AIML CAPSTONE PROJECT

PROJECT NAME : CAPSTONE PROJECT - NLP

GROUP NAME : GROUP 12-NLP2-CAPSTONE-DEC22A

GROUP MEMBERS : DEVESH PAREEK
NAVITHA G
PRASHANTAGARWAL
SUKRITI MODGIL
VIJI
VISHWANATH SHEBINAKATTI

TABLE OF CONTENTS

INTRODUCTION :

Machine translation (MT) is the process of translating text in one language to another language with the use of software by incorporating both computational and linguistic knowledge.

NMT is a recently formulated method for automatic translation with the help of deep neural networks. NMT uses a single large neural network for training. This structure comprised of encoder and decoder networks where the encoder consumes the input sentences to produce a vector representation, and the decoder takes this vector and outputs the target language words. Generally, both encoder and decoder networks are designed using the recurrent neural networks (RNN) or long short-term memory (LSTM) or gated recurrent unit (GRU) or bidirectional RNN, which are the alternatives to RNN. Even though RNN, especially LSTM, is theoretically proven for handling long-term dependencies in the sentences

Here, in this case study we use a simple RNN model combined with lstm nodes for our translation task.

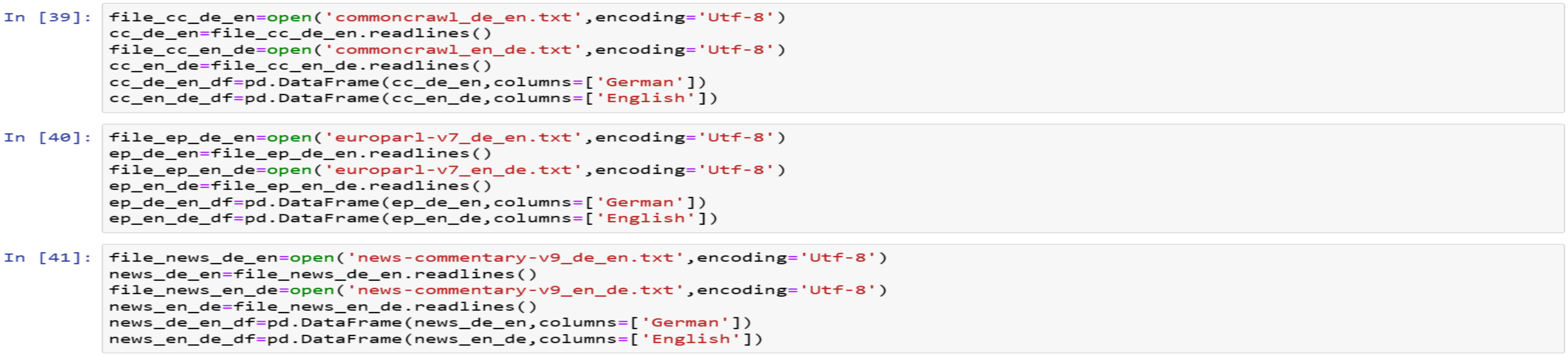
1. Dataset Description

The data contains three pairs,

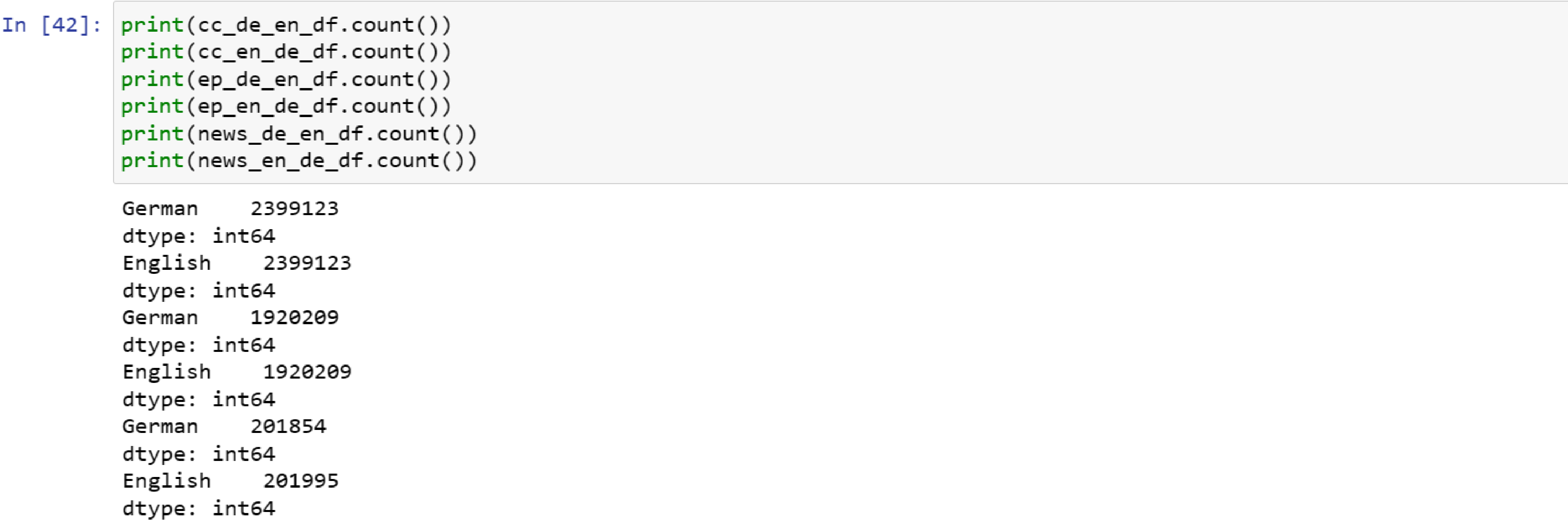
europarl-v7_en_de.txt file - 1920209 text
europarl-v7_de_en.txt - 1920209 text

commoncrawl_en_de.txt - 2399123 text
commoncrawl_de_en.txt – 2399123 text

news-commentary-v9_en_de.txt – 201854 text
news-commentary-v9_de_en.txt – 201854 text



```
In [39]: file_cc_de_en=open('commoncrawl_de_en.txt',encoding='Utf-8')
         cc_de_en=file_cc_de_en.readlines()
         file_cc_en_de=open('commoncrawl_en_de.txt',encoding='Utf-8')
         cc_en_de=file_cc_en_de.readlines()
         cc_de_en_df=pd.DataFrame(cc_de_en,columns=['German'])
         cc_en_de_df=pd.DataFrame(cc_en_de,columns=['English'])

In [40]: file_ep_de_en=open('europarl-v7_de_en.txt',encoding='Utf-8')
         ep_de_en=file_ep_de_en.readlines()
         file_ep_en_de=open('europarl-v7_en_de.txt',encoding='Utf-8')
         ep_en_de=file_ep_en_de.readlines()
         ep_de_en_df=pd.DataFrame(ep_de_en,columns=['German'])
         ep_en_de_df=pd.DataFrame(ep_en_de,columns=['English'])

In [41]: file_news_de_en=open('news-commentary-v9_de_en.txt',encoding='Utf-8')
         news_de_en=file_news_de_en.readlines()
         file_news_en_de=open('news-commentary-v9_en_de.txt',encoding='Utf-8')
         news_en_de=file_news_en_de.readlines()
         news_de_en_df=pd.DataFrame(news_de_en,columns=['German'])
         news_en_de_df=pd.DataFrame(news_en_de,columns=['English'])
```

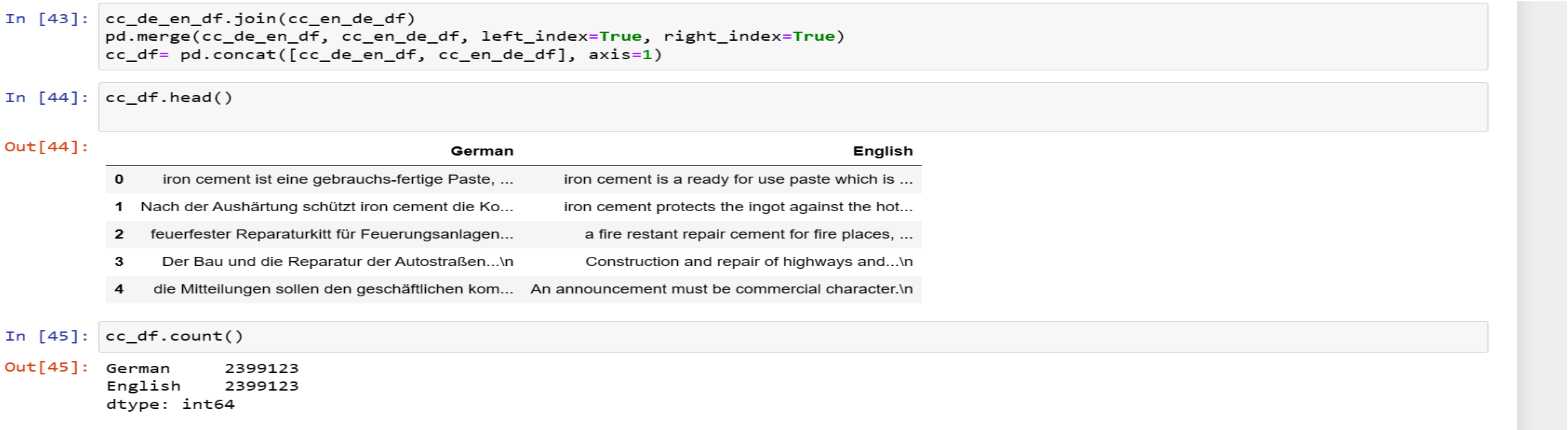


```
In [42]: print(cc_de_en_df.count())
         print(cc_en_de_df.count())
         print(ep_de_en_df.count())
         print(ep_en_de_df.count())
         print(news_de_en_df.count())
         print(news_en_de_df.count())
```

```
German    2399123
dtype: int64
English    2399123
dtype: int64
German    1920209
dtype: int64
English    1920209
dtype: int64
German    201854
dtype: int64
English    201995
dtype: int64
```

All the 6 files we have both german and english content alternatively. These are our datasets by using this information we going to split up english and german language.

Merging all the three datasets :



In [43]:
```
cc_de_en_df.join(cc_en_de_df)
pd.merge(cc_de_en_df, cc_en_de_df, left_index=True, right_index=True)
cc_df= pd.concat([cc_de_en_df, cc_en_de_df], axis=1)
```

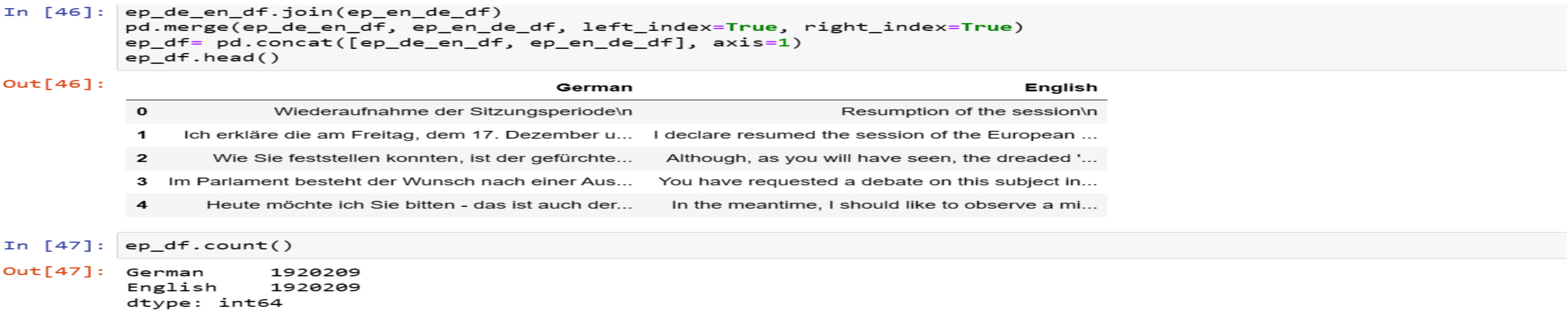
In [44]:
```
cc_df.head()
```

Out[44]:

	German	English
0	iron cement ist eine gebrauchs-fertige Paste, ...	iron cement is a ready for use paste which is ...
1	Nach der Aushärtung schützt iron cement die Ko...	iron cement protects the ingot against the hot...
2	feuerfester Reparaturkitt für Feuerungsanlagen...	a fire restant repair cement for fire places, ...
3	Der Bau und die Reparatur der Autostraßen...\n	Construction and repair of highways and...\n
4	die Mitteilungen sollen den geschäftlichen kom...	An announcement must be commercial character.\n

In [45]:
```
cc_df.count()
```

Out[45]:
```
German      2399123
English     2399123
dtype: int64
```



In [46]:
```
ep_de_en_df.join(ep_en_de_df)
pd.merge(ep_de_en_df, ep_en_de_df, left_index=True, right_index=True)
ep_df= pd.concat([ep_de_en_df, ep_en_de_df], axis=1)
ep_df.head()
```

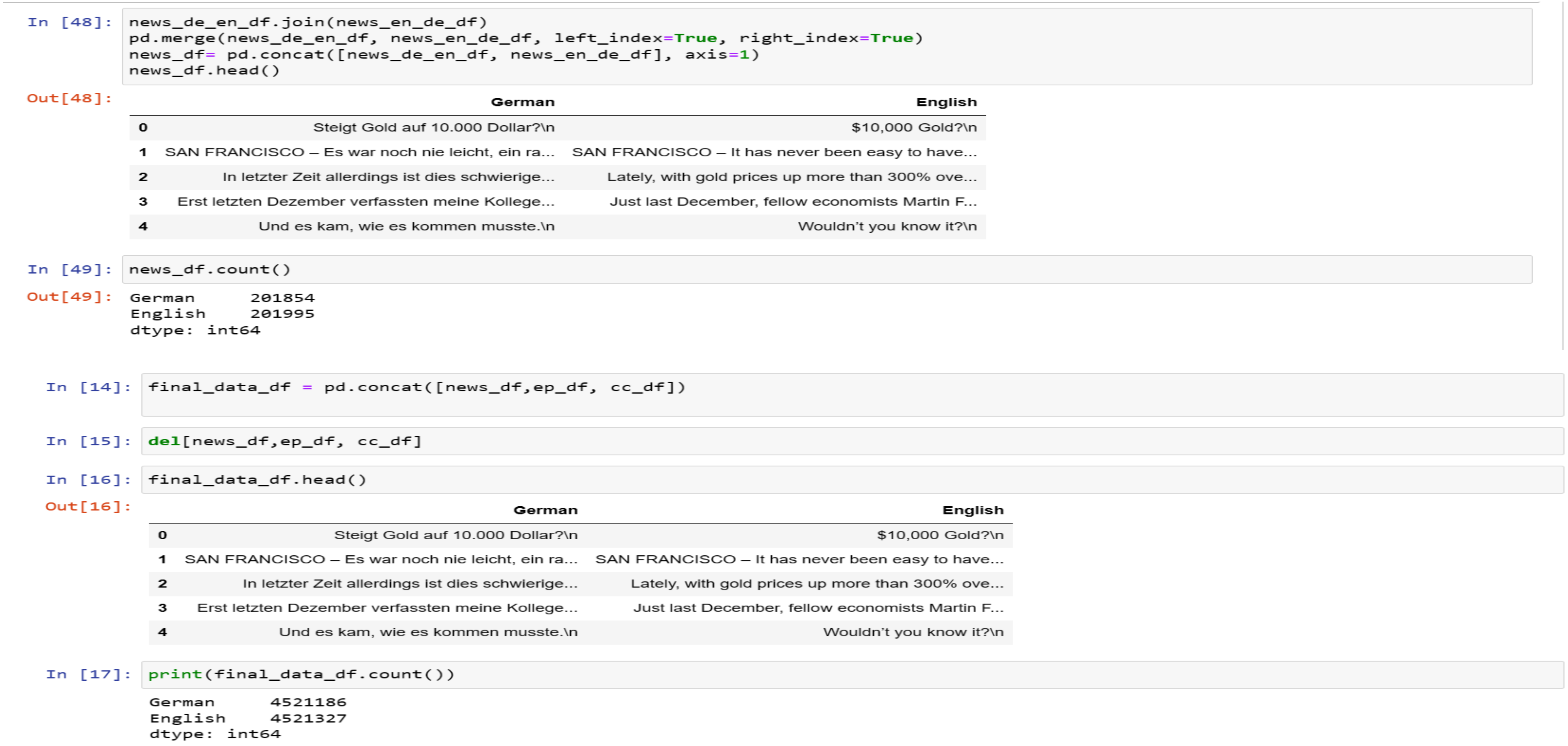
Out[46]:

	German	English
0	Wiederaufnahme der Sitzungsperiode\n	Resumption of the session\n
1	Ich erkläre die am Freitag, dem 17. Dezember u...	I declare resumed the session of the European ...
2	Wie Sie feststellen konnten, ist der gefürchte...	Although, as you will have seen, the dreaded '...
3	Im Parlament besteht der Wunsch nach einer Aus...	You have requested a debate on this subject in...
4	Heute möchte ich Sie bitten - das ist auch der...	In the meantime, I should like to observe a mi...

In [47]:
```
ep_df.count()
```

Out[47]:
```
German      1920209
English     1920209
dtype: int64
```

Concatenated all the dataframes into a single dataframe



In [48]:
```
news_de_en_df.join(news_en_de_df)
pd.merge(news_de_en_df, news_en_de_df, left_index=True, right_index=True)
news_df= pd.concat([news_de_en_df, news_en_de_df], axis=1)
news_df.head()
```

Out[48]:

	German	English
0	Steigt Gold auf 10.000 Dollar?\n	$10,000 Gold?\n
1	SAN FRANCISCO – Es war noch nie leicht, ein ra...	SAN FRANCISCO – It has never been easy to have...
2	In letzter Zeit allerdings ist dies schwierige...	Lately, with gold prices up more than 300% ove...
3	Erst letzten Dezember verfassten meine Kollege...	Just last December, fellow economists Martin F...
4	Und es kam, wie es kommen musste.\n	Wouldn't you know it?\n

In [49]:
```
news_df.count()
```

Out[49]:
```
German      201854
English     201995
dtype: int64
```

In [14]:
```
final_data_df = pd.concat([news_df,ep_df, cc_df])
```

In [15]:
```
del[news_df,ep_df, cc_df]
```

In [16]:
```
final_data_df.head()
```

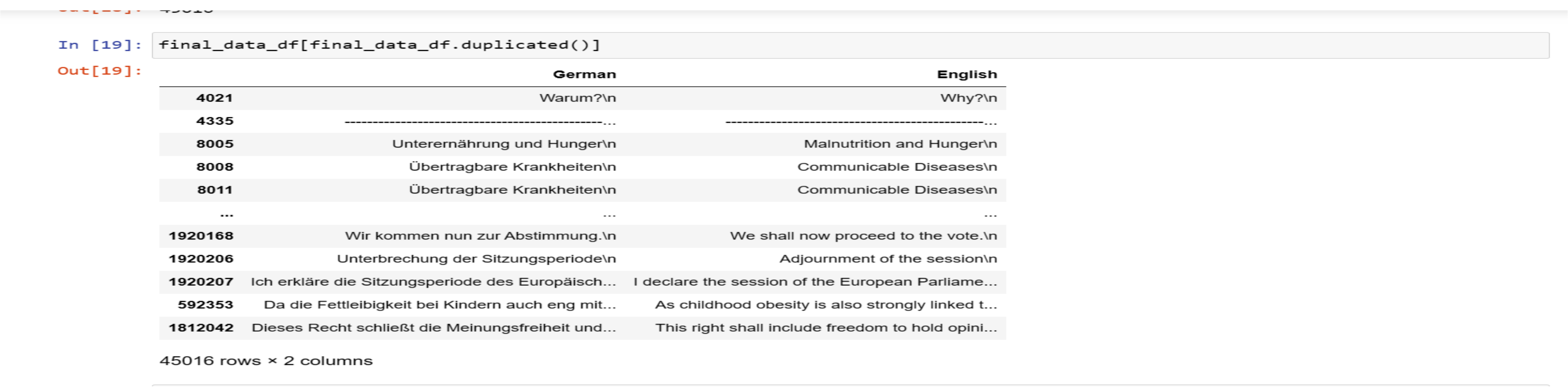
Out[16]:

	German	English
0	Steigt Gold auf 10.000 Dollar?\n	$10,000 Gold?\n
1	SAN FRANCISCO – Es war noch nie leicht, ein ra...	SAN FRANCISCO – It has never been easy to have...
2	In letzter Zeit allerdings ist dies schwierige...	Lately, with gold prices up more than 300% ove...
3	Erst letzten Dezember verfassten meine Kollege...	Just last December, fellow economists Martin F...
4	Und es kam, wie es kommen musste.\n	Wouldn't you know it?\n

In [17]:
```
print(final_data_df.count())
```

```
German      4521186
English     4521327
dtype: int64
```

FINDING DUPLICATE VALUES

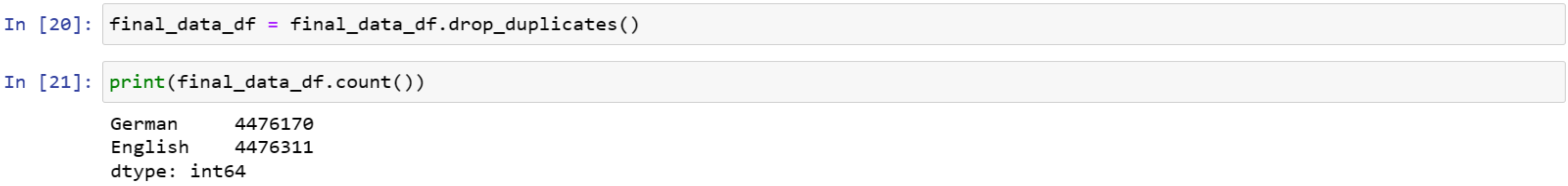


```
In [19]: final_data_df[final_data_df.duplicated()]
Out[19]:
```

	German	English
4021	Warum?\n	Why?\n
4335	---...	---...
8005	Unterernährung und Hunger\n	Malnutrition and Hunger\n
8008	Übertragbare Krankheiten\n	Communicable Diseases\n
8011	Übertragbare Krankheiten\n	Communicable Diseases\n
...	...	...
1920168	Wir kommen nun zur Abstimmung.\n	We shall now proceed to the vote.\n
1920206	Unterbrechung der Sitzungsperiode\n	Adjournment of the session\n
1920207	Ich erkläre die Sitzungsperiode des Europäisch...	I declare the session of the European Parliame...
592353	Da die Fettleibigkeit bei Kindern auch eng mit...	As childhood obesity is also strongly linked t...
1812042	Dieses Recht schließt die Meinungsfreiheit und...	This right shall include freedom to hold opini...

45016 rows × 2 columns

DROPPING DUPLICATES



```
In [20]: final_data_df = final_data_df.drop_duplicates()

In [21]: print(final_data_df.count())

German     4476170
English    4476311
dtype: int64
```

Summary of the Approach to EDA and Pre-processing:

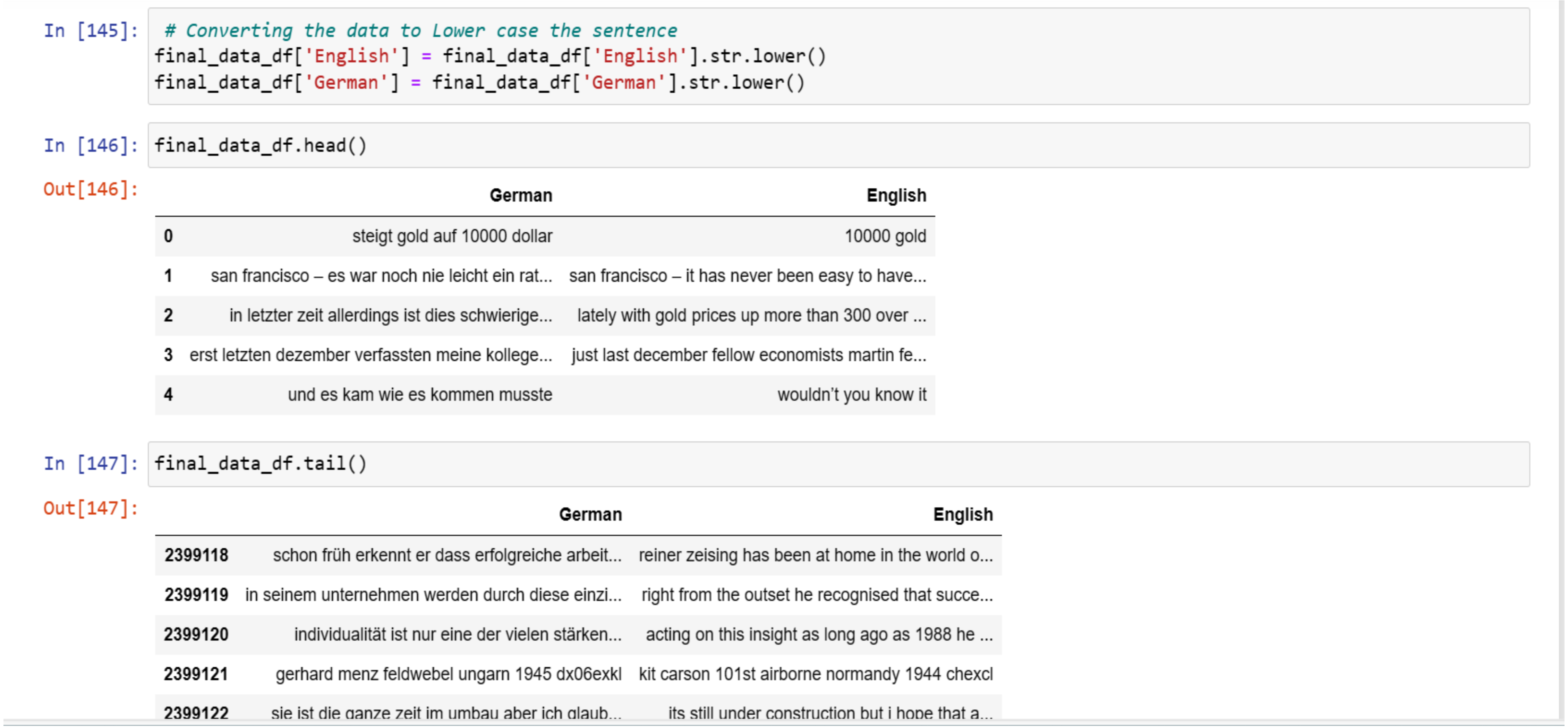
Our agenda is to convert one language into another language. we have the text data required for the process, so we not doing any visualization feature here. As part of the preprocessing, we have followed the following steps.

- Merge, Load and examined the data
- Cleansing and processing the data
- Tokenization
- Padding

Cleansing and processing the data:

- The data has been converted into lower case.
- Duplicating records have been dropped
- Punctuation has been replaced by empty string.
- Missing values are identified
- Special characters like’\n’ has been removed

Converting data into lower case:



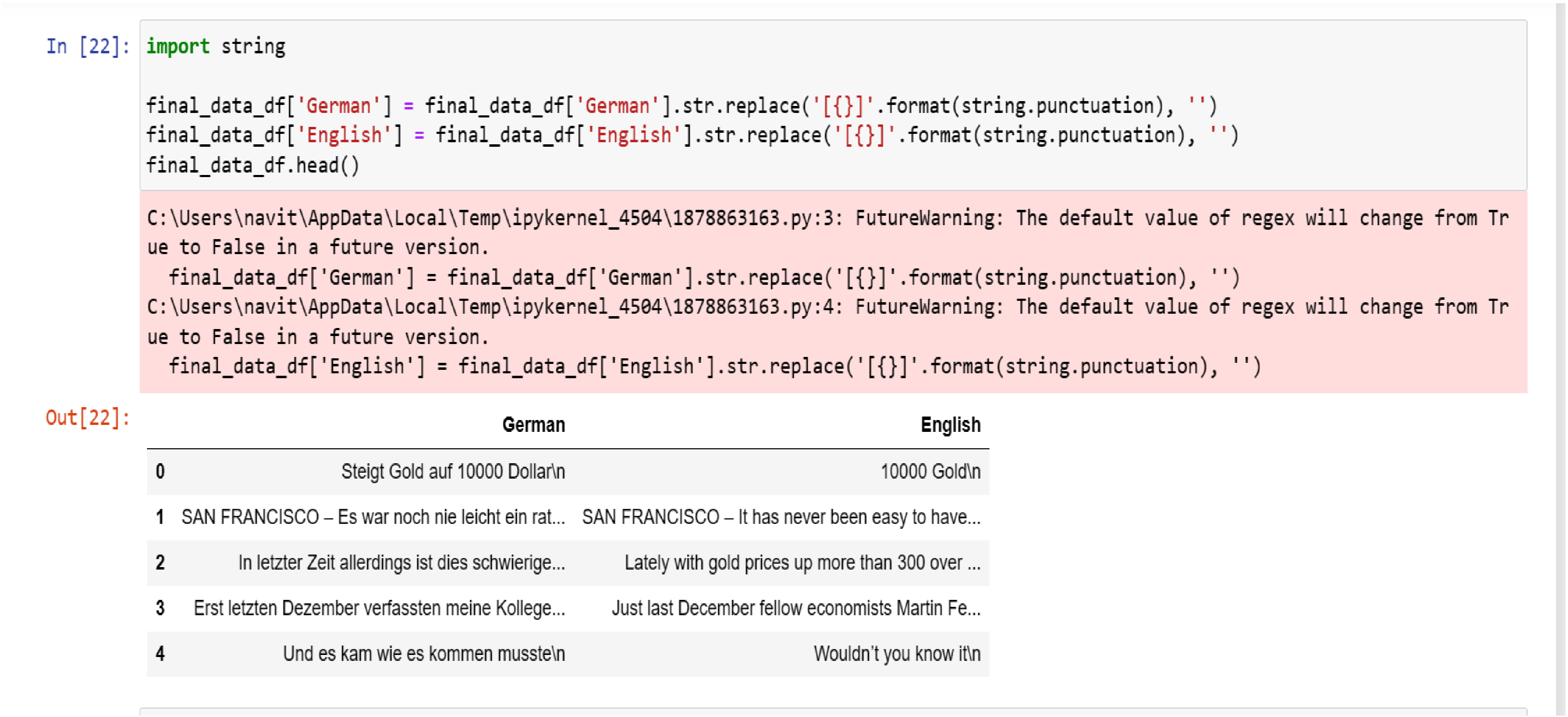
```
In [145]:  # Converting the data to Lower case the sentence
           final_data_df['English'] = final_data_df['English'].str.lower()
           final_data_df['German'] = final_data_df['German'].str.lower()

In [146]:  final_data_df.head()

Out[146]:
```

	German	English
0	steigt gold auf 10000 dollar	10000 gold
1	san francisco – es war noch nie leicht ein rat...	san francisco – it has never been easy to have...
2	in letzter zeit allerdings ist dies schwierige...	lately with gold prices up more than 300 over ...
3	erst letzten dezember verfassten meine kollege...	just last december fellow economists martin fe...
4	und es kam wie es kommen musste	wouldn't you know it

```
In [147]:  final_data_df.tail()

Out[147]:
```

	German	English
2399118	schon früh erkennt er dass erfolgreiche arbeit...	reiner zeising has been at home in the world o...
2399119	in seinem unternehmen werden durch diese einzi...	right from the outset he recognised that succe...
2399120	individualität ist nur eine der vielen stärken...	acting on this insight as long ago as 1988 he ...
2399121	gerhard menz feldwebel ungarn 1945 dx06exkl	kit carson 101st airborne normandy 1944 chexcl
2399122	sie ist die ganze zeit im umbau aber ich glaub...	its still under construction but i hope that a...

Punctuation has been replaced by empty string:



In [22]:

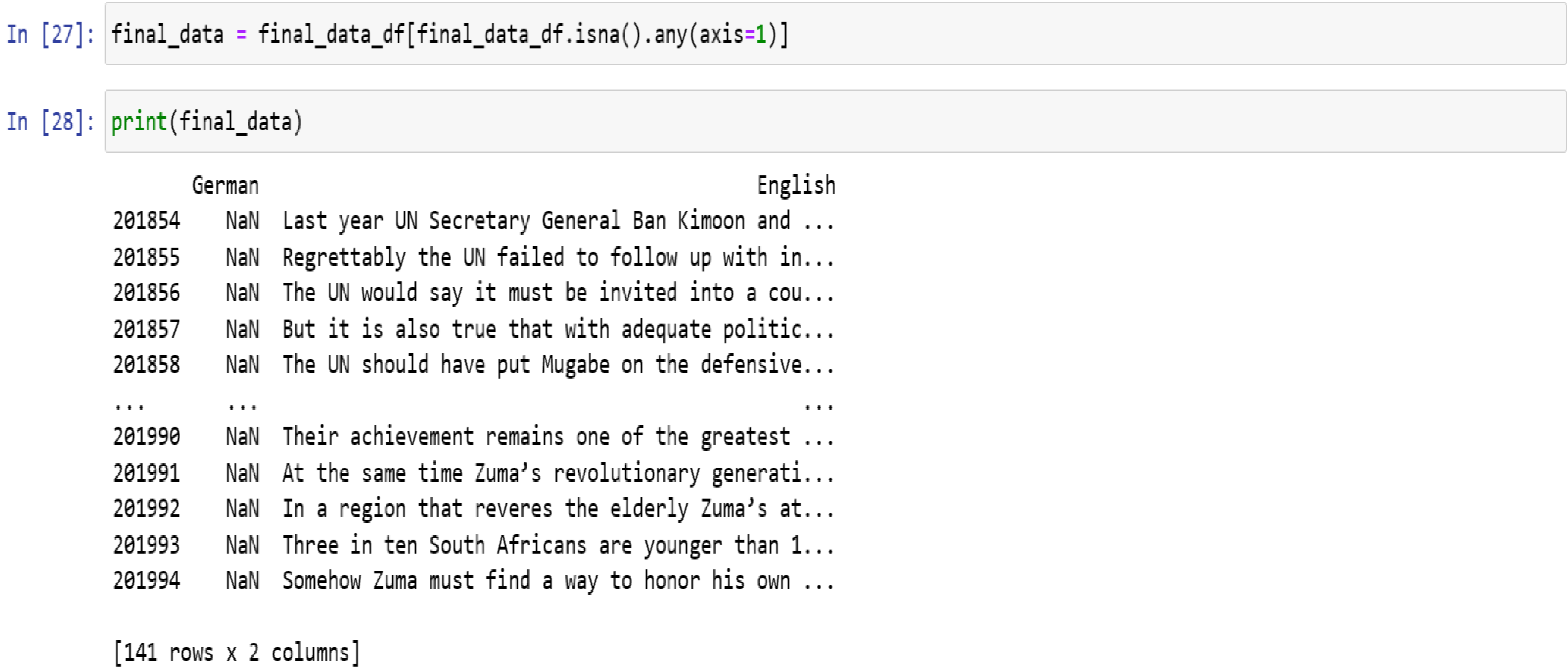
```
import string

final_data_df['German'] = final_data_df['German'].str.replace('[{}]'.format(string.punctuation), '')
final_data_df['English'] = final_data_df['English'].str.replace('[{}]'.format(string.punctuation), '')
final_data_df.head()
```

```
C:\Users\navit\AppData\Local\Temp\ipykernel_4504\1878863163.py:3: FutureWarning: The default value of regex will change from Tr
ue to False in a future version.
  final_data_df['German'] = final_data_df['German'].str.replace('[{}]'.format(string.punctuation), '')
C:\Users\navit\AppData\Local\Temp\ipykernel_4504\1878863163.py:4: FutureWarning: The default value of regex will change from Tr
ue to False in a future version.
  final_data_df['English'] = final_data_df['English'].str.replace('[{}]'.format(string.punctuation), '')
```

Out[22]:

	German	English
0	Steigt Gold auf 10000 Dollar\n	10000 Gold\n
1	SAN FRANCISCO – Es war noch nie leicht ein rat...	SAN FRANCISCO – It has never been easy to have...
2	In letzter Zeit allerdings ist dies schwierige...	Lately with gold prices up more than 300 over ...
3	Erst letzten Dezember verfassten meine Kollege...	Just last December fellow economists Martin Fe...
4	Und es kam wie es kommen musste\n	Wouldn’t you know it\n

Missing values are identified :

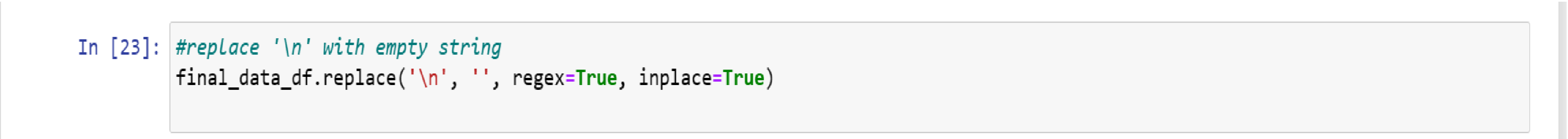


```
In [27]: final_data = final_data_df[final_data_df.isna().any(axis=1)]
```

```
In [28]: print(final_data)
```

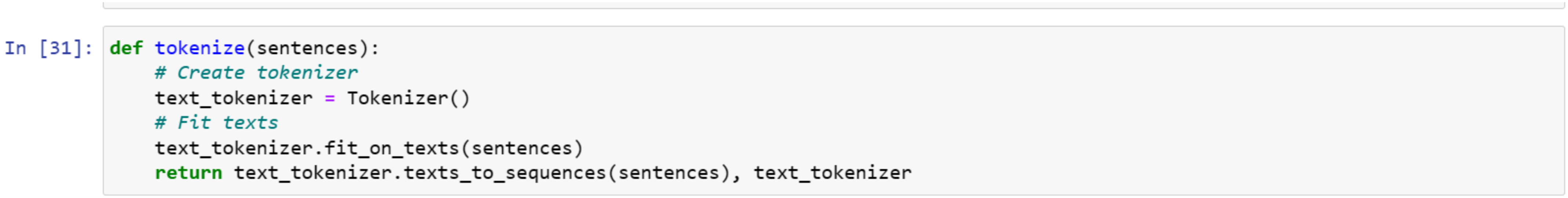
```
       German                                            English
201854    NaN  Last year UN Secretary General Ban Kimoon and ...
201855    NaN  Regrettably the UN failed to follow up with in...
201856    NaN  The UN would say it must be invited into a cou...
201857    NaN  But it is also true that with adequate politic...
201858    NaN  The UN should have put Mugabe on the defensive...
...       ...                                                ...
201990    NaN  Their achievement remains one of the greatest ...
201991    NaN  At the same time Zuma’s revolutionary generati...
201992    NaN  In a region that reveres the elderly Zuma’s at...
201993    NaN  Three in ten South Africans are younger than 1...
201994    NaN  Somehow Zuma must find a way to honor his own ...

[141 rows x 2 columns]
```

Special characters like’\n’ has been removed :

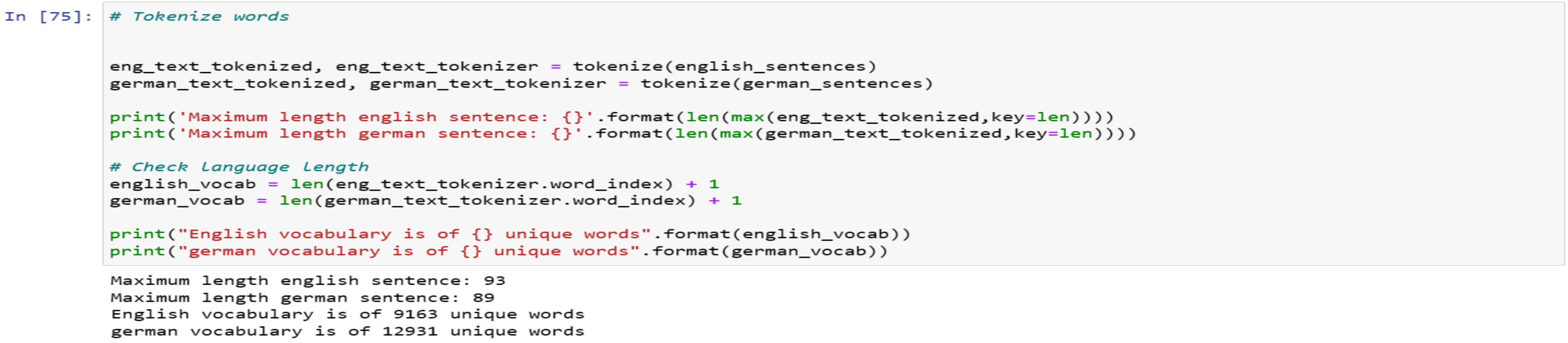


In [23]:
```
#replace '\n' with empty string
final_data_df.replace('\n', '', regex=True, inplace=True)
```

Tokenization :



In [31]:
```
def tokenize(sentences):
    # Create tokenizer
    text_tokenizer = Tokenizer()
    # Fit texts
    text_tokenizer.fit_on_texts(sentences)
    return text_tokenizer.texts_to_sequences(sentences), text_tokenizer
```

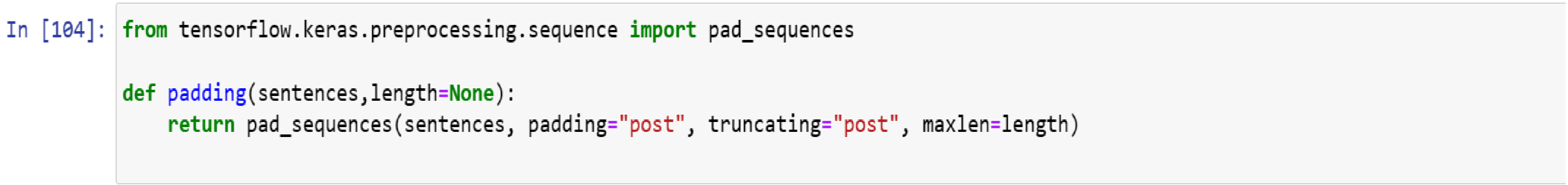


In [75]:
```
# Tokenize words


eng_text_tokenized, eng_text_tokenizer = tokenize(english_sentences)
german_text_tokenized, german_text_tokenizer = tokenize(german_sentences)

print('Maximum length english sentence: {}'.format(len(max(eng_text_tokenized,key=len))))
print('Maximum length german sentence: {}'.format(len(max(german_text_tokenized,key=len))))

# Check language length
english_vocab = len(eng_text_tokenizer.word_index) + 1
german_vocab = len(german_text_tokenizer.word_index) + 1

print("English vocabulary is of {} unique words".format(english_vocab))
print("german vocabulary is of {} unique words".format(german_vocab))
```

```
Maximum length english sentence: 93
Maximum length german sentence: 89
English vocabulary is of 9163 unique words
german vocabulary is of 12931 unique words
```

Padding :

When the sequence of word id’s feed into the model, each sequence should have the same length. Padding is added to the shorter sequence with the max length



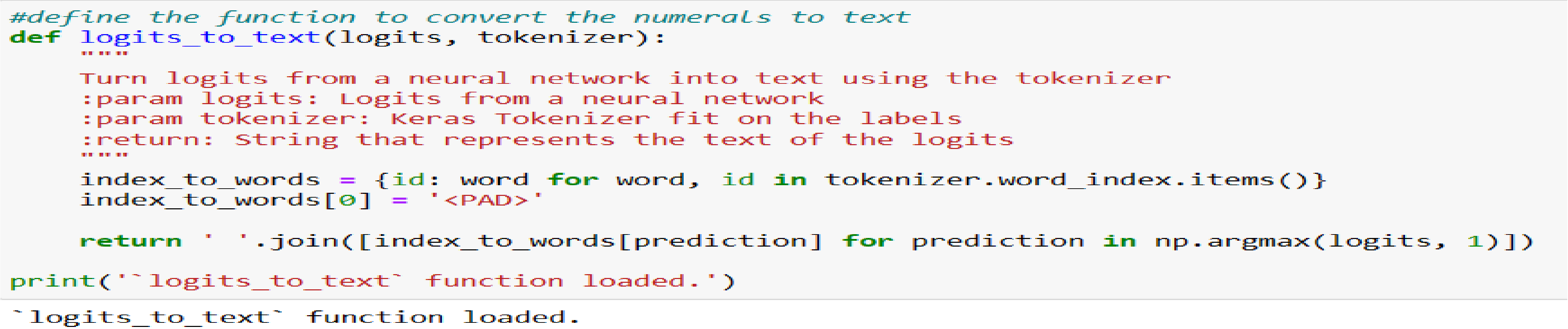
```
from tensorflow.keras.preprocessing.sequence import pad_sequences

def padding(sentences,length=None):
    return pad_sequences(sentences, padding="post", truncating="post", maxlen=length)
```

DESIGN :

Deciding Models and Model Building :

We build the model using LSTM and RNN, by using these techniques we handled designing, training and testing.

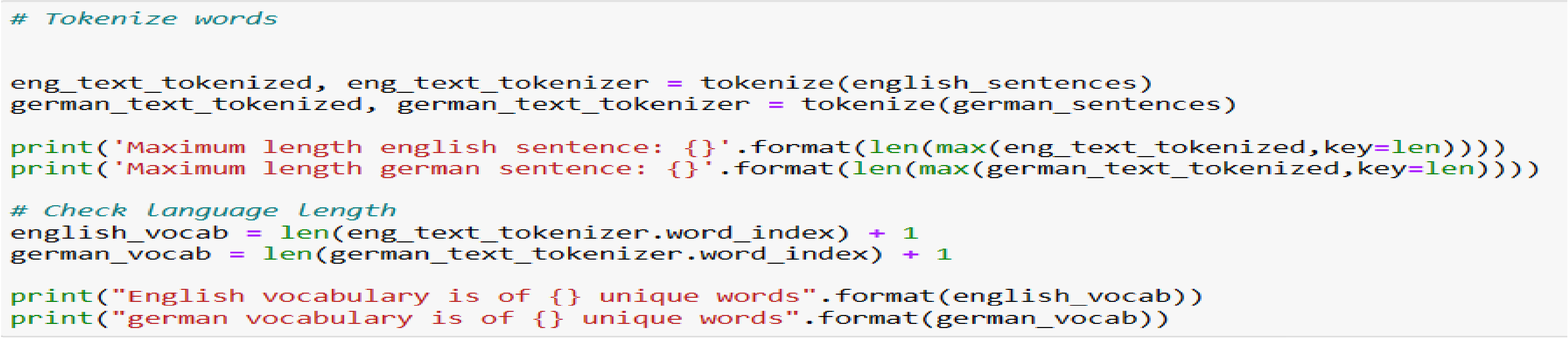
Tokenized index has been converted into texts



```
#define the function to convert the numerals to text
def logits_to_text(logits, tokenizer):
    """
    Turn logits from a neural network into text using the tokenizer
    :param logits: Logits from a neural network
    :param tokenizer: Keras Tokenizer fit on the labels
    :return: String that represents the text of the logits
    """
    index_to_words = {id: word for word, id in tokenizer.word_index.items()}
    index_to_words[0] = '<PAD>'

    return ' '.join([index_to_words[prediction] for prediction in np.argmax(logits, 1)])

print('`logits_to_text` function loaded.')
```

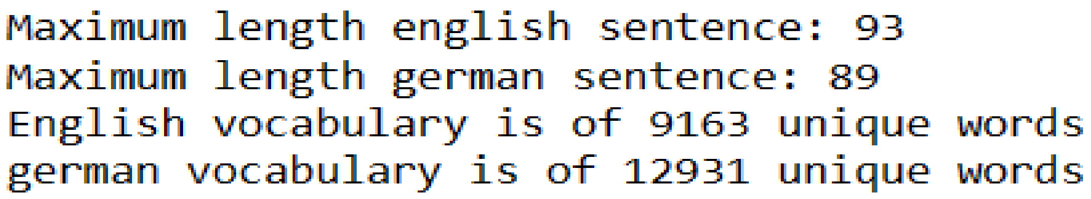
```
`logits_to_text` function loaded.
```

Execution of the tokenize function. This creates the maximum length of the sequence and the number of words in the vocabulary list



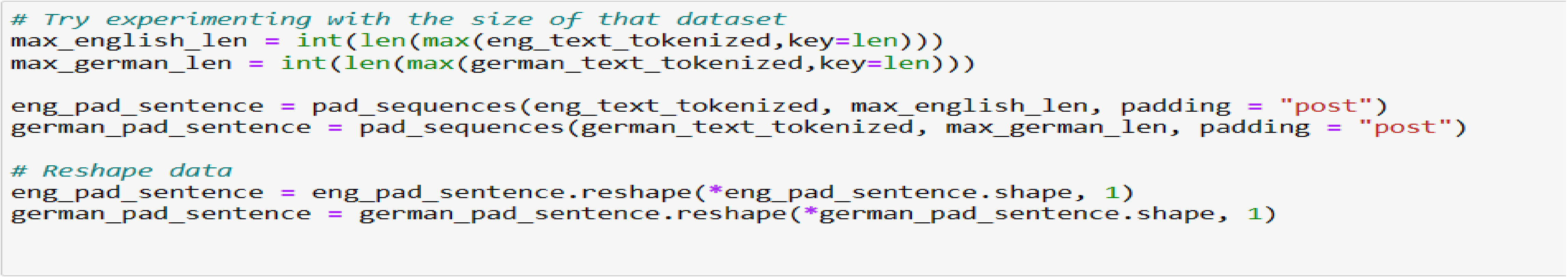
```
# Tokenize words


eng_text_tokenized, eng_text_tokenizer = tokenize(english_sentences)
german_text_tokenized, german_text_tokenizer = tokenize(german_sentences)

print('Maximum length english sentence: {}'.format(len(max(eng_text_tokenized,key=len))))
print('Maximum length german sentence: {}'.format(len(max(german_text_tokenized,key=len))))

# Check language length
english_vocab = len(eng_text_tokenizer.word_index) + 1
german_vocab = len(german_text_tokenizer.word_index) + 1

print("English vocabulary is of {} unique words".format(english_vocab))
print("german vocabulary is of {} unique words".format(german_vocab))
```

Output of the tokenized data



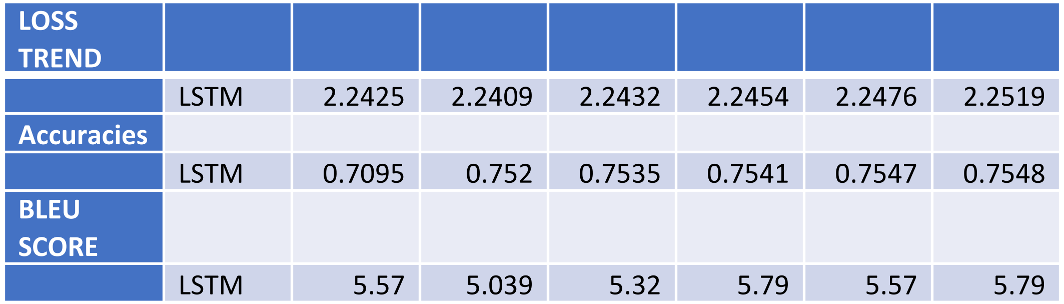
```
Maximum length english sentence: 93
Maximum length german sentence: 89
English vocabulary is of 9163 unique words
german vocabulary is of 12931 unique words
```

Data needs to be reshaped in order to execute the model. The model would need 3-dimensional data. Shape of the data has been converted to fit the model



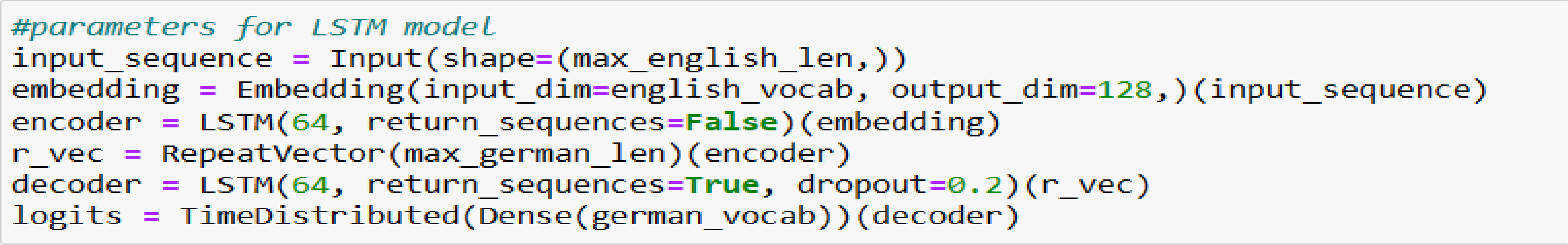
```
# Try experimenting with the size of that dataset
max_english_len = int(len(max(eng_text_tokenized,key=len)))
max_german_len = int(len(max(german_text_tokenized,key=len)))

eng_pad_sentence = pad_sequences(eng_text_tokenized, max_english_len, padding = "post")
german_pad_sentence = pad_sequences(german_text_tokenized, max_german_len, padding = "post")

# Reshape data
eng_pad_sentence = eng_pad_sentence.reshape(*eng_pad_sentence.shape, 1)
german_pad_sentence = german_pad_sentence.reshape(*german_pad_sentence.shape, 1)
```

Training the model : LSTM model



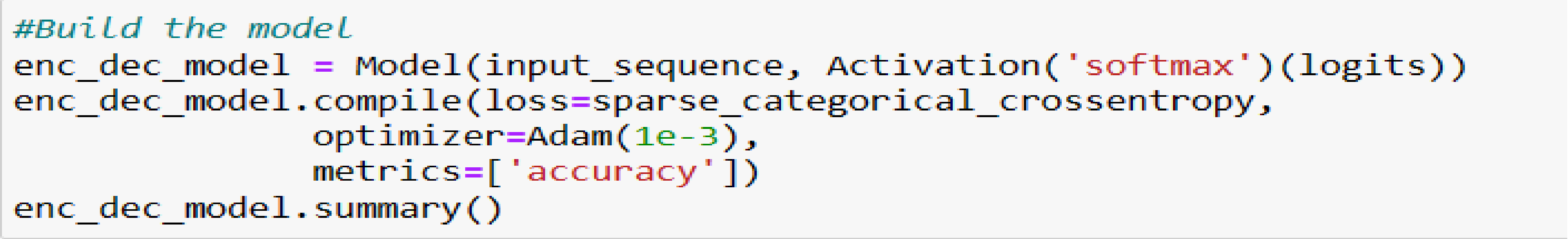
LOSS TREND							
	LSTM	2.2425	2.2409	2.2432	2.2454	2.2476	2.2519
Accuracies							
	LSTM	0.7095	0.752	0.7535	0.7541	0.7547	0.7548
BLEU SCORE							
	LSTM	5.57	5.039	5.32	5.79	5.57	5.79

Implementing a simple LSTM model using the encoder and decoder methods



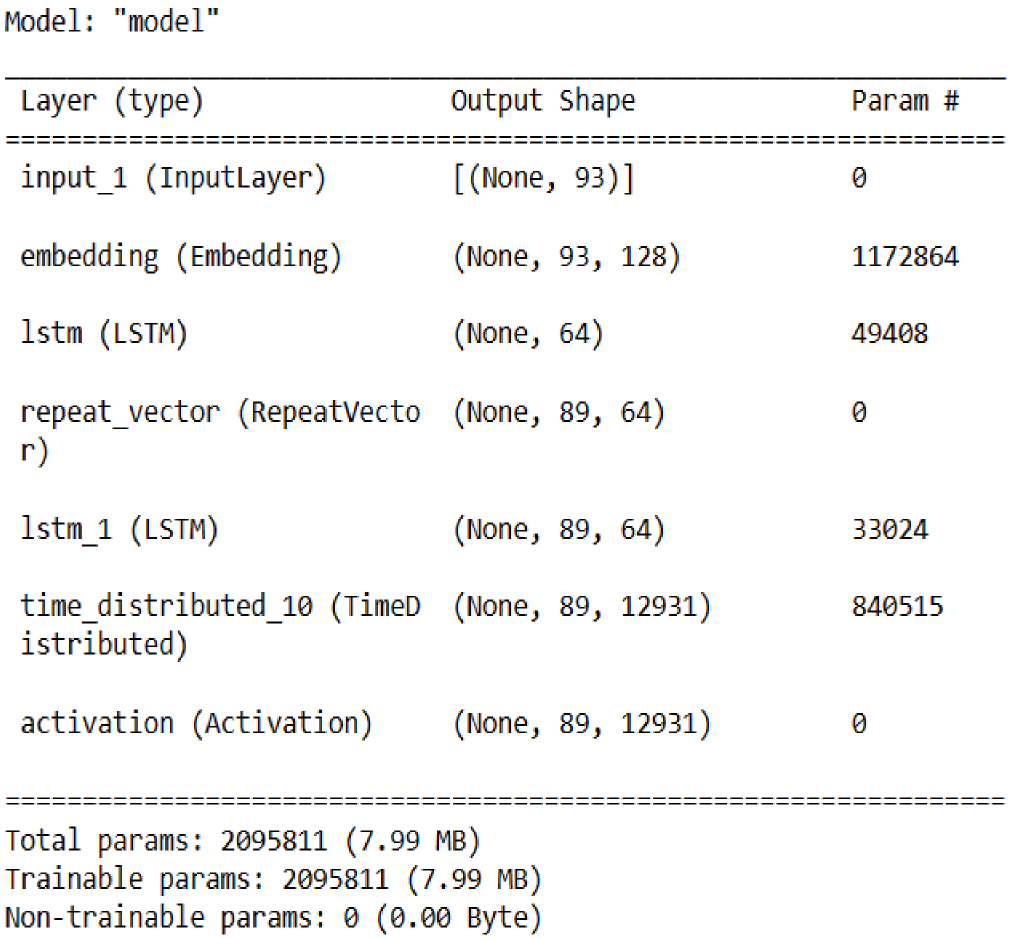
```
#parameters for LSTM model
input_sequence = Input(shape=(max_english_len,))
embedding = Embedding(input_dim=english_vocab, output_dim=128,)(input_sequence)
encoder = LSTM(64, return_sequences=False)(embedding)
r_vec = RepeatVector(max_german_len)(encoder)
decoder = LSTM(64, return_sequences=True, dropout=0.2)(r_vec)
logits = TimeDistributed(Dense(german_vocab))(decoder)
```

Softmax activation has been implemented with accuracy metric. Summary of the model is printed



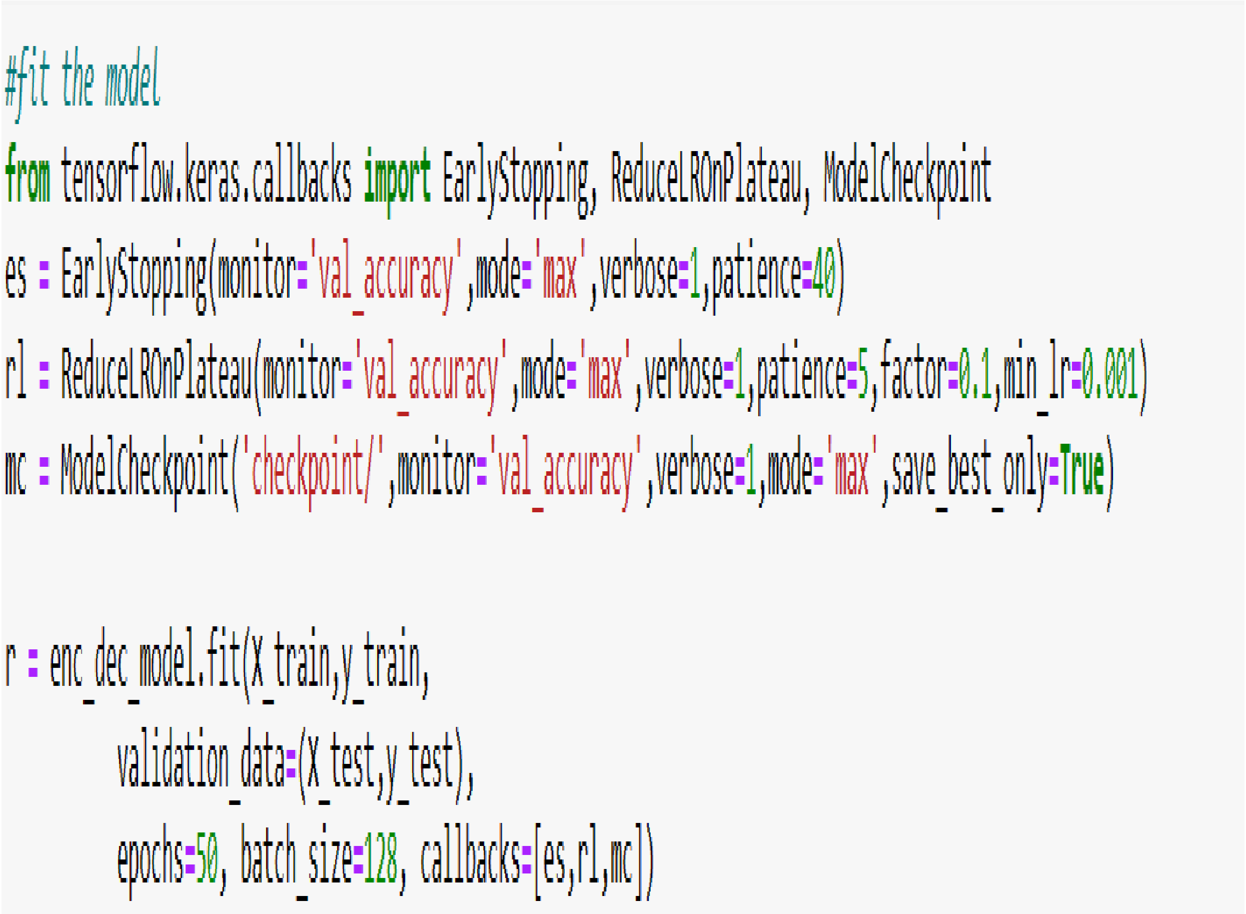
```
#Build the model
enc_dec_model = Model(input_sequence, Activation('softmax')(logits))
enc_dec_model.compile(loss=sparse_categorical_crossentropy,
              optimizer=Adam(1e-3),
              metrics=['accuracy'])
enc_dec_model.summary()
```

Summary model of the LSTM model is given below



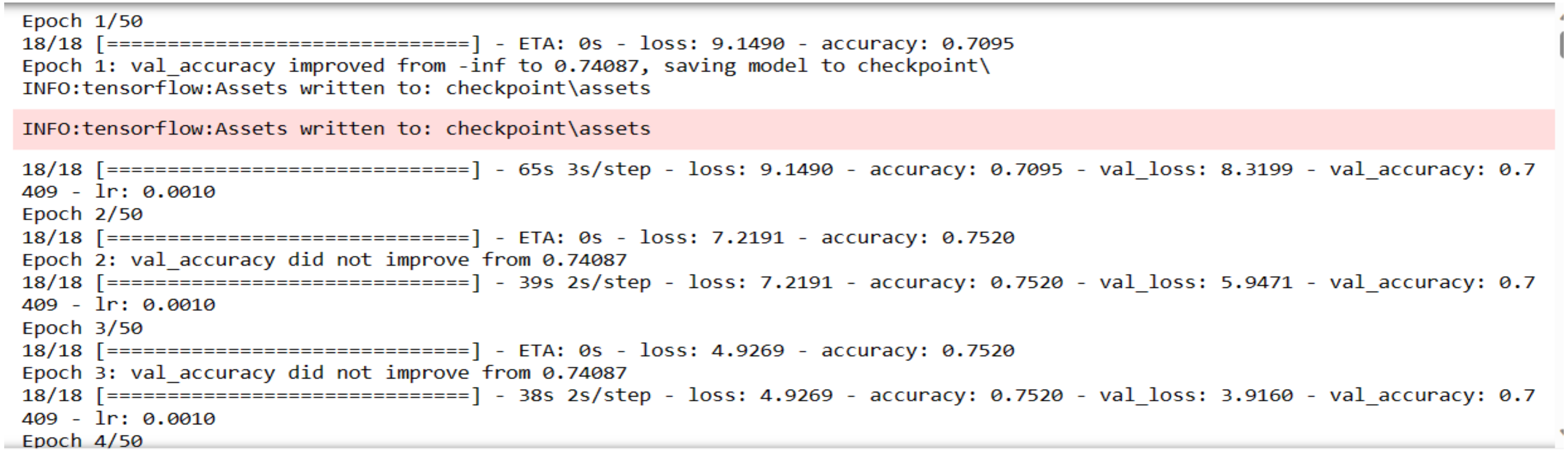
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 93)]              0

 embedding (Embedding)       (None, 93, 128)           1172864

 lstm (LSTM)                 (None, 64)                49408

 repeat_vector (RepeatVecto  (None, 89, 64)            0
 r)

 lstm_1 (LSTM)               (None, 89, 64)            33024

 time_distributed_10 (TimeD  (None, 89, 12931)         840515
 istributed)

 activation (Activation)     (None, 89, 12931)         0

=================================================================
Total params: 2095811 (7.99 MB)
Trainable params: 2095811 (7.99 MB)
Non-trainable params: 0 (0.00 Byte)
```

Fit the model with early stopping based on the test and train data split with a epoch of 50 and batch size 128



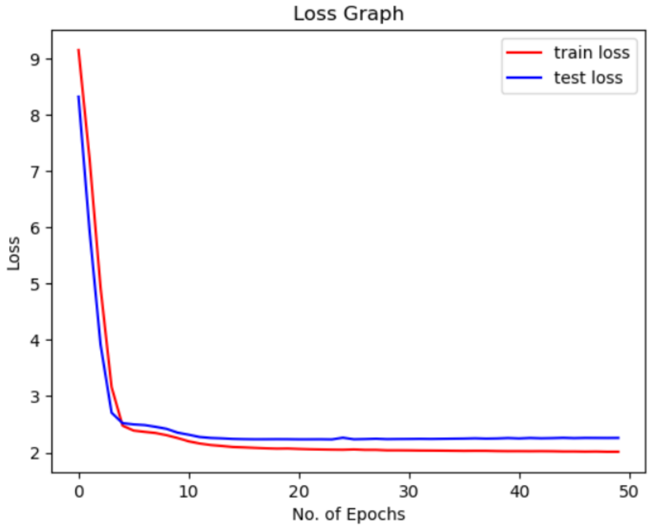
```
#fit the model
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
es = EarlyStopping(monitor='val_accuracy',mode='max',verbose=1,patience=40)
rl = ReduceLROnPlateau(monitor='val_accuracy',mode='max',verbose=1,patience=5,factor=0.1,min_lr=0.001)
mc = ModelCheckpoint('checkpoint/',monitor='val_accuracy',verbose=1,mode='max',save_best_only=True)


r = enc_dec_model.fit(X_train,y_train,
        validation_data=(X_test,y_test),
        epochs=50, batch_size=128, callbacks=[es,rl,mc])
```

Execution output with the above metrics

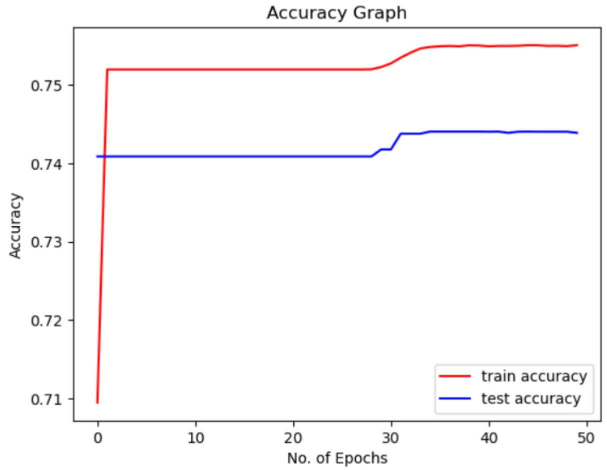


```
Epoch 1/50
18/18 [==============================] - ETA: 0s - loss: 9.1490 - accuracy: 0.7095
Epoch 1: val_accuracy improved from -inf to 0.74087, saving model to checkpoint\
INFO:tensorflow:Assets written to: checkpoint\assets

INFO:tensorflow:Assets written to: checkpoint\assets

18/18 [==============================] - 65s 3s/step - loss: 9.1490 - accuracy: 0.7095 - val_loss: 8.3199 - val_accuracy: 0.7
409 - lr: 0.0010
Epoch 2/50
18/18 [==============================] - ETA: 0s - loss: 7.2191 - accuracy: 0.7520
Epoch 2: val_accuracy did not improve from 0.74087
18/18 [==============================] - 39s 2s/step - loss: 7.2191 - accuracy: 0.7520 - val_loss: 5.9471 - val_accuracy: 0.7
409 - lr: 0.0010
Epoch 3/50
18/18 [==============================] - ETA: 0s - loss: 4.9269 - accuracy: 0.7520
Epoch 3: val_accuracy did not improve from 0.74087
18/18 [==============================] - 38s 2s/step - loss: 4.9269 - accuracy: 0.7520 - val_loss: 3.9160 - val_accuracy: 0.7
409 - lr: 0.0010
Epoch 4/50
```

Loss graph



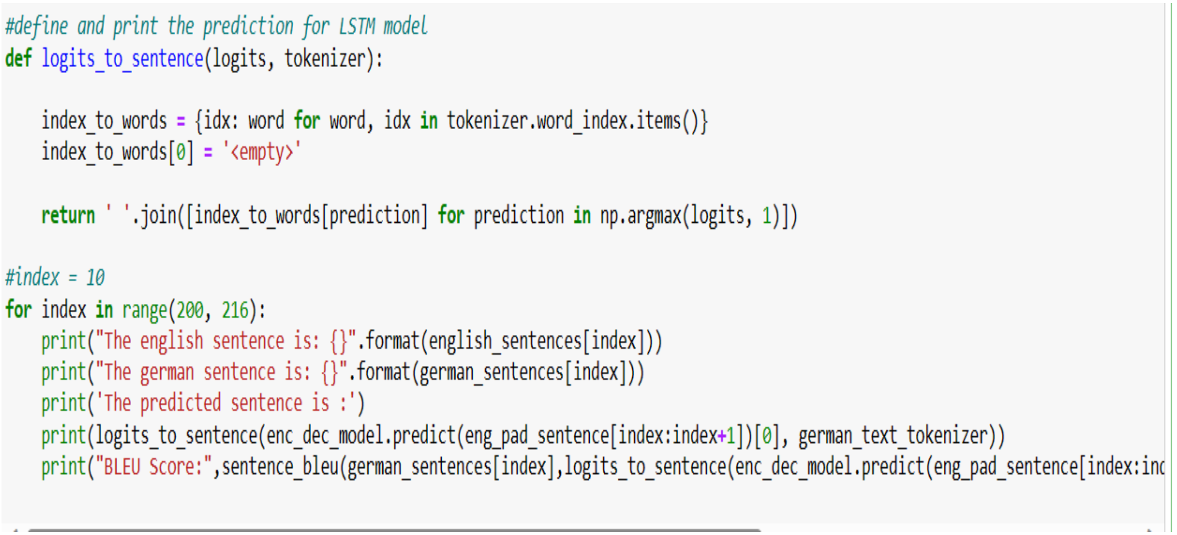
Accuracy graph of the models



Testing the model : SimpleRNN model

Logit function and the model prediction of the Simple RNN model

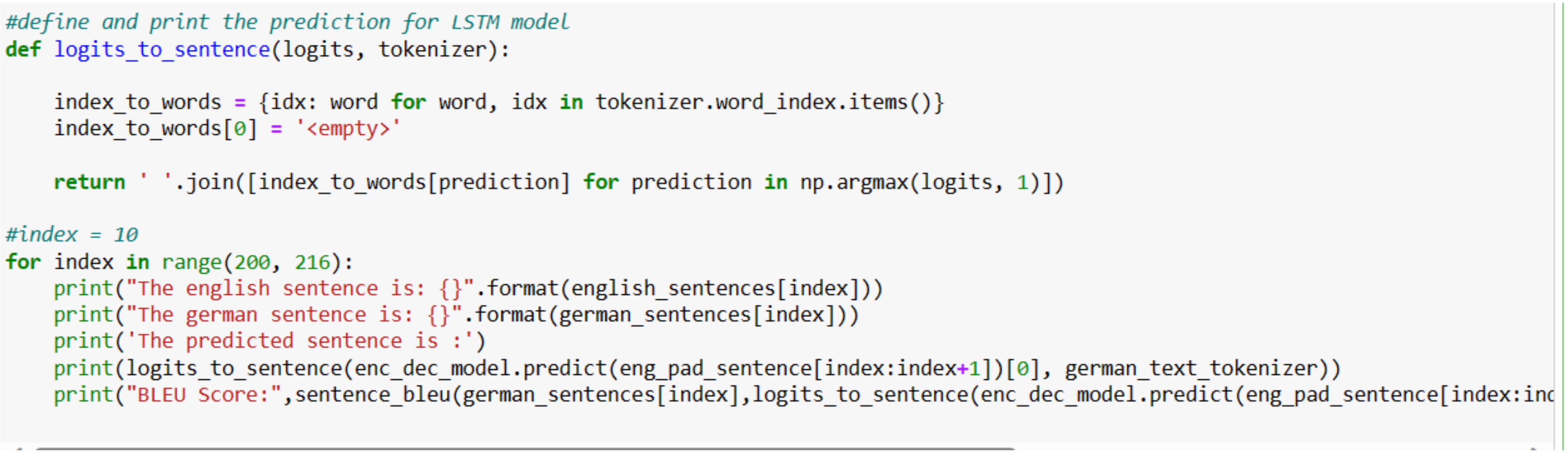
LOSS TREND							
	RNN	2936140	2936107	2936105	2936103	2936105	2936104



```
#define and print the prediction for LSTM model
def logits_to_sentence(logits, tokenizer):

    index_to_words = {idx: word for word, idx in tokenizer.word_index.items()}
    index_to_words[0] = '<empty>'

    return ' '.join([index_to_words[prediction] for prediction in np.argmax(logits, 1)])

#index = 10
for index in range(200, 216):
    print("The english sentence is: {}".format(english_sentences[index]))
    print("The german sentence is: {}".format(german_sentences[index]))
    print('The predicted sentence is :')
    print(logits_to_sentence(enc_dec_model.predict(eng_pad_sentence[index:index+1])[0], german_text_tokenizer))
    print("BLEU Score:",sentence_bleu(german_sentences[index],logits_to_sentence(enc_dec_model.predict(eng_pad_sentence[index:inc
```

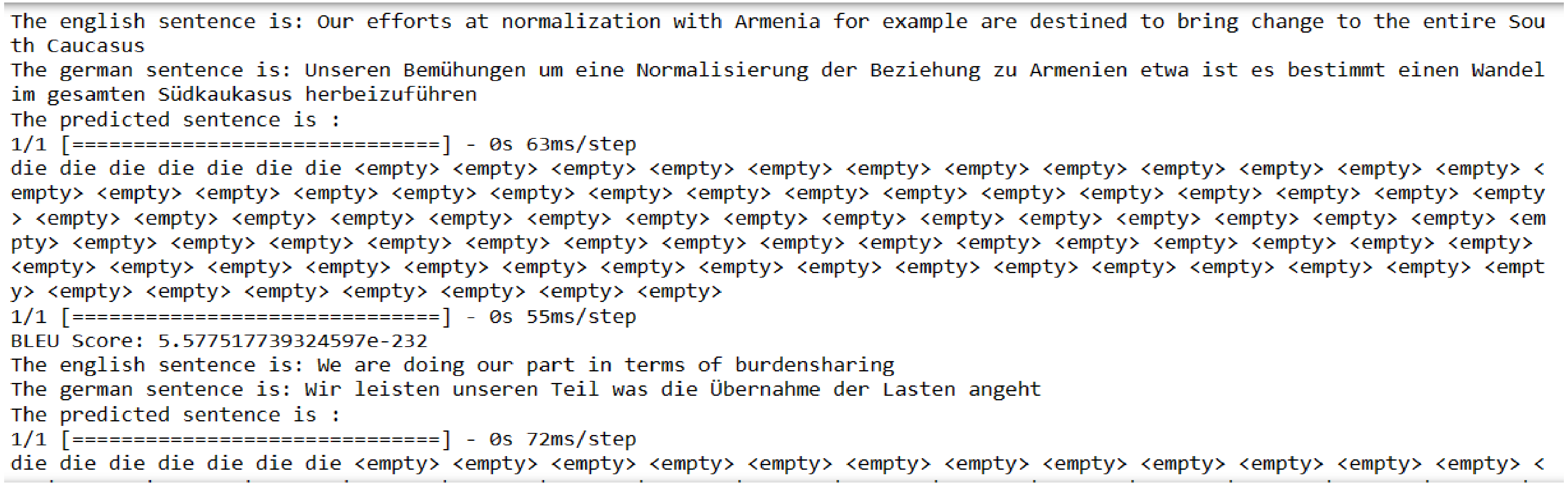
Predict the simple RNN models by calling the logits function to convert the index to text

Testing the model : LSTM model



```
#define and print the prediction for LSTM model
def logits_to_sentence(logits, tokenizer):

    index_to_words = {idx: word for word, idx in tokenizer.word_index.items()}
    index_to_words[0] = '<empty>'

    return ' '.join([index_to_words[prediction] for prediction in np.argmax(logits, 1)])

#index = 10
for index in range(200, 216):
    print("The english sentence is: {}".format(english_sentences[index]))
    print("The german sentence is: {}".format(german_sentences[index]))
    print('The predicted sentence is :')
    print(logits_to_sentence(enc_dec_model.predict(eng_pad_sentence[index:index+1])[0], german_text_tokenizer))
    print("BLEU Score:",sentence_bleu(german_sentences[index],logits_to_sentence(enc_dec_model.predict(eng_pad_sentence[index:inc
```

Output of the predicted model



```
The english sentence is: Our efforts at normalization with Armenia for example are destined to bring change to the entire Sou
th Caucasus
The german sentence is: Unseren Bemühungen um eine Normalisierung der Beziehung zu Armenien etwa ist es bestimmt einen Wandel
im gesamten Südkaukasus herbeizuführen
The predicted sentence is :
1/1 [==============================] - 0s 63ms/step
die die die die die die die <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <
empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty
> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <em
pty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty>
<empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empt
y> <empty> <empty> <empty> <empty> <empty> <empty> <empty>
1/1 [==============================] - 0s 55ms/step
BLEU Score: 5.577517739324597e-232
The english sentence is: We are doing our part in terms of burdensharing
The german sentence is: Wir leisten unseren Teil was die Übernahme der Lasten angeht
The predicted sentence is :
1/1 [==============================] - 0s 72ms/step
die die die die die die die <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <empty> <
```