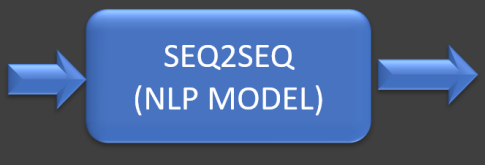
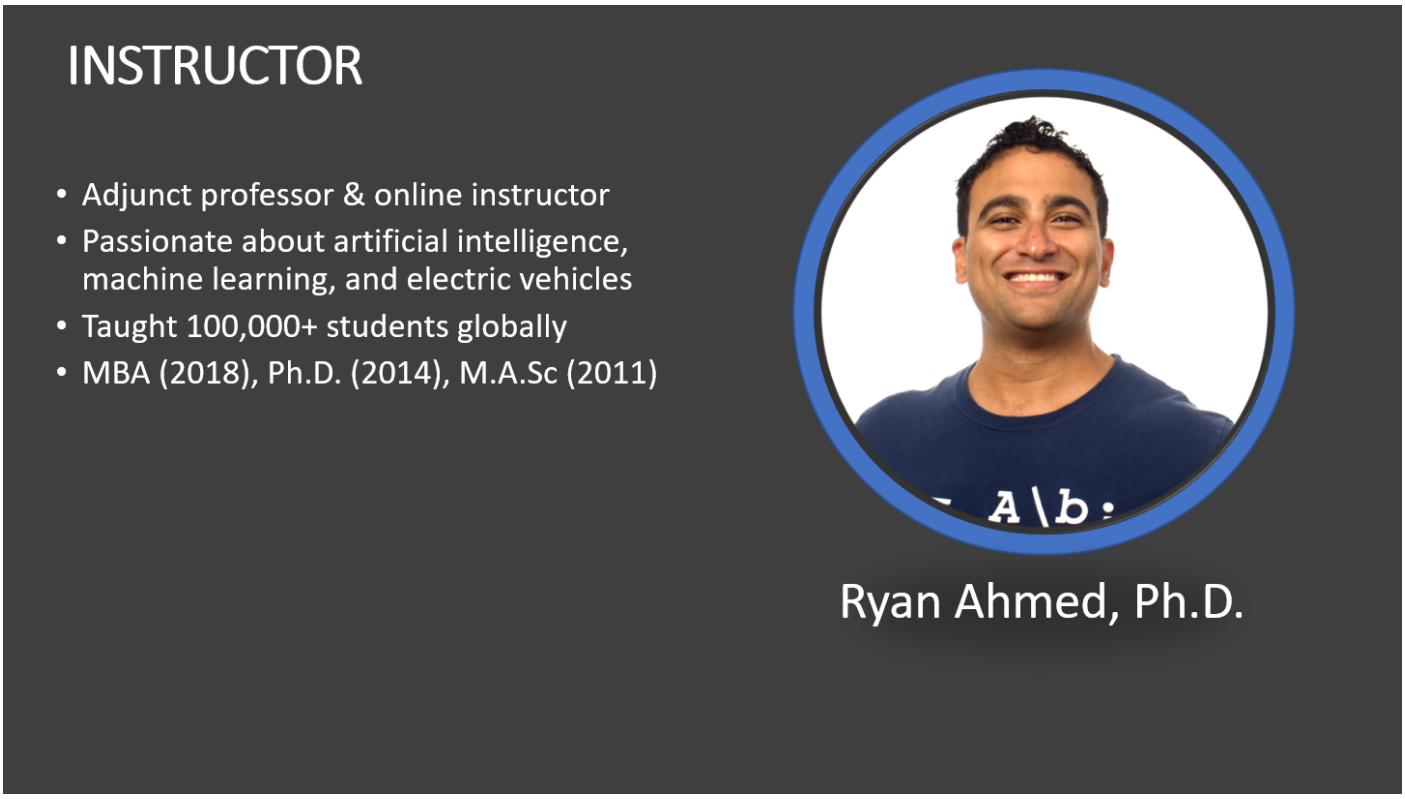
TASK #1: UNDERSTAND THE PROBLEM STATMENT AND BUSINESS CASE

ENGLISH TO FRENCH TRANSLATION:

- Language translation is a key service that is needed by people who are travelling as well as for people who are settling in a new country.
- In this hands-on project, we will build a seq2seq model that translates English sentence to French sentence.
- Natural language processors (NLP) work by converting words (text) into numbers.
- These numbers are then used to train an AI/ML model to make predictions.
- AI/ML-based Neural Machine Translation has been used by Google Translate, which has been used by more than 1 billion users.





TASK #2: IMPORT LIBRARIES AND DATASETS

In [1]:
```
!pip install --upgrade tensorflow-gpu==2.0
```

```
Requirement already up-to-date: tensorflow-gpu==2.0 in c:\users\administrator
\anaconda31\lib\site-packages (2.0.0)
Requirement already satisfied, skipping upgrade: tensorboard<2.1.0,>=2.0.0 in
c:\users\administrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.
0) (2.0.2)
Requirement already satisfied, skipping upgrade: gast==0.2.2 in c:\users\admi
nistrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (0.2.2)
Requirement already satisfied, skipping upgrade: protobuf>=3.6.1 in c:\users
\administrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (3.11.
3)
Requirement already satisfied, skipping upgrade: wrapt>=1.11.1 in c:\users\ad
ministrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (1.11.2)
Requirement already satisfied, skipping upgrade: grpcio>=1.8.6 in c:\users\ad
ministrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (1.28.1)
Requirement already satisfied, skipping upgrade: astor>=0.6.0 in c:\users\adm
inistrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (0.8.1)
Requirement already satisfied, skipping upgrade: numpy<2.0,>=1.16.0 in c:\use
rs\administrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (1.1
8.1)
Requirement already satisfied, skipping upgrade: google-pasta>=0.1.6 in c:\us
ers\administrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (0.
2.0)
Requirement already satisfied, skipping upgrade: absl-py>=0.7.0 in c:\users\a
dministrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (0.9.0)
Requirement already satisfied, skipping upgrade: keras-applications>=1.0.8 in
c:\users\administrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.
0) (1.0.8)
Requirement already satisfied, skipping upgrade: opt-einsum>=2.3.2 in c:\user
s\administrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (3.2.
1)
Requirement already satisfied, skipping upgrade: termcolor>=1.1.0 in c:\users
\administrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (1.1.
0)
Requirement already satisfied, skipping upgrade: wheel>=0.26 in c:\users\admi
nistrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (0.34.2)
Requirement already satisfied, skipping upgrade: six>=1.10.0 in c:\users\admi
nistrator\anaconda31\lib\site-packages (from tensorflow-gpu==2.0) (1.14.0)
Requirement already satisfied, skipping upgrade: keras-preprocessing>=1.0.5 i
n c:\users\administrator\anaconda31\lib\site-packages (from tensorflow-gpu==
2.0) (1.1.2)
Requirement already satisfied, skipping upgrade: tensorflow-estimator<2.1.0,>
=2.0.0 in c:\users\administrator\anaconda31\lib\site-packages (from tensorflo
w-gpu==2.0) (2.0.1)
Requirement already satisfied, skipping upgrade: requests<3,>=2.21.0 in c:\us
ers\administrator\anaconda31\lib\site-packages (from tensorboard<2.1.0,>=2.0.
0->tensorflow-gpu==2.0) (2.22.0)
Requirement already satisfied, skipping upgrade: setuptools>=41.0.0 in c:\use
rs\administrator\anaconda31\lib\site-packages (from tensorboard<2.1.0,>=2.0.0
->tensorflow-gpu==2.0) (45.2.0.post20200210)
Requirement already satisfied, skipping upgrade: werkzeug>=0.11.15 in c:\user
s\administrator\anaconda31\lib\site-packages (from tensorboard<2.1.0,>=2.0.0-
>tensorflow-gpu==2.0) (1.0.0)
Requirement already satisfied, skipping upgrade: google-auth-oauthlib<0.5,>=
0.4.1 in c:\users\administrator\anaconda31\lib\site-packages (from tensorboar
d<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (0.4.1)
Requirement already satisfied, skipping upgrade: markdown>=2.6.8 in c:\users
\administrator\anaconda31\lib\site-packages (from tensorboard<2.1.0,>=2.0.0->
```

```
tensorflow-gpu==2.0) (3.2.1)
Requirement already satisfied, skipping upgrade: google-auth<2,>=1.6.3 in
c:\users\administrator\anaconda31\lib\site-packages (from tensorboard<2.1.0,>
=2.0.0->tensorflow-gpu==2.0) (1.14.1)
Requirement already satisfied, skipping upgrade: h5py in c:\users\administrat
or\anaconda31\lib\site-packages (from keras-applications>=1.0.8->tensorflow-g
pu==2.0) (2.10.0)
Requirement already satisfied, skipping upgrade: chardet<3.1.0,>=3.0.2 in
c:\users\administrator\anaconda31\lib\site-packages (from requests<3,>=2.21.0
->tensorboard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (3.0.4)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in c:\use
rs\administrator\anaconda31\lib\site-packages (from requests<3,>=2.21.0->tens
orboard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (2019.11.28)
Requirement already satisfied, skipping upgrade: urllib3!=1.25.0,!=1.25.1,<1.
26,>=1.21.1 in c:\users\administrator\anaconda31\lib\site-packages (from requ
ests<3,>=2.21.0->tensorboard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (1.25.8)
Requirement already satisfied, skipping upgrade: idna<2.9,>=2.5 in c:\users\a
dministrator\anaconda31\lib\site-packages (from requests<3,>=2.21.0->tensorbo
ard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (2.8)
Requirement already satisfied, skipping upgrade: requests-oauthlib>=0.7.0 in
c:\users\administrator\anaconda31\lib\site-packages (from google-auth-oauthli
b<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (1.3.0)
Requirement already satisfied, skipping upgrade: rsa<4.1,>=3.1.4 in c:\users
\administrator\anaconda31\lib\site-packages (from google-auth<2,>=1.6.3->tens
orboard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (4.0)
Requirement already satisfied, skipping upgrade: pyasn1-modules>=0.2.1 in
c:\users\administrator\anaconda31\lib\site-packages (from google-auth<2,>=1.
6.3->tensorboard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (0.2.8)
Requirement already satisfied, skipping upgrade: cachetools<5.0,>=2.0.0 in
c:\users\administrator\anaconda31\lib\site-packages (from google-auth<2,>=1.
6.3->tensorboard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (4.1.0)
Requirement already satisfied, skipping upgrade: oauthlib>=3.0.0 in c:\users
\administrator\anaconda31\lib\site-packages (from requests-oauthlib>=0.7.0->g
oogle-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.1.0,>=2.0.0->tensorflow-gpu==
2.0) (3.1.0)
Requirement already satisfied, skipping upgrade: pyasn1>=0.1.3 in c:\users\ad
ministrator\anaconda31\lib\site-packages (from rsa<4.1,>=3.1.4->google-auth<
2,>=1.6.3->tensorboard<2.1.0,>=2.0.0->tensorflow-gpu==2.0) (0.4.8)
```

In [2]:
```
# install nltk
!pip install nltk
# install gensim
!pip install gensim
# install spacy
!pip install spacy
!pip install plotly
```

```
Requirement already satisfied: nltk in c:\users\administrator\anaconda31\lib
\site-packages (3.4.5)
Requirement already satisfied: six in c:\users\administrator\anaconda31\lib\s
ite-packages (from nltk) (1.14.0)
Requirement already satisfied: gensim in c:\users\administrator\anaconda31\li
b\site-packages (3.8.3)
Requirement already satisfied: scipy>=0.18.1 in c:\users\administrator\anacon
da31\lib\site-packages (from gensim) (1.4.1)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\administrator\an
aconda31\lib\site-packages (from gensim) (2.1.0)
Requirement already satisfied: Cython==0.29.14 in c:\users\administrator\anac
onda31\lib\site-packages (from gensim) (0.29.14)
Requirement already satisfied: six>=1.5.0 in c:\users\administrator\anaconda3
1\lib\site-packages (from gensim) (1.14.0)
Requirement already satisfied: numpy>=1.11.3 in c:\users\administrator\anacon
da31\lib\site-packages (from gensim) (1.18.1)
Requirement already satisfied: boto in c:\users\administrator\anaconda31\lib
\site-packages (from smart-open>=1.8.1->gensim) (2.49.0)
Requirement already satisfied: requests in c:\users\administrator\anaconda31
\lib\site-packages (from smart-open>=1.8.1->gensim) (2.22.0)
Requirement already satisfied: boto3 in c:\users\administrator\anaconda31\lib
\site-packages (from smart-open>=1.8.1->gensim) (1.14.30)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\administrator\anaco
nda31\lib\site-packages (from requests->smart-open>=1.8.1->gensim) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\administrator\a
naconda31\lib\site-packages (from requests->smart-open>=1.8.1->gensim) (2019.
11.28)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\administrator\anaconda31\lib\site-packages (from requests->smart-ope
n>=1.8.1->gensim) (1.25.8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\administrato
r\anaconda31\lib\site-packages (from requests->smart-open>=1.8.1->gensim) (3.
0.4)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in c:\users\administrat
or\anaconda31\lib\site-packages (from boto3->smart-open>=1.8.1->gensim) (0.1
0.0)
Requirement already satisfied: s3transfer<0.4.0,>=0.3.0 in c:\users\administr
ator\anaconda31\lib\site-packages (from boto3->smart-open>=1.8.1->gensim) (0.
3.3)
Requirement already satisfied: botocore<1.18.0,>=1.17.30 in c:\users\administ
rator\anaconda31\lib\site-packages (from boto3->smart-open>=1.8.1->gensim)
(1.17.30)
Requirement already satisfied: docutils<0.16,>=0.10 in c:\users\administrator
\anaconda31\lib\site-packages (from botocore<1.18.0,>=1.17.30->boto3->smart-o
pen>=1.8.1->gensim) (0.15.2)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in c:\users\admini
strator\anaconda31\lib\site-packages (from botocore<1.18.0,>=1.17.30->boto3->
smart-open>=1.8.1->gensim) (2.8.1)
Requirement already satisfied: spacy in c:\users\administrator\anaconda31\lib
\site-packages (2.3.2)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in c:\users\administrato
r\anaconda31\lib\site-packages (from spacy) (3.0.2)
Requirement already satisfied: numpy>=1.15.0 in c:\users\administrator\anacon
da31\lib\site-packages (from spacy) (1.18.1)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in c:\users\administrator
\anaconda31\lib\site-packages (from spacy) (0.7.1)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in c:\users\administrator
```

```
\anaconda31\lib\site-packages (from spacy) (4.42.1)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in c:\users\administrator
\anaconda31\lib\site-packages (from spacy) (2.0.3)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in c:\users\administrator\a
naconda31\lib\site-packages (from spacy) (0.4.1)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in c:\users\administrator
\anaconda31\lib\site-packages (from spacy) (1.0.2)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in c:\users\administra
tor\anaconda31\lib\site-packages (from spacy) (1.0.0)
Requirement already satisfied: setuptools in c:\users\administrator\anaconda3
1\lib\site-packages (from spacy) (45.2.0.post20200210)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in c:\users\administra
tor\anaconda31\lib\site-packages (from spacy) (2.22.0)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in c:\users\administrator\a
naconda31\lib\site-packages (from spacy) (1.1.3)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in c:\users\administ
rator\anaconda31\lib\site-packages (from spacy) (1.0.2)
Requirement already satisfied: thinc==7.4.1 in c:\users\administrator\anacond
a31\lib\site-packages (from spacy) (7.4.1)
Requirement already satisfied: importlib-metadata>=0.20; python_version < "3.
8" in c:\users\administrator\anaconda31\lib\site-packages (from catalogue<1.
1.0,>=0.0.7->spacy) (1.5.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
c:\users\administrator\anaconda31\lib\site-packages (from requests<3.0.0,>=2.
13.0->spacy) (1.25.8)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\administrator\a
naconda31\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2019.11.2
8)
Requirement already satisfied: idna<2.9,>=2.5 in c:\users\administrator\anaco
nda31\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2.8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\administrato
r\anaconda31\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (3.0.4)
Requirement already satisfied: zipp>=0.5 in c:\users\administrator\anaconda31
\lib\site-packages (from importlib-metadata>=0.20; python_version < "3.8"->ca
talogue<1.1.0,>=0.0.7->spacy) (2.2.0)
Requirement already satisfied: plotly in c:\users\administrator\anaconda31\li
b\site-packages (4.9.0)
Requirement already satisfied: six in c:\users\administrator\anaconda31\lib\s
ite-packages (from plotly) (1.14.0)
Requirement already satisfied: retrying>=1.3.3 in c:\users\administrator\anac
onda31\lib\site-packages (from plotly) (1.3.3)
```

In [3]:
```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\administrator\anaconda31
\lib\site-packages (1.7.0)
Requirement already satisfied: matplotlib in c:\users\administrator\anaconda3
1\lib\site-packages (from wordcloud) (3.1.3)
Requirement already satisfied: numpy>=1.6.1 in c:\users\administrator\anacond
a31\lib\site-packages (from wordcloud) (1.18.1)
Requirement already satisfied: pillow in c:\users\administrator\anaconda31\li
b\site-packages (from wordcloud) (7.0.0)
Requirement already satisfied: cycler>=0.10 in c:\users\administrator\anacond
a31\lib\site-packages (from matplotlib->wordcloud) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\administrator\anaconda31\lib\site-packages (from matplotlib->wordclo
ud) (2.4.6)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\administrator
\anaconda31\lib\site-packages (from matplotlib->wordcloud) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\administrator\an
aconda31\lib\site-packages (from matplotlib->wordcloud) (1.1.0)
Requirement already satisfied: six in c:\users\administrator\anaconda31\lib\s
ite-packages (from cycler>=0.10->matplotlib->wordcloud) (1.14.0)
Requirement already satisfied: setuptools in c:\users\administrator\anaconda3
1\lib\site-packages (from kiwisolver>=1.0.1->matplotlib->wordcloud) (45.2.0.p
ost20200210)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Administrator\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[4]:
```
True
```

In [5]:
```
from collections import Counter
import operator
import plotly.express as px
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
import nltk
import re
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
import gensim
import tensorflow as tf
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from tensorflow.keras.preprocessing.text import one_hot, Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, TimeDistributed, RepeatVec
tor, Embedding, Input, LSTM, Conv1D, MaxPool1D, Bidirectional
from tensorflow.keras.models import Model
# from jupyterthemes import jtplot
# jtplot.style(theme='monokai', context='notebook', ticks=True, grid=False)
# setting the style of the notebook to be monokai theme
# this line of code is important to ensure that we are able to see the x and y
axes clearly
# If you don't run this code line, you will notice that the xlabel and ylabel
 on any plot is black on black and it will be hard to see them.
```

In [6]:
```
# load the data
df_english = pd.read_csv('small_vocab_en.csv', sep = '/t', names = ['english'
])
df_french = pd.read_csv('small_vocab_fr.csv', sep = '/t', names = ['french'])
```

```
C:\Users\Administrator\anaconda31\lib\site-packages\ipykernel_launcher.py:2:
ParserWarning: Falling back to the 'python' engine because the 'c' engine doe
s not support regex separators (separators > 1 char and different from '\s+'
are interpreted as regex); you can avoid this warning by specifying engine='p
ython'.

C:\Users\Administrator\anaconda31\lib\site-packages\ipykernel_launcher.py:3:
ParserWarning: Falling back to the 'python' engine because the 'c' engine doe
s not support regex separators (separators > 1 char and different from '\s+'
are interpreted as regex); you can avoid this warning by specifying engine='p
ython'.
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

MINI CHALLENGE #1:

- Explore the 'english' and 'french' data and indicate how many samples are included.
- Do we have Null elements? What are the memory usage for both dataframes?

In [7]:
```
df_english
```

Out[7]:

	english
0	new jersey is sometimes quiet during autumn , ...
1	the united states is usually chilly during jul...
2	california is usually quiet during march , and...
3	the united states is sometimes mild during jun...
4	your least liked fruit is the grape , but my l...
...	...
137855	france is never busy during march , and it is ...
137856	india is sometimes beautiful during spring , a...
137857	india is never wet during summer , but it is s...
137858	france is never chilly during january , but it...
137859	the orange is her favorite fruit , but the ban...

137860 rows × 1 columns

In [8]:
```
df_french
```

Out[8]:

	french
0	new jersey est parfois calme pendant l' automn...
1	les Ã©tats-unis est gÃ©nÃ©ralement froid en ju...
2	california est gÃ©nÃ©ralement calme en mars , ...
3	les Ã©tats-unis est parfois lÃ©gÃ¨re en juin ,...
4	votre moins aimÃ© fruit est le raisin , mais m...
...	...
137855	la france est jamais occupÃ©e en mars , et il ...
137856	l' inde est parfois belle au printemps , et il...
137857	l' inde est jamais mouillÃ© pendant l' Ã©tÃ© ,...
137858	la france est jamais froid en janvier , mais i...
137859	l'orange est son fruit prÃ©fÃ©rÃ© , mais la ba...

137860 rows × 1 columns

In [9]:
```
df_english.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 137860 entries, 0 to 137859
Data columns (total 1 columns):
 #   Column   Non-Null Count   Dtype 
---  ------   --------------   ----- 
 0   english  137860 non-null  object
dtypes: object(1)
memory usage: 1.1+ MB
```

In [10]:
```
df_french.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 137860 entries, 0 to 137859
Data columns (total 1 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   french  137860 non-null  object
dtypes: object(1)
memory usage: 1.1+ MB
```

MINI CHALLENGE #2:

- Concatenate both dataframes and indicate how many records are present
- Print out the following: "Total English Records = xx, Total French Records = xx"

In [11]:
```
#concatenates both dataframes column wise
df = pd.concat([df_english, df_french], axis=1)
```

In [12]:
```
df
```

Out[12]:

	english	french
0	new jersey is sometimes quiet during autumn , ...	new jersey est parfois calme pendant l' automn...
1	the united states is usually chilly during jul...	les Ã©tats-unis est gÃ©nÃ©ralement froid en ju...
2	california is usually quiet during march , and...	california est gÃ©nÃ©ralement calme en mars , ...
3	the united states is sometimes mild during jun...	les Ã©tats-unis est parfois lÃ©gÃ¨re en juin ,...
4	your least liked fruit is the grape , but my l...	votre moins aimÃ© fruit est le raisin , mais m...
...	...	...
137855	france is never busy during march , and it is ...	la france est jamais occupÃ©e en mars , et il ...
137856	india is sometimes beautiful during spring , a...	l' inde est parfois belle au printemps , et il...
137857	india is never wet during summer , but it is s...	l' inde est jamais mouillÃ© pendant l' Ã©tÃ© ,...
137858	france is never chilly during january , but it...	la france est jamais froid en janvier , mais i...
137859	the orange is her favorite fruit , but the ban...	l'orange est son fruit prÃ©fÃ©rÃ© , mais la ba...

137860 rows × 2 columns

In [13]:
```
print(f"Total English records {len(df['english'])}")
print(f"Total French records {len(df['french'])}")
```

```
Total English records 137860
Total French records 137860
```

TASK #3: PERFORM DATA CLEANING

In [14]:
```
# download nltk packages
nltk.download('punkt')

# download stopwords
nltk.download("stopwords")
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Administrator\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Administrator\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[14]: True

In [15]:
```
# function to remove punctuations
#regular expression that substitutes all the punctuations with an empty string
def remove_punc(x):
  return re.sub('[!#?,.:";]', '', x)
```

In [16]:
```
df['french'] = df['french'].apply(remove_punc)
df['english'] = df['english'].apply(remove_punc)
#this is how you call a function for a pandas dataframe
```

In [17]:
```
english_words = []
french_words  = []
```

MINI CHALLENGE #3:

- How many unique words are available in the english and french dictionairies?

In [18]:
```
#puts all unique words into an array
def get_unique_words(x, word_list):
    for word in x.split(): #splits the words into an array so it can be looped
over
        if word not in word_list:
            word_list.append(word)

df['english'].apply(lambda x: get_unique_words(x, english_words))
len(english_words)
```

Out[18]: 199

In [19]:
```
# number of unique words in french
df['french'].apply(lambda x: get_unique_words(x, french_words))
len(french_words)
```

Out[19]: 350

TASK #4: VISUALIZE CLEANED UP DATASET

In [20]:
```
# Obtain list of all words in the dataset
words = []
for i in df['english']:
  for word in i.split():
    words.append(word)

words
```

Out[20]:
```
['new',
 'jersey',
 'is',
 'sometimes',
 'quiet',
 'during',
 'autumn',
 'and',
 'it',
 'is',
 'snowy',
 'in',
 'april',
 'the',
 'united',
 'states',
 'is',
 'usually',
 'chilly',
 'during',
 'july',
 'and',
 'it',
 'is',
 'usually',
 'freezing',
 'in',
 'november',
 'california',
 'is',
 'usually',
 'quiet',
 'during',
 'march',
 'and',
 'it',
 'is',
 'usually',
 'hot',
 'in',
 'june',
 'the',
 'united',
 'states',
 'is',
 'sometimes',
 'mild',
 'during',
 'june',
 'and',
 'it',
 'is',
 'cold',
 'in',
 'september',
 'your',
 'least',
```

```
'liked',
'fruit',
'is',
'the',
'grape',
'but',
'my',
'least',
'liked',
'is',
'the',
'apple',
'his',
'favorite',
'fruit',
'is',
'the',
'orange',
'but',
'my',
'favorite',
'is',
'the',
'grape',
'paris',
'is',
'relaxing',
'during',
'december',
'but',
'it',
'is',
'usually',
'chilly',
'in',
'july',
'new',
'jersey',
'is',
'busy',
'during',
'spring',
'and',
'it',
'is',
'never',
'hot',
'in',
'march',
'our',
'least',
'liked',
'fruit',
'is',
'the',
'lemon',
'but',
```

```
'my',
'least',
'liked',
'is',
'the',
'grape',
'the',
'united',
'states',
'is',
'sometimes',
'busy',
'during',
'january',
'and',
'it',
'is',
'sometimes',
'warm',
'in',
'november',
'the',
'lime',
'is',
'her',
'least',
'liked',
'fruit',
'but',
'the',
'banana',
'is',
'my',
'least',
'liked',
'he',
'saw',
'a',
'old',
'yellow',
'truck',
'india',
'is',
'rainy',
'during',
'june',
'and',
'it',
'is',
'sometimes',
'warm',
'in',
'november',
'that',
'cat',
'was',
'my',
```

```
'most',
'loved',
'animal',
'he',
'dislikes',
'grapefruit',
'limes',
'and',
'lemons',
'her',
'least',
'liked',
'fruit',
'is',
'the',
'lemon',
'but',
'his',
'least',
'liked',
'is',
'the',
'grapefruit',
'california',
'is',
'never',
'cold',
'during',
'february',
'but',
'it',
'is',
'sometimes',
'freezing',
'in',
'june',
'china',
'is',
'usually',
'pleasant',
'during',
'autumn',
'and',
'it',
'is',
'usually',
'quiet',
'in',
'october',
'paris',
'is',
'never',
'freezing',
'during',
'november',
'but',
'it',
```

```
                'is',
                'wonderful',
                'in',
                'october',
                'the',
                'united',
                'states',
                'is',
                'never',
                'rainy',
                'during',
                'january',
                'but',
                'it',
                'is',
                'sometimes',
                'mild',
                'in',
                'october',
                'china',
                'is',
                'usually',
                'pleasant',
                'during',
                'november',
                'and',
                'it',
                'is',
                'never',
                'quiet',
                'in',
                'october',
                'the',
                'united',
                'states',
                'is',
                'never',
                'nice',
                'during',
                'february',
                'but',
                'it',
                'is',
                'sometimes',
                'pleasant',
                'in',
                'april',
                'india',
                'is',
                'never',
                'busy',
                'during',
                'autumn',
                'and',
                'it',
                'is',
                'mild',
```

```
'in',
'spring',
'paris',
'is',
'mild',
'during',
'summer',
'but',
'it',
'is',
'usually',
'busy',
'in',
'april',
'france',
'is',
'never',
'cold',
'during',
'september',
'and',
'it',
'is',
'snowy',
'in',
'october',
'california',
'is',
'never',
'cold',
'during',
'may',
'and',
'it',
'is',
'sometimes',
'chilly',
'in',
'march',
'he',
'dislikes',
'lemons',
'grapes',
'and',
'mangoes',
'their',
'favorite',
'fruit',
'is',
'the',
'mango',
'but',
'our',
'favorite',
'is',
'the',
'pear',
```

```
'france',
'is',
'sometimes',
'quiet',
'during',
'may',
'and',
'it',
'is',
'never',
'chilly',
'in',
'august',
'paris',
'is',
'never',
'pleasant',
'during',
'september',
'and',
'it',
'is',
'beautiful',
'in',
'autumn',
'he',
'dislikes',
'apples',
'peaches',
'and',
'grapes',
'california',
'is',
'usually',
'freezing',
'during',
'december',
'and',
'it',
'is',
'busy',
'in',
'april',
'your',
'most',
'feared',
'animal',
'is',
'that',
'shark',
'paris',
'is',
'usually',
'wet',
'during',
'august',
'and',
```

```
'it',
'is',
'never',
'dry',
'in',
'november',
'paris',
'is',
'usually',
'beautiful',
'during',
'september',
'and',
'it',
'is',
'usually',
'snowy',
'in',
'november',
'the',
'united',
'states',
'is',
'never',
'wet',
'during',
'january',
'but',
'it',
'is',
'usually',
'hot',
'in',
'october',
'we',
'like',
'oranges',
'mangoes',
'and',
'grapes',
'they',
'like',
'pears',
'apples',
'and',
'mangoes',
'she',
'dislikes',
'that',
'little',
'red',
'truck',
'the',
'grapefruit',
'is',
'my',
'most',
```

```
'loved',
'fruit',
'but',
'the',
'banana',
'is',
'her',
'most',
'loved',
'france',
'is',
'snowy',
'during',
'may',
'and',
'it',
'is',
'never',
'busy',
'in',
'autumn',
'china',
'is',
'usually',
'mild',
'during',
'winter',
'but',
'it',
'is',
'never',
'busy',
'in',
'february',
'china',
'is',
'never',
'nice',
'during',
'july',
'but',
'it',
'is',
'usually',
'snowy',
'in',
'spring',
'california',
'is',
'busy',
'during',
'november',
'but',
'it',
'is',
'rainy',
'in',
```

```
'autumn',
'china',
'is',
'warm',
'during',
'spring',
'and',
'it',
'is',
'sometimes',
'cold',
'in',
'february',
'california',
'is',
'usually',
'beautiful',
'during',
'winter',
'but',
'it',
'is',
'never',
'busy',
'in',
'february',
'france',
'is',
'wonderful',
'during',
'november',
'but',
'it',
'is',
'sometimes',
'hot',
'in',
'september',
'india',
'is',
'usually',
'pleasant',
'during',
'november',
'but',
'it',
'is',
'never',
'relaxing',
'in',
'july',
'the',
'united',
'states',
'is',
'never',
'freezing',
```

```
'during',
'autumn',
'but',
'it',
'is',
'never',
'busy',
'in',
'june',
'paris',
'is',
'sometimes',
'warm',
'during',
'june',
'but',
'it',
'is',
'usually',
'hot',
'in',
'july',
'paris',
'is',
'never',
'hot',
'during',
'summer',
'and',
'it',
'is',
'usually',
'mild',
'in',
'winter',
'she',
'disliked',
'a',
'rusty',
'yellow',
'car',
'france',
'is',
'usually',
'quiet',
'during',
'november',
'but',
'it',
'is',
'sometimes',
'warm',
'in',
'february',
'new',
'jersey',
'is',
```

```
'never',
'wet',
'during',
'november',
'and',
'it',
'is',
'mild',
'in',
'august',
'we',
'like',
'peaches',
'pears',
'and',
'strawberries',
'the',
'orange',
'is',
'her',
'least',
'liked',
'fruit',
'but',
'the',
'grapefruit',
'is',
'their',
'least',
'liked',
'china',
'is',
'never',
'rainy',
'during',
'november',
'and',
'it',
'is',
'quiet',
'in',
'january',
'china',
'is',
'relaxing',
'during',
'march',
'but',
'it',
'is',
'sometimes',
'snowy',
'in',
'september',
'paris',
'is',
'wonderful',
```

```
'during',
'march',
'but',
'it',
'is',
'usually',
'pleasant',
'in',
'june',
'new',
'jersey',
'is',
'chilly',
'during',
'autumn',
'and',
'it',
'is',
'sometimes',
'pleasant',
'in',
'spring',
'california',
'is',
'never',
'freezing',
'during',
'october',
'but',
'it',
'is',
'usually',
'quiet',
'in',
'june',
'new',
'jersey',
'is',
'freezing',
'during',
'winter',
'but',
'it',
'is',
'sometimes',
'wonderful',
'in',
'january',
'i',
'like',
'grapes',
'pears',
'and',
'strawberries',
'the',
'lemon',
'is',
```

```
'my',
'most',
'loved',
'fruit',
'but',
'the',
'strawberry',
'is',
'our',
'most',
'loved',
'china',
'is',
'usually',
'dry',
'during',
'march',
'but',
'it',
'is',
'nice',
'in',
'november',
'paris',
'is',
'pleasant',
'during',
'december',
'but',
'it',
'is',
'never',
'nice',
'in',
'november',
'china',
'is',
'freezing',
'during',
'july',
'but',
'it',
'is',
'relaxing',
'in',
'january',
'the',
'apple',
'is',
'our',
'least',
'favorite',
'fruit',
'but',
'the',
'orange',
'is',
```

```
'her',
'least',
'favorite',
'he',
'dislikes',
'grapes',
'grapefruit',
'and',
'bananas',
'he',
'dislikes',
'apples',
'mangoes',
'and',
'strawberries',
'china',
'is',
'hot',
'during',
'july',
'but',
'it',
'is',
'never',
'pleasant',
'in',
'january',
'india',
'is',
'usually',
'dry',
'during',
'april',
'and',
'it',
'is',
'freezing',
'in',
'february',
'she',
'is',
'going',
'to',
'the',
'united',
'states',
'next',
'summer',
'france',
'is',
'sometimes',
'rainy',
'during',
'february',
'and',
'it',
'is',
```

```
'usually',
'quiet',
'in',
'spring',
'i',
'plan',
'to',
'visit',
'california',
'next',
'may',
'california',
'is',
'never',
'wet',
'during',
'november',
'and',
'it',
'is',
'sometimes',
'pleasant',
'in',
'september',
'they',
'like',
'lemons',
'limes',
'and',
'grapefruit',
'the',
'united',
'states',
'is',
'never',
'beautiful',
'during',
'march',
'and',
'it',
'is',
'usually',
'relaxing',
'in',
'summer',
'elephants',
'were',
'his',
'most',
'feared',
'animals',
'the',
'strawberry',
'is',
'their',
'least',
'favorite',
```

```
'fruit',
'but',
'the',
'apple',
'is',
'our',
'least',
'favorite',
'they',
'are',
'going',
'to',
'france',
'next',
'june',
'he',
'likes',
'strawberries',
'oranges',
'and',
'limes',
'california',
'is',
'never',
'pleasant',
'during',
'winter',
'and',
'it',
'is',
'sometimes',
'wonderful',
'in',
'december',
'the',
'apple',
'is',
'our',
'least',
'favorite',
'fruit',
'but',
'the',
'mango',
'is',
'their',
'least',
'favorite',
'she',
'likes',
'strawberries',
'oranges',
'and',
'bananas',
'california',
'is',
'beautiful',
```

```
 'during',
 'january',
 'and',
 'it',
 'is',
 'pleasant',
 'in',
 'february',
 'they',
 'dislike',
 'grapes',
 'mangoes',
 'and',
 'limes',
 'she',
 'dislikes',
 'lemons',
 'grapes',
 'and',
 'oranges',
 'our',
 'least',
 'favorite',
 'fruit',
 'is',
 'the',
 'banana',
 'but',
 'your',
 'least',
 'favorite',
 ...]
```

In [21]:
```
# Obtain the total count of words
english_words_counts = Counter(words)
english_words_counts
```

Out[21]:
```
Counter({'new': 12197,
         'jersey': 11225,
         'is': 205858,
         'sometimes': 37746,
         'quiet': 8693,
         'during': 74933,
         'autumn': 9004,
         'and': 59850,
         'it': 75137,
         'snowy': 8898,
         'in': 75525,
         'april': 8954,
         'the': 67628,
         'united': 11270,
         'states': 11270,
         'usually': 37507,
         'chilly': 8770,
         'july': 8956,
         'freezing': 8928,
         'november': 8951,
         'california': 11250,
         'march': 9023,
         'hot': 8639,
         'june': 9133,
         'mild': 8743,
         'cold': 8878,
         'september': 8958,
         'your': 9734,
         'least': 27564,
         'liked': 14046,
         'fruit': 27192,
         'grape': 4848,
         'but': 63987,
         'my': 9700,
         'apple': 4848,
         'his': 9700,
         'favorite': 28332,
         'orange': 4848,
         'paris': 11334,
         'relaxing': 8696,
         'december': 8945,
         'busy': 8791,
         'spring': 9102,
         'never': 37500,
         'our': 8932,
         'lemon': 4848,
         'january': 9090,
         'warm': 8890,
         'lime': 4848,
         'her': 9700,
         'banana': 4848,
         'he': 10786,
         'saw': 648,
         'a': 1944,
         'old': 972,
         'yellow': 972,
         'truck': 1944,
```

```
'india': 11277,
'rainy': 8761,
'that': 2712,
'cat': 192,
'was': 1867,
'most': 14934,
'loved': 14166,
'animal': 2304,
'dislikes': 7314,
'grapefruit': 10692,
'limes': 5844,
'lemons': 5844,
'february': 8942,
'china': 10953,
'pleasant': 8916,
'october': 8910,
'wonderful': 8808,
'nice': 8984,
'summer': 8948,
'france': 11170,
'may': 8995,
'grapes': 5844,
'mangoes': 5844,
'their': 8932,
'mango': 4848,
'pear': 4848,
'august': 8789,
'beautiful': 8915,
'apples': 5844,
'peaches': 5844,
'feared': 768,
'shark': 192,
'wet': 8726,
'dry': 8794,
'we': 2532,
'like': 4588,
'oranges': 5844,
'they': 3222,
'pears': 5844,
'she': 10786,
'little': 1016,
'red': 972,
'winter': 9038,
'disliked': 648,
'rusty': 972,
'car': 1944,
'strawberries': 5844,
'i': 2664,
'strawberry': 4848,
'bananas': 5844,
'going': 666,
'to': 5166,
'next': 1666,
'plan': 714,
'visit': 1224,
'elephants': 64,
'were': 384,
```

```
'animals': 768,
'are': 870,
'likes': 7314,
'dislike': 4444,
'fall': 9134,
'driving': 1296,
'peach': 4848,
'drives': 648,
'blue': 972,
'you': 2414,
'bird': 192,
'horses': 64,
'mouse': 192,
'went': 378,
'last': 781,
'horse': 192,
'automobile': 1944,
'dogs': 64,
'white': 972,
'elephant': 192,
'black': 972,
'think': 240,
'difficult': 260,
'translate': 480,
'between': 540,
'spanish': 312,
'portuguese': 312,
'big': 1016,
'green': 972,
'translating': 300,
'fun': 260,
'where': 12,
'dog': 192,
'why': 240,
'might': 378,
'go': 1386,
'this': 768,
'drove': 648,
'shiny': 972,
'sharks': 64,
'monkey': 192,
'how': 67,
'weather': 33,
'lion': 192,
'plans': 476,
'bear': 192,
'rabbit': 192,
"it's": 240,
'chinese': 312,
'when': 144,
'eiffel': 57,
'tower': 57,
'did': 204,
'grocery': 57,
'store': 57,
'wanted': 378,
'does': 24,
```

```
'football': 57,
'field': 57,
'wants': 252,
"didn't": 60,
'snake': 192,
'snakes': 64,
'do': 84,
'easy': 260,
'thinks': 360,
'english': 312,
'french': 312,
'would': 48,
"aren't": 36,
'cats': 64,
'rabbits': 64,
'has': 24,
'been': 36,
'monkeys': 64,
'lake': 57,
'bears': 64,
'school': 57,
'birds': 64,
'want': 126,
"isn't": 24,
'lions': 64,
'am': 24,
'mice': 64,
'have': 12})
```

In [22]:
```
# sort the dictionary by values
#reverse=True - descending order
#operator.itemgetter(1) gets the item at index 1 ig (the value of each key)
english_words_counts = sorted(english_words_counts.items(), key = operator.ite
mgetter(1), reverse = True)
```

In [23]:
```
english_words_counts
```

Out[23]:
```
[('is', 205858),
 ('in', 75525),
 ('it', 75137),
 ('during', 74933),
 ('the', 67628),
 ('but', 63987),
 ('and', 59850),
 ('sometimes', 37746),
 ('usually', 37507),
 ('never', 37500),
 ('favorite', 28332),
 ('least', 27564),
 ('fruit', 27192),
 ('most', 14934),
 ('loved', 14166),
 ('liked', 14046),
 ('new', 12197),
 ('paris', 11334),
 ('india', 11277),
 ('united', 11270),
 ('states', 11270),
 ('california', 11250),
 ('jersey', 11225),
 ('france', 11170),
 ('china', 10953),
 ('he', 10786),
 ('she', 10786),
 ('grapefruit', 10692),
 ('your', 9734),
 ('my', 9700),
 ('his', 9700),
 ('her', 9700),
 ('fall', 9134),
 ('june', 9133),
 ('spring', 9102),
 ('january', 9090),
 ('winter', 9038),
 ('march', 9023),
 ('autumn', 9004),
 ('may', 8995),
 ('nice', 8984),
 ('september', 8958),
 ('july', 8956),
 ('april', 8954),
 ('november', 8951),
 ('summer', 8948),
 ('december', 8945),
 ('february', 8942),
 ('our', 8932),
 ('their', 8932),
 ('freezing', 8928),
 ('pleasant', 8916),
 ('beautiful', 8915),
 ('october', 8910),
 ('snowy', 8898),
 ('warm', 8890),
 ('cold', 8878),
```

```
('wonderful', 8808),
('dry', 8794),
('busy', 8791),
('august', 8789),
('chilly', 8770),
('rainy', 8761),
('mild', 8743),
('wet', 8726),
('relaxing', 8696),
('quiet', 8693),
('hot', 8639),
('dislikes', 7314),
('likes', 7314),
('limes', 5844),
('lemons', 5844),
('grapes', 5844),
('mangoes', 5844),
('apples', 5844),
('peaches', 5844),
('oranges', 5844),
('pears', 5844),
('strawberries', 5844),
('bananas', 5844),
('to', 5166),
('grape', 4848),
('apple', 4848),
('orange', 4848),
('lemon', 4848),
('lime', 4848),
('banana', 4848),
('mango', 4848),
('pear', 4848),
('strawberry', 4848),
('peach', 4848),
('like', 4588),
('dislike', 4444),
('they', 3222),
('that', 2712),
('i', 2664),
('we', 2532),
('you', 2414),
('animal', 2304),
('a', 1944),
('truck', 1944),
('car', 1944),
('automobile', 1944),
('was', 1867),
('next', 1666),
('go', 1386),
('driving', 1296),
('visit', 1224),
('little', 1016),
('big', 1016),
('old', 972),
('yellow', 972),
('red', 972),
('rusty', 972),
```

```
('blue', 972),
('white', 972),
('black', 972),
('green', 972),
('shiny', 972),
('are', 870),
('last', 781),
('feared', 768),
('animals', 768),
('this', 768),
('plan', 714),
('going', 666),
('saw', 648),
('disliked', 648),
('drives', 648),
('drove', 648),
('between', 540),
('translate', 480),
('plans', 476),
('were', 384),
('went', 378),
('might', 378),
('wanted', 378),
('thinks', 360),
('spanish', 312),
('portuguese', 312),
('chinese', 312),
('english', 312),
('french', 312),
('translating', 300),
('difficult', 260),
('fun', 260),
('easy', 260),
('wants', 252),
('think', 240),
('why', 240),
("it's", 240),
('did', 204),
('cat', 192),
('shark', 192),
('bird', 192),
('mouse', 192),
('horse', 192),
('elephant', 192),
('dog', 192),
('monkey', 192),
('lion', 192),
('bear', 192),
('rabbit', 192),
('snake', 192),
('when', 144),
('want', 126),
('do', 84),
('how', 67),
('elephants', 64),
('horses', 64),
('dogs', 64),
```

```
 ('sharks', 64),
 ('snakes', 64),
 ('cats', 64),
 ('rabbits', 64),
 ('monkeys', 64),
 ('bears', 64),
 ('birds', 64),
 ('lions', 64),
 ('mice', 64),
 ("didn't", 60),
 ('eiffel', 57),
 ('tower', 57),
 ('grocery', 57),
 ('store', 57),
 ('football', 57),
 ('field', 57),
 ('lake', 57),
 ('school', 57),
 ('would', 48),
 ("aren't", 36),
 ('been', 36),
 ('weather', 33),
 ('does', 24),
 ('has', 24),
 ("isn't", 24),
 ('am', 24),
 ('where', 12),
 ('have', 12)]
```

In [24]:
```
# append the values to a list for visualization purposes
english_words = []
english_counts = []
for i in range(len(english_words_counts)):
  english_words.append(english_words_counts[i][0])
  english_counts.append(english_words_counts[i][1])
```

In [25]:
```
english_words
```

Out[25]:
```
['is',
 'in',
 'it',
 'during',
 'the',
 'but',
 'and',
 'sometimes',
 'usually',
 'never',
 'favorite',
 'least',
 'fruit',
 'most',
 'loved',
 'liked',
 'new',
 'paris',
 'india',
 'united',
 'states',
 'california',
 'jersey',
 'france',
 'china',
 'he',
 'she',
 'grapefruit',
 'your',
 'my',
 'his',
 'her',
 'fall',
 'june',
 'spring',
 'january',
 'winter',
 'march',
 'autumn',
 'may',
 'nice',
 'september',
 'july',
 'april',
 'november',
 'summer',
 'december',
 'february',
 'our',
 'their',
 'freezing',
 'pleasant',
 'beautiful',
 'october',
 'snowy',
 'warm',
 'cold',
```

```
'wonderful',
'dry',
'busy',
'august',
'chilly',
'rainy',
'mild',
'wet',
'relaxing',
'quiet',
'hot',
'dislikes',
'likes',
'limes',
'lemons',
'grapes',
'mangoes',
'apples',
'peaches',
'oranges',
'pears',
'strawberries',
'bananas',
'to',
'grape',
'apple',
'orange',
'lemon',
'lime',
'banana',
'mango',
'pear',
'strawberry',
'peach',
'like',
'dislike',
'they',
'that',
'i',
'we',
'you',
'animal',
'a',
'truck',
'car',
'automobile',
'was',
'next',
'go',
'driving',
'visit',
'little',
'big',
'old',
'yellow',
'red',
'rusty',
```

```
                'blue',
                'white',
                'black',
                'green',
                'shiny',
                'are',
                'last',
                'feared',
                'animals',
                'this',
                'plan',
                'going',
                'saw',
                'disliked',
                'drives',
                'drove',
                'between',
                'translate',
                'plans',
                'were',
                'went',
                'might',
                'wanted',
                'thinks',
                'spanish',
                'portuguese',
                'chinese',
                'english',
                'french',
                'translating',
                'difficult',
                'fun',
                'easy',
                'wants',
                'think',
                'why',
                "it's",
                'did',
                'cat',
                'shark',
                'bird',
                'mouse',
                'horse',
                'elephant',
                'dog',
                'monkey',
                'lion',
                'bear',
                'rabbit',
                'snake',
                'when',
                'want',
                'do',
                'how',
                'elephants',
                'horses',
                'dogs',
```

```
'sharks',
'snakes',
'cats',
'rabbits',
'monkeys',
'bears',
'birds',
'lions',
'mice',
"didn't",
'eiffel',
'tower',
'grocery',
'store',
'football',
'field',
'lake',
'school',
'would',
"aren't",
'been',
'weather',
'does',
'has',
"isn't",
'am',
'where',
'have']
```

In [26]:
```
english_counts
```

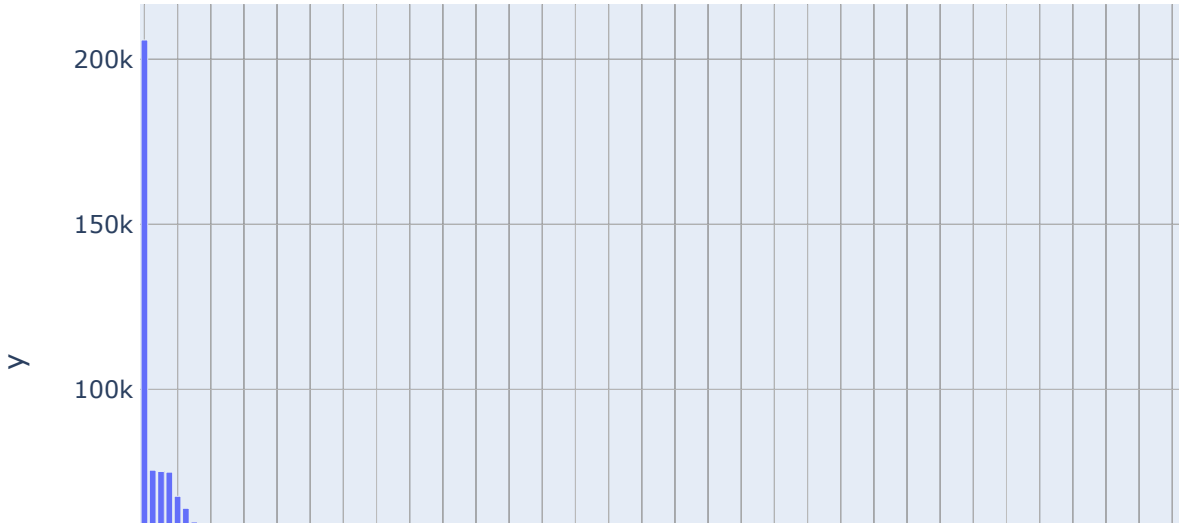
Out[26]:
```
[205858,
 75525,
 75137,
 74933,
 67628,
 63987,
 59850,
 37746,
 37507,
 37500,
 28332,
 27564,
 27192,
 14934,
 14166,
 14046,
 12197,
 11334,
 11277,
 11270,
 11270,
 11250,
 11225,
 11170,
 10953,
 10786,
 10786,
 10692,
 9734,
 9700,
 9700,
 9700,
 9134,
 9133,
 9102,
 9090,
 9038,
 9023,
 9004,
 8995,
 8984,
 8958,
 8956,
 8954,
 8951,
 8948,
 8945,
 8942,
 8932,
 8932,
 8928,
 8916,
 8915,
 8910,
 8898,
 8890,
 8878,
```

```
8808,
8794,
8791,
8789,
8770,
8761,
8743,
8726,
8696,
8693,
8639,
7314,
7314,
5844,
5844,
5844,
5844,
5844,
5844,
5844,
5844,
5844,
5844,
5844,
5166,
4848,
4848,
4848,
4848,
4848,
4848,
4848,
4848,
4848,
4848,
4848,
4588,
4444,
3222,
2712,
2664,
2532,
2414,
2304,
1944,
1944,
1944,
1944,
1867,
1666,
1386,
1296,
1224,
1016,
1016,
972,
972,
972,
972,
```

```
            972,
            972,
            972,
            972,
            972,
            870,
            781,
            768,
            768,
            768,
            714,
            666,
            648,
            648,
            648,
            648,
            540,
            480,
            476,
            384,
            378,
            378,
            378,
            360,
            312,
            312,
            312,
            312,
            312,
            300,
            260,
            260,
            260,
            252,
            240,
            240,
            240,
            204,
            192,
            192,
            192,
            192,
            192,
            192,
            192,
            192,
            192,
            192,
            192,
            192,
            144,
            126,
            84,
            67,
            64,
            64,
            64,
```

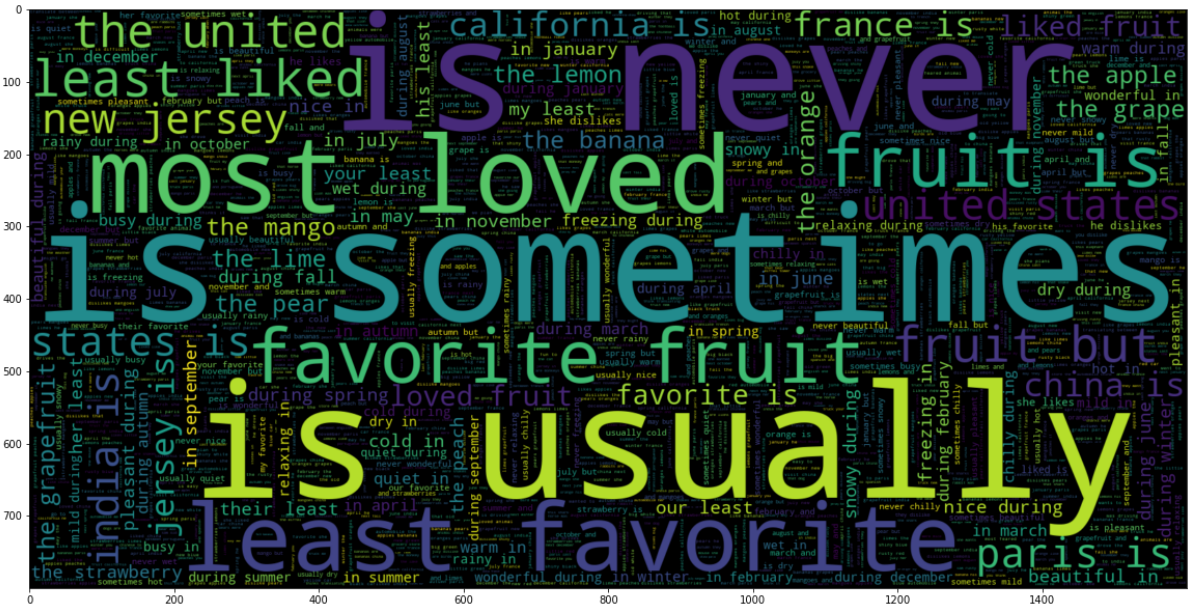
```
64,
64,
64,
64,
64,
64,
64,
64,
64,
64,
60,
57,
57,
57,
57,
57,
57,
57,
57,
57,
48,
36,
36,
33,
24,
24,
24,
24,
12,
12]
```

In [27]:
```
# Plot barplot using plotly
fig = px.bar(x=english_words, y=english_counts)
fig.show()
```



In [28]:
```
# plot the word cloud for text that is Real
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000, width = 1600, height = 800 ).generate(" ".joi
n(df.english))
plt.imshow(wc, interpolation = 'bilinear')
```

Out[28]: <matplotlib.figure.AxesImage at 0x23aa2ebc108>



In [29]:
```
#gets first sentence
df.english[0]
#this splits the sentence into an array
nltk.word_tokenize(df.english[0])
```

Out[29]:
```
['new',
 'jersey',
 'is',
 'sometimes',
 'quiet',
 'during',
 'autumn',
 'and',
 'it',
 'is',
 'snowy',
 'in',
 'april']
```

In [30]:
```
# Maximum length (number of words) per document. We will need it later for embeddings
maxlen_english = -1
#tokenizes each sentence and obtains max length of a sentence in the dictionary
for doc in df.english:
    tokens = nltk.word_tokenize(doc)
    if(maxlen_english < len(tokens)):
        maxlen_english = len(tokens)
print("The maximum number of words in any document = ", maxlen_english)
```

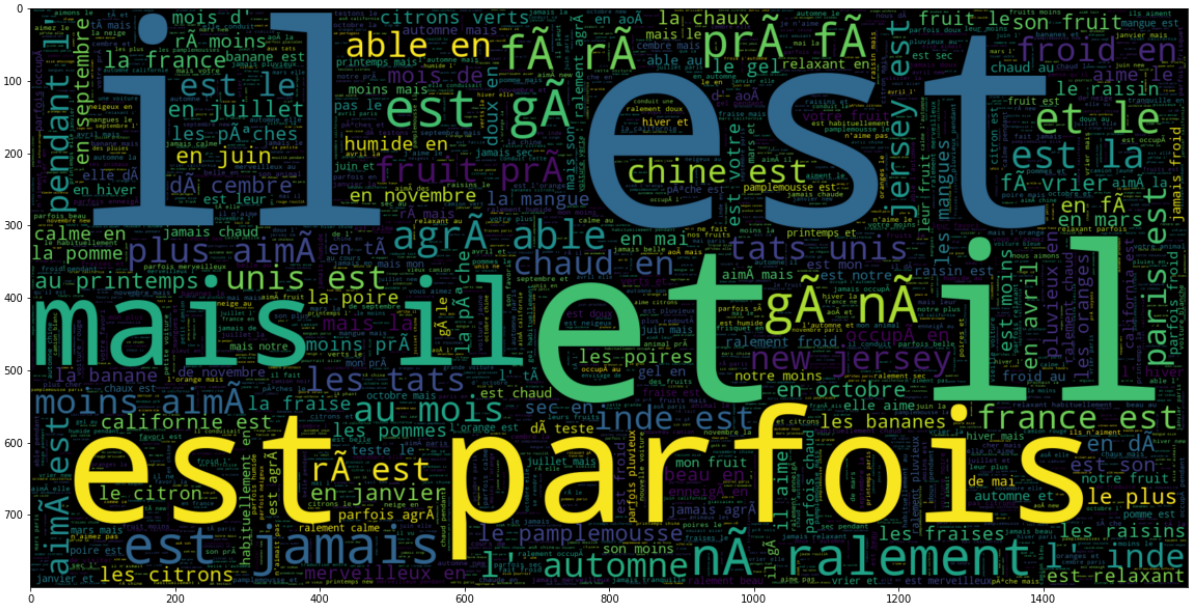
```
The maximum number of words in any document =  15
```

MINI CHALLENGE #4 (QUIZ!):

- Perform similar data visualizations but for the french language instead
- What are the top 3 common french words?!
- What is the maximum number of words in any french document?

In [31]:
```
# obtain the count of french words
words = []
for i in df['french']:
  for word in i.split():
    words.append(word)
words

french_words_counts = Counter(words)
french_words_counts


# sort the dictionary by values
french_words_counts = sorted(french_words_counts.items(), key = operator.itemg
etter(1), reverse = True)

french_words_counts

# append the values to a list for visuaization purpose
french_words = []
french_counts = []
for i in range(len(french_words_counts)):
  french_words.append(french_words_counts[i][0])
  french_counts.append(french_words_counts[i][1])

fig = px.bar(x = french_words, y = french_counts)
fig.show()


# plot the word cloud for French
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".jo
in(df.french))
plt.imshow(wc, interpolation = 'bilinear')


# Maximum length (number of words) per document. We will need it later for emb
eddings
maxlen_french = -1
for doc in df.french:
    tokens = nltk.word_tokenize(doc)
    if(maxlen_french < len(tokens)):
        maxlen_french = len(tokens)
print("The maximum number of words in any document = ", maxlen_french)
```


```
The maximum number of words in any document =  24
```



TASK #5: PREPARE THE DATA BY PERFORMING TOKENIZATION AND PADDING

TOKENIZER

- Tokenizer allows us to vectorize text corpus by turning each text into a sequence of integers.
- SENTENCE:

" budget fight looms republicans flip fiscal script Washington Reuters head conservative republican faction congress voted month …"

- TOKENS:

[3138, 3581, 2895, 27, 5354, 22457, 3505, 9, 3138, 35, 2895, 208, 213, 3581, 29, 71, 5354, 22457, 1275, 335, 2, 619, 2903, 27, 10461, 43213, 4908, …]

In [32]:
```
#to train lstm we need to tokenize all the words
def tokenize_and_pad(x, maxlen):
  #  a tokenier to tokenize the words and create sequences of tokenized words
  tokenizer = Tokenizer(char_level = False)
  #use the tokenizer on x
  tokenizer.fit_on_texts(x)
  #returns tokenized version of the text
  sequences = tokenizer.texts_to_sequences(x)
  #make all the data words of equal size
  padded = pad_sequences(sequences, maxlen = maxlen, padding = 'post')
  return tokenizer, sequences, padded
```

In [33]:
```
# tokenize and padding to the data
x_tokenizer, x_sequences, x_padded = tokenize_and_pad(df.english, maxlen_engli
sh)
y_tokenizer, y_sequences, y_padded = tokenize_and_pad(df.french,  maxlen_frenc
h)
```

In [34]:
```
print("The tokenized version for document\n", df.english[-1:].item(),"\n is :
 ", x_padded[-1:])
```

```
The tokenized version for document
 the orange is her favorite fruit  but the banana is your favorite
 is :  [[ 5 84  1 32 11 13  6  5 87  1 29 11  0  0  0]]
```

In [35]:
```
print("The tokenized version for document\n", df.french[-1:].item(),"\n is : "
, y_padded[-1:])
```

```
The tokenized version for document
 l'orange est son fruit prÃ©fÃ©rÃ©  mais la banane est votre favori
 is :  [[84  1 20 16 17  5  7 87  1 40 93  0  0  0  0  0  0  0  0  0  0  0  0
0]]
```

In [36]:
```
# function to obtain the text from padded variables
def pad_to_text(padded, tokenizer):

    id_to_word = {id: word for word, id in tokenizer.word_index.items()}
    id_to_word[0] = ''

    return ' '.join([id_to_word[j] for j in padded])
```

In [37]:
```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\users\administrator\anaconda31\l
ib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\administrator\anacond
a31\lib\site-packages (from sklearn) (0.22.1)
Requirement already satisfied: scipy>=0.17.0 in c:\users\administrator\anacon
da31\lib\site-packages (from scikit-learn->sklearn) (1.4.1)
Requirement already satisfied: numpy>=1.11.0 in c:\users\administrator\anacon
da31\lib\site-packages (from scikit-learn->sklearn) (1.18.1)
Requirement already satisfied: joblib>=0.11 in c:\users\administrator\anacond
a31\lib\site-packages (from scikit-learn->sklearn) (0.14.1)
```

In [38]:
```
# Train test split
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_padded, y_padded, test_s
ize = 0.1)
```

MINI CHALLENGE #5:

- Change the padding length so that both english and french have the same length

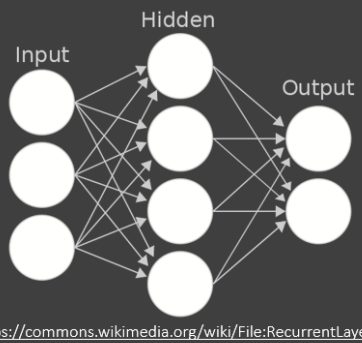
In [39]:
```
x_tokenizer, x_sequences, x_padded = tokenize_and_pad(df.english, maxlen_frenc
h)
y_tokenizer, y_sequences, y_padded = tokenize_and_pad(df.french,  maxlen_frenc
h)
```

TASK #6: UNDERSTAND THE THEORY AND INTUITION BEHIND RECURRENT NEURAL NETWORKS AND LSTM

RECURRENT NEURAL NETWORKS (RNN): WHAT ARE THEY?

- We covered Feedforward Neural Networks (vanilla networks) that map a fixed size input (such as image) to a fixed size output (classes or probabilities).
- A drawback in Feedforward networks is that they do not have any time dependency or memory effect.
- A RNN is a type of ANN that is designed to take temporal dimension into consideration by having a memory (internal state) (feedback loop).

FEED FORWARD ANN



RECURRENT NEURAL NETWORK

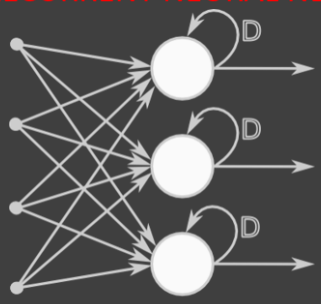


Photo Credit: https://commons.wikimedia.org/wiki/File:RecurrentLayerNeuralNetwork_english.png
Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

CHECK OUT THIS MOVIE WRITTEN BY AI!

- Let's watch a movie written by AI!

https://arstechnica.com/gaming/2016/06/an-ai-wrote-this-movie-and-its-strangely-moving/

- The movie was written by an LSTM recurrent neural network
- The LSTM network was trained with a corpus of dozens of sci-fi screenplays from movies from the 1980s and 90s.

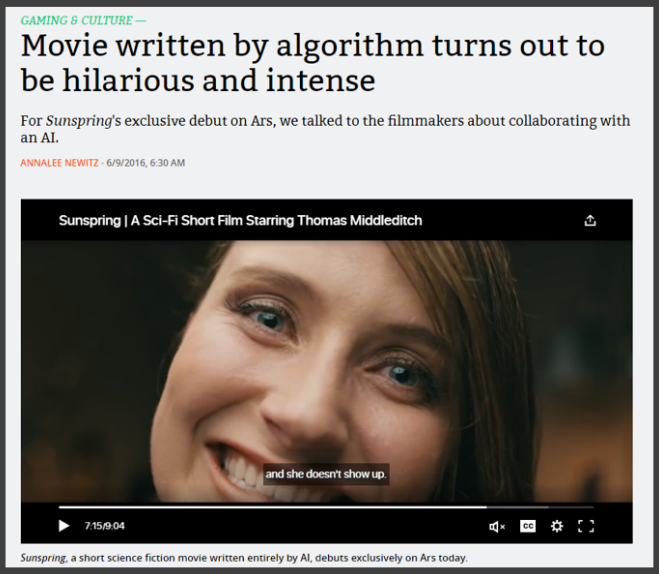


Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

RNN ARCHITECTURE

- A RNN contains a temporal loop in which the hidden layer not only gives an output but it feeds itself as well.
- An extra dimension is added which is time!
- RNN can recall what happened in the previous time stamp so it works great with sequence of text.

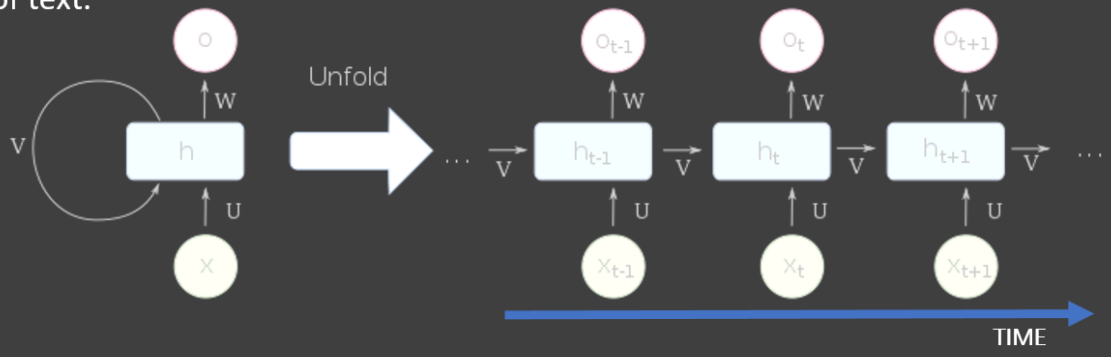
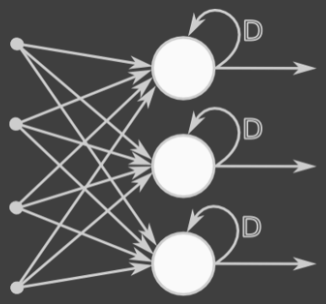


Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

WHAT MAKE RNNs SO SPECIAL?

- Feedforward ANNs are so constrained with their fixed number of input and outputs.
- For example, a CNN will have fixed size image (28x28) and generates a fixed output (class or probabilities).
- Feedforward ANN have a fixed configuration, i.e.: same number of hidden layers and weights.
- Recurrent Neural Networks offer huge advantage over feedforward ANN and they are much more fun!
- RNN allow us to work with a sequence of vectors:
  - Sequence in inputs
  - Sequence in outputs
  - Sequence in both!

RECURRENT NEURAL NETWORK



Source: The Unreasonable Effectiveness of Recurrent Neural Networks by Andrej Karpathy http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Photo Credit: https://commons.wikimedia.org/wiki/File:RecurrentLayerNeuralNetwork_english.png

WHAT MAKE RNNs SO SPECIAL?

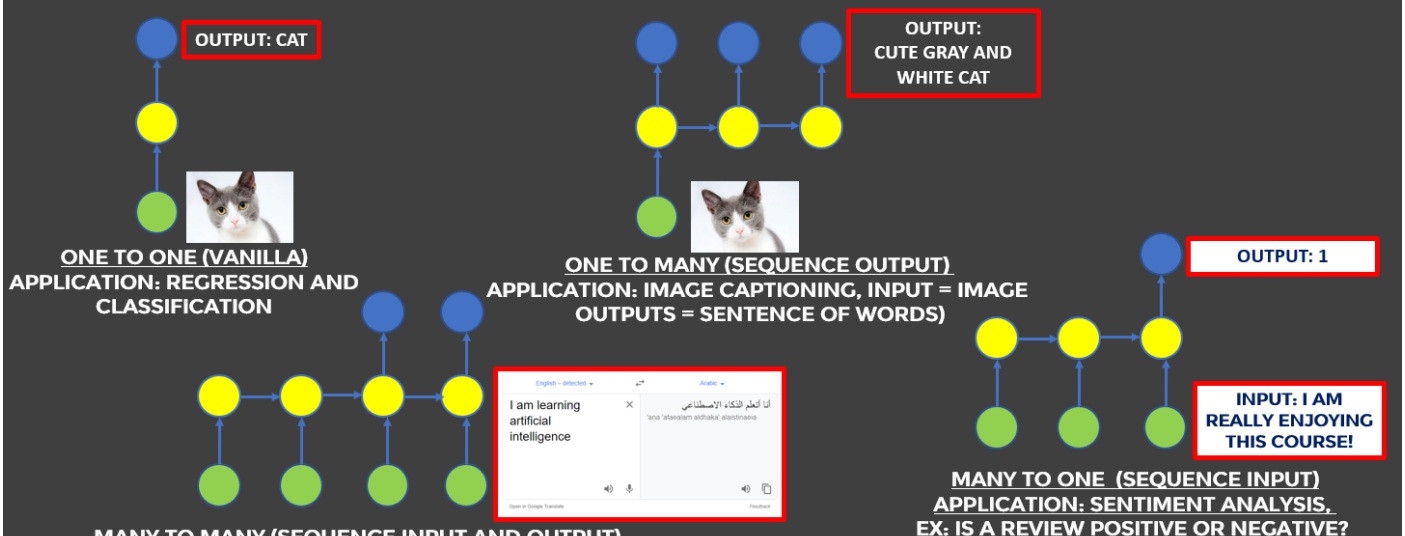
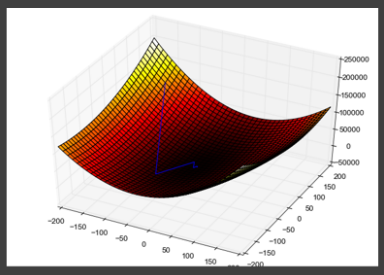


Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

GRADIENT DESCENT

- Gradient descent is an optimization algorithm used to obtain the optimized network weight and bias values.
- It works by iteratively trying to minimize the cost function.
- It works by calculating the gradient of the cost function and moving in the negative direction until the local/global minimum is achieved.
- If the positive of the gradient is taken, local/global maximum is achieved.
- The size of the steps taken are called the learning rate.
- If learning rate increases, the area covered in the search space will increase so we might reach global minimum faster. However, we can overshoot the target.
- For small learning rates, training will take much longer to reach optimized weight values



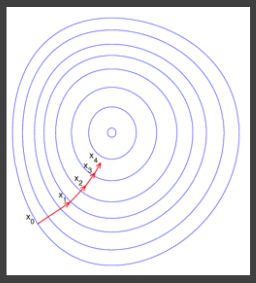


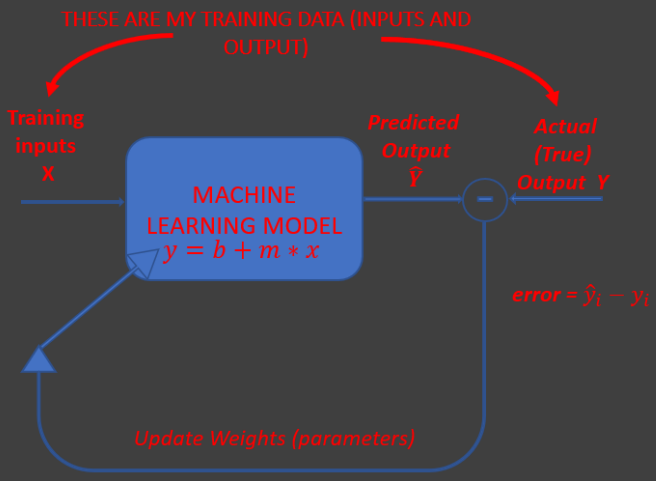
Photo Credit: https://commons.wikimedia.org/wiki/File:Gradient_descent_method.png
Photo Credit: https://commons.wikimedia.org/wiki/File:Gradient_descent.png

GRADIENT DESCENT

- Let's assume that we want to obtain the optimal values for parameters 'm' and 'b'.

$$y = b + m * x$$

GOAL IS TO FIND BEST PARAMETERS

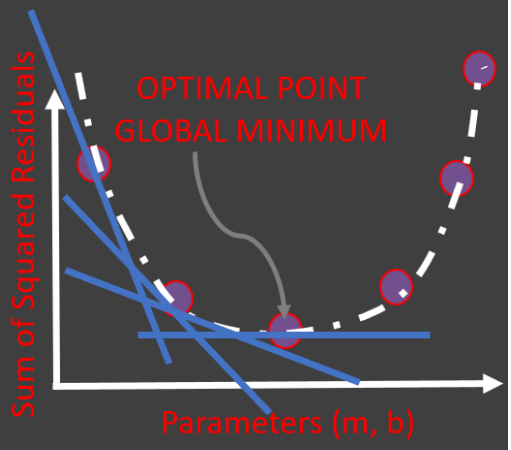


- We need to first formulate a loss function as follows:

$$Cost\ Function\ f(m,b) = \frac{1}{N}\sum_{i=1}^{n}(error)^2 = \frac{1}{N}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

GRADIENT DESCENT

$$Loss\ Function\ f(m,b) = \frac{1}{N}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$



GRADIENT DESCENT WORKS AS FOLLOWS:

1. Calculate the gradient (derivative) of the Loss function $\frac{\partial loss}{\partial w}$
2. Pick random values for weights (m, b) and substitute
3. Calculate the step size (how much are we going to update the parameters?)

$$Step\ size\ = learning\ rate * gradient = \alpha * \frac{\partial loss}{\partial w}$$

4. Update the parameters and repeat

$$new\ weight\ = old\ weight - step\ size$$

$$w_{new} = w_{old} - \alpha * \frac{\partial loss}{\partial w}$$

*Note: in reality, this graph is 3D and has three axes, one for m, b and sum of squared residuals

VANISHING GRADIENT PROBLEM

- LSTM networks work much better compared to vanilla RNN since they overcome the vanishing gradient problem.
- The error has to propagate through all the previous layers resulting in a vanishing gradient.
- As the gradient goes smaller, the network weights are no longer updated.
- As more layers are added, the gradients of the loss function approaches zero, making the network hard to train.

EACH LAYER DEPENDS ON THE OUTPUT FROM THE PREVIOUS LAYERS, THE "V" IS MULTIPLIED SEVERAL TIMES RESULTING IN VANISHING GRADIENT

0.1 * 0.1 * 0.1 *.....*0.1 =1e-10

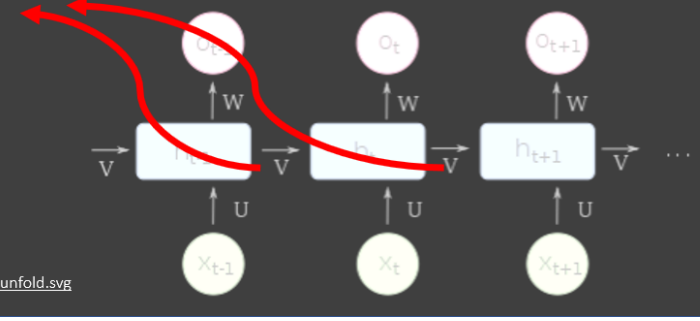
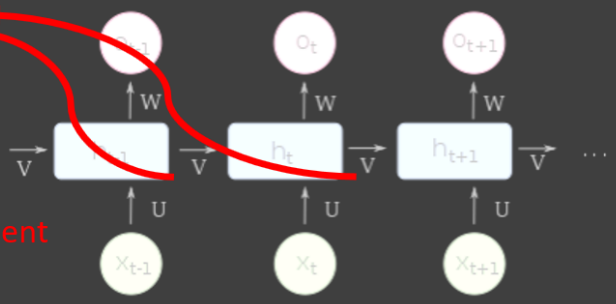


Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

VANISHING GRADIENT PROBLEM

- ANN gradients are calculated during backpropagation.
- In backpropagation, we calculate the derivatives of the network by moving from the outermost layer (close to output) back to the initial layers (close to inputs).
- The chain rule is used during this calculation in which the derivatives from the final layers are multiplied by the derivatives from early layers.
- The gradients keeps diminishing exponentially and therefore the weights and biases are no longer being updated.

EACH LAYER DEPENDS ON THE OUTPUT FROM THE PREVIOUS LAYERS, THE "V" IS MULTIPLIED SEVERAL TIMES RESULTING IN VANISHING GRADIENT, (ex: 0.1 * 0.1 * 0.1 *.....*0.1 =1e-10)



New Weight = Old Weight – Learning rate * gradient
9.09999 = 10.1 – 1 * 0.001

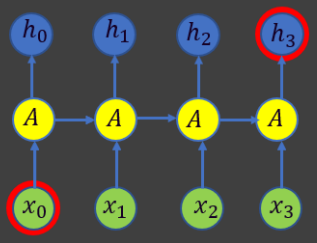
Photo Credit: https://fr.wikipedia.org/wiki/Fichier:Recurrent_neural_network_unfold.svg

TASK #7: UNDERSTAND THE INTUITION BEHIND LONG SHORT TERM MEMORY (LSTM) NETWORKS

LSTM INTUITION

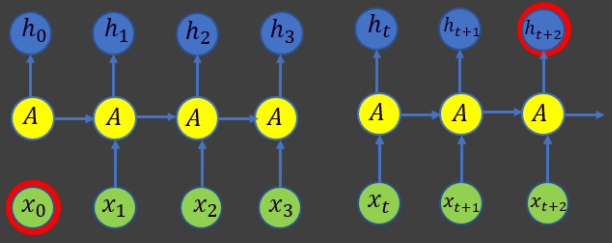
- LSTM networks work better compared to vanilla RNN since they overcome vanishing gradient problem.
- In practice, RNN fail to establish long term dependencies.
- Reference: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The tree color is "green"



RNN PERFORMS WELL SINCE THE GAP BETWEEN THE PREDICTION "GREEN" AND THE NECESSARY CONTEXT INFORMATION "TREE" IS SMALL

I live in Quebec in Northern Canada……where I live, the weather is generally "cold" most of the year



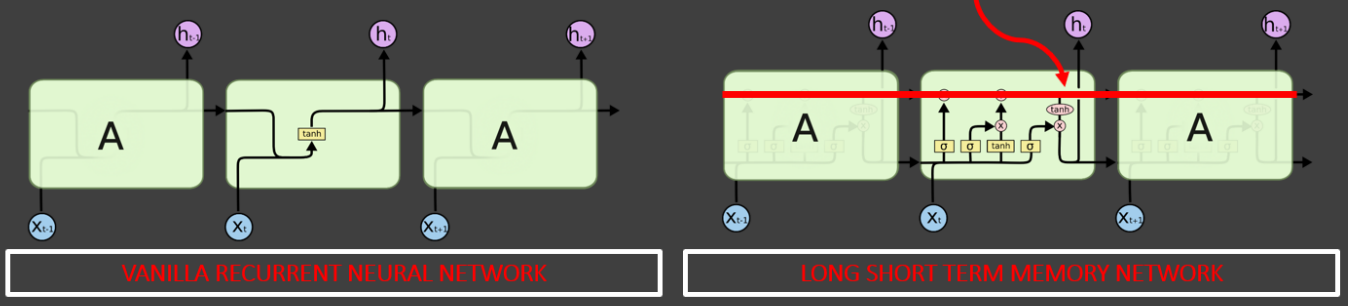
RNN PERFORMS POORLY WHEN THE GAP BETWEEN THE PREDICTION "COLD" AND THE NECESSARY CONTEXT INFORMATION "CANADA" IS LARGE

Reference: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM INTUITION

- LSTM networks are type of RNN that are designed to remember long term dependencies by default.
- LSTM can remember and recall information for a prolonged period of time.
- Recall that each line represents a full vector.

THIS HORIZONTAL LINE (MEMORY) OR CELL STATE ENABLES LSTM TO REMEMBER VERY OLD INFORMTION



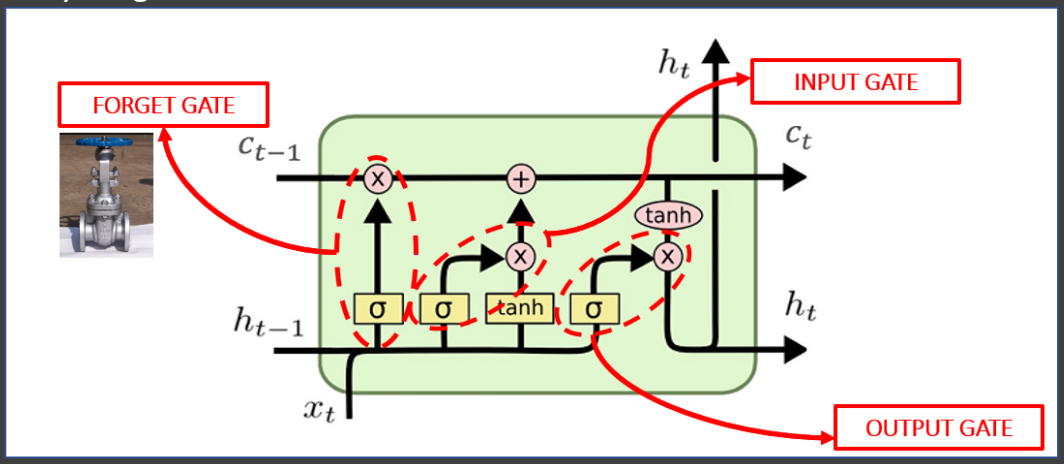
VANILLA RECURRENT NEURAL NETWORK

LONG SHORT TERM MEMORY NETWORK

Reference and Photo Credit: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM INTUITION – GATES

- LSTM contains gates that can allow or block information from passing by.
- Gates consist of a sigmoid neural net layer along with a pointwise multiplication operation.
- Sigmoid output ranges from 0 to 1:
  - 0 = Don't allow any data to flow
  - 1 = Allow everything to flow!



Reference and Photo Credit: https://colah.github.io/posts/2015-08-Understanding-LSTMs/
Photo Credit: https://commons.wikimedia.org/wiki/File:LSTM.png

LSTM INTUITION – MATH

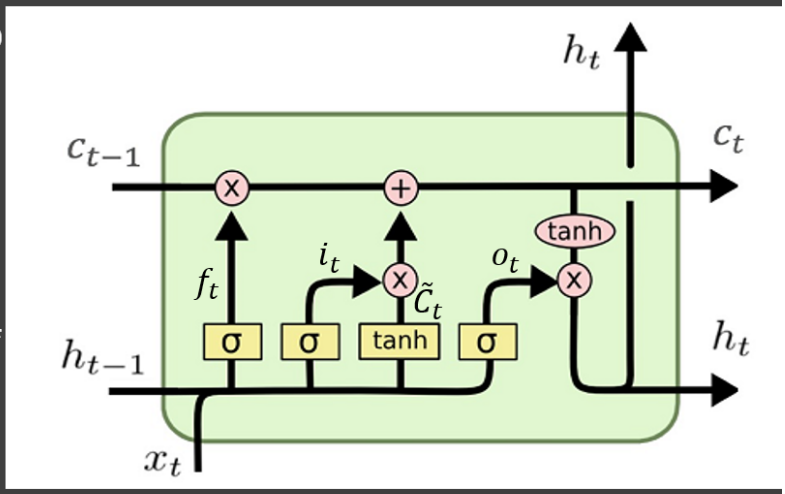
$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$\tilde{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

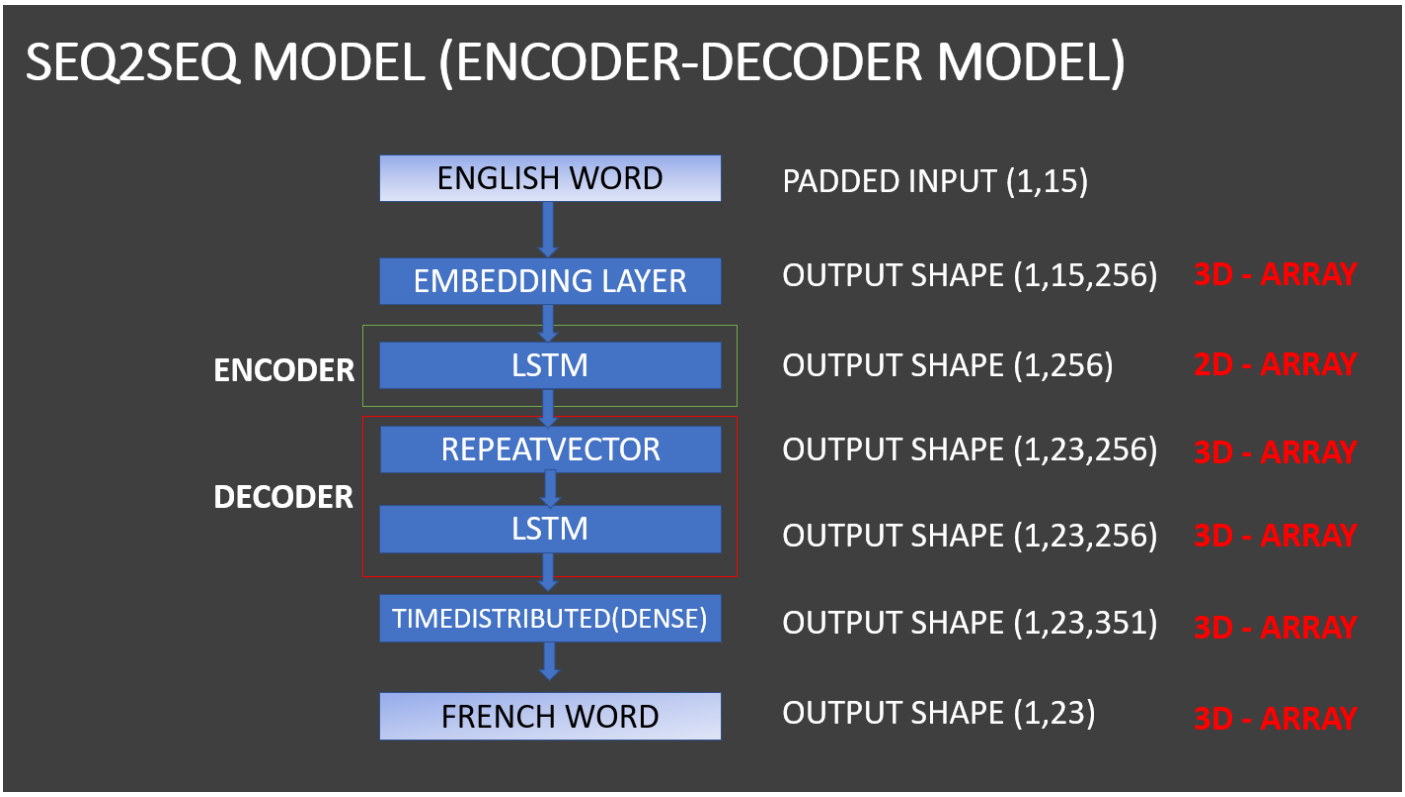


Reference and Photo Credit: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

TASK #8: BUILD AND TRAIN THE MODEL

EMBEDDING LAYER

- Embedding layers learn the low-dimensional continuous representation of input discrete variables.
- For example, let assume that we have 100,000 unique values in our data and want to train the model with this data. Even though we can train the model to generate accurate results, it would require more resources to train.
- Alternatively, by introducing an embedding layer, you can specify the number of low-dimensional features that you would need to represent the input data, in this case let's take the value to be 200.
- Now, what happens is the embedding layer learns the way to represent 100,000 variables with 200 variables only (*think of it as PCA or Autoencoder*).
- This in-turn helps subsequent layers learn more effectively with less compute resources.


SEQ2SEQ MODEL (ENCODER-DECODER MODEL)
ENGLISH WORD
PADDED INPUT (1,15)
EMBEDDING LAYER
OUTPUT SHAPE (1,15,256)
3D - ARRAY
ENCODER
LSTM
OUTPUT SHAPE (1,256)
2D - ARRAY
DECODER
REPEATVECTOR
OUTPUT SHAPE (1,23,256)
3D - ARRAY
LSTM
OUTPUT SHAPE (1,23,256)
3D - ARRAY
TIMEDISTRIBUTED(DENSE)
OUTPUT SHAPE (1,23,351)
3D - ARRAY
FRENCH WORD
OUTPUT SHAPE (1,23)
3D - ARRAY

In [40]:
```
#total vocab size, since we added padding we add 1 to the total word count
tf.test.is_gpu_available( cuda_only=False, min_cuda_compute_capability=None )

english_vocab_size = len(english_words) + 1
french_vocab_size = len(french_words) + 1
# Sequential Model
model = Sequential()
# embedding layer
model.add(tf.keras.layers.Embedding(english_vocab_size, 256, input_length = ma
xlen_english, mask_zero = True))
# encoder
model.add(LSTM(256))
# decoder
# repeatvector repeats the input for the desired number of times to change
# 2D-array to 3D array. For example: (1,256) to (1,23,256)
model.add(RepeatVector(maxlen_french))
model.add(LSTM(256, return_sequences= True ))
model.add(TimeDistributed(Dense(french_vocab_size, activation ='softmax')))
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metric
s=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 15, 256)           51200
_________________________________________________________________
lstm (LSTM)                  (None, 256)               525312
_________________________________________________________________
repeat_vector (RepeatVector) (None, 24, 256)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 24, 256)           525312
_________________________________________________________________
time_distributed (TimeDistri (None, 24, 351)           90207
=================================================================
Total params: 1,192,031
Trainable params: 1,192,031
Non-trainable params: 0
_________________________________________________________________
```

In [41]:
```
# change the shape of target from 2D to 3D
y_train = np.expand_dims(y_train, axis = 2)
y_train.shape
```

Out[41]:
```
(124074, 24, 1)
```

In [42]:
```
# train the model
model.fit(x_train, y_train, batch_size=1024, validation_split= 0.1, epochs=1)
```

```
Train on 111666 samples, validate on 12408 samples
111666/111666 [==============================] - 553s 5ms/sample - loss: 2.59
85 - accuracy: 0.5185 - val_loss: 2.0164 - val_accuracy: 0.5599
```

Out[42]:
```
<tensorflow.python.keras.callbacks.History at 0x23a9d331ac8>
```

In [43]:
```
# save the model
model.save("weights.h5")
```

MINI CHALLENGE #6:

- Train the model with different embedding output dimension and comment on model performance during training

In [ ]:
```

```

TASK #9: ASSESS TRAINED MODEL PERFORMANCE

In [44]:
```
# function to make prediction
def prediction(x, x_tokenizer = x_tokenizer, y_tokenizer = y_tokenizer):
    predictions = model.predict(x)[0]
    id_to_word = {id: word for word, id in y_tokenizer.word_index.items()}
    id_to_word[0] = ''
    return ' '.join([id_to_word[j] for j in np.argmax(predictions,1)])
```

In [45]:
```
for i in range(5):

  print('Original English word - {}\n'.format(pad_to_text(x_test[i], x_tokeniz
er)))
  print('Original French word - {}\n'.format(pad_to_text(y_test[i], y_tokenize
r)))
  print('Predicted French word - {}\n\n\n\n'.format(prediction(x_test[i:i+1
])))
```

```
Original English word - they like grapefruit peaches and strawberries

Original French word - ils aiment le pamplemousse les pãªches et les fraises

Predicted French word - les les les les les les les les les



Original English word - i dislike apples lemons and limes

Original French word - je n'aime pas les pommes les citrons et les citrons ve
rts

Predicted French word - les les les les les les les les les



Original English word - paris is cold during january but it is sometimes rain
y in summer

Original French word - paris est froid en janvier mais il est parfois pluvieu
x en ã©tã©

Predicted French word - est est est est est est est il est est est en



Original English word - has she been to the school

Original French word - elle a ã©tã© ã  l'ã©cole

Predicted French word - les les les les les



Original English word - i dislike bananas strawberries and oranges

Original French word - je n'aime les bananes les fraises et les oranges

Predicted French word - les les les les les les les les les les
```

CONGRATULATIONS!

MINI CHALLENGE #1:

- Explore the 'english' and 'french' data and indicate how many samples is included.

In [ ]:
```
# data containing english text
df_english
# data containing frensh text
df_french

# dataframe information
df_english.info
# dataframe information
df_french.info()

# check for null values
df_english.isnull().sum()
# check for null values
df_french.isnull().sum()
```

MINI CHALLENGE #2:

- Concatenate both dataframes and indicate how many records are present

In [ ]:
```
# Concatenate Real and Fake News
df = pd.concat([df_english, df_french], axis = 1)
df
```

In [ ]:
```
print("Total English Records = {}".format(len(df['english'])))
print("Total French Records = {}".format(len(df['french'])))
```

MINI CHALLENGE #3:

- How many unique words are available in the english and french dictionairies?

In [ ]:
```
# function to get the list of unique words
def get_label_superset(x, word_list):
    for label in x.split():
        if label not in word_list:
            word_list.append(label)


df['english'].apply(lambda x: get_label_superset(x, english_words))
df['french'].apply(lambda x: get_label_superset(x, french_words))

# number of unique words in english
total_english_words  = len(english_words)
total_english_words

# number of unique words in french
total_french_words = len(french_words)
total_french_words
```

MINI CHALLENGE #4 (QUIZ!):

- Perform similar data visualizations but for the french language instead
- What are the top 3 common french words?!
- What is the maximum number of words in any french document?

In [ ]:
```
# obtain the count of french words
words = []
for i in df['french']:
  for word in i.split():
    words.append(word)
words

french_words_counts = Counter(words)
french_words_counts


# sort the dictionary by values
french_words_counts = sorted(french_words_counts.items(), key = operator.itemg
etter(1), reverse = True)

french_words_counts

# append the values to a list for visuaization purpose
french_words = []
french_counts = []
for i in range(len(french_words_counts)):
  french_words.append(french_words_counts[i][0])
  french_counts.append(french_words_counts[i][1])

fig = px.bar(x = french_words, y = french_counts)
fig.show()


# plot the word cloud for French
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 2000 , width = 1600 , height = 800).generate(" ".jo
in(df.french))
plt.imshow(wc, interpolation = 'bilinear')


# Maximum length (number of words) per document. We will need it later for emb
eddings
maxlen_french = -1
for doc in df.french:
    tokens = nltk.word_tokenize(doc)
    if(maxlen_french < len(tokens)):
        maxlen_french = len(tokens)
print("The maximum number of words in any document = ", maxlen_french)
```

MINI CHALLENGE #5:

- Change the padding length so that both english and french have the same length and retrain the model

In [ ]:
```
# tokenize and padding to the data
x_tokenizer, x_sequences, x_padded = tokenize_and_pad(df.english, maxlen_frenc
h)
y_tokenizer, y_sequences, y_padded = tokenize_and_pad(df.french,  maxlen_frenc
h)
```

MINI CHALLENGE #6

- Train the model with different embedding output dimension and comment on model

In [ ]:
```
# Sequential Model
model = Sequential()

# embedding layer
model.add(Embedding(english_vocab_size, 128, input_length = maxlen_english, ma
sk_zero = True))
model.add(LSTM(256))
model.add(RepeatVector(maxlen_french))
model.add(LSTM(256, return_sequences= True ))
model.add(TimeDistributed(Dense(french_vocab_size, activation ='softmax')))
model.compile(optimizer = Adam(lr = 1e-3), loss = 'sparse_categorical_crossent
ropy', metrics=['accuracy'])
model.summary()
```