FAKE NEWS DETECTION BY NLP:

PHASE 3

NAME	KAMALESHWARAN R
REGISTER NUMBER	61772221T304
DATE	21/10/2023

CONTENTS:

- DATA IMPORTING
- DATA CLEANING
- DATA VISUALIZATION
- DATA ANALYSIS

DATA IMPORTING:

```
# Fake news detection by NLP
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import nltk
import re
import string

from bs4 import BeautifulSoup
from nltk.corpus import stopwords

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

import keras
from keras.preprocessing import text,sequence
from keras.models import Sequential
from keras.layers import Dense,Embedding,LSTM,Dropout

import warnings
warnings.filterwarnings('ignore')
```

```
# DATA IMPORT
```

In [2]:
```
#data importing
real_news = pd.read_csv('True_News.csv')
fake_news = pd.read_csv('Fake_News.csv')
#here we import the data set of both the true and fake news
```

Reading CSV Files: Here the pd.read_csv() function from Pandas to read data from CSV files. CSV (Comma-Separated Values) files are a common data format for storing structured data.

- real_news = pd.read_csv('True_News.csv'): This line reads the data from a CSV file named 'True_News.csv' and stores it in a Pandas Data Frame ca led real_news. This file is expected to contain data related to "true" or real news articles.
- fake_news = pd.read_csv('Fake_News.csv'): Similarly, this line reads data from a CSV file named 'Fake_News.csv' and stores it in another Pandas Data Frame called fake news. This file is expected to contain data related to "fake" news articles.

DATA CLEANING:

del data['title']: This removes the 'title' column from the Data Frame. It's removed from the Data Frame's structure, and you will no longer be able to access this column's data.

Similarly, this line removes the 'subject' column from the Data Frame. This line removes the 'date' column from the Data Frame.

After running these lines, the Data Frame data wi l no longer contain the 'title', 'subject', and 'date' columns. This is often done when we want to focus on specific columns or when those columns are no longer needed for the plan to perform.

The code you provided contains a series of functions and a final function called cleaning that are used to preprocess text data in a Pandas Data Frame. The purpose of this preprocessing is to clean and prepare the text for various natural language processing (NLP) tasks, such as text classification or sentiment analysis.

DATA CLEANING

In [10]:
```
#the colums of title,subject,date are deleted as they are not used.
data['text']= data['subject'] + " " + data['title'] + " " + data['text']
del data['title']
del data['subject']
del data['date']
data.head()
```

Out[10]:

	text	target
0	politicsNews As U.S. budget fight looms, Repub...	0
1	politicsNews U.S. military to accept transgend...	0
2	politicsNews Senior U.S. Republican senator: '...	0
3	politicsNews FBI Russia probe helped by Austra...	0
4	politicsNews Trump wants Postal Service to cha...	0

This function combines the previous functions to create a comprehensive cleaning process. It takes a text as input and applies the following steps in order:

- Removes HTML content.
- Removes punctuation marks and special characters.
- Removes non-alphabet characters.

- Removes stop words and lemmatizes the remaining words.
- Returns the cleaned text.

In [22]:
```
#Removal of HTML Contents
def remove_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

#Removal of Punctuation Marks
def remove_punctuations(text):
    return re.sub('\[[^]]*\]', '', text)

# Removal of Special Characters
def remove_characters(text):
    return re.sub("[^a-zA-Z]"," ",text)

#Removal of stopwords
def remove_stopwords_and_lemmatization(text):
    final_text = []
    text = text.lower()
    text = nltk.word_tokenize(text)

    for word in text:
        if word not in set(stopwords.words('english')):
            lemma = nltk.WordNetLemmatizer()
            word = lemma.lemmatize(word)
            final_text.append(word)
    return " ".join(final_text)

#Total function
def cleaning(text):
    text = remove_html(text)
    text = remove_punctuations(text)
    text = remove_characters(text)
    text = remove_stopwords_and_lemmatization(text)
    return text

#Apply function on text column
data['text']=data['text'].apply(cleaning)
```

DATA VISUALIZATION:

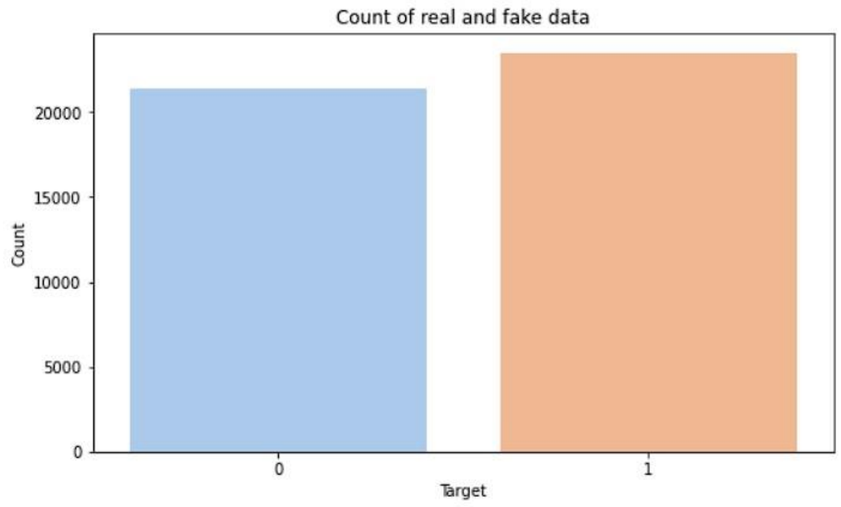
This prints the counts of each unique value in the "target" column. It will show you how many instances belong to each class, which is useful for understanding the class distribution.

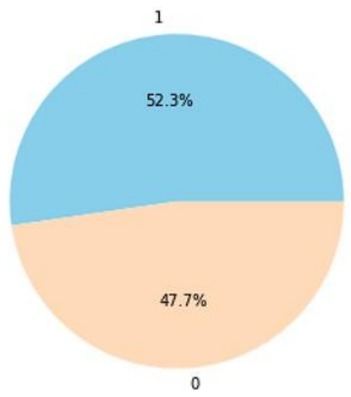
This code creates a pie chart in the second subplot. It uses the values and labels from the count of "target" classes to create the chart.

DATA VISUALIZATION

In [8]:
```
#we express the dataset of both true and fake in terms of a bar chart and a pie chart.
print(data["target"].value_counts())
fig, ax = plt.subplots(1,2, figsize=(19, 5))
g1 = sns.countplot(data.target,ax=ax[0],palette="pastel");
g1.set_title("Count of real and fake data")
g1.set_ylabel("Count")
g1.set_xlabel("Target")
g2 = plt.pie(data["target"].value_counts().values,explode=[0,0],labels=data.target.value_counts().index, autopct='%1.1f%%',colors
fig.show()
```

```
1    23481
0    21417
Name: target, dtype: int64
```





- X ="subject": This sets the values on the x-axis to come from the "subject" column, which represents the different subjects.
- Hue ='target': It adds color differentiation based on the "target" column. The plot wi l have two bars (real and fake) for each subject, with dif erent colors.

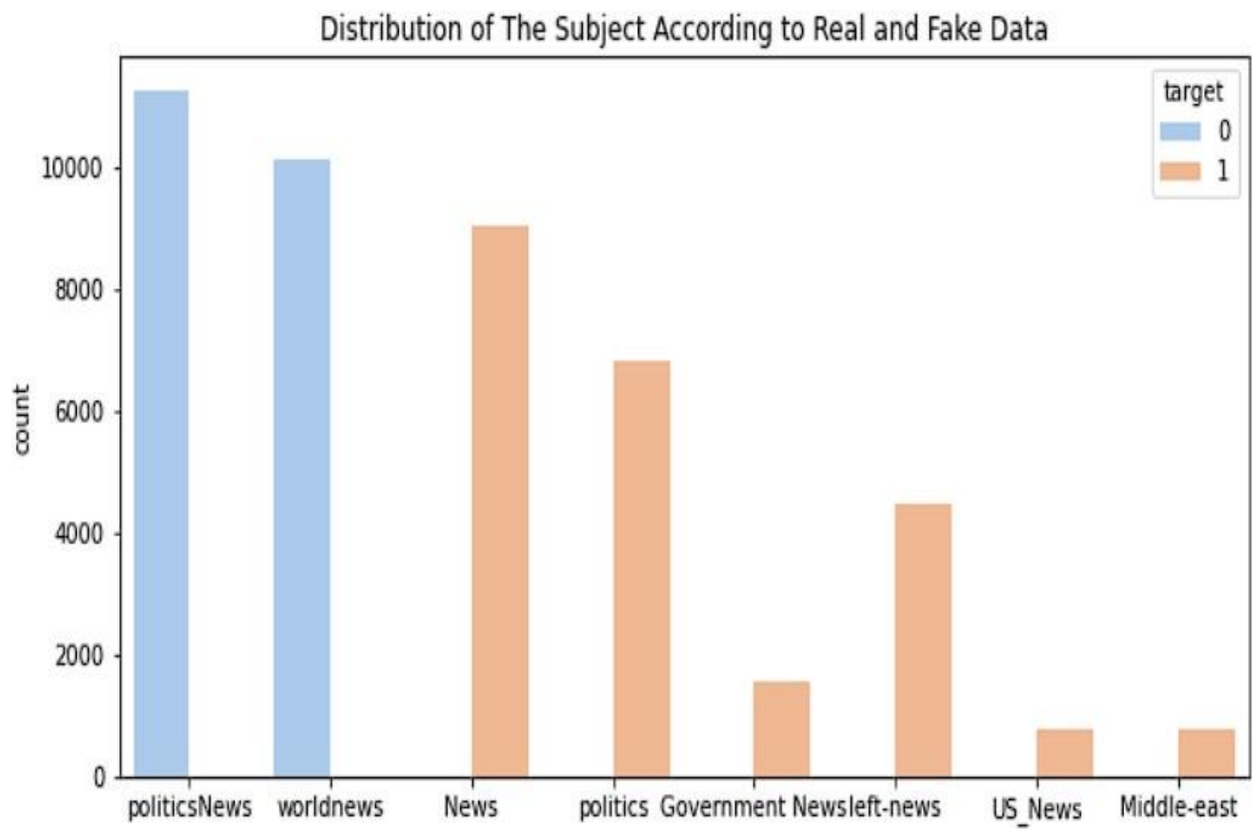
- Data =data: Specifies the DataFrame from which the data for the plot is retrieved.
- Palette ="pastel": Sets the color palette to "pastel," giving the plot a more colorful and visua ly appealing style.

```
In [9]: #here we get to see the contents of the subject and the count of news in each subject
        print(data.subject.value_counts())
        plt.figure(figsize=(10, 5))

        ax = sns.countplot(x="subject",  hue='target', data=data, palette="pastel")
        plt.title("Distribution of The Subject According to Real and Fake Data")
```

```
politicsNews       11272
worldnews          10145
News                9050
politics            6841
left-news           4459
Government News     1570
US_News              783
Middle-east          778
Name: subject, dtype: int64
```

```
Out[9]: Text(0.5, 1.0, 'Distribution of The Subject According to Real and Fake Data')
```



WORD CLOUD:

This code imports the Word Cloud class from the word cloud library and the STOPWORDS set, which contains common English stop words.

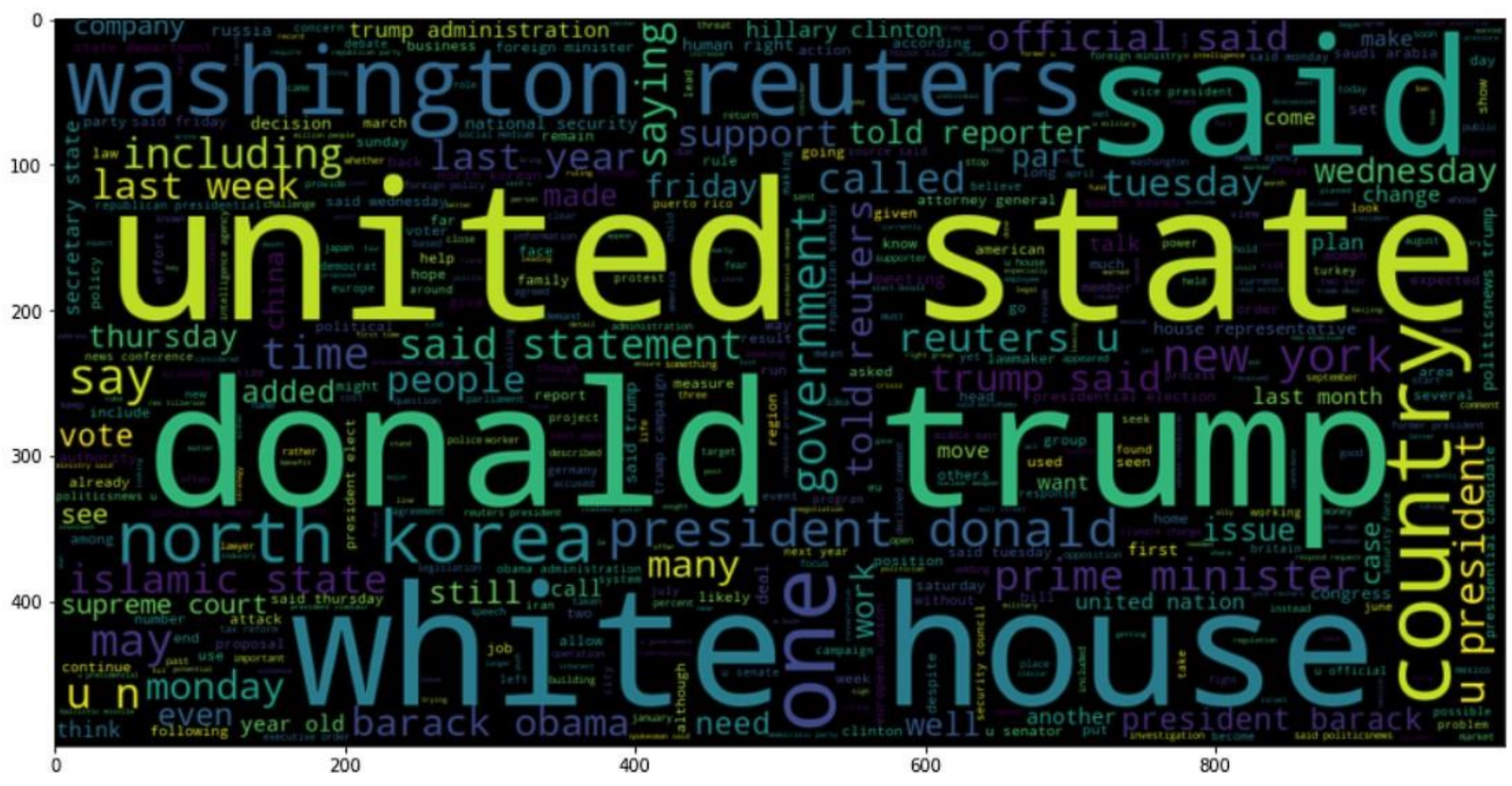
- Maximum number of words to include in the word cloud is set.
- Width and height specify the dimensions of the word cloud image.

REAL DATASET

 # 1. real news

```
In [26]: from wordcloud import WordCloud,STOPWORDS
plt.figure(figsize = (15,15))
wc = WordCloud(max_words = 500 , width = 1000 , height = 500 , stopwords = STOPWORDS).generate(" ".join(data[data.target == 0]
plt.imshow(wc , interpolation = 'bilinear')
```

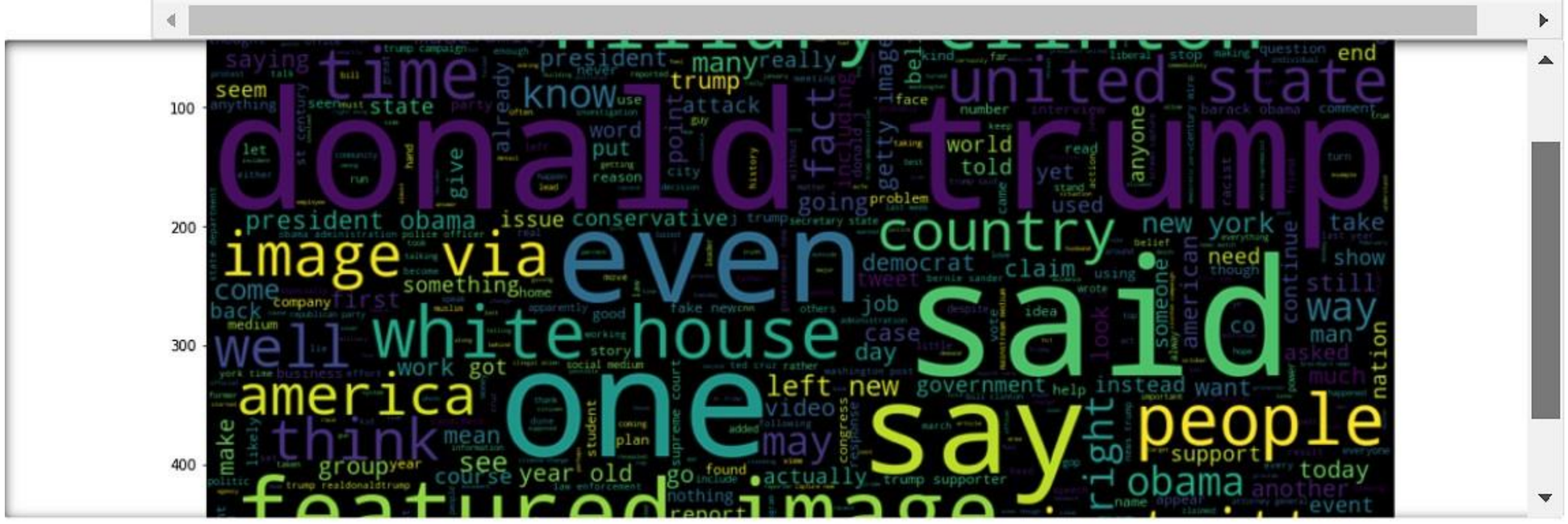
```
Out[26]: <matplotlib.image.AxesImage at 0x1f000997310>
```



FAKE DATASET :

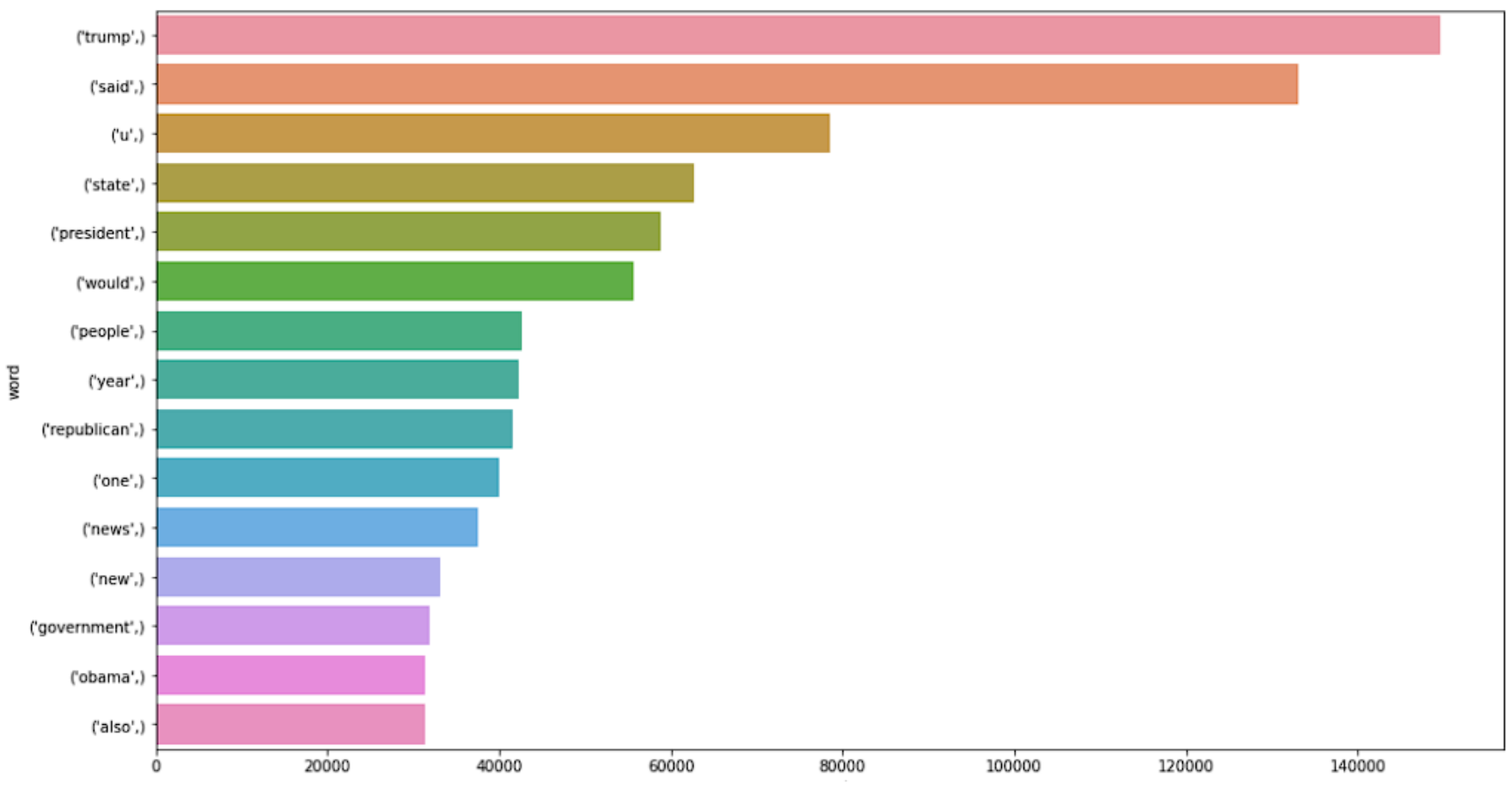
2. fake news

In [27]:
```
#the word cloud is used to visualize the most used words.
from wordcloud import WordCloud,STOPWORDS
plt.figure(figsize = (15,15))
wc = WordCloud(max_words = 500 , width = 1000 , height = 500 , stopwords = STOPWORDS).generate(" ".join(data[data.target == 1].te
plt.imshow(wc , interpolation = 'bilinear')
```



DATA ANA LYSI

Out[29]: <AxesSubplot:xlabel='count', ylabel='word'>



Finally, it creates a bar plot using Sea born to visualize the frequency of these n-grams, with the x-axis displaying the count and the y-axis showing the individual n-grams.