# Installation of Software

* Which should you use?
  + R/RStudio: mostly for users who are good with R and want to keep using RStudio because you've been using it a lot for other classes
  + start a rmarkdown assignment and make the assignment o Anaconda/Jupyter/Spyder: you are already use python in this system a lot
  + Start a jupyter notebook and make the assignment o Datalore: is for people who are newer at python OR bad at installing stuff on their computer OR have an older computer OR have spaces in your username OR you want to be able to work together with your team mates or me
  + start a new notebook and make the assignment
* Notes:
* o You only need keras and tensorflow if you want to try deep learning in the classification section

# Pycharm

* Install!
* Click learn
* New course
* In marketplace > introduction to python
* Take a screen when you are done with the sections required (not all due at once)
* Be sure you can see the TERMINAL at the bottom in that screen shot

# Processing Text

* On class assignments, we all turn in the the same code and data source.

## Libraries

# import just a function   
from urllib.request import urlopen  
from bs4 import BeautifulSoup  
  
# import a whole library as a new name  
import pandas as pd  
  
# import package as its own name  
import nltk  
  
nltk.download('punkt')  
  
# other packages  
import re  
  
  
# rest of stuff  
import textacy.preprocessing as tprep  
  
# install pyspellchecker !!!  
from spellchecker import SpellChecker  
  
import spacy  
  
import textacy  
  
from itertools import chain  
from collections import Counter  
  
  
  
# impurity function  
RE\_SUSPICIOUS = re.compile(r'[&#<>{}\[\]\\]')  
  
def impurity(text, min\_len=10):  
 """returns the share of suspicious characters in a text"""  
 if text == None or len(text) < min\_len:  
 return 0  
 else:  
 return len(RE\_SUSPICIOUS.findall(text))/len(text)  
  
  
  
def normalize(text):  
 text = tprep.normalize.hyphenated\_words(text)  
 text = tprep.normalize.quotation\_marks(text)  
 text = tprep.normalize.unicode(text)  
 text = tprep.remove.accents(text)  
 text = tprep.replace.phone\_numbers(text)  
 text = tprep.replace.urls(text)  
 text = tprep.replace.emails(text)  
 text = tprep.replace.user\_handles(text)  
 text = tprep.replace.emojis(text)  
 return text  
   
  
spell = SpellChecker()  
  
nlp = spacy.load("en\_core\_web\_sm")

[nltk\_data] Downloading package punkt to  
[nltk\_data] C:\Users\user7\AppData\Roaming\nltk\_data...  
[nltk\_data] Package punkt is already up-to-date!

## Find Text

* As a class, we will find a text source to analyze. This text source usually will consist of a webpage or other dataset to examine and clean.
* Import the text into your report.
* If the text is one big long string, first break into sentence segments and store it in a Pandas DataFrame.

myurl = "https://www.foxnews.com/sports/patrick-mahomes-fiery-message-win-bills-they-got-what-they-asked-for"  
#myurl = "https://www.foxnews.com/lifestyle/newly-elected-school-board-pennsylvania-reclaims-indigenous-mascot-rejects-cancel-culture"  
  
html = urlopen(myurl).read()  
  
soupified = BeautifulSoup(html, "html.parser")  
# soupified  
  
# just try get\_text()  
try\_text = soupified.get\_text()  
try\_text[0:100]

"\n\n\nPatrick Mahomes' fiery message after win over Bills: 'They got what they asked for' | Fox News\n\n\n"

* Regular expressions

# find an exact match for the first time this occurs  
text = try\_text[  
 # everything from the end of this sentence and on  
 re.search("To access the content, check your email and follow the instructions provided.", try\_text).end():  
 # now the end  
 re.search("CLICK HERE TO GET THE FOX NEWS APP", try\_text).start()  
]

* • Breaking down into sentences

# break down into sentences and put into DF  
sentences = nltk.sent\_tokenize(text)  
type(sentences)  
  
# convert to dataframe  
DF = pd.DataFrame(sentences, columns = ["sentence"])  
DF.head()

sentence  
0 \n Having trouble?  
1 Click here.  
2 Kansas City Chiefs quarterback Patrick Mahomes...  
3 Mahomes, surrounded by Travis Kelce, Chris Jon...  
4 They have the Baltimore Ravens up next.

* We've used:
* o One big string (one variable)
* o A list which uses []
* o Dictionaries {}
* o Tuples ()
* o DataFrame from pandas

## Length for Proposal

# do this on the full text not broken into sentences  
len(nltk.word\_tokenize(text))  
# be sure to import nltk in the proposal

554

## Fix Errors

* Examine the text for errors or problems by looking at the text.
* o Legit, just look at the text.
* o Looking for any type of "garbage" - dependent on what you are doing.
* Use the “impurity” function from class to examine the text for potential issues.

DF['score'] = DF['sentence'].apply(impurity)  
DF

sentence score  
0 \n Having trouble? 0.000000  
1 Click here. 0.000000  
2 Kansas City Chiefs quarterback Patrick Mahomes... 0.000000  
3 Mahomes, surrounded by Travis Kelce, Chris Jon... 0.000000  
4 They have the Baltimore Ravens up next. 0.000000  
5 "Hey, they asked for it, and they got what the... 0.000000  
6 "But he (Andy Reid) said it. 0.000000  
7 This s--- ain’t done. 0.000000  
8 We come back next week ready to f---ing go." 0.000000  
9 CLICK HERE FOR MORE SPORTS COVERAGE ON FOXNEWS... 0.000000  
10 VIEW THE MOMENT ON X.Reid said it’s not time f... 0.000000  
11 They still have to get through the Ravens and ... 0.000000  
12 The Ravens are going to be the Chiefs’ toughes... 0.000000  
13 Kansas City will have to do it on the road. 0.000000  
14 The Ravens are hosting a conference championsh... 0.002222  
15 Baltimore boasts one of the toughest defenses ... 0.000000  
16 "There’s no weakness there," the star quarterb... 0.000000  
17 "It’s going to take our best effort. 0.000000  
18 Defense, offense, special teams. 0.000000  
19 They do it all. 0.000000  
20 It’s always a great challenge, and that stadiu... 0.000000  
21 So, we’re excited for the challenge. 0.000000  
22 "On Monday, Mahomes added that he praised the ... 0.000000  
23 "For three quarters offensively, we were movin... 0.000000  
24 "I went over to the defense and told them: ‘Y’... 0.000000  
25 We’ll get to the AFC championship game." 0.000000  
26 Kansas City Chiefs quarterback Patrick Mahomes... 0.000000

* Remove the noise with the regex function.
* Re-examine the impurity to determine if the data has been mostly cleaned.
* o Not necessary because it looks fine.
* Normalize the rest of the text by using textacy.

DF['clean'] = DF['sentence'].apply(normalize)  
DF

sentence score \  
0 \n Having trouble? 0.000000   
1 Click here. 0.000000   
2 Kansas City Chiefs quarterback Patrick Mahomes... 0.000000   
3 Mahomes, surrounded by Travis Kelce, Chris Jon... 0.000000   
4 They have the Baltimore Ravens up next. 0.000000   
5 "Hey, they asked for it, and they got what the... 0.000000   
6 "But he (Andy Reid) said it. 0.000000   
7 This s--- ain’t done. 0.000000   
8 We come back next week ready to f---ing go." 0.000000   
9 CLICK HERE FOR MORE SPORTS COVERAGE ON FOXNEWS... 0.000000   
10 VIEW THE MOMENT ON X.Reid said it’s not time f... 0.000000   
11 They still have to get through the Ravens and ... 0.000000   
12 The Ravens are going to be the Chiefs’ toughes... 0.000000   
13 Kansas City will have to do it on the road. 0.000000   
14 The Ravens are hosting a conference championsh... 0.002222   
15 Baltimore boasts one of the toughest defenses ... 0.000000   
16 "There’s no weakness there," the star quarterb... 0.000000   
17 "It’s going to take our best effort. 0.000000   
18 Defense, offense, special teams. 0.000000   
19 They do it all. 0.000000   
20 It’s always a great challenge, and that stadiu... 0.000000   
21 So, we’re excited for the challenge. 0.000000   
22 "On Monday, Mahomes added that he praised the ... 0.000000   
23 "For three quarters offensively, we were movin... 0.000000   
24 "I went over to the defense and told them: ‘Y’... 0.000000   
25 We’ll get to the AFC championship game." 0.000000   
26 Kansas City Chiefs quarterback Patrick Mahomes... 0.000000   
  
 clean   
0 \n Having trouble?   
1 Click here.   
2 Kansas City Chiefs quarterback Patrick Mahomes...   
3 Mahomes, surrounded by Travis Kelce, Chris Jon...   
4 They have the Baltimore Ravens up next.   
5 "Hey, they asked for it, and they got what the...   
6 "But he (Andy Reid) said it.   
7 This s--- ain't done.   
8 We come back next week ready to f---ing go."   
9 CLICK HERE FOR MORE SPORTS COVERAGE ON FOXNEWS...   
10 VIEW THE MOMENT ON X.Reid said it's not time f...   
11 They still have to get through the Ravens and ...   
12 The Ravens are going to be the Chiefs' toughes...   
13 Kansas City will have to do it on the road.   
14 The Ravens are hosting a conference championsh...   
15 Baltimore boasts one of the toughest defenses ...   
16 "There's no weakness there," the star quarterb...   
17 "It's going to take our best effort.   
18 Defense, offense, special teams.   
19 They do it all.   
20 It's always a great challenge, and that stadiu...   
21 So, we're excited for the challenge.   
22 "On Monday, Mahomes added that he praised the ...   
23 "For three quarters offensively, we were movin...   
24 "I went over to the defense and told them: 'Y'...   
25 We'll get to the AFC championship game."   
26 Kansas City Chiefs quarterback Patrick Mahomes...

* Examine spelling errors in at least one row of the dataset.
* o Any time you have stuff with names, please do not do spelling.
* o Mostly, only do this if you have a specific goals.

# find all the unique tokens  
# set is find unique  
# nltk.word\_tokenize is break down into words  
# " ".join is combine into one long text  
# .to\_list() is a function to convert to list   
clean\_tokens = set(nltk.word\_tokenize(" ".join(DF['clean'].to\_list())))  
  
# what is wrong?   
misspelled = spell.unknown(clean\_tokens)  
  
for word in misspelled:  
 # what's the word  
 print(word)  
 print("\n")  
 # Get the one `most likely` answer  
 print(spell.correction(word))  
  
 # Get a list of `likely` options  
 print(spell.candidates(word))  
   
# make a dictionary of the misspelled word and the correction  
# use find and replace in re to fix them

ii  
  
  
i  
{'in', 'if', 'is', 'id', 'i', 'ai', 'bi', 'hi', 'it', 'ti', 'qi', 'si', 'ki', 'oi', 'pi', 'li', 'mi', 'xi'}  
baltimore  
  
  
None  
None  
photo/frank  
  
  
None  
None  
photo/adrian  
  
  
None  
None  
lamar  
  
  
lama  
{'lazar', 'lama', 'damar', 'lamer', 'lamas'}  
jim  
  
  
him  
{'jib', 'him', 'jam', 'rim', 'jig', 'nim', 'vim', 'mim', 'aim', 'dim'}  
's  
  
  
is  
{'is', 'vs', 'cs', 'ts', 'ls', 'ps', 's', 'ss', 'gs', 'ms', 'es', 'rs', 'ks', 'as', 'us'}  
're  
  
  
are  
{'ore', 'ire', 'ere', 're', 'are'}  
travis  
  
  
traves  
{'traves'}  
monday  
  
  
money  
{'money', 'moray', 'monads', 'moldy', 'moody', 'mayday', 'monody', 'monas', 'modal', 'today', 'monde', 'monad', 'midday', 'fondly', 'monkey', 'noonday'}  
chris  
  
  
chis  
{'chrism', 'chis'}  
kelce  
  
  
else  
{'kelt', 'ketch', 'fence', 'deuce', 'kelter', 'belle', 'kale', 'else', 'veloce', 'kerne', 'kedge', 'kelpie', 'kyle', 'helve', 'belch', 'kelly', 'pence', 'belie', 'dolce', 'hence', 'welch', 'recce', 'deice', 'keel', 'peace', 'delve', 'keck', 'kente', 'kelp', 'terce', 'dele'}  
jan.  
  
  
jane  
{'jane'}  
harbaugh  
  
  
None  
None  
nfl  
  
  
nil  
{'nil'}  
brittany  
  
  
dittany  
{'dittany'}  
jones  
  
  
ones  
{'zones', 'cones', 'jokes', 'nones', 'ones', 'bones', 'tones', 'pones', 'hones', 'sones'}  
``  
  
  
i  
{'is', 'hg', 'mb', 'it', 'h', 'd', 'am', 'ut', 'uh', 'kb', 'ln', 'k', 'la', 'em', 'ka', 'j', 'jo', 'if', 'ha', 'ye', 'vs', 'ls', 'or', 'w', 'xu', 'at', 'id', 'dm', 'v', 'we', 'mi', 'go', 'p', 'q', 't', 'ps', 'de', 'fa', 'a', 'ow', 're', 'as', 'i', 's', 'jg', 'ne', 'ks', 'lx', 'um', 'er', 'ai', 'mu', 'cs', 'ah', 'od', 'ti', 'm', 'en', 'lm', 'me', 'be', 'ja', 'el', 'of', 'g', 'wo', 'b', 'dg', 'on', 'us', 'dc', 'to', 'kw', 'bi', 'hi', 'no', 'mf', 'so', 'hm', 'my', 'n', 'yo', 'o', 'ax', 'db', 'ox', 'ad', 'rs', 'eh', 'ta', 'sh', 'pa', 'si', 'y', 'oi', 'pi', 'ms', 'by', 'cw', 'r', 'do', 'ho', 'qi', 'nu', 'es', 'hl', 'ss', 'l', 'x', 'oh', 'ts', 'ex', 'u', 'c', 'kc', 'ya', 'an', 'om', 'lo', 'ki', 'li', 'et', 'z', 'in', 'pe', 'f', 'gs', 'ma', 'up', 'e', 'oo', 'he', 'ay', 'xi'}  
x.reid  
  
  
nereid  
{'nereid'}  
-ing  
  
  
sing  
{'sing', 'ting', 'wing', 'ding', 'zing', 'ping', 'ring', 'king'}  
n't  
  
  
not  
{'net', "an't", 'not', 'nut', 'nit'}  
kansas  
  
  
canvas  
{'tankas', 'pandas', 'cannas', 'kinas', 'canvas', 'manias', 'paisas', 'kwanzas', 'manses', 'anoas', 'vandas', 'kashas', 'ganjas', 'mantas', 'manas', 'kana', 'kantar', 'kavas', 'anas', 'kappas', 'balsas', 'pangas', 'kanzus', 'annas', 'salsas'}  
sunday  
  
  
sundae  
{'sundae', 'sundry'}  
reid  
  
  
read  
{'reis', 'raid', 'redid', 'rend', 'rid', 'reed', 'read', 'rebid', 'redd', 'rein', 'red'}  
ap  
  
  
a  
{'tap', 'am', 'nap', 'a', 'as', 'amp', 'cap', 'rap', 'sap', 'alp', 'ai', 'lap', 'ah', 'ape', 'ax', 'hap', 'an', 'ad', 'map', 'asp', 'zap', 'at', 'apt', 'bap', 'pa', 'up', 'yap', 'gap', 'pap', 'dap', 'ay', 'p'}  
'll  
  
  
all  
{'ell', 'all', 'ill'}  
andy  
  
  
and  
{'handy', 'and', 'any', 'dandy', 'randy', 'candy', 'bandy', 'sandy'}  
--  
  
  
i  
{'is', 'hg', 'mb', 'it', 'h', 'd', 'am', 'ut', 'uh', 'kb', 'ln', 'k', 'la', 'em', 'j', 'ka', 'jo', 'if', 'ha', 'ye', 'vs', 'ls', 'or', 'w', 'xu', 'at', 'dm', 'id', 'v', 'we', 'mi', 'go', 'p', 'q', 't', 'de', 'ps', 'fa', 'a', 'ow', 're', 'as', 'i', 's', 'jg', 'ne', 'ks', 'lx', 'um', 'er', 'ai', 'mu', 'cs', 'ah', 'od', 'ti', 'm', 'en', 'lm', 'me', 'be', 'ja', 'el', 'of', 'g', 'wo', 'b', 'dg', 'on', 'us', 'dc', 'to', 'kw', 'bi', 'hi', 'mf', 'no', 'so', 'hm', 'my', 'n', 'yo', 'o', 'ax', 'db', 'ox', 'ad', 'rs', 'eh', 'ta', 'sh', 'pa', 'y', 'si', 'oi', 'ms', 'pi', 'by', 'cw', 'r', 'do', 'ho', 'qi', 'nu', 'es', 'hl', 'ss', 'l', 'x', 'oh', 'ts', 'ex', 'u', 'c', 'om', 'kc', 'ya', 'an', 'ki', 'lo', 'li', 'et', 'z', 'in', 'pe', 'f', 'gs', 'ma', 'up', 'e', 'oo', 'he', 'ay', 'xi'}  
night.chargers  
  
  
None  
None  
n.y.  
  
  
nay  
{'nay', 'nays'}  
afc  
  
  
aft  
{'aft', 'arc'}  
''  
  
  
i  
{'is', 'hg', 'mb', 'it', 'h', 'd', 'am', 'ut', 'uh', 'kb', 'ln', 'k', 'la', 'em', 'j', 'ka', 'jo', 'if', 'ha', 'ye', 'vs', 'ls', 'or', 'w', 'xu', 'at', 'dm', 'id', 'v', 'we', 'mi', 'go', 'p', 'q', 't', 'de', 'ps', 'fa', 'a', 'ow', 're', 'as', 'i', 's', 'ne', 'jg', 'ks', 'lx', 'er', 'um', 'ai', 'cs', 'mu', 'ah', 'od', 'ti', 'm', 'en', 'lm', 'me', 'be', 'ja', 'el', 'of', 'g', "h'm", 'wo', 'b', 'dg', 'on', 'us', 'dc', 'to', 'kw', 'bi', 'hi', 'no', 'mf', 'so', 'hm', 'my', 'n', 'yo', 'o', 'ax', 'db', 'ox', 'ad', 'rs', 'eh', 'ta', 'sh', 'pa', 'y', 'si', 'oi', 'ms', 'pi', 'by', 'cw', 'r', 'do', 'ho', 'nu', 'qi', 'es', 'hl', 'ss', 'l', 'x', 'oh', 'ts', 'ex', 'u', 'c', 'kc', 'om', 'an', 'ya', 'ki', 'lo', 'li', 'et', 'z', 'in', 'pe', 'f', 'gs', 'ma', 'up', 'e', 'oo', 'he', 'ay', 'xi'}  
mvp  
  
  
map  
{'mop', 'map'}  
kraus  
  
  
krauts  
{'krauts', 'kraut'}  
jackson  
  
  
jacks  
{'jacks', 'jackpot'}  
mahomes  
  
  
homes  
{'homes', 'manholes', 'mahouts', 'radomes'}  
patrick  
  
  
trick  
{'pathic', 'iatric', 'patriot', 'strick', 'trick', 'hayrick', 'prick'}  
foxnews.com  
  
  
None  
None

## Pre-Processing

* Using spacy and textacy, pre-process the text to end up with a list of tokenized lists.

output = []  
  
# only the tagger and lemmatizer  
for doc in nlp.pipe(DF['clean'].tolist(), disable=["tok2vec", "ner", "parser"]):  
 tokens = textacy.extract.words(doc,  
 filter\_stops = True, # default True, no stopwords  
 filter\_punct = True, # default True, no punctuation  
 filter\_nums = True, # default False, no numbers  
 include\_pos = None, # default None = include all  
 exclude\_pos = None, # default None = exclude none  
 min\_freq = 1) # minimum frequency of words  
 output.append([str(word) for word in tokens]) # close output append   
  
output[0:1]

[['Having', 'trouble']]

• Create a frequency table of each of the tokens returned in this output. Below is some example code to get us started.

# all items  
type(output)  
# first list  
type(output[0])  
# first list, first item (this is the issue!)  
type(output[0][0])  
  
Counter(chain.from\_iterable(output))

Counter({'Mahomes': 9,  
 'Chiefs': 6,  
 'game': 6,  
 'Kansas': 5,  
 'City': 5,  
 'quarterback': 5,  
 'Bills': 5,  
 'AFC': 5,  
 'Sunday': 5,  
 'said': 5,  
 'Patrick': 4,  
 'Buffalo': 4,  
 'Ravens': 4,  
 'championship': 3,  
 'got': 3,  
 'divisional': 3,  
 'playoff': 3,  
 'Jan.': 3,  
 'Orchard': 3,  
 'Park': 3,  
 'N.Y.': 3,  
 'AP': 3,  
 'Photo': 3,  
 'going': 3,  
 'Kelce': 2,  
 'Andy': 2,  
 'Reid': 2,  
 'Baltimore': 2,  
 'asked': 2,  
 'week': 2,  
 'Adrian': 2,  
 'Kraus': 2,  
 'added': 2,  
 'time': 2,  
 'win': 2,  
 'toughest': 2,  
 'NFL': 2,  
 'offense': 2,  
 'challenge': 2,  
 'defense': 2,  
 'shut': 2,  
 'quarter': 2,  
 'Having': 1,  
 'trouble': 1,  
 'Click': 1,  
 'delivered': 1,  
 'epic': 1,  
 'pep': 1,  
 'talk': 1,  
 'teammates': 1,  
 'team': 1,  
 'defeated': 1,  
 'advance': 1,  
 'surrounded': 1,  
 'Travis': 1,  
 'Chris': 1,  
 'Jones': 1,  
 'agreed': 1,  
 'coach': 1,  
 'job': 1,  
 'Hey': 1,  
 's---': 1,  
 'ai': 1,  
 'come': 1,  
 'ready': 1,  
 'f': 1,  
 'ing': 1,  
 'CLICK': 1,  
 'SPORTS': 1,  
 'COVERAGE': 1,  
 'FOXNEWS.COM': 1,  
 'reacts': 1,  
 'beating': 1,  
 'case': 1,  
 'know': 1,  
 'VIEW': 1,  
 'MOMENT': 1,  
 'X.Reid': 1,  
 'dancing': 1,  
 'Super': 1,  
 'Bowl': 1,  
 'celebrations': 1,  
 'happen': 1,  
 'matchup': 1,  
 'road': 1,  
 'hosting': 1,  
 'conference': 1,  
 'history': 1,  
 'M&T': 1,  
 'Bank': 1,  
 'Stadium': 1,  
 'night': 1,  
 'CHARGERS': 1,  
 'PURSUIT': 1,  
 'JIM': 1,  
 'HARBAUGH': 1,  
 'ADVANCING': 1,  
 'FINAL': 1,  
 'STAGES': 1,  
 'REPORT': 1,  
 'Brittany': 1,  
 'right': 1,  
 'hugs': 1,  
 'knows': 1,  
 'work': 1,  
 'cut': 1,  
 'boasts': 1,  
 'defenses': 1,  
 'likely': 1,  
 'MVP': 1,  
 'Lamar': 1,  
 'Jackson': 1,  
 'weakness': 1,  
 'star': 1,  
 'best': 1,  
 'effort': 1,  
 'Defense': 1,  
 'special': 1,  
 'teams': 1,  
 'great': 1,  
 'stadium': 1,  
 'rocking': 1,  
 'excited': 1,  
 'Monday': 1,  
 'praised': 1,  
 'coming': 1,  
 'clutch': 1,  
 'quarters': 1,  
 'offensively': 1,  
 'moving': 1,  
 'ball': 1,  
 'field': 1,  
 'went': 1,  
 'told': 1,  
 "Y'all": 1,  
 'football': 1,  
 'rolls': 1,  
 'pocket': 1,  
 'Frank': 1,  
 'Franklin': 1,  
 'II': 1})

## Summarize

Write a paragraph explaining the process of cleaning data for an NLP pipeline. You should explain the errors you found in the dataset and how you fixed them. Explain the information that is gathered by using spacy and textacy and the final output. What did you learn from your frequency table? What is the text document about?

To begin the data cleaning process for our NLP pipeline, we first encountered an error related to the NLTK library, where the punkt tokenizer models had not been downloaded, resulting in a LookupError. This issue was resolved by downloading the necessary punkt package using nltk.download('punkt'), which allowed us to tokenize our text data into sentences successfully.

Digging deeper with tools like spacy and textacy, we got into the nitty-gritty of our text, uncovering the roles of words, spotting key terms and phrases, and mapping out how words connect to each other. This deeper dive into the language mechanics helped us spot and smooth out any rough patches or unclear bits in our text, giving us a clearer, more consistent dataset to work with.

The final output of our cleaning process was a well-structured dataset, ready for more advanced NLP tasks. From the frequency table generated, we gained insights into the most common words and phrases, which helped us understand the underlying themes and topics of our text document.

Overall, the process taught us the importance of thorough data cleaning and preparation in NLP. By addressing errors upfront and utilizing powerful linguistic tools, we were able to enhance the quality of our dataset, paving the way for more accurate and insightful natural language processing applications.