Phase3

DEVELOPMENT-PART I

Project Name : CREATE A CHATBOT IN PYTHON

Team ID : 8932

Introduction:

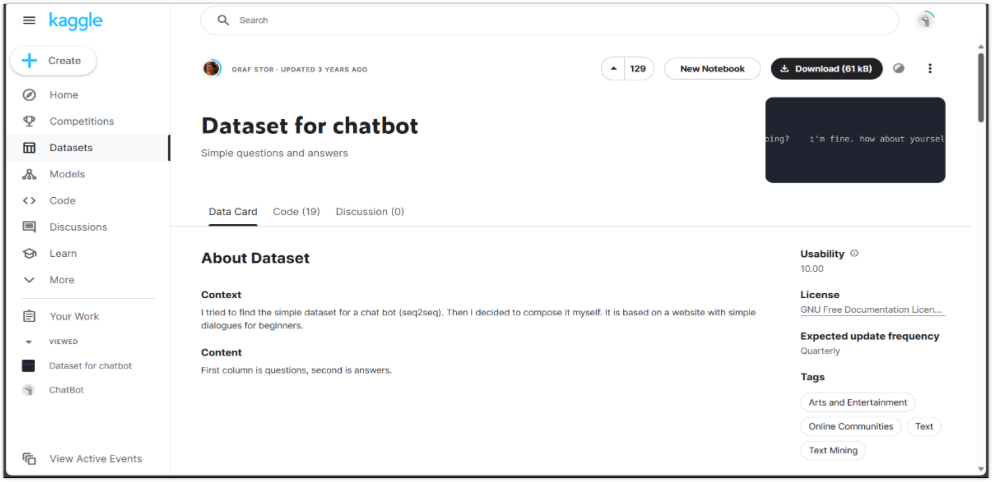
Chatbot often powered by sophisticated language models like GPT, have gained popularity in natural language processing tasks. These models can generate human-like responses based on input text and are trained on vast amounts of diverse textual data.

To effectively train such models, a crucial step is the preprocessing of the dataset. In this Python script, we outline a systematic approach to prepare a dataset for training a chatbot.

Set up a development environment:

To get started, we need to set up our development environment.we are using Jupyter Notebook (Python3).

Download corresponding dataset:



Dataset for chatbot

Simple questions and answers

About Dataset

Context

I tried to find the simple dataset for a chat bot (seq2seq). Then I decided to compose it myself. It is based on a website with simple dialogues for beginners.

Content

First column is questions, second is answers.

Usability 10.00

License GNU Free Documentation Licen...

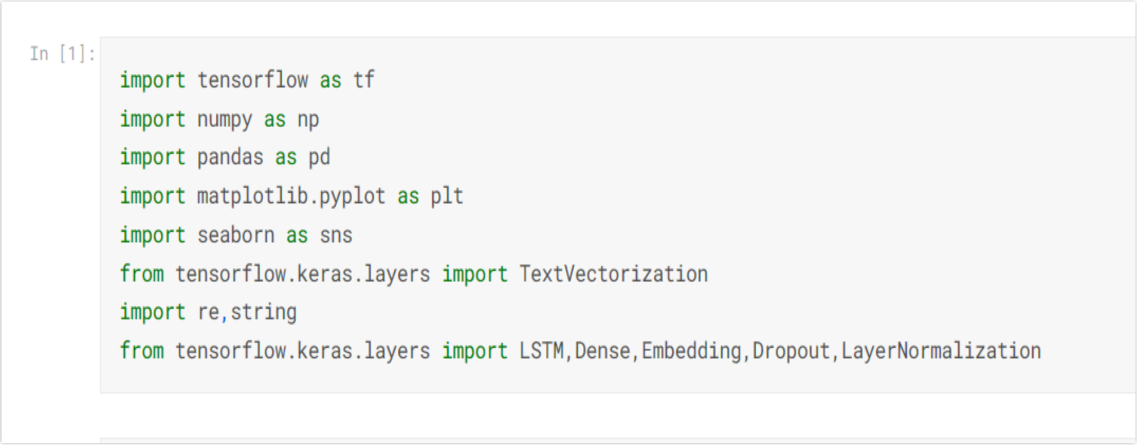
Expected update frequency Quarterly

Tags: Arts and Entertainment, Online Communities, Text, Text Mining

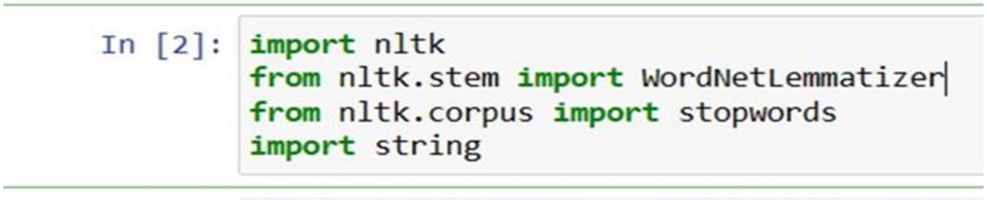
Data preprocessing:

Data preprocessing is a crucial step in the data mining and data analysis process that involves transforming raw data into a format that can be understood and analyzed by computers and machine learning algorithms.

Importing Libraries:



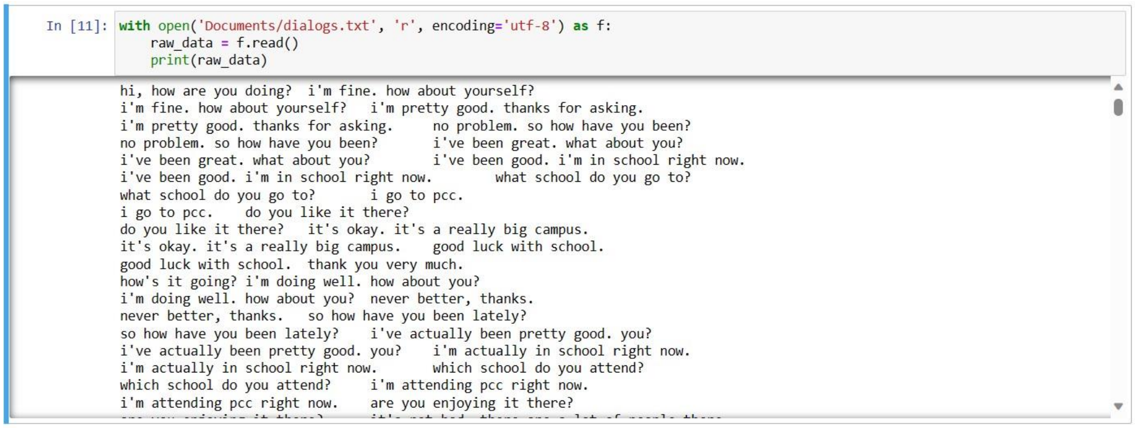
```
In [1]:
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.layers import TextVectorization
import re,string
from tensorflow.keras.layers import LSTM,Dense,Embedding,Dropout,LayerNormalization
```



```
In [2]: import nltk
        from nltk.stem import WordNetLemmatizer
        from nltk.corpus import stopwords
        import string
```

Statement segmentation:

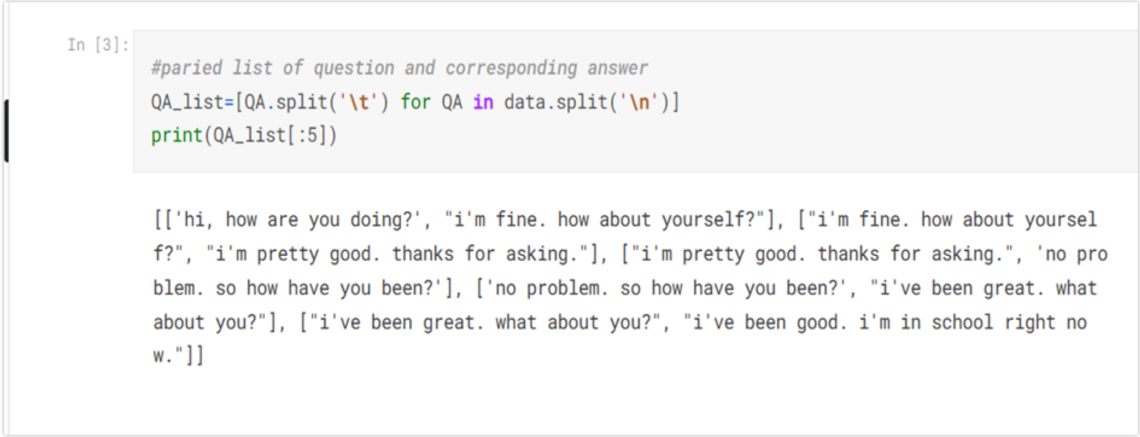
Reading the dataset



```
In [11]: with open('Documents/dialogs.txt', 'r', encoding='utf-8') as f:
             raw_data = f.read()
             print(raw_data)
```

```
hi, how are you doing?  i'm fine. how about yourself?
i'm fine. how about yourself?   i'm pretty good. thanks for asking.
i'm pretty good. thanks for asking.     no problem. so how have you been?
no problem. so how have you been?       i've been great. what about you?
i've been great. what about you?        i've been good. i'm in school right now.
i've been good. i'm in school right now.        what school do you go to?
what school do you go to?       i go to pcc.
i go to pcc.    do you like it there?
do you like it there?   it's okay. it's a really big campus.
it's okay. it's a really big campus.    good luck with school.
good luck with school.  thank you very much.
how's it going? i'm doing well. how about you?
i'm doing well. how about you?  never better, thanks.
never better, thanks.   so how have you been lately?
so how have you been lately?    i've actually been pretty good. you?
i've actually been pretty good. you?    i'm actually in school right now.
i'm actually in school right now.       which school do you attend?
which school do you attend?     i'm attending pcc right now.
i'm attending pcc right now.    are you enjoying it there?
```

Question and Answers:



```
In [3]:
        #paried list of question and corresponding answer
        QA_list=[QA.split('\t') for QA in data.split('\n')]
        print(QA_list[:5])
```

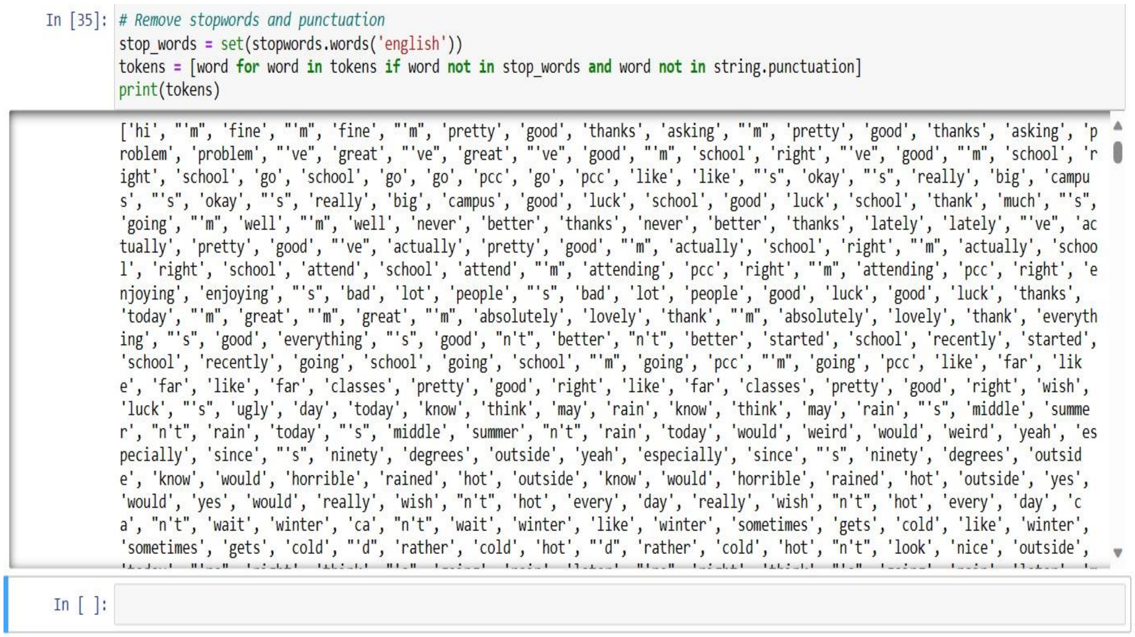
```
[['hi, how are you doing?', "i'm fine. how about yourself?"], ["i'm fine. how about yoursel
f?", "i'm pretty good. thanks for asking."], ["i'm pretty good. thanks for asking.", 'no pro
blem. so how have you been?'], ['no problem. so how have you been?', "i've been great. what
about you?"], ["i've been great. what about you?", "i've been good. i'm in school right no
w."]]
```

Normalization:

The goal of normalizing text is to *group related tokens together*, where tokens are usually the words in the text.

Depending on the text you are working with and the type of analysis you are doing, you might not need all of the normalization techniques in this post.

Removing Punctuations:

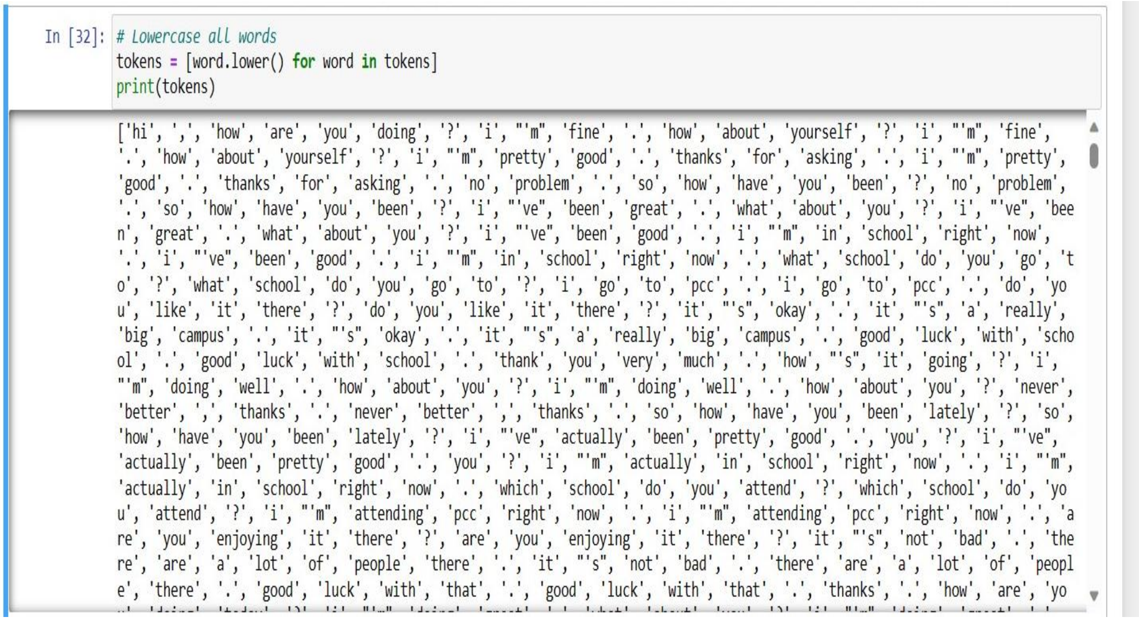


In [35]:
```
# Remove stopwords and punctuation
stop_words = set(stopwords.words('english'))
tokens = [word for word in tokens if word not in stop_words and word not in string.punctuation]
print(tokens)
```

```
['hi', "'m", 'fine', "'m", 'fine', "'m", 'pretty', 'good', 'thanks', 'asking', "'m", 'pretty', 'good', 'thanks', 'asking', 'p
roblem', 'problem', "'ve", 'great', "'ve", 'great', "'ve", 'good', "'m", 'school', 'right', "'ve", 'good', "'m", 'school', 'r
ight', 'school', 'go', 'school', 'go', 'go', 'pcc', 'go', 'pcc', 'like', 'like', "'s", 'okay', "'s", 'really', 'big', 'campu
s', "'s", 'okay', "'s", 'really', 'big', 'campus', 'good', 'luck', 'school', 'good', 'luck', 'school', 'thank', 'much', "'s",
'going', "'m", 'well', "'m", 'well', 'never', 'better', 'thanks', 'never', 'better', 'thanks', 'lately', 'lately', "'ve", 'ac
tually', 'pretty', 'good', "'ve", 'actually', 'pretty', 'good', "'m", 'actually', 'school', 'right', "'m", 'actually', 'schoo
l', 'right', 'school', 'attend', 'school', 'attend', "'m", 'attending', 'pcc', 'right', "'m", 'attending', 'pcc', 'right', 'e
njoying', 'enjoying', "'s", 'bad', 'lot', 'people', "'s", 'bad', 'lot', 'people', 'good', 'luck', 'good', 'luck', 'thanks',
'today', "'m", 'great', "'m", 'great', "'m", 'absolutely', 'lovely', 'thank', "'m", 'absolutely', 'lovely', 'thank', 'everyth
ing', "'s", 'good', 'everything', "'s", 'good', "n't", 'better', "n't", 'better', 'started', 'school', 'recently', 'started',
'school', 'recently', 'going', 'school', 'going', 'school', "'m", 'going', 'pcc', "'m", 'going', 'pcc', 'like', 'far', 'lik
e', 'far', 'like', 'far', 'classes', 'pretty', 'good', 'right', 'like', 'far', 'classes', 'pretty', 'good', 'right', 'wish',
'luck', "'s", 'ugly', 'day', 'today', 'know', 'think', 'may', 'rain', 'know', 'think', 'may', 'rain', "'s", 'middle', 'summe
r', "n't", 'rain', 'today', "'s", 'middle', 'summer', "n't", 'rain', 'today', 'would', 'weird', 'would', 'weird', 'yeah', 'es
pecially', 'since', "'s", 'ninety', 'degrees', 'outside', 'yeah', 'especially', 'since', "'s", 'ninety', 'degrees', 'outsid
e', 'know', 'would', 'horrible', 'rained', 'hot', 'outside', 'know', 'would', 'horrible', 'rained', 'hot', 'outside', 'yes',
'would', 'yes', 'would', 'really', 'wish', "n't", 'hot', 'every', 'day', 'really', 'wish', "n't", 'hot', 'every', 'day', 'c
a', "n't", 'wait', 'winter', 'ca', "n't", 'wait', 'winter', 'like', 'winter', 'sometimes', 'gets', 'cold', 'like', 'winter',
'sometimes', 'gets', 'cold', "'d", 'rather', 'cold', 'hot', "'d", 'rather', 'cold', 'hot', "n't", 'look', 'nice', 'outside',
```

In [ ]:

Converting Uppercase to Lowercase:

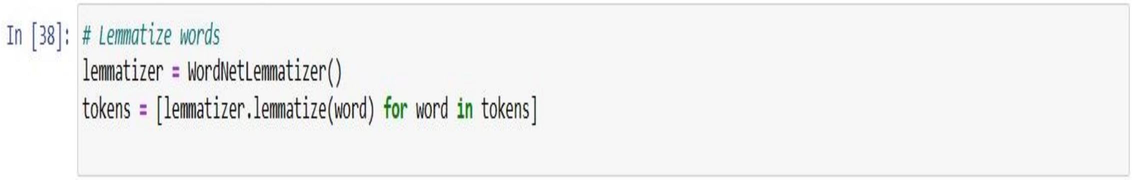


In [32]:
```
# Lowercase all words
tokens = [word.lower() for word in tokens]
print(tokens)
```

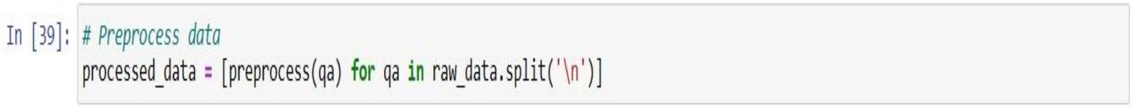
```
['hi', ',', 'how', 'are', 'you', 'doing', '?', 'i', "'m", 'fine', '.', 'how', 'about', 'yourself', '?', 'i', "'m", 'fine',
'.', 'how', 'about', 'yourself', '?', 'i', "'m", 'pretty', 'good', '.', 'thanks', 'for', 'asking', '.', 'i', "'m", 'pretty',
'good', '.', 'thanks', 'for', 'asking', '.', 'no', 'problem', '.', 'so', 'how', 'have', 'you', 'been', '?', 'no', 'problem',
'.', 'so', 'how', 'have', 'you', 'been', '?', 'i', "'ve", 'been', 'great', '.', 'what', 'about', 'you', '?', 'i', "'ve", 'bee
n', 'great', '.', 'what', 'about', 'you', '?', 'i', "'ve", 'been', 'good', '.', 'i', "'m", 'in', 'school', 'right', 'now',
'.', 'i', "'ve", 'been', 'good', '.', 'i', "'m", 'in', 'school', 'right', 'now', '.', 'what', 'school', 'do', 'you', 'go', 't
o', '?', 'what', 'school', 'do', 'you', 'go', 'to', '?', 'i', 'go', 'to', 'pcc', '.', 'i', 'go', 'to', 'pcc', '.', 'do', 'yo
u', 'like', 'it', 'there', '?', 'do', 'you', 'like', 'it', 'there', '?', 'it', "'s", 'okay', '.', 'it', "'s", 'a', 'really',
'big', 'campus', '.', 'it', "'s", 'okay', '.', 'it', "'s", 'a', 'really', 'big', 'campus', '.', 'good', 'luck', 'with', 'scho
ol', '.', 'good', 'luck', 'with', 'school', '.', 'thank', 'you', 'very', 'much', '.', 'how', "'s", 'it', 'going', '?', 'i',
"'m", 'doing', 'well', '.', 'how', 'about', 'you', '?', 'i', "'m", 'doing', 'well', '.', 'how', 'about', 'you', '?', 'never',
'better', ',', 'thanks', '.', 'never', 'better', ',', 'thanks', '.', 'so', 'how', 'have', 'you', 'been', 'lately', '?', 'so',
'how', 'have', 'you', 'been', 'lately', '?', 'i', "'ve", 'actually', 'been', 'pretty', 'good', '.', 'you', '?', 'i', "'ve",
'actually', 'been', 'pretty', 'good', '.', 'you', '?', 'i', "'m", 'actually', 'in', 'school', 'right', 'now', '.', 'i', "'m",
'actually', 'in', 'school', 'right', 'now', '.', 'which', 'school', 'do', 'you', 'attend', '?', 'which', 'school', 'do', 'yo
u', 'attend', '?', 'i', "'m", 'attending', 'pcc', 'right', 'now', '.', 'i', "'m", 'attending', 'pcc', 'right', 'now', '.', 'a
re', 'you', 'enjoying', 'it', 'there', '?', 'are', 'you', 'enjoying', 'it', 'there', '?', 'it', "'s", 'not', 'bad', '.', 'the
re', 'are', 'a', 'lot', 'of', 'people', 'there', '.', 'it', "'s", 'not', 'bad', '.', 'there', 'are', 'a', 'lot', 'of', 'peopl
e', 'there', '.', 'good', 'luck', 'with', 'that', '.', 'good', 'luck', 'with', 'that', '.', 'thanks', '.', 'how', 'are', 'yo
```

Tokenization:

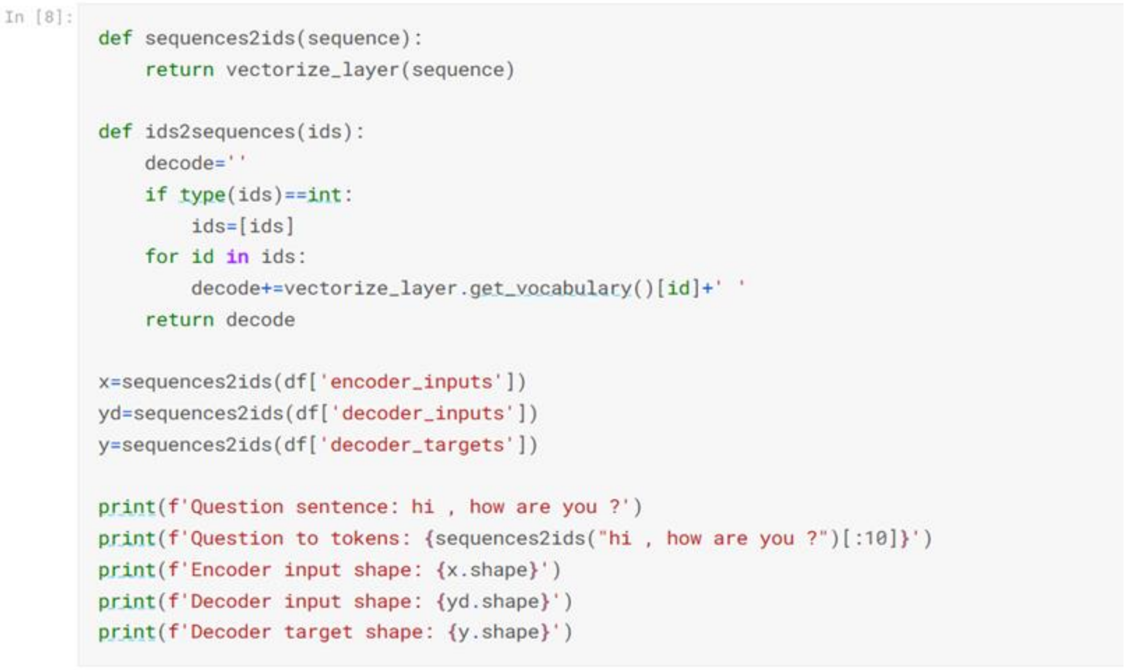
Tokenization, when applied to data security, is the process of substituting a sensitive data element with a non-sensitive equivalent, referred to as a token, that has no intrinsic or exploitable meaning or value.



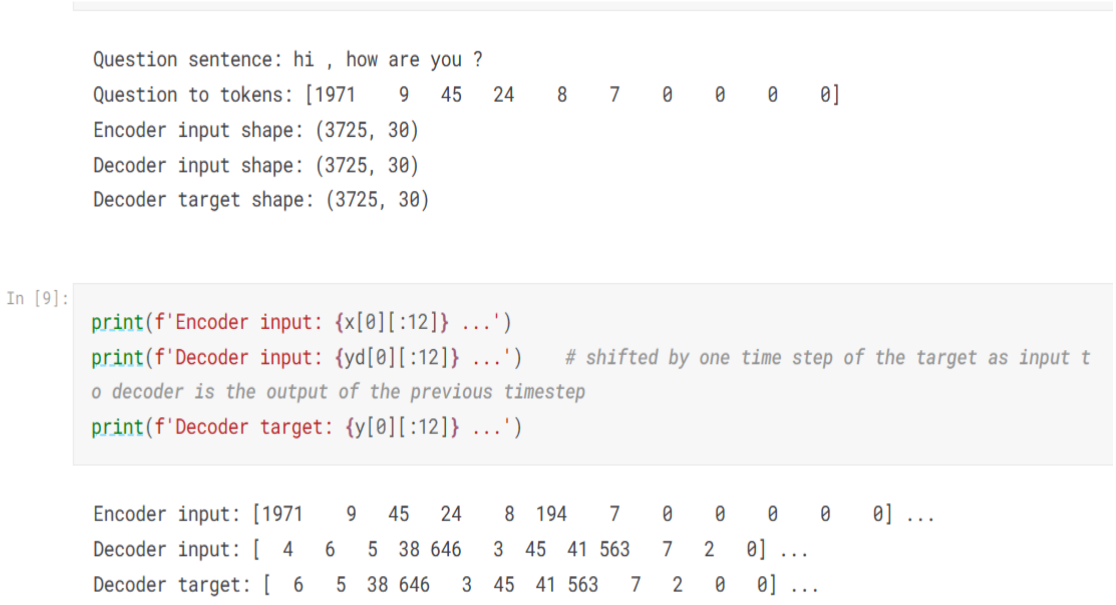
```
In [38]: # Lemmatize words
         lemmatizer = WordNetLemmatizer()
         tokens = [lemmatizer.lemmatize(word) for word in tokens]
```



```
In [39]: # Preprocess data
         processed_data = [preprocess(qa) for qa in raw_data.split('\n')]
```



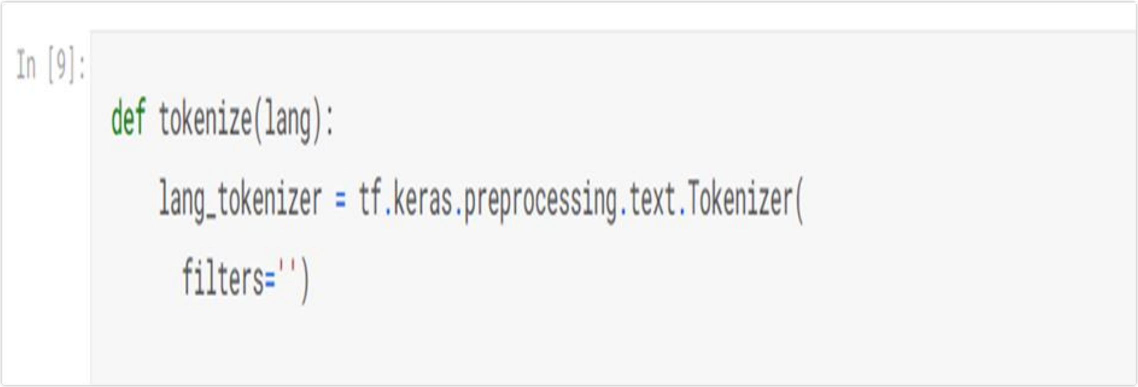
```
In [8]:
        def sequences2ids(sequence):
            return vectorize_layer(sequence)

        def ids2sequences(ids):
            decode=''
            if type(ids)==int:
                ids=[ids]
            for id in ids:
                decode+=vectorize_layer.get_vocabulary()[id]+' '
            return decode

        x=sequences2ids(df['encoder_inputs'])
        yd=sequences2ids(df['decoder_inputs'])
        y=sequences2ids(df['decoder_targets'])

        print(f'Question sentence: hi , how are you ?')
        print(f'Question to tokens: {sequences2ids("hi , how are you ?")[:10]}')
        print(f'Encoder input shape: {x.shape}')
        print(f'Decoder input shape: {yd.shape}')
        print(f'Decoder target shape: {y.shape}')
```



```
Question sentence: hi , how are you ?
Question to tokens: [1971    9   45   24    8    7    0    0    0    0]
Encoder input shape: (3725, 30)
Decoder input shape: (3725, 30)
Decoder target shape: (3725, 30)
```

```
In [9]:
print(f'Encoder input: {x[0][:12]} ...')
print(f'Decoder input: {yd[0][:12]} ...')    # shifted by one time step of the target as input t
o decoder is the output of the previous timestep
print(f'Decoder target: {y[0][:12]} ...')
```

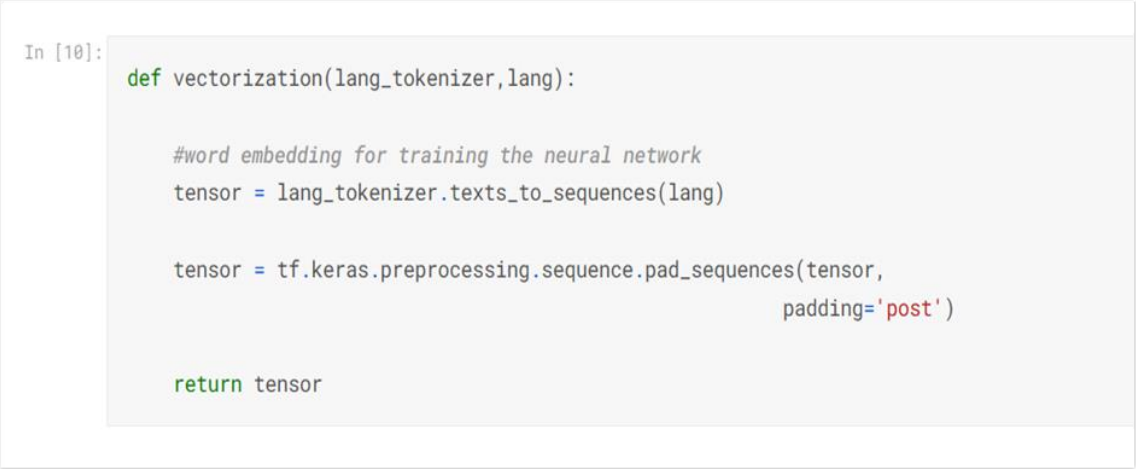
```
Encoder input: [1971    9   45   24    8  194    7    0    0    0    0    0] ...
Decoder input: [   4    6    5   38  646    3   45   41  563    7    2    0] ...
Decoder target: [   6    5   38  646    3   45   41  563    7    2    0    0] ...
```



```
In [9]:
def tokenize(lang):
    lang_tokenizer = tf.keras.preprocessing.text.Tokenizer(
      filters='')
```

Word Embedding:

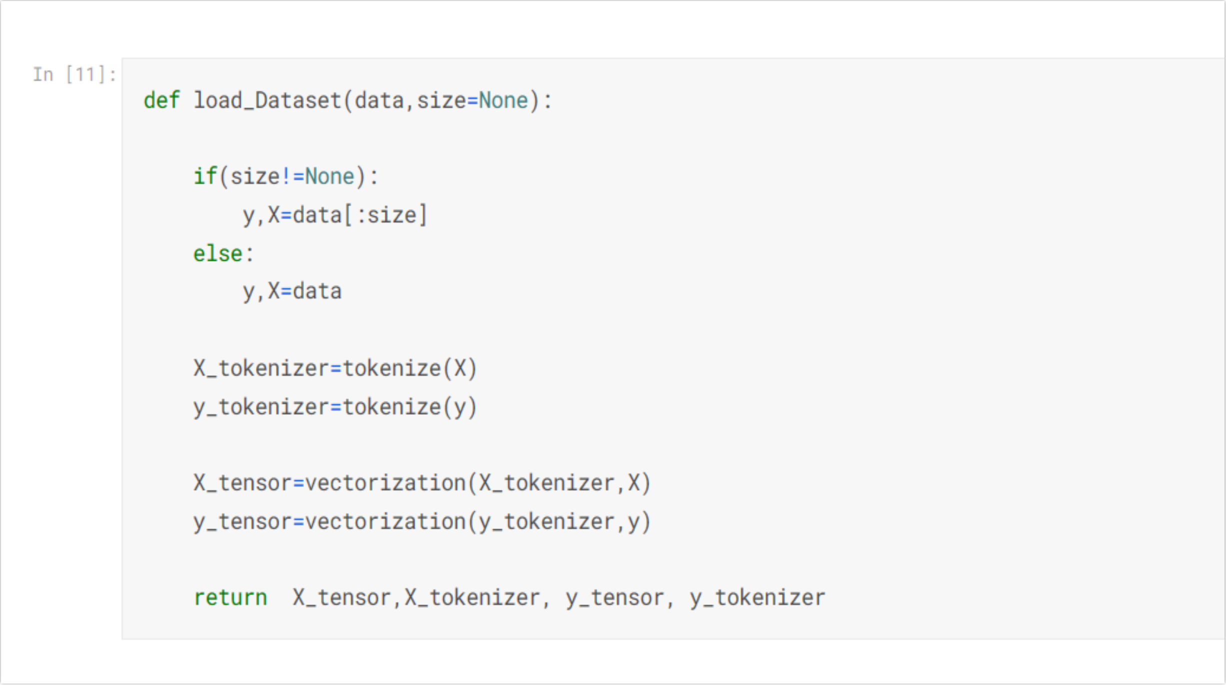
It is an approach for representing words and documents. Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. It allows words with similar meanings to have a similar representation. They can also approximate meaning. A word vector with 50 values can represent 50 unique features.



In [10]:

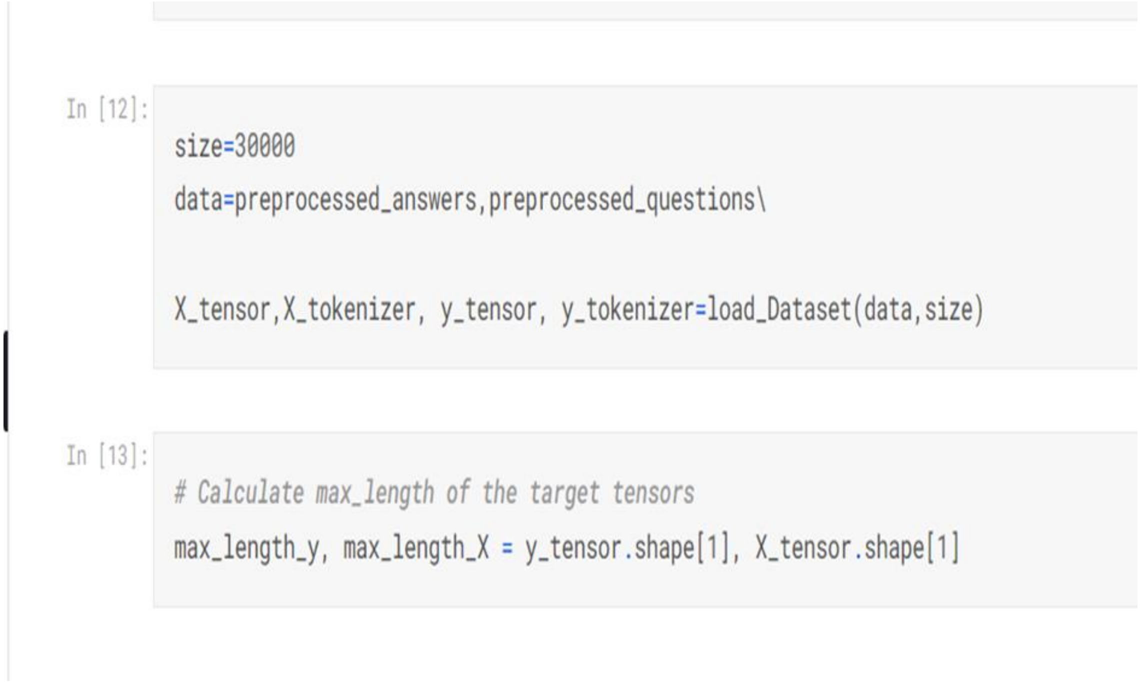
```
def vectorization(lang_tokenizer,lang):

    #word embedding for training the neural network
    tensor = lang_tokenizer.texts_to_sequences(lang)

    tensor = tf.keras.preprocessing.sequence.pad_sequences(tensor,
                                                           padding='post')

    return tensor
```

Loading dataset :



In [11]:

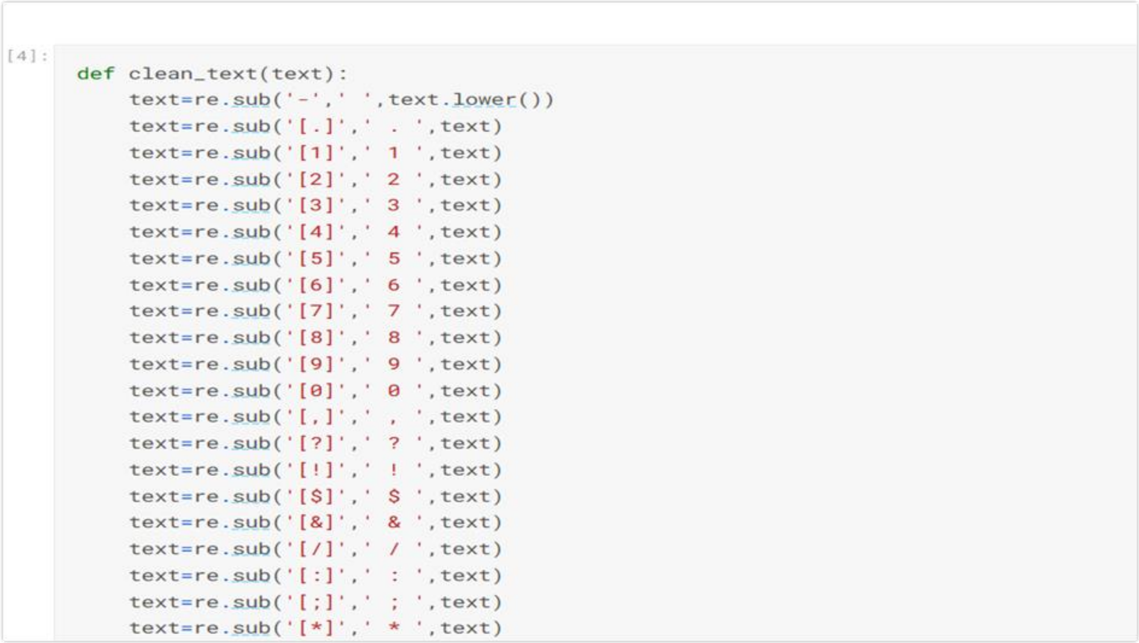
```
def load_Dataset(data,size=None):

    if(size!=None):
        y,X=data[:size]
    else:
        y,X=data

    X_tokenizer=tokenize(X)
    y_tokenizer=tokenize(y)

    X_tensor=vectorization(X_tokenizer,X)
    y_tensor=vectorization(y_tokenizer,y)

    return  X_tensor,X_tokenizer, y_tensor, y_tokenizer
```



```
In [12]:
size=30000
data=preprocessed_answers,preprocessed_questions\


X_tensor,X_tokenizer, y_tensor, y_tokenizer=load_Dataset(data,size)
```

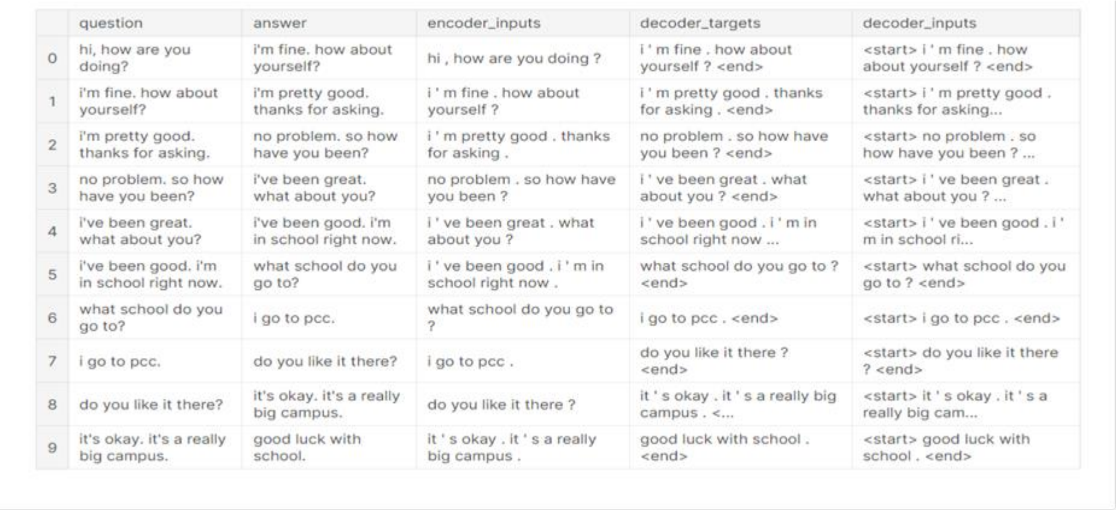
```
In [13]:
# Calculate max_length of the target tensors
max_length_y, max_length_X = y_tensor.shape[1], X_tensor.shape[1]
```

Text Cleaning:



```
[4]:
def clean_text(text):
    text=re.sub('-',' ',text.lower())
    text=re.sub('[.]',' . ',text)
    text=re.sub('[1]',' 1 ',text)
    text=re.sub('[2]',' 2 ',text)
    text=re.sub('[3]',' 3 ',text)
    text=re.sub('[4]',' 4 ',text)
    text=re.sub('[5]',' 5 ',text)
    text=re.sub('[6]',' 6 ',text)
    text=re.sub('[7]',' 7 ',text)
    text=re.sub('[8]',' 8 ',text)
    text=re.sub('[9]',' 9 ',text)
    text=re.sub('[0]',' 0 ',text)
    text=re.sub('[,]',' , ',text)
    text=re.sub('[?]',' ? ',text)
    text=re.sub('[!]',' ! ',text)
    text=re.sub('[$]',' $ ',text)
    text=re.sub('[&]',' & ',text)
    text=re.sub('[/]',' / ',text)
    text=re.sub('[:]',' : ',text)
    text=re.sub('[;]',' ; ',text)
    text=re.sub('[*]',' * ',text)
```

Output:



	question	answer	encoder_inputs	decoder_targets	decoder_inputs
0	hi, how are you doing?	i'm fine. how about yourself?	hi , how are you doing ?	i ' m fine . how about yourself ? <end>	<start> i ' m fine . how about yourself ? <end>
1	i'm fine. how about yourself?	i'm pretty good. thanks for asking.	i ' m fine . how about yourself ?	i ' m pretty good . thanks for asking . <end>	<start> i ' m pretty good . thanks for asking...
2	i'm pretty good. thanks for asking.	no problem. so how have you been?	i ' m pretty good . thanks for asking .	no problem . so how have you been ? <end>	<start> no problem . so how have you been ? ...
3	no problem. so how have you been?	i've been great. what about you?	no problem . so how have you been ?	i ' ve been great . what about you ? <end>	<start> i ' ve been great . what about you ? ...
4	i've been great. what about you?	i've been good. i'm in school right now.	i ' ve been great . what about you ?	i ' ve been good . i ' m in school right now ...	<start> i ' ve been good . i ' m in school ri...
5	i've been good. i'm in school right now.	what school do you go to?	i ' ve been good . i ' m in school right now .	what school do you go to ? <end>	<start> what school do you go to ? <end>
6	what school do you go to?	i go to pcc.	what school do you go to ?	i go to pcc . <end>	<start> i go to pcc . <end>
7	i go to pcc.	do you like it there?	i go to pcc .	do you like it there ? <end>	<start> do you like it there ? <end>
8	do you like it there?	it's okay. it's a really big campus.	do you like it there ?	it ' s okay . it ' s a really big campus . <...	<start> it ' s okay . it ' s a really big cam...
9	it's okay. it's a really big campus.	good luck with school.	it ' s okay . it ' s a really big campus .	good luck with school . <end>	<start> good luck with school . <end>

Conclusion:

Preprocessing is an essential step in chatbot development. It involves cleaning and normalizing the data, removing irrelevant information, and tokenizing the text. Preprocessing helps optimize inputs and outputs for better results . Once data is collected for training a chatbot, it’s important to pre-process it to ensure it’s clean and ready for use. And creation of virtual environment is an most important part of building chatbot.