TASK 3.1(Pratham Kitawat)

Text Preprocessing: What It Is and Why We Do It

Let's talk about text preprocessing. It's a crucial step when we're dealing with text data in natural language processing (NLP) and text analysis. Basically, it's all about cleaning up and transforming raw text so we can analyze it better. In this document, we'll go through the different steps involved, explain why each one is important, and show you some code examples.

Steps in Text Preprocessing

1. Lowercasing

What it is: Lowercasing is pretty simple - we're just turning all the letters in our text to lowercase. So "Hello World" becomes "hello world".

Why we do it:

- It keeps things consistent. We don't want our computer to think "The" and "the" are different words.
- It makes our data smaller and easier to handle. Instead of having separate entries for "Cat", "CAT", and "cat", we just have one: "cat".
- It helps when we're trying to match words or search for patterns. We don't have to worry about whether something is capitalized or not.
- Many NLP tools work better when everything's in lowercase.

Here's how we do it in code: First , we are importing all the necessary libraries and performing some basic operations like storing our data in a dataframe and dropping unnecessary columns and then we are converting all our data to lowercase by using apply and .lower() function

```
import numpy as np
import pandas as pd
import string
from nltk.tokenize import word_tokenize
import nltk
import re
```

```
df = pd.read_csv("/content/Twitter Sentiments.csv")
df.head(5)
df = df.drop(columns = ["id","label"],axis = 1)
df.head(5)
#dropping the id and label column as they are not needed
```

	tweet
0	@user when a father is dysfunctional and is s...
1	@user @user thanks for #lyft credit i can't us...
2	bihday your majesty
3	#model i love u take with u all the time in ...
4	factsguide: society now #motivation

```
#converting to lowercase
df["cleanText"] = df["tweet"].apply(lambda x : x.lower())
#creating a new column clean text to store the data after text processing
df.head(5)
```

	tweet	cleanText
0	@user when a father is dysfunctional and is s...	@user when a father is dysfunctional and is s...
1	@user @user thanks for #lyft credit i can't us...	@user @user thanks for #lyft credit i can't us...
2	bihday your majesty	bihday your majesty
3	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	factsguide: society now #motivation	factsguide: society now #motivation

2. Removing HTML Tags

What it is: This is where we get rid of any HTML stuff that might be in our text. For example, " This is a paragraph " just becomes "This is a paragraph".

Why we do it:

- HTML tags aren't usually part of the actual content we want to analyze. They're just extra noise.
- If we leave them in, our NLP tools might get confused and think they're real words.
- It makes the text easier for humans to read too.
- It puts all our text in the same format, no matter where it came from.

Here's how we do it in code: Though , this dataset probably doesn't have a html tag we are gonna do it just in case. Here , we remove all the html tags with the help of regex (regular expressions). We perform this step before removing special characters and before tokenization for two reasons:

1. If we removed special characters first, the "<" and ">" symbols would be eliminated, leaving us with incomplete HTML structures like "body" or "title" in the tweets. These could then be mistaken for regular words.
2. If we performed tokenization first, each element of an HTML tag would be segregated into different parts. For example, ["html", "body"] might result from splitting an HTML tag, making it more difficult to identify and remove later.

```
def removeHTML(text):
  pattern = r"<.*?>"
  return re.sub(pattern,"",text)
df["cleanText"] = df["cleanText"].apply(lambda x : removeHTML(x))
df
```

	tweet	cleanText
0	@user when a father is dysfunctional and is s...	@user when a father is dysfunctional and is s...
1	@user @user thanks for #lyft credit i can't us...	@user @user thanks for #lyft credit i can't us...
2	bihday your majesty	bihday your majesty
3	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	factsguide: society now #motivation	factsguide: society now #motivation
...	...	...
31957	ate @user isz that youuu?ðððððððððð...	ate @user isz that youuu?ðððððððððð...
31958	to see nina turner on the airwaves trying to...	to see nina turner on the airwaves trying to...
31959	listening to sad songs on a monday morning otw...	listening to sad songs on a monday morning otw...
31960	@user #sikh #temple vandalised in in #calgary,...	@user #sikh #temple vandalised in in #calgary,...
31961	thank you @user for you follow	thank you @user for you follow

3. Removing Punctuation and Special Characters

What it is: Here, we're taking out all the punctuation marks (like periods, commas, exclamation marks) and special characters (like @, #, $). So "Hello, world!" becomes "Hello world".

Why we do it:

- Most of the time, these marks and characters don't add much meaning, and they can mess up our analysis.
- It makes the text simpler and easier for our computer to process.
- It helps keep things consistent, especially when we're dealing with text from different sources.
- It can lead to better word splitting (tokenization) later on.
- It makes our data smaller, which can help our models run faster.

Here's how we do it: First, we store all the punctuation marks in a list using the punctuation method from the string module. Then, using the translate and maketrans functions, we eliminate all these punctuation marks from our text. Next, with the help of regex

(regular expressions), we replace all special characters with a space. However, this process creates extra spaces in our text. To address this, we use another regex operation to replace all consecutive spaces with a single space.

```
#Remove puncutuations and special characters
def remove_punc_special(text):
  punctuations = string.punctuation
  #stores all the punctuation characters(,.?etc)
  t = str.maketrans("","",punctuations)
  text = text.translate(t)
  text = re.sub(r"[^a-zA-Z0-9]"," ",text)
  text = re.sub(r"[\s]+"," ",text)
  return text
df["cleanText"] = df["cleanText"].apply(lambda x : remove_punc_special(x))
df
```

	tweet	cleanText
0	@user when a father is dysfunctional and is s...	user when a father is dysfunctional and is so...
1	@user @user thanks for #lyft credit i can't us...	user user thanks for lyft credit i cant use ca...
2	bihday your majesty	bihday your majesty
3	#model i love u take with u all the time in ...	model i love u take with u all the time in ur
4	factsguide: society now #motivation	factsguide society now motivation
...	...	...
31957	ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð□□□ð...	ate user isz that youuu
31958	to see nina turner on the airwaves trying to...	to see nina turner on the airwaves trying to ...
31959	listening to sad songs on a monday morning otw...	listening to sad songs on a monday morning otw...
31960	@user #sikh #temple vandalised in in #calgary,...	user sikh temple vandalised in in calgary wso ...
31961	thank you @user for you follow	thank you user for you follow

1962 rows × 2 columns

4. Tokenization

What it is: Tokenization is just a fancy word for splitting our text into individual words. So the sentence "I love NLP" would become a list: ["I", "love", "NLP"].

Why we do it:

- It lets us analyze text at the word level, which is super important for many NLP tasks.
- It's necessary for a lot of other preprocessing steps, like removing stop words,stemming,etc
- It helps us count words and do things like create "bag of words" models.
- It's the first step in understanding the structure of a sentence.

Here's the code: First , we downdload the necessary packages required for tokenization and then we convert all our text into tokens with the help of word_tokenize function

```
#tokenization
nltk.download('punkt')
df["cleanText"] = df["cleanText"].apply(lambda x : word_tokenize(x))
df
```

	tweet	cleanText
0	@user when a father is dysfunctional and is s...	[user, when, a, father, is, dysfunctional, and...
1	@user @user thanks for #lyft credit i can't us...	[user, user, thanks, for, lyft, credit, i, can...
2	bihday your majesty	[bihday, your, majesty]
3	#model i love u take with u all the time in ...	[model, i, love, u, take, with, u, all, the, t...
4	factsguide: society now #motivation	[factsguide, society, now, motivation]
...	...	...

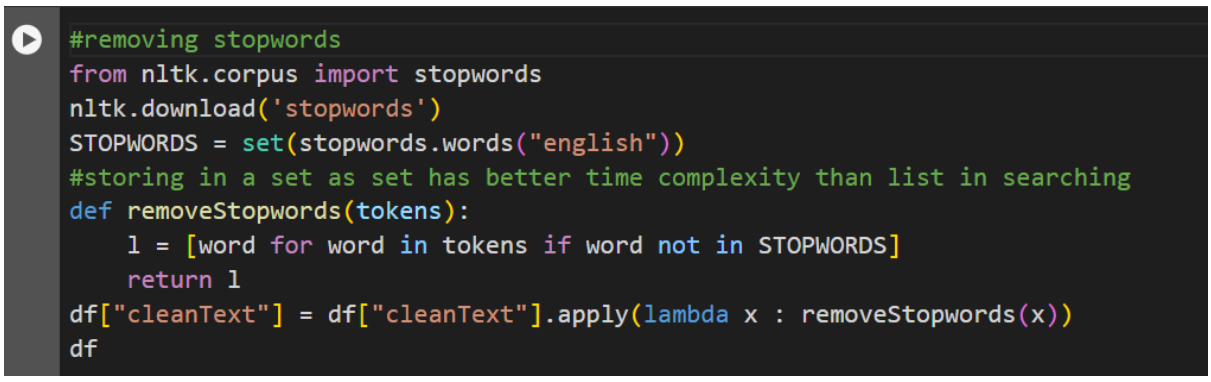
5. Removing Stop Words

What it is: This is where we take out really common words that don't usually add much meaning to a sentence like "the", "is", "in", "and". So "The cat is on the mat" might become just "cat mat".

Why we do it:

- It helps us focus on the words that really matter.
- It cuts down on the "noise" in our data.
- It makes our dataset smaller, which means faster processing and less memory use.
- Many NLP models work better without these common words cluttering things up.

Here's how we do it: First , we import and download the necessary packages required and then we store all our stopwords in a set because we mostly just want to perform searching operations and sets are very efficient for searching and then with the help of removeStopwords , we remove all the stopwords in the tweets and store the new result with apply method.



```
#removing stopwords
from nltk.corpus import stopwords
nltk.download('stopwords')
STOPWORDS = set(stopwords.words("english"))
#storing in a set as set has better time complexity than list in searching
def removeStopwords(tokens):
    l = [word for word in tokens if word not in STOPWORDS]
    return l
df["cleanText"] = df["cleanText"].apply(lambda x : removeStopwords(x))
df
```

	tweet	cleanText
0	@user when a father is dysfunctional and is s...	[user, father, dysfunctional, selfish, drags, ...
1	@user @user thanks for #lyft credit i can't us...	[user, user, thanks, lyft, credit, cant, use, ...
2	bihday your majesty	[bihday, majesty]
3	#model i love u take with u all the time in ...	[model, love, u, take, u, time, ur]
4	factsguide: society now #motivation	[factsguide, society, motivation]
...	...	...
31957	ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð...	[ate, user, isz, youuu]
31958	to see nina turner on the airwaves trying to...	[see, nina, turner, airwaves, trying, wrap, ma...
31959	listening to sad songs on a monday morning otw...	[listening, sad, songs, monday, morning, otw, ...
31960	@user #sikh #temple vandalised in in #calgary,...	[user, sikh, temple, vandalised, calgary, wso,...
31961	thank you @user for you follow	[thank, user, follow]

31962 rows × 2 columns

6. Removing Frequent Words

What it is: This is similar to removing stop words, but instead of using a pre-defined list, we're looking at our specific dataset and removing the words that show up most often. For example, in a bunch of movie reviews, we might remove words like "movie" or "film" if they appear too much.

Why we do it:

- It helps us clean up our text in a way that's specific to our data.
- It lets us focus on words that help distinguish one piece of text from another.
- Sometimes, it can help our models perform better.
- It can cut down on words that are so common in our specific data that they're just noise.
- It makes our data even smaller, which can be good for many machine learning algorithms.

  Here's how we do it: First, we count the occurrence of all words in our dataset using the Counter class. Then, we store

the three most common words in a set and remove these words using the replaceFrequentWords method. Finally, we store the new results.

```
#removing frequent words
from collections import Counter
counter = Counter()
for lists in df["cleanText"]:
  for word in lists:
    counter[word]+=1
frequentWords = set(word for word, count in counter.most_common(3))
print(counter.most_common(3))
def replaceFrequentWords(tokens):
    return [word for word in tokens if word not in frequentWords]
df["cleanText"] = df["cleanText"].apply(lambda x : replaceFrequentWords(x))
df
```

```
[('user', 17509), ('love', 2732), ('day', 2259)]
```

	tweet	cleanText
0	@user when a father is dysfunctional and is s...	[father, dysfunctional, selfish, drags, kids, ...
1	@user @user thanks for #lyft credit i can't us...	[thanks, lyft, credit, cant, use, cause, dont,...
2	bihday your majesty	[bihday, majesty]
3	#model i love u take with u all the time in ...	[model, u, take, u, time, ur]
4	factsguide: society now #motivation	[factsguide, society, motivation]
...	...	...
31957	ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð...	[ate, isz, youuu]
31958	to see nina turner on the airwaves trying to...	[see, nina, turner, airwaves, trying, wrap, ma...
31959	listening to sad songs on a monday morning otw...	[listening, sad, songs, monday, morning, otw, ...
31960	@user #sikh #temple vandalised in in #calgary,...	[sikh, temple, vandalised, calgary, wso, conde...
31961	thank you @user for you follow	[thank, follow]

31962 rows × 2 columns

7. Stemming and Lemmatization

What They Are:

Both stemming and lemmatization are techniques used to reduce words to their base or root form. However, they differ in their approaches and results.

Stemming: Stemming is a process that cuts off prefixes and suffixes to reduce a word to its root form. It uses algorithms that apply a set of rules to strip affixes from words. This process can sometimes produce words that are not in the dictionary

What it is: Stemming is a quick and straightforward method to chop off the ends of words. For example, "running," "runs," and "ran" will all be reduced to "run".

Why we do it:

- It's faster and simpler, making it suitable for large datasets.
- Reduces the number of unique words, saving memory and computational resources.
- Useful in applications where approximate results are acceptable.

Lemmatization: Lemmatization reduces words to their base or dictionary form, known as a lemma. It considers the context and the word's part of speech to achieve this. Lemmatization ensures that the root word is a valid word in the language.

What it is: Lemmatization is a more sophisticated technique that finds the dictionary form of a word. For example, "better" would be reduced to "good".

Why we do it:

- Produces accurate and meaningful root words, improving the quality of text analysis.
- Useful in applications where precise linguistic analysis is required.
- Handles different inflected forms correctly based on context.

Here's how we do it: In the code we're looking at, I went with stemming instead of lemmatization. Here's why:

1. It's faster, which is great when we're dealing with lots of data.
2. It's simpler and doesn't need as much setup.
3. For a lot of tasks, especially with informal text like tweets, stemming works well enough.

We reduce each word with the help of stem method and then create a new column to show the differences between original and stemmed text

```
[32] #stemming
     from nltk.stem import PorterStemmer
     ps = PorterStemmer()
     def stemWords(tokens):
         l = [ps.stem(word) for word in tokens]
         return l
     df["stemText"]  = df["cleanText"].apply(lambda x : stemWords(x))
     #creating a new column to show the difference between the two
     df
```

	tweet	cleanText	stemText
0	@user when a father is dysfunctional and is s...	[father, dysfunctional, selfish, drags, kids, ...	[father, dysfunct, selfish, drag, kid, dysfunc...
1	@user @user thanks for #lyft credit i can't us...	[thanks, lyft, credit, cant, use, cause, dont,...	[thank, lyft, credit, cant, use, caus, dont, o...
2	bihday your majesty	[bihday, majesty]	[bihday, majesti]
3	#model i love u take with u all the time in ...	[model, u, take, u, time, ur]	[model, u, take, u, time, ur]
4	factsguide: society now #motivation	[factsguide, society, motivation]	[factsguid, societi, motiv]
...	...	...	...
31957	ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð...	[ate, isz, youuu]	[ate, isz, youuu]
31958	to see nina turner on the airwaves trying to...	[see, nina, turner, airwaves, trying, wrap, ma...	[see, nina, turner, airwav, tri, wrap, mantl, ...
31959	listening to sad songs on a monday morning otw...	[listening, sad, songs, monday, morning, otw, ...	[listen, sad, song, monday, morn, otw, work, sad]
31960	@user #sikh #temple vandalised in in #calgary,...	[sikh, temple, vandalised, calgary, wso, conde...	[sikh, templ, vandalis, calgari, wso, condemn,...
31961	thank you @user for you follow	[thank, follow]	[thank, follow]

31962 rows × 3 columns

Then , we store the result of stemText column in cleanText and delete the stemText column . After that , we convert our tokens back to string for display purposes .

```
df["cleanText"] = df["stemText"]
#deleting the stem text column
df = df.drop("stemText",axis = 1)
```

```
#coverting the tokens back to string for display
df["cleanText"] = df["cleanText"].apply(lambda x : " ".join(x))
df
```

Here is the final result:

	tweet	cleanText
0	@user when a father is dysfunctional and is s...	father dysfunct selfish drag kid dysfunct run
1	@user @user thanks for #lyft credit i can't us...	thank lyft credit cant use caus dont offer whe...
2	bihday your majesty	bihday majesti
3	#model i love u take with u all the time in ...	model u take u time ur
4	factsguide: society now #motivation	factsguid societi motiv
...	...	...
31957	ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð...	ate isz youuu
31958	to see nina turner on the airwaves trying to...	see nina turner airwav tri wrap mantl genuin h...
31959	listening to sad songs on a monday morning otw...	listen sad song monday morn otw work sad
31960	@user #sikh #temple vandalised in in #calgary,...	sikh templ vandalis calgari wso condemn act
31961	thank you @user for you follow	thank follow

31962 rows × 2 columns

Wrapping Up

So, that's text preprocessing in a nutshell. Each step helps clean up our text, cut down on noise, and focus on the important stuff. Depending on what we're trying to do and what kind of text we're working with, we might use different steps or do them in a different order. The key is to think about how each step affects our text and choose the ones that work best for our specific task.

References : Link to my colab notebook :

https://colab.research.google.com/drive/1_2LPGk8DNt0i1K12iAq1aZvtIXXjHo-8?usp=sharing

Text Preprocessing in NLP | Python (youtube.com)

Tokenization | NLP | Python (youtube.com)

Apply method in python pandas| Use lambda function and defined functions to preprocess your data (youtube.com)

Python Tutorials - String Methods | translate() | maketrans() - YouTube

Regular Expression Tutorial Python | Python Regex Tutorial (youtube.com)

Tutorial 4- Stemming And Lemmatization And Its Implementation NLP|Krish NAik - YouTube