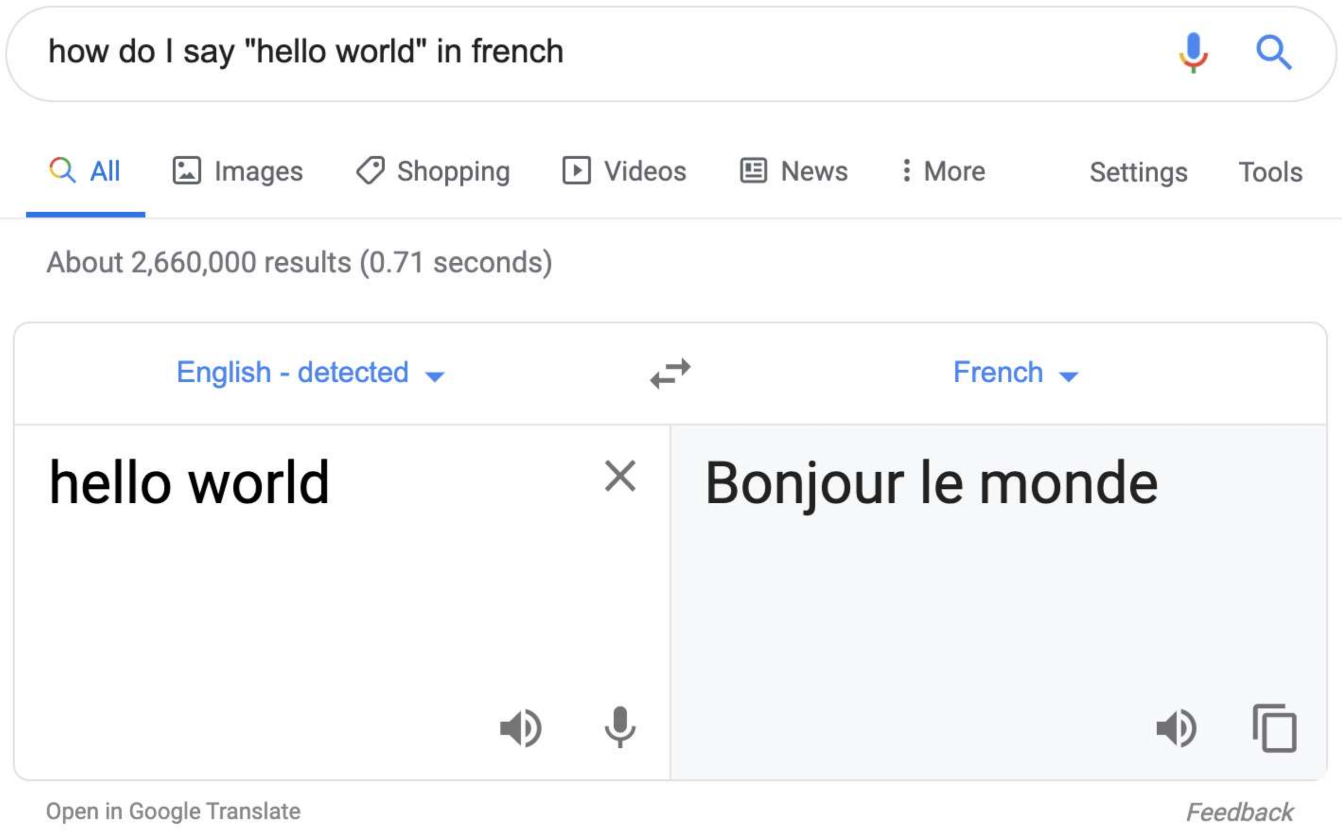
English French-Translator

Savez-vous comment créer une application de traduction de langue?
Did you understand the above sentence?
Well after googling it, I found its meaning as:
Do you know how to create a language translator app?

We all know about Google Translate which allows us to convert from one language to another and it’s very useful for learning and understanding new languages.


how do I say "hello world" in french

About 2,660,000 results (0.71 seconds)

hello world

Bonjour le monde

In this project we aim to convert English phrases to French using RNN on Deep Learning Neural Network

Introduction

In this notebook, you will build a deep neural network that functions as part of an end-to-end machine translation pipeline. Your completed pipeline will accept English text as input and return the French translation.

Preprocess - You'll convert text to sequence of integers. Models Create models which accepts a sequence of integers as input and returns a probability distribution over possible translations. After learning about the basic types of neural networks that are often used for machine translation, you will engage in your own investigations, to design your own model! Prediction Run the model on English text.

Now let's start by importing necessary libraries.

- Importing numpy for working with arrays
- It then defines a Tokenizer object that will be used to split text into individual words using tensorflow tokenizer. Refer:https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer
- pad_sequences is used to ensure that all sequences in a list have the same length. Refer:https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/sequence/pad_sequences
- Keras model represents the actual neural network model.
- the Sequential model is a linear stack of layers.
- Importing GRU,Input,Dense,TimeDistributed,Activation,RepeatVector,Bidirectional,Dropout,LSTM,Embedding from tensorflow layers. Refer:https://www.tensorflow.org/api_docs/python/tf/keras/layers

```
#Now import above mentioned libraries
```

Load Data

The data is located in data/small_vocab_en and data/small_vocab_fr. The small_vocab_en file contains English sentences with their French translations in the small_vocab_fr file. Load the English and French data from these files from running the cell below.

```
#load data here
#english data

#french data
```

- The OS module in Python provides functions for interacting with the operating system
- The code loads the data from a file called input_file. The code then splits the string of text into an array using split().

Then, it uses list comprehension to create a list with each line in the array as its own element.

```
#import os

#define a function with one parameter path

  #join the path with join keyword

  #open file and read  as f

    #read file

  #return with data split("\n")
```

Now loading all english and french data into variables.

```
#Now loading english data

#Now loading french data
```

Analysis of Dataset

Let us look at a few examples in the dataset of both language

```
#iterate over range of 5 in english and french


  #print english sentences

  #print french sentences
```

```
Sample : 0
new jersey is sometimes quiet during autumn , and it is snowy in april .
new jersey est parfois calme pendant l' automne , et il est neigeux en avril .
--------------------------------------------------
Sample : 1
the united states is usually chilly during july , and it is usually freezing in novem
les états-unis est généralement froid en juillet , et il gèle habituellement en novem
--------------------------------------------------
Sample : 2
california is usually quiet during march , and it is usually hot in june .
california est généralement calme en mars , et il est généralement chaud en juin .
--------------------------------------------------
Sample : 3
the united states is sometimes mild during june , and it is cold in september .
les états-unis est parfois légère en juin , et il fait froid en septembre .
```

```
--------------------------------------------------
Sample : 4
your least liked fruit is the grape , but my least liked is the apple .
votre moins aimé fruit est le raisin , mais mon moins aimé est la pomme .
--------------------------------------------------
```

Convert to Vocabulary

The complexity of the problem is determined by the complexity of the vocabulary. A more complex vocabulary is a more complex problem. Let's look at the complexity of the dataset we'll be working with.

A counter is a container that stores elements as dictionary keys, and their counts are stored as dictionary values.
Refer:https://docs.python.org/3/library/collections.html

```
#importing collections
```

```
#we will check for english vocabulary
#first we will iterate through english sentence
#then we will split that words and then we will use counter function
#refer more information in documentation
#we done for english vocab
#english_words_counter = collections.Counter([word for sentence in english_sentences for w
```

```
English Vocab: 227
French Vocab: 355
```

Tokenize (IMPLEMENTATION)

For a neural network to predict on text data, it first has to be turned into data it can understand. Text data like "dog" is a sequence of ASCII character encodings. Since a neural network is a series of multiplication and addition operations, the input data needs to be numbers.

We can turn each character into a number or each word into a number. These are called character and word ids, respectively. Character ids are used for character level models that generate text predictions for each character. A word level model uses word ids that generate text predictions for each word. Word level models tend to learn better, since they are lower in complexity, so we'll use those.

Turn each sentence into a sequence of words ids using Keras's Tokenizer function. Use this function to tokenize english_sentences and french_sentences in the cell below.

```
#The code tokenizes a string and returns the tokens as well as the text that was tokenized
#define a function

  #create a object of tokenizer

  #fit the data

  #return the  tokenizer.texts_to_sequences(x), tokenizer
```

- The code starts by tokenizing the text_sentences list into individual sentences.Then, it prints out the word index of each sentence in the text_tokenized list.
  Next, it iterates through each sentence and prints out a sample output for that sentence.

```
# Tokenize Sample output
text_sentences = [
    'The quick brown fox jumps over the lazy dog .',
    'By Jove , my quick study of lexicography won a prize .',
    'This is a short sentence .']

#pass the sample text into tokenize[text_tokenized, text_tokenizer = tokenize(text_sentenc

#print text_tokenizer.word_index


#iterate over sample text and text_tokeinzed

  #print sequence

  #print input

  #print Output
```

```
{'the': 1, 'quick': 2, 'a': 3, 'brown': 4, 'fox': 5, 'jumps': 6, 'over': 7, 'lazy': 8

Sequence 1 in x
  Input:  The quick brown fox jumps over the lazy dog .
  Output: [1, 2, 4, 5, 6, 7, 1, 8, 9]
Sequence 2 in x
  Input:  By Jove , my quick study of lexicography won a prize .
  Output: [10, 11, 12, 2, 13, 14, 15, 16, 3, 17]
Sequence 3 in x
  Input:  This is a short sentence .
  Output: [18, 19, 3, 20, 21]
```

Padding (IMPLEMENTATION)

When batching the sequence of word ids together, each sequence needs to be the same length. Since sentences are dynamic in length, we can add padding to the end of the sequences to make them the same length.

Make sure all the English sequences have the same length and all the French sequences have the same length by adding padding to the end of each sequence using Keras's pad_sequences function.

```
#define a pad function and pass the x parameter

  #return pad_sequences with x and maxlength and keep padding =post
```

- The code is used to preprocess the input data set. - The tokenize function splits the text into individual tokens, which are then passed to a function called pad that takes in a list of tokens and pads them with a specified character (in this case, spaces).

```
#define preproces function with x and y

    #preproces the data of x

    #preproces the data of y


    #padding the data x

    #padding the data y


    # Keras's sparse_categorical_crossentropy function requires the labels to be in 3 dime
    #Expanding dimensions


    #return preprocess_x, preprocess_y, x_tk, y_tk


#preproc_english_sentences, preproc_french_sentences, english_tokenizer, french_tokenizer
    #preprocess(english_sentences, french_sentences)


#print Max English sentence length

#print Max french sentence length

#print len of english vocabulary

#print len of french vocabulary
```

```
Data Preprocessed
Max English sentence length: 15
Max French sentence length: 21
English vocabulary size: 199
French vocabulary size: 344
```

Create Model

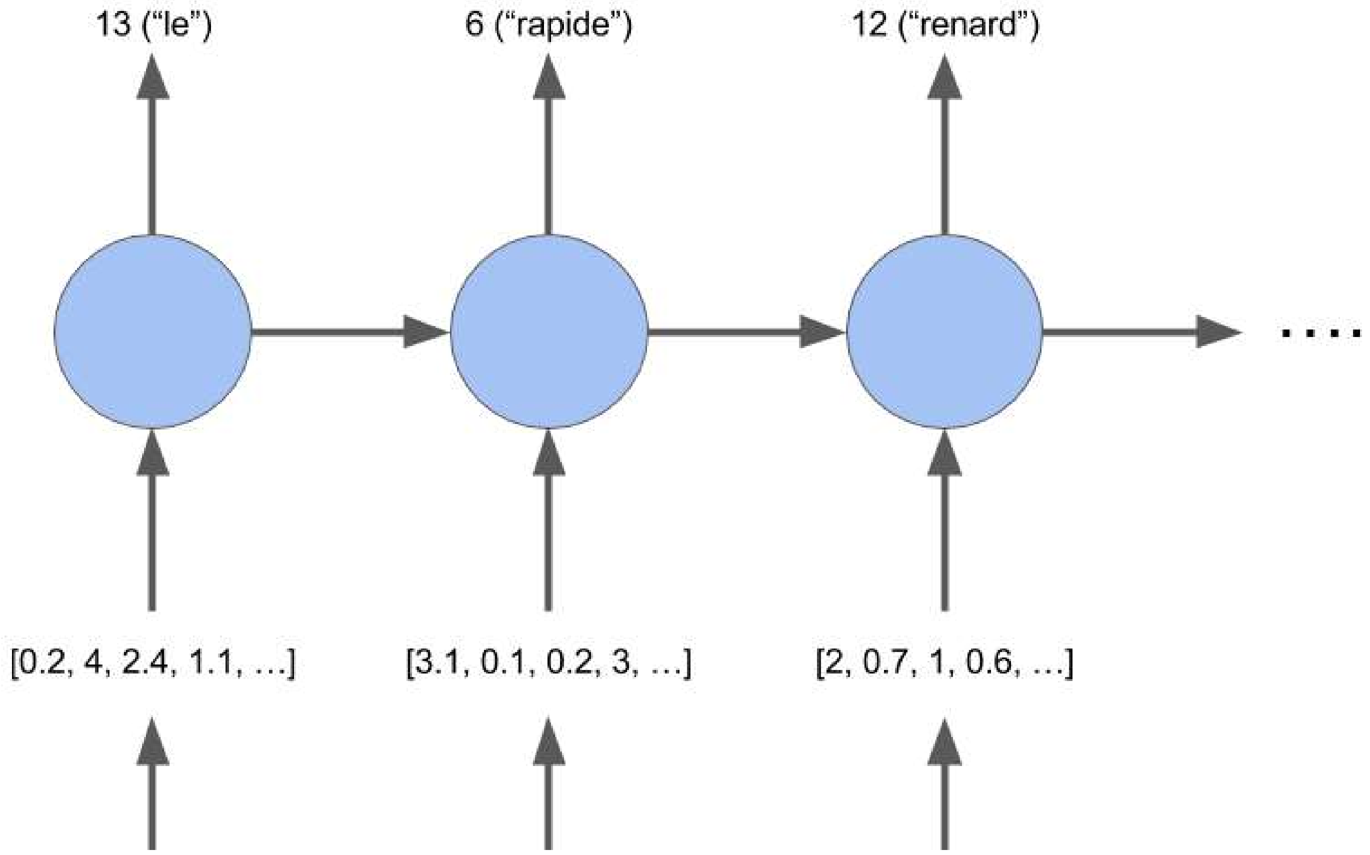
In this section, you will experiment with various neural network architectures. You will begin by training four relatively simple architectures.

Model 1 is a simple RNN Model 2 is a RNN with Embedding Model 3 is a Bidirectional RNN Model 4 is an optional Encoder-Decoder RNN After experimenting with the four simple architectures, you will construct a deeper architecture that is designed to outperform all four models.

Ids Back to Text

The neural network will be translating the input to words ids, which isn't the final form we want. We want the French translation. The function logits_to_text will bridge the gab between the logits from the neural network to the French translation. You'll be using this function to better understand the output of the neural network.

```
#define a function of logits to text with 2 parameters logits, tokenizer


  #vThe code takes in an index_to_words dictionary which maps each word to its correspondi
  #Then, it uses this dictionary to find all the words with ids 0-9 and prints them

  #make index_to_words[0] = '<PAD>'


  #So basically we are predicting output for a given word and then selecting best answer
  #Then selecting that label we reverse-enumerate the word from id
```



Building Model

Here we use RNN model combined with GRU nodes for translation

- The code starts by defining the input shape and output sequence length.Next, it defines the number of unique English words in the dataset and French words in the dataset.The code then builds a Keras model using word embedding on x and y.It also sets hyperparameters for learning rate, which is 0.005, as well as building layers for this model.Finally, it compiles this model with sparse_categorical_crossentropy loss function and Adam optimizer with learning rate set to 0.005. The code will create a Keras model that has been trained to recognize words in English and French.

```
def embed_model(input_shape, output_sequence_length, english_vocab_size, french_vocab_size
    """
    Build and train a RNN model using word embedding on x and y
    :param input_shape: Tuple of input shape
    :param output_sequence_length: Length of output sequence
    :param english_vocab_size: Number of unique English words in the dataset
    :param french_vocab_size: Number of unique French words in the dataset
    :return: Keras model built, but not trained
    """
    # TODO: Implement

    # Hyperparameters
    #learning_rate = 0.005


    # TODO: Build the layers
    #create model Sequential

    #add embedding layer with english_vocab_size, 256, input_length=input_shape[1], input_

    #add GRU layer of 256

    #add TimeDistribute layer dense of 1024 and applying of activation function of relu

    #adding dropout layer

    #add TimeDistributed Dense layer of french_vocab_size and applying softmax activation


    # Compile model
    # with loss function of activation function with adam optimizer and accuaray metrix



    #return model



# Reshaping the input to work with a basic RNN
#tmp_x = pad(preproc_english_sentences, preproc_french_sentences.shape[1])

#tmp_x = tmp_x.reshape((-1, preproc_french_sentences.shape[-2]))
```

Finally calling the model function

```
#calling our model
#with tmp_x.shape, preproc_french_sentences.shape[1], len(english_tokenizer.word_index)+1,
```

Printing model summary

```
#print Model summary
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 21, 256)           51200
_________________________________________________________________
gru_1 (GRU)                  (None, 21, 256)           394752
_________________________________________________________________
time_distributed_2 (TimeDist (None, 21, 1024)          263168
_________________________________________________________________
dropout_1 (Dropout)          (None, 21, 1024)          0
_________________________________________________________________
time_distributed_3 (TimeDist (None, 21, 345)           353625
=================================================================
Total params: 1,062,745
Trainable params: 1,062,745
Non-trainable params: 0
_________________________________________________________________
```

Training the model

Here we start to train the model and pass the english text and the max_sequence_length, with vocab size for both english and french text

```
#model train with tmp_x, preproc_french_sentences, batch_size=1024, epochs=20, validation_
```

```
Train on 110288 samples, validate on 27572 samples
Epoch 1/20
110288/110288 [==============================] - 18s 166us/sample - loss: 1.3575 - ac
Epoch 2/20
110288/110288 [==============================] - 15s 138us/sample - loss: 0.3948 - ac
Epoch 3/20
110288/110288 [==============================] - 14s 131us/sample - loss: 0.2839 - ac
Epoch 4/20
110288/110288 [==============================] - 15s 132us/sample - loss: 0.2372 - ac
Epoch 5/20
110288/110288 [==============================] - 14s 131us/sample - loss: 0.2172 - ac
Epoch 6/20
110288/110288 [==============================] - 14s 131us/sample - loss: 0.2031 - ac
Epoch 7/20
110288/110288 [==============================] - 15s 132us/sample - loss: 0.1936 - ac
Epoch 8/20
110288/110288 [==============================] - 14s 130us/sample - loss: 0.1873 - ac
Epoch 9/20
110288/110288 [==============================] - 14s 130us/sample - loss: 0.1820 - ac
Epoch 10/20
110288/110288 [==============================] - 14s 130us/sample - loss: 0.1777 - ac
Epoch 11/20
110288/110288 [==============================] - 14s 130us/sample - loss: 0.1741 - ac
Epoch 12/20
110288/110288 [==============================] - 14s 130us/sample - loss: 0.1751 - ac
Epoch 13/20
110288/110288 [==============================] - 14s 130us/sample - loss: 0.1731 - ac
Epoch 14/20
110288/110288 [==============================] - 14s 130us/sample - loss: 0.1694 - ac
Epoch 15/20
110288/110288 [==============================] - 14s 129us/sample - loss: 0.1692 - ac
Epoch 16/20
110288/110288 [==============================] - 14s 129us/sample - loss: 0.1720 - ac
Epoch 17/20
110288/110288 [==============================] - 14s 129us/sample - loss: 0.1676 - ac
Epoch 18/20
110288/110288 [==============================] - 14s 129us/sample - loss: 0.1626 - ac
Epoch 19/20
```

```
110288/110288 [==============================] - 14s 130us/sample - loss: 0.1636 - ac
Epoch 20/20
110288/110288 [==============================] - 14s 129us/sample - loss: 0.1631 - ac
```

Saving our model

```
#save our model
```

Arbitrary Predictions

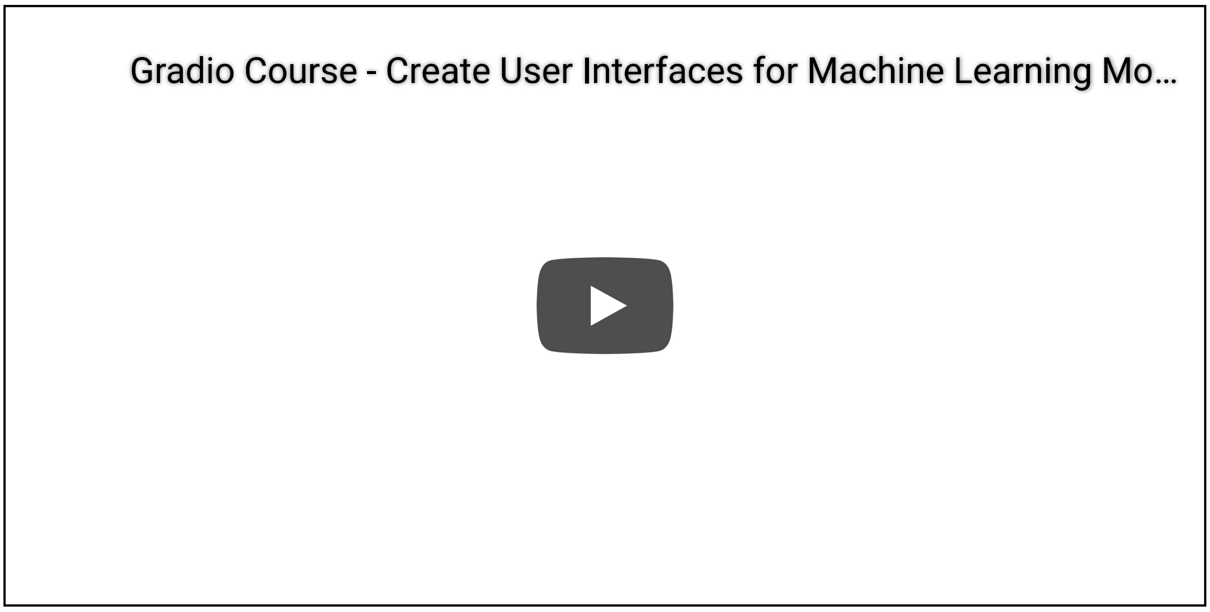
Performing predictions on the models using User Input.

```
#define a function for final_predictions with text timetable.


  #converting y_ids to words [y_id_to_word = {value: key for key, value in french_tokenize

  #y_id_to_word[0] = '<PAD>'


  #spliting our english word and tokezing

  #padding our sentences

  #converting into text

  #avoiding of <PAD> part in output
  #create a variable of text

  #iterate over text split

    #if i == <pad>

      #then break

    #else

      #text2=text2+" "+i

  #return text
```

Implementation

Enter your input here to get predictions. We will using Gradio for implementation part. Refer video for detailed information

```
####Refer Video
from IPython.display import YouTubeVideo
YouTubeVideo('RiCQzBluTxU', width=600, height=300)
```


Gradio Course - Create User Interfaces for Machine Learning Mo…

```
#pip install gradio
```

```
    Requirement already satisfied: gradio in /usr/local/lib/python3.7/dist-packages (2.7
    Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (
    Requirement already satisfied: markdown2 in /usr/local/lib/python3.7/dist-packages (f
    Requirement already satisfied: paramiko in /usr/local/lib/python3.7/dist-packages (fr
    Requirement already satisfied: aiohttp in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: python-multipart in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (fr
    Requirement already satisfied: ffmpy in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: uvicorn in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: pydub in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: fastapi in /usr/local/lib/python3.7/dist-packages (fro
    Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: pycryptodome in /usr/local/lib/python3.7/dist-packages
    Requirement already satisfied: analytics-python in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from
    Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3
    Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/python3.7/d
    Requirement already satisfied: asynctest==0.13.0 in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.7/dist-p
    Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.7/dist-pac
    Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.7/dist-packag
    Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.7/dist-package
    Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3
    Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.7/dist-pack
    Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.7/dist-packages (f
    Requirement already satisfied: backoff==1.10.0 in /usr/local/lib/python3.7/dist-packa
    Requirement already satisfied: monotonic>=1.5 in /usr/local/lib/python3.7/dist-packag
    Requirement already satisfied: python-dateutil>2.1 in /usr/local/lib/python3.7/dist-p
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: starlette==0.17.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pydantic!=1.7,!=1.7.1,!=1.7.2,!=1.7.3,!=1.8,!=1.8.1,<2
Requirement already satisfied: anyio<4,>=3.0.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: bcrypt>=3.1.3 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cryptography>=2.5 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pynacl>=1.0.1 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cffi>=1.1 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: click>=7.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: asgiref>=3.4.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: h11>=0.8 in /usr/local/lib/python3.7/dist-packages (fr
```

After installing, we import gradio as gr

```
#impprt gradio as gr
```

- The code creates an interface with a function called final_predictions. - The inputs of the interface are a textbox that has two lines and a placeholder, which is "Text to translate". - The outputs of the interface are "text". - The code launches the program in debug mode.

```
#creating interface for gradio
```

```
#lanuch the interface
```

```
Colab notebook detected. This cell will run indefinitely so that you can see errors a
Running on public URL: https://34921.gradio.app

This share link expires in 72 hours. For free permanent hosting, check out Spaces (ht
```

gradio

No interface is running right now

```
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/gradio/app.py", line 200, in predict
    app.interface.process, raw_input
  File "/usr/local/lib/python3.7/dist-packages/starlette/concurrency.py", line 39, in
    return await anyio.to_thread.run_sync(func, *args)
  File "/usr/local/lib/python3.7/dist-packages/anyio/to_thread.py", line 29, in run_s
    limiter=limiter)
  File "/usr/local/lib/python3.7/dist-packages/anyio/_backends/_asyncio.py", line 818
    return await future
  File "/usr/local/lib/python3.7/dist-packages/anyio/_backends/_asyncio.py", line 754
    result = context.run(func, *args)
  File "/usr/local/lib/python3.7/dist-packages/gradio/interface.py", line 531, in pro
    processed_input, return_duration=True
  File "/usr/local/lib/python3.7/dist-packages/gradio/interface.py", line 487, in run
    prediction = predict_fn(*processed_input)
  File "<ipython-input-44-e58e1c4dcc89>", line 5, in final_predictions
    sentence = [english_tokenizer.word_index[word] for word in text.split()]
  File "<ipython-input-44-e58e1c4dcc89>", line 5, in <listcomp>
    sentence = [english_tokenizer.word_index[word] for word in text.split()]
KeyError: 'm'
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/gradio/app.py", line 200, in predict
    app.interface.process, raw_input
  File "/usr/local/lib/python3.7/dist-packages/starlette/concurrency.py", line 39, in
```

We have come to an end of this project but don't stop here, try as many projects of the similar type to get a better understanding of the use cases. Solve the practice sheet of this project to test yourself.!!