```
In [1]: import numpy as np # linear algebra
        import pandas as pd
```

```
In [2]: # dowload dataset
        fake = pd.read_csv('ISOT Dataset/Fake.csv')
        true = pd.read_csv("ISOT Dataset/True.csv")
```

```
In [3]: # Add flag to track fake and real
        fake['target'] = 1
        true['target'] = 0
```

```
In [4]: data = pd.concat([fake, true]).reset_index(drop = True)
        data.shape

Out[4]: (44898, 5)
```

```
In [5]: from sklearn.utils import shuffle
        data = shuffle(data)
        data = data.reset_index(drop=True)
```

```
In [6]: data.head()
```

Out[6]:

	title	text	subject	date	target
0	Republican ex-Treasury chief Paulson slams Tru...	WASHINGTON (Reuters) - Henry Paulson, a Republ...	politicsNews	June 25, 2016	0
1	SHOCK POLL In MUST WIN State Of FLORIDA: Hispa...	Apparently the Black Lives Matter terror group...	left-news	Jul 11, 2016	1
2	MEDALS OF VALOR: President Trump Honored Agent...	It s great to have a president who appreciates...	politics	Jul 27, 2017	1
3	Newsweek Just Made Their BEST Cover Ever And ...	Newsweek has never been a publication to shy a...	News	November 9, 2017	1
4	Trump says he believes Cuba responsible for at...	WASHINGTON (Reuters) - President Donald Trump ...	politicsNews	October 16, 2017	0

```
In [7]: ## Data Preparation
        data = data[data['text'].notna()]
        data = data[data['title'].notna()]
        data = data[data['subject'].notna()]
```

```
In [8]: import matplotlib.pyplot as plt
        import seaborn as sns

        import nltk
        nltk.download('stopwords')
        nltk.download('wordnet')
        [nltk_data] Downloading package stopwords to
Out[8]: True
```

```
In [9]: # Let's do some statistics of the text columns
        txt_len = data.text.str.split().str.len()
        txt_len.describe()
```

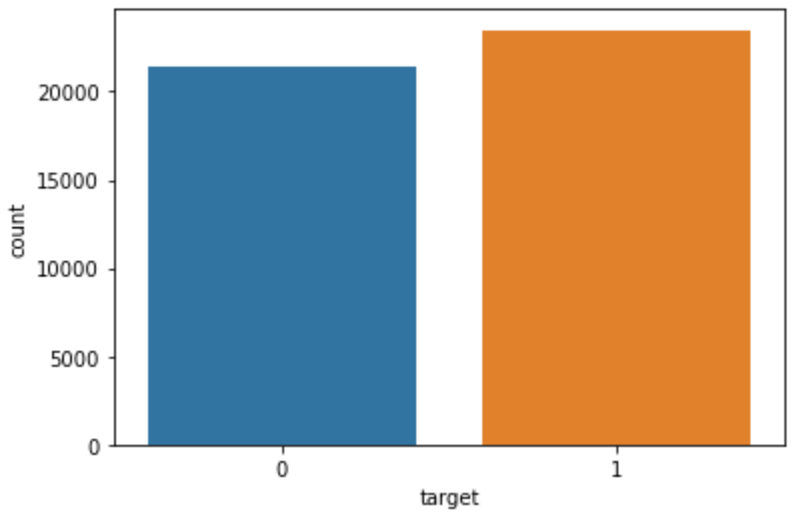
```
Out[9]: count    44898.000000
        mean       405.282284
        std        351.265595
        min          0.000000
        25%        203.000000
        50%        362.000000
        75%        513.000000
        max       8135.000000
        Name: text, dtype: float64
```

```
In [10]: # Let's do some statistics of the title columns
         title_len = data.title.str.split().str.len()
         title_len.describe()
```

```
Out[10]: count    44898.000000
         mean        12.453472
         std          4.111476
         min          1.000000
         25%         10.000000
         50%         11.000000
         75%         14.000000
         max         42.000000
         Name: title, dtype: float64
```

```
In [11]: # Class Distribution
         # 1: Unreliable
         # 2: Reliable
         sns.countplot(x='target', data= data)
```

```
Out[11]: <AxesSubplot:xlabel='target', ylabel='count'>
```



```
In [12]: print(data.target.value_counts())
         print()
         print(round(data.target.value_counts(normalize=True),2)*100)
```

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

1    52.0
0    48.0
```

In [13]:
```
data.isnull().sum()
```

Out[13]:
```
title      0
text       0
subject    0
date       0
target     0
dtype: int64
```

In [14]:
```
column_n = ['date', 'title', 'subject', 'text', 'target']
remove_c = ['subject','date']
categorical_features = []
target_col = ['target']
text_f = ['title', 'text']
```

In [15]:
```
# cleaning
import nltk
from nltk.corpus import stopwords
import re
from nltk.stem.porter import PorterStemmer
from collections import Counter

ps = PorterStemmer()
wnl = nltk.stem.WordNetLemmatizer()

stop_words = stopwords.words('english')
stopwords_dict = Counter(stop_words)

# remove unused columns
def remove_unused_c(df, column_n=remove_c):
    df = df.drop(column_n, axis=1)
    return df

# impute null values with none
def null_process(feature_df):
    for col in text_f:
        feature_df.loc[feature_df[col].isnull(),col] = "None"
    return feature_df

# clean_data
def clean_dataset(df):
    # remove unused column
    df = remove_unused_c(df)
    #impute null value
    df = null_process(df)

    return df

# Cleaning text from unused characters
def clean_text(text):
    text = str(text).replace(r'http[\w:/\.]+', ' ')  # removing urls
    text = str(text).replace(r'[^\.\w\s]', ' ')  # remove everything
    text = str(text).replace('[^a-zA-Z]', ' ')
    text = str(text).replace(r'\s\s+', ' ')
    text = text.lower().strip()
    #text = ' '.join(text)
    return text

## Nltk Preprocessing include:
# Stop words, Stemming and Lemmetization
# For our project we use only Stop word removal
def nltk_preprocess(text):
    text = clean_text(text)
    wordlist = re.sub(r'[^\w\s]', '', text).split()
    text = ' '.join([wnl.lemmatize(word) for word in wordlist if word
    return  text
```

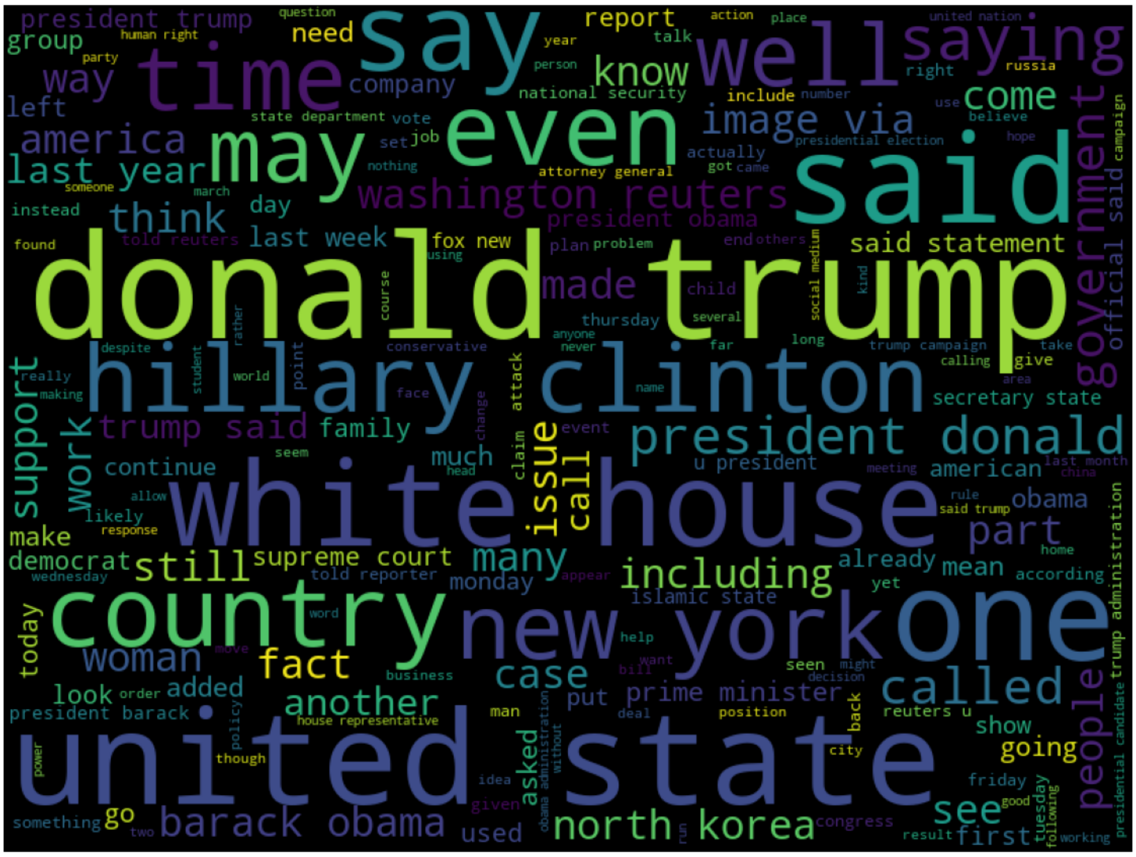
In [16]:
```
df = clean_dataset(data)
df['text'] = df.text.apply(nltk_preprocess)
df['title'] = df.title.apply(nltk_preprocess)
```

In [17]:
```
df.head()
```

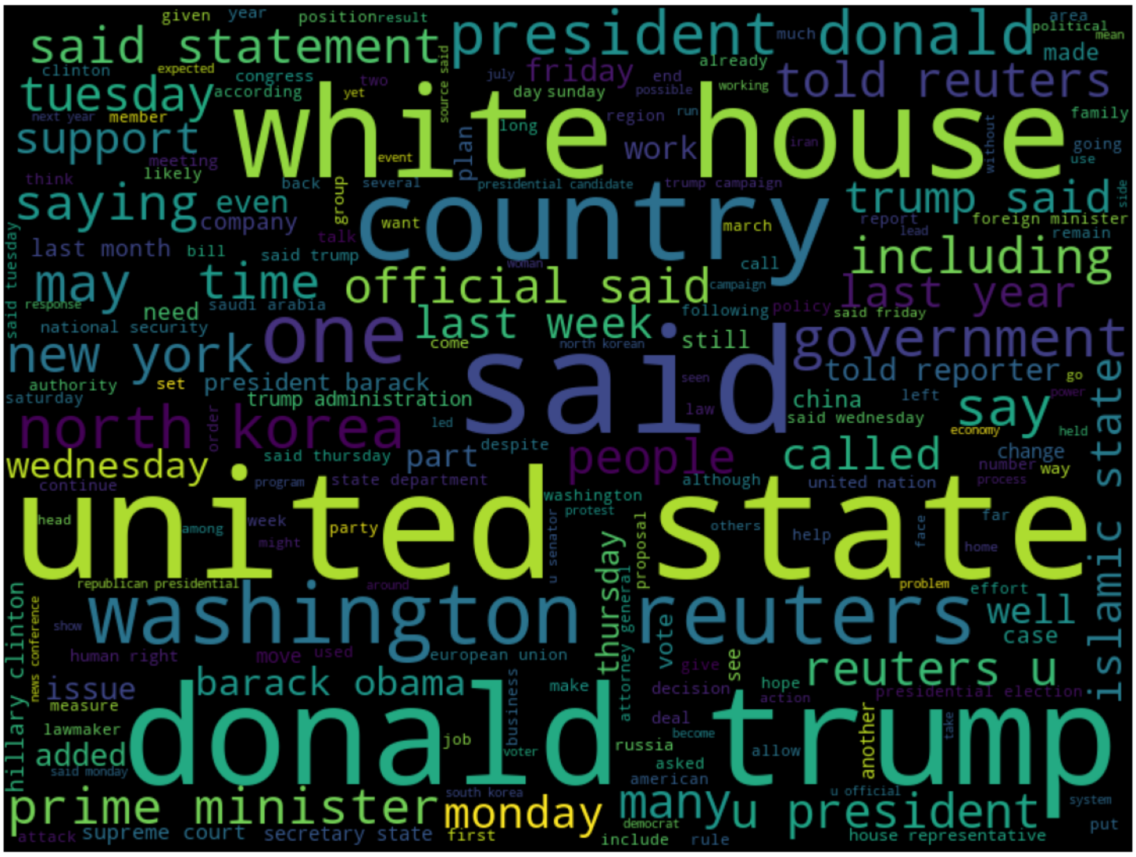
Out[17]:

	title	text	target
0	republican extreasury chief paulson slam trump...	washington reuters henry paulson republican u ...	0
1	shock poll must win state florida hispanic tur...	apparently black life matter terror group mana...	1
2	medal valor president trump honored agent offi...	great president appreciates special agent poli...	1
3	newsweek made best cover ever people freaking	newsweek never publication shy away controvers...	1
4	trump say belief cuba responsible attack hurt ...	washington reuters president donald trump said...	0

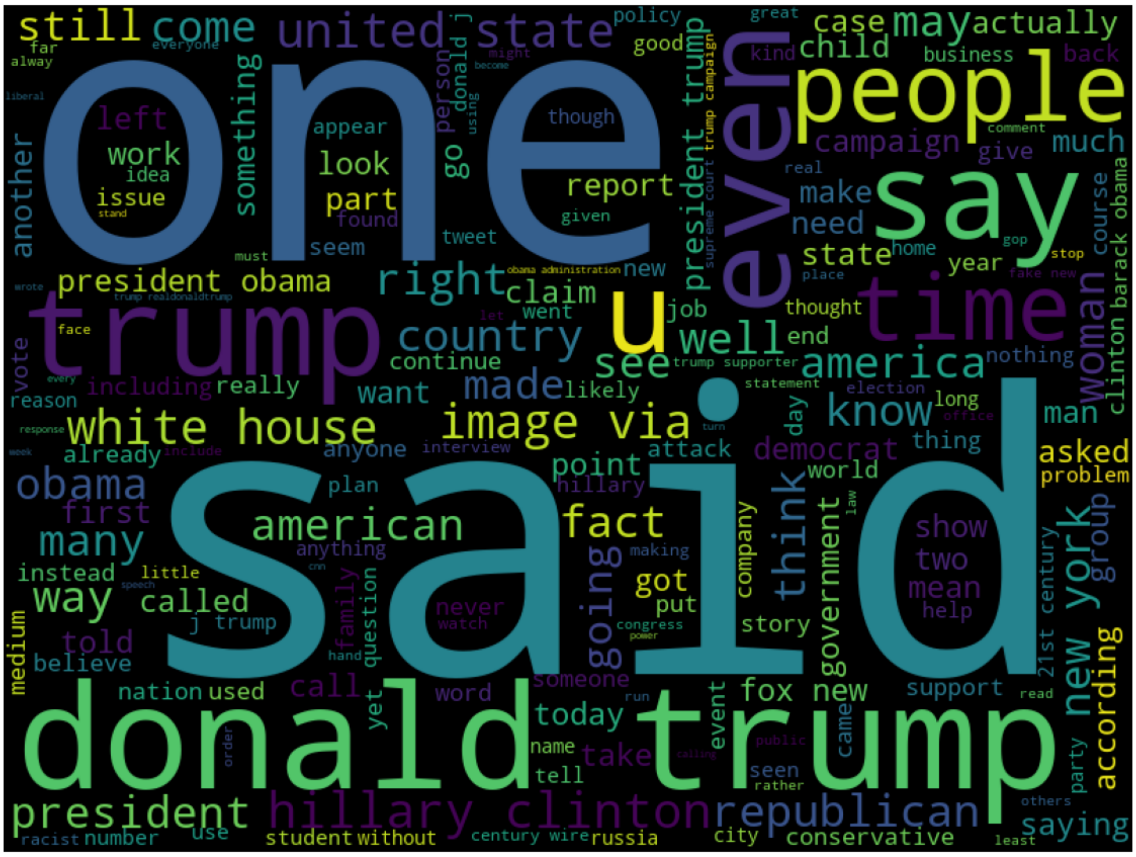
In [18]:
```
from wordcloud import WordCloud, STOPWORDS

# initialize the word cloud
wordcloud = WordCloud(background_color='black', width=800, height=60
# generate the word cloud
text_cloud = wordcloud.generate(" ".join(df['text']))
# plotting the word cloud
plt.figure(figsize=(20,30))
plt.imshow(text_cloud)
plt.axis('off')
plt.show()
```



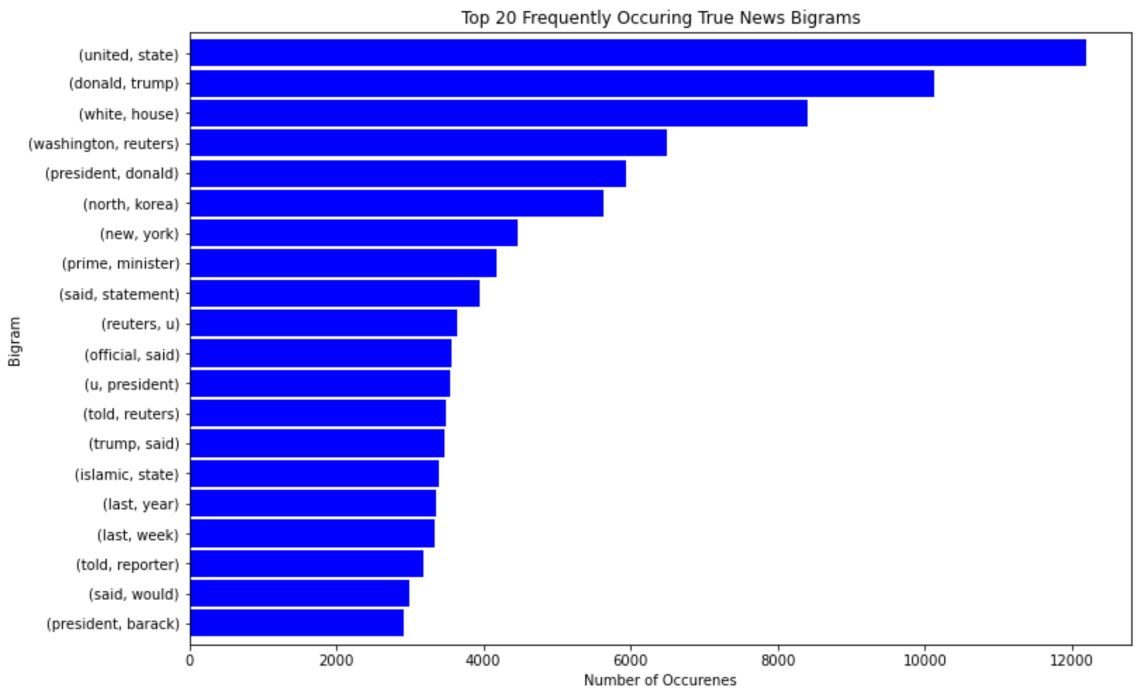
In [19]:
```
# reliable news (0)
reliable_news = " ".join(df[df['target']==0]['text'])
wc = wordcloud.generate(reliable_news)
plt.figure(figsize=(20,30))
plt.imshow(wc)
plt.axis('off')
plt.show()
```



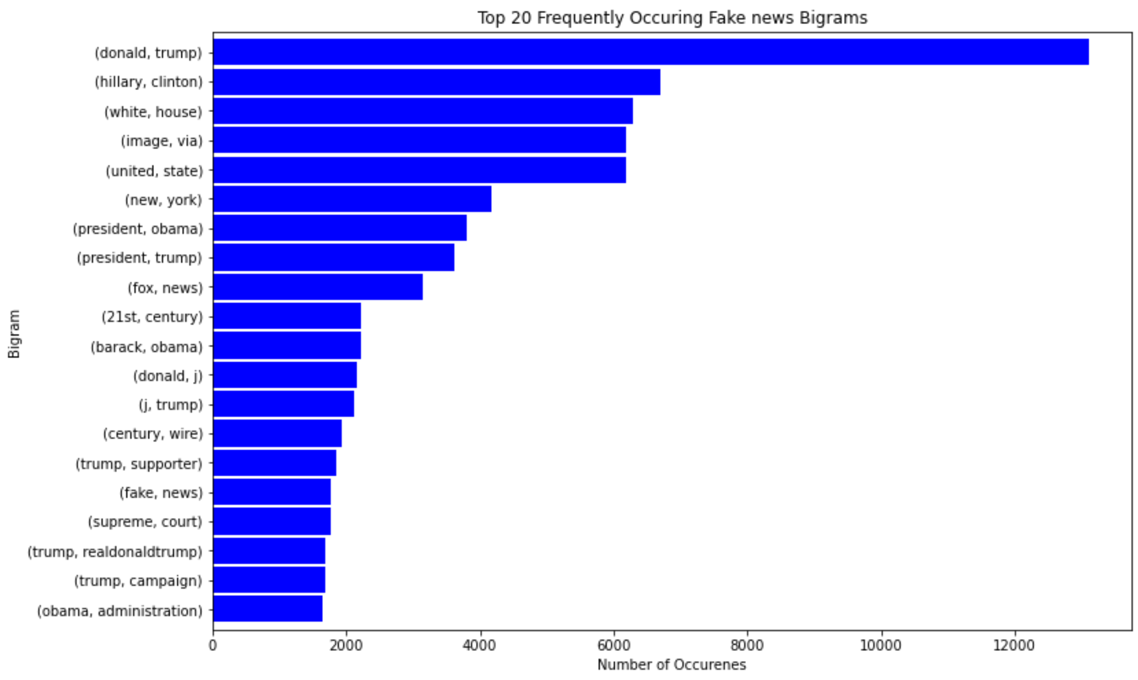
```
In [20]: # unreliable news (1)
         unreliable_news  = ' '.join(df[df['target']==1]['text'])
         wc= wordcloud.generate(unreliable_news)
         plt.figure(figsize=(20,30))
         plt.imshow(wc)
         plt.axis('off')
         plt.show()
```



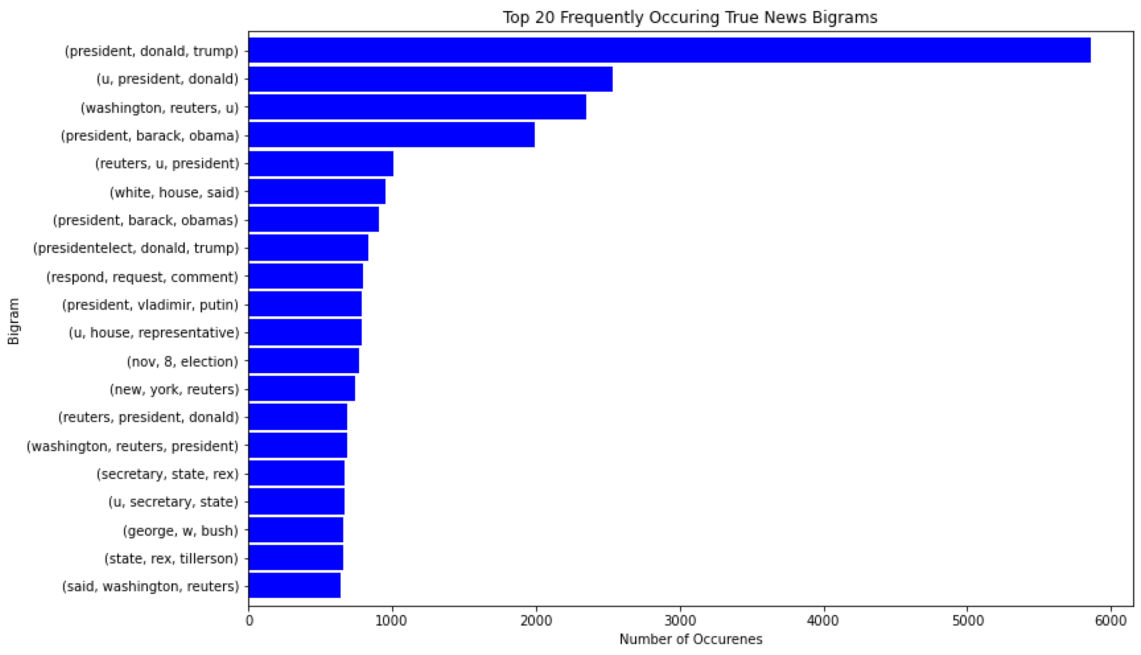
In [21]:
```
# Bigram

def plot_top_ngrams(corpus, title, ylabel, xlabel="Number of Occurene
    true_b = (pd.Series(nltk.ngrams(corpus.split(), n)).value_counts(
    true_b.sort_values().plot.barh(color='blue', width=.9, figsize=(1
    plt.title(title)
    plt.ylabel(ylabel)
    plt.xlabel(xlabel)
    plt.show()


plot_top_ngrams(reliable_news, "Top 20 Frequently Occuring True News
```



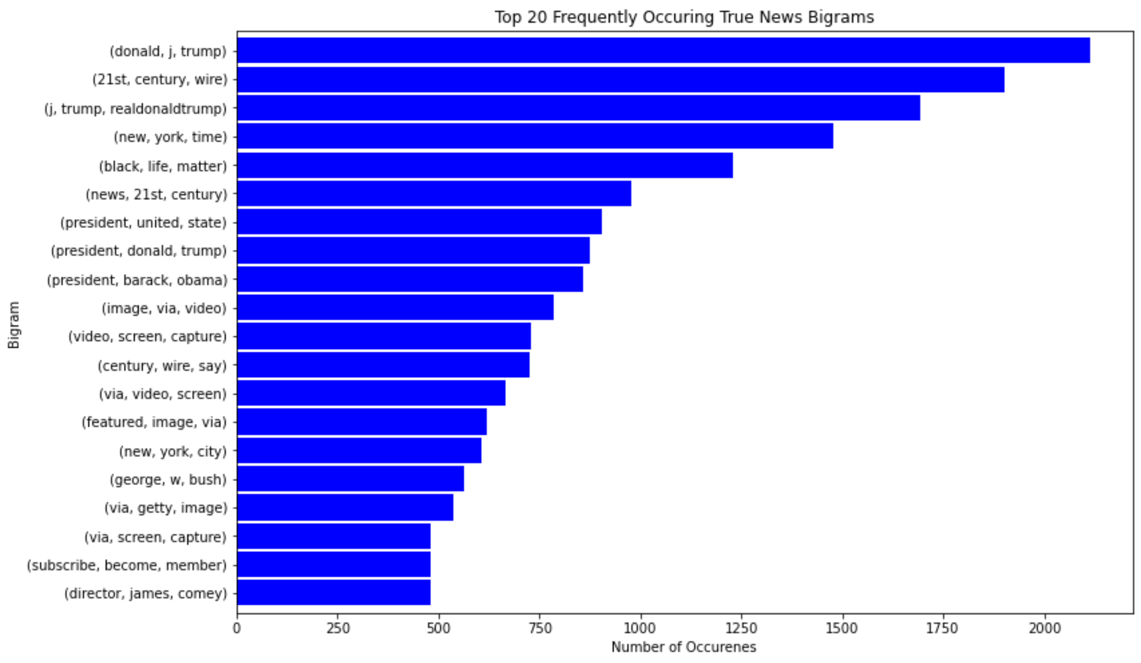
In [22]:
```
plot_top_ngrams(unreliable_news, 'Top 20 Frequently Occuring Fake new
```



In [23]:
```
# Trigram
plot_top_ngrams(reliable_news, "Top 20 Frequently Occuring True News
```



In [24]:
```
plot_top_ngrams(unreliable_news, "Top 20 Frequently Occuring True New
```



In [25]:
```
!pip install transformers
```

```
Requirement already satisfied: transformers in /home/administrator/
anaconda3/lib/python3.9/site-packages (4.18.0)
Requirement already satisfied: sacremoses in /home/administrator/an
aconda3/lib/python3.9/site-packages (from transformers) (0.0.49)
Requirement already satisfied: regex!=2019.12.17 in /home/administr
ator/anaconda3/lib/python3.9/site-packages (from transformers) (202
1.8.3)
Requirement already satisfied: packaging>=20.0 in /home/administrat
or/anaconda3/lib/python3.9/site-packages (from transformers) (21.0)
Requirement already satisfied: tqdm>=4.27 in /home/administrator/an
aconda3/lib/python3.9/site-packages (from transformers) (4.62.3)
Requirement already satisfied: requests in /home/administrator/anac
onda3/lib/python3.9/site-packages (from transformers) (2.26.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.13,>=0.11.1 in
/home/administrator/anaconda3/lib/python3.9/site-packages (from tra
nsformers) (0.12.1)
Requirement already satisfied: numpy>=1.17 in /home/administrator/a
naconda3/lib/python3.9/site-packages (from transformers) (1.20.3)
Requirement already satisfied: pyyaml>=5.1 in /home/administrator/a
naconda3/lib/python3.9/site-packages (from transformers) (6.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in /home
/administrator/anaconda3/lib/python3.9/site-packages (from transfor
mers) (0.5.1)
Requirement already satisfied: filelock in /home/administrator/anac
onda3/lib/python3.9/site-packages (from transformers) (3.3.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /home/
administrator/anaconda3/lib/python3.9/site-packages (from huggingfa
ce-hub<1.0,>=0.1.0->transformers) (3.10.0.2)
Requirement already satisfied: pyparsing>=2.0.2 in /home/administra
tor/anaconda3/lib/python3.9/site-packages (from packaging>=20.0->tr
ansformers) (3.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/admin
istrator/anaconda3/lib/python3.9/site-packages (from requests->tran
sformers) (1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/a
dministrator/anaconda3/lib/python3.9/site-packages (from requests->
transformers) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /home/administrator/
anaconda3/lib/python3.9/site-packages (from requests->transformers)
(3.2)
```

In [26]:
```
import torch
from transformers.file_utils import is_tf_available, is_torch_availab
from transformers import BertTokenizerFast, BertForSequenceClassifica
from transformers import Trainer, TrainingArguments
from sklearn.model_selection import train_test_split
import random
```

In [27]:
```
import tensorflow as tf
with tf.device('GPU:1'):
    def set_seed(seed: int):
        """
        Helper function for reproducible behavior to set the seed in
        installed).

        Args:
            seed (:obj:`int`): The seed to set.
        """
        random.seed(seed)
        np.random.seed(seed)
        if is_torch_available():
            torch.manual_seed(seed)
            torch.cuda.manual_seed_all(seed)
            # ^^ safe to call this function even if cuda is not avail
        if is_tf_available():
            import tensorflow as tf

            tf.random.set_seed(seed)

    set_seed(123)
```

```
2022-05-16 14:38:42.403950: I tensorflow/core/platform/cpu_feature_
guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep
Neural Network Library (oneDNN) to use the following CPU instructio
ns in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the app
ropriate compiler flags.
2022-05-16 14:38:43.367928: I tensorflow/core/common_runtime/gpu/gp
u_device.cc:1532] Created device /job:localhost/replica:0/task:0/de
vice:GPU:0 with 14637 MB memory:  -> device: 0, name: Tesla V100-PC
IE-16GB, pci bus id: 0000:3b:00.0, compute capability: 7.0
2022-05-16 14:38:43.368594: I tensorflow/core/common_runtime/gpu/gp
u_device.cc:1532] Created device /job:localhost/replica:0/task:0/de
vice:GPU:1 with 14637 MB memory:  -> device: 1, name: Tesla V100-PC
IE-16GB, pci bus id: 0000:d8:00.0, compute capability: 7.0
```

In [28]:
```
with tf.device('GPU:1'):
    model_name = "bert-base-uncased"
    max_length= 512
```

In [29]:
```
with tf.device('GPU:1'):
    tokenizer = BertTokenizerFast.from_pretrained(model_name, do_lowe
```

In [30]:
```
data.head()
```

Out[30]:

	title	text	subject	date	target
0	Republican ex-Treasury chief Paulson slams Tru...	WASHINGTON (Reuters) - Henry Paulson, a Republ...	politicsNews	June 25, 2016	0
1	SHOCK POLL In MUST WIN State Of FLORIDA: Hispa...	Apparently the Black Lives Matter terror group...	left-news	Jul 11, 2016	1
2	MEDALS OF VALOR: President Trump Honored Agent...	It s great to have a president who appreciates...	politics	Jul 27, 2017	1
3	Newsweek Just Made Their BEST Cover Ever And ...	Newsweek has never been a publication to shy a...	News	November 9, 2017	1
4	Trump says he believes Cuba responsible for at...	WASHINGTON (Reuters) - President Donald Trump ...	politicsNews	October 16, 2017	0

In [31]:
```
with tf.device('GPU:1'):
    ## Data Preparation
    data = data[data['text'].notna()]
```

In [33]:
```
with tf.device('GPU:1'):
    def prepare_data(df, test_size=0.2, include_title=True, include_a
        texts = []
        labels = []

        for i in range(len(df)):
            text = df['text'].iloc[i]
            label = df['target'].iloc[i]

            if text and label in [0,1]:
                texts.append(text)
                labels.append(label)

        return train_test_split(texts, labels, test_size=test_size)

    train_texts, valid_texts, train_labels, valid_labels = prepare_da
```

In [34]:
```
print(len(train_texts), len(train_labels))
print(len(valid_texts), len(valid_labels))
```

```
35918 35918
8980 8980
```

In [35]:
```
with tf.device('GPU:1'):
    # tokenizing the dataset
    train_encodings = tokenizer(train_texts, truncation=True, padding
    valid_encodings = tokenizer(valid_texts, truncation=True, padding
```

In [36]:
```
with tf.device('GPU:1'):
    # converting the encoding into a PyTorch datset
    class NewsGroupsDataset(torch.utils.data.Dataset):
        def __init__(self, encodings, labels):
            self.encodings = encodings
            self.labels = labels

        def __getitem__(self, idx):
            item = {k: torch.tensor(v[idx]) for k, v in self.encoding
            item['labels'] = torch.tensor([self.labels[idx]])
            return item

        def __len__(self):
            return len(self.labels)

    # convert tokenize data into torch dataset
    train_dataset = NewsGroupsDataset(train_encodings, train_labels)
    valid_dataset = NewsGroupsDataset(valid_encodings, valid_labels)
```

In [37]:
```
with tf.device('GPU:1'):
    model = BertForSequenceClassification.from_pretrained(model_name,
Some weights of the model checkpoint at bert-base-uncased were not
used when initializing BertForSequenceClassification: ['cls.seq_rel
ationship.bias', 'cls.predictions.transform.dense.bias', 'cls.predi
ctions.decoder.weight', 'cls.predictions.transform.LayerNorm.bias',
'cls.seq_relationship.weight', 'cls.predictions.transform.dense.wei
ght', 'cls.predictions.bias', 'cls.predictions.transform.LayerNorm.
weight']
- This IS expected if you are initializing BertForSequenceClassific
ation from the checkpoint of a model trained on another task or wit
```

In [38]:
```
with tf.device('GPU:1'):
    from sklearn.metrics import precision_recall_fscore_support
    from sklearn.metrics import accuracy_score
    def computer_metrics(pred):
        labels = pred.label_ids
        preds = pred.predictions.argmax(-1)
        precision, recall, f1, _ = precision_recall_fscore_support(la
        acc = accuracy_score(labels, preds)
        return {
            'accuracy': acc,
            'f1': f1,
            'precision': precision,
            'recall': recall
        }
```

In [39]:
```
with tf.device('GPU:1'):
    training_args = TrainingArguments(
        output_dir='./results',          # output directory
        num_train_epochs=1,              # total number of training e
        per_device_train_batch_size=10,  # batch size per device duri
        per_device_eval_batch_size=20,   # batch size for evaluation
        warmup_steps=100,                # number of warmup steps for
        logging_dir='./logs',            # directory for storing logs
        load_best_model_at_end=True,     # load the best model when t
        # but you can specify `metric_for_best_model` argument to cha
        logging_steps=200,               # log & save weights each lo
        save_steps=200,
        evaluation_strategy="steps",     # evaluate each `logging_ste
    )
```

In [40]:
```
with tf.device('GPU:1'):
    trainer = Trainer(
            model = model,
            args = training_args,
            train_dataset=train_dataset,
            eval_dataset=valid_dataset,
            compute_metrics=computer_metrics,
        )
```

In [41]:
```
with tf.device('GPU:1'):
    trainer.train()
```

```
/home/administrator/anaconda3/lib/python3.9/site-packages/transform
ers/optimization.py:306: FutureWarning: This implementation of Adam
W is deprecated and will be removed in a future version. Use the Py
Torch implementation torch.optim.AdamW instead, or set `no_deprecat
ion_warning=True` to disable this warning
  warnings.warn(
***** Running training *****
  Num examples = 35918
  Num Epochs = 1
  Instantaneous batch size per device = 10
  Total train batch size (w. parallel, distributed & accumulation)
```

In [46]:

```
with tf.device('GPU:1'):
    # evaluate the current model after training
    trainer.evaluate()
```

```
***** Running Evaluation *****
  Num examples = 8980
  Batch size = 20
/home/administrator/anaconda3/lib/python3.9/site-packages/torch/nn/
parallel/_functions.py:68: UserWarning: Was asked to gather along d
imension 0, but all input tensors were scalars; will instead unsque
eze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '

Attempted to log scalar metric eval_loss:
9.067665814654902e-05
Attempted to log scalar metric eval_accuracy:
1.0
Attempted to log scalar metric eval_f1:
1.0
Attempted to log scalar metric eval_precision:
1.0
Attempted to log scalar metric eval_recall:
1.0
Attempted to log scalar metric eval_runtime:
56.0933
Attempted to log scalar metric eval_samples_per_second:
160.09
Attempted to log scalar metric eval_steps_per_second:
4.011
Attempted to log scalar metric epoch:
1.0
```

In [43]:

```
with tf.device('GPU:1'):
# saving the fine tuned model & tokenizer
    model_path = "fake-news-bert-base-uncased"
    model.save_pretrained(model_path)
    tokenizer.save_pretrained(model_path)
```

```
Configuration saved in fake-news-bert-base-uncased/config.json
Model weights saved in fake-news-bert-base-uncased/pytorch_model.bi
n
tokenizer config file saved in fake-news-bert-base-uncased/tokenize
r_config.json
Special tokens file saved in fake-news-bert-base-uncased/special_to
kens_map.json
```

In [44]:

```
def get_prediction(text, convert_to_label=False):
    # prepare our text into tokenized sequence
    inputs = tokenizer(text, padding=True, truncation=True, max_lengt
    # perform inference to our model
    outputs = model(**inputs)
    # get output probabilities by doing softmax
    probs = outputs[0].softmax(1)
    # executing argmax function to get the candidate label
    d = {
        0: "reliable",
        1: "fake"
    }
    if convert_to_label:
        return d[int(probs.argmax())]
    else:
        return int(probs.argmax())
```

In [45]:

```
real_news = """
 Donald Trump Sends Out Embarrassing New Year’s Eve Message; This is
"""

get_prediction(real_news, convert_to_label=True)
```

Out[45]:

```
'fake'
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: