In [4]:

```
import pandas as pd
import wordninja
import re
import nltk
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')
nltk.download('stopwords')
from sklearn.model_selection import train_test_split
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/summerai/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package punkt to /Users/summerai/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/summerai/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [5]:

```
# import data
df = pd.read_csv('combined_fulltext.csv')
```

In [6]:

```
# select necessary data
df = df[['uid','full_text','classifiers']]
df = df[df['full_text'].notnull()].reset_index(drop=True)
```

In [9]:

```
# split multiple lables to rows
data = df.set_index(['uid', 'full_text']) \
                .apply(lambda x: x.str.split('|').explode()).reset_index()
```

In [12]:

```
# create a function to clean the text
def fulltext_clean(string):
    #PREPARATION
    # step 1
    # repalce all characters with a white space except these three char -  ,  . among di
gits/letters;
    # Eg. keep 2,000, 3.00, covid-19
    remove_char = re.sub(r"(?!(?<=[a-zA-Z0-9])[\,\.\-](?=[a-zA-Z0-9]))[^a-zA-Z0-9 \n]","
", string)
    # if there are more than one white spaces between words, reduce to one
    remove_spaces = re.sub('\s+', " ", remove_char).strip()

    # step 2
    # if a word matches this pattern or is in the list then we don't want to pass it to w
ordninja
    # if there is hyphen, combination of letters and digits or pure capitalized letters,
don't pass
    wordninja_filter = re.compile(r"-|([A-Za-z]+\d+\w*|\d+[A-Za-z]+\w*)|^[^a-z]*$")
    # if a word is in the list, don't pass it to wordninja because it can't handle the wo
rd well
    words_pass = ['qanon', 'covid']

    # step 3
    # set up for lemmatize
    def get_wordnet_pos(word):
```

```
        """Map POS tag to first character lemmatize() accepts"""
        tag = nltk.pos_tag([word])[0][1][0].upper()
        tag_dict = {"J": wordnet.ADJ,
                    "N": wordnet.NOUN,
                    "V": wordnet.VERB,
                    "R": wordnet.ADV}
        return tag_dict.get(tag, wordnet.NOUN)

    lemmatizer = WordNetLemmatizer()

    # step4
    # remove stop words
    stop_words = set(stopwords.words('english'))

    # CLEANING
    # split the string by a white space
    string_isolated = remove_spaces.split()

    # check the string word by word to detect necessary split, lemmatize and remove stop
word
    words_split = ''
    for el in string_isolated:
        # step 2
        # if the word matches the pattern or is in the list, then we don't pass it to wor
dnijia to split
        if wordninja_filter.search(el) or el.lower() in words_pass:
            temp = el
        # all the other words will be checked and be split if necessary
        else:
            temp = ' '.join(wordninja.split(el))

        # step 3: lemmatize the word
        words_lemmatized = lemmatizer.lemmatize(temp, get_wordnet_pos(temp))

        # step 4 & step 5
        if words_lemmatized.lower() not in stop_words:
             words_split += ' ' + words_lemmatized.lower()

    words_split = words_split.strip()

    return words_split
```

In [13]:

```
# apply the function to the whole dataset
for i in range(len(data)):
    string = data.iloc[i,-2]
    data.iloc[i,-2] = fulltext_clean(string)
```

In [14]:

```
# apply one hot encoding
one_hot = pd.get_dummies(data['classifiers'])
data.drop(columns = 'classifiers', axis=1, inplace=True)
final_data = data.join(one_hot)
```

In [16]:

```
# output the data into csv format
final_data.to_csv('fulltext_cleaned.csv', index=False)
```

In [ ]:

```

```