,
      eg: `{'lol' : 'laughing out loud'}, {'uno' : 'one'}`
    - Prompt the user for a term, and if the requested abbreviation is in the program's dictionary, then it prints out the definition.
      If the abbreviation is not in the dictionary, the program prints an apologetic message saying that it could not find a definition. Furthermore, the data structure can be updated to include the new term defined by the user.

    Warning: *When working with Python, since the structure is called a dictionary and this activity is basically simulating a dictionary, students may develop the misconception that this is the lone functionality of the data structure.*

  - Word Counter:
    - Low-stakes assignment to expose students to the idea of cleaning up and tokenizing data.
      - Given a sample statement, such as, *"Baseball is a fun sport. The sport is not new, it was invented in 1839. There are two professional baseball teams in New York."*
      - Have students find how many times words such as "baseball", "sport", and "new" exist in the text.
      - Students can then investigate a text of their interest.

Work Leading Up to the Summative (cont.)

- Dictionaries/HashMaps
  - To-Do List:
    - Students get exposed to the idea that the data structure does not necessarily have to have key-value pairs that are primitives.
    - A structure such as `{'Sunday':['family time','cook','finalize lesson plans']}`
    - Students get experience searching and accessing data:
      - Find all days where __(some task)__ is required.
      - Print all of the tasks on __(some day)__.
      - Clear my schedule on __(some day)__.
      - Add __(some task)__ to __(some day)__.

  - Top Words:
    - Low-stakes assignment to expose students to the idea of stop words.
    - For example, in *Green Eggs and Ham*, the top 5 words are: *i, 22; like, 17; not, 13; do, 11; them, 12*
      - Plant the seed of multiple rounds of cleaning up the data before it can be used to answer the investigative question.

  - CollegeBoard published a lab for the AP CS A curriculum focused on Chatbots, another application of NLP. Student Guide, CollegeBoard Lab Resource Page, CS Awesome Runestone

Useful Python Libraries, Packages, and Modules

- **Natural Language Toolkit** (NLTK) - Python platform with several useful libraries and packages for projects in NLP and data science like tokenizers, stemmers, lemmatizers, data cleaning, grammar and spell checks
  - **Snowball Stemmer -** stemmer tool
  - **Word Tokenize**- create a list of tokenized words
  - **Sentence Tokenize** - creates a list of tokenized sentences
  - **Stopwords** - scans text for most common words like that “a”, “the”, “and”

- **Beautiful Soup** (BS) - Python library for web scraping (pulling text from a website)

**Will need to install NLTK and BS before any modules or packages can be used!

Resources Used

- N-Grams and Markov Chains, Coding Challenge #42 - Daniel Shiffman (videos)
- Natural Language Processing in Python - Alice Zhao (video)
- Natural Language Processing in Python - Alice Zhao (Github repo to go along with the above video)
- Natural Language Processing Tutorial with Python & NLTK - freeCodeCamp (video)
- How to use NLP in Python: a Practical Step-by-Step Example - Lianne & Justin (article)
- Matplotlib - Python Library for Visualizations
- NLTK - Natural Language Toolkit

Other Resources

- Natural Language Processing with Python - Steven Bird, Ewan Klein, Edward Loper (textbook)
- Sentiment Analysis Using Machine Learning and Python - randerson112358 (article)
- Extract text from a website using Python and Beautiful Soup
- Web scraping with Python
- Creating a text summarizer tutorial
-