Team :Anirudh Sundara Rajan, Mayank

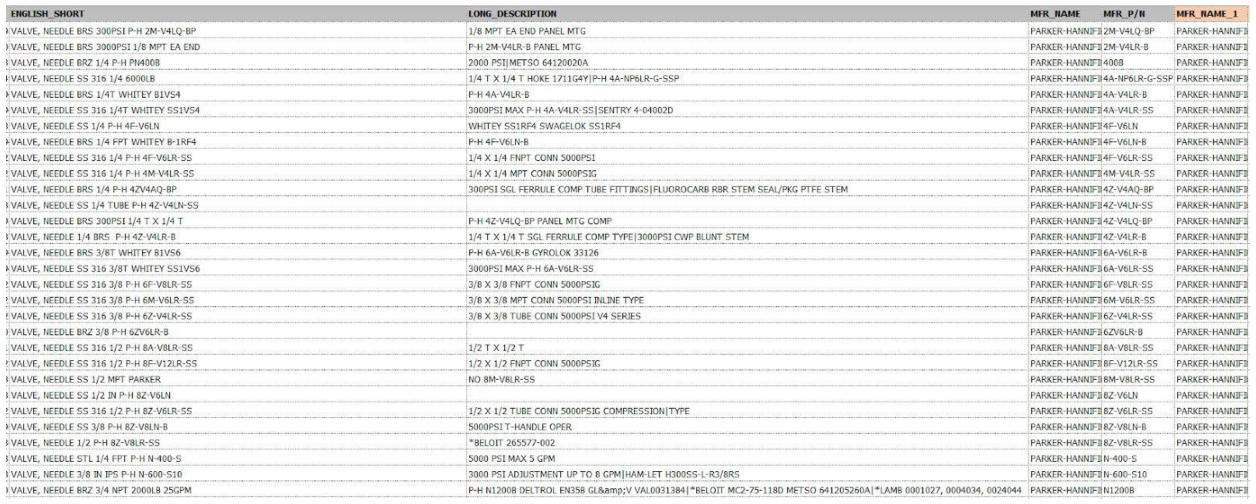
Negi 10 June 2020

Performing Named Entity Recognition Using a Recurrent Neural Network Model

Overview

This description shows the thought process and the code behind the Named Entity Recognition implemented using Keras and TensorFlow.This report will discuss the dataset and the model in detail.A validation accuracy of around 80% has been achieved through this model.

The Dataset. The dataset is a complex dataset which has been modified to help with the preprocessing.



ENGLISH_SHORT	LONG_DESCRIPTION	MFR_NAME	MFR_P/N	MFR_NAME_1
VALVE, NEEDLE BRS 300PSI P-H 2M-V4LQ-BP	1/8 MPT EA END PANEL MTG	PARKER-HANNIFI	2M-V4LQ-BP	PARKER-HANNIFI
VALVE, NEEDLE BRS 3000PSI 1/8 MPT EA END	P-H 2M-V4LR-B PANEL MTG	PARKER-HANNIFI	2M-V4LR-B	PARKER-HANNIFI
VALVE, NEEDLE BRZ 1/4 P-H PN400B	2000 PSI|METSO 64120020A	PARKER-HANNIFI	400B	PARKER-HANNIFI
VALVE, NEEDLE SS 316 1/4 6000LB	1/4 T X 1/4 T HOKE 1711G4Y|P-H 4A-NP6LR-G-SSP	PARKER-HANNIFI	4A-NP6LR-G-SSP	PARKER-HANNIFI
VALVE, NEEDLE BRS 1/4T WHITEY B1VS4	P-H 4A-V4LR-B	PARKER-HANNIFI	4A-V4LR-B	PARKER-HANNIFI
VALVE, NEEDLE SS 316 1/4T WHITEY SS1VS4	3000PSI MAX P-H 4A-V4LR-SS|SENTRY 4-04002D	PARKER-HANNIFI	4A-V4LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 1/4 P-H 4F-V6LN	WHITEY SS1RF4 SWAGELOK SS1RF4	PARKER-HANNIFI	4F-V6LN	PARKER-HANNIFI
VALVE, NEEDLE BRS 1/4 FPT WHITEY B-1RF4	P-H 4F-V6LN-B	PARKER-HANNIFI	4F-V6LN-B	PARKER-HANNIFI
VALVE, NEEDLE SS 316 1/4 P-H 4F-V6LR-SS	1/4 X 1/4 FNPT CONN 5000PSI	PARKER-HANNIFI	4F-V6LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 316 1/4 P-H 4M-V4LR-SS	1/4 X 1/4 MPT CONN 5000PSIG	PARKER-HANNIFI	4M-V4LR-SS	PARKER-HANNIFI
VALVE, NEEDLE BRS 1/4 P-H 4ZV4AQ-BP	300PSI SGL FERRULE COMP TUBE FITTINGS|FLUOROCARB RBR STEM SEAL/PKG PTFE STEM	PARKER-HANNIFI	4Z-V4AQ-BP	PARKER-HANNIFI
VALVE, NEEDLE SS 1/4 TUBE P-H 4Z-V4LN-SS		PARKER-HANNIFI	4Z-V4LN-SS	PARKER-HANNIFI
VALVE, NEEDLE BRS 300PSI 1/4 T X 1/4 T	P-H 4Z-V4LQ-BP PANEL MTG COMP	PARKER-HANNIFI	4Z-V4LQ-BP	PARKER-HANNIFI
VALVE, NEEDLE 1/4 BRS P-H 4Z-V4LR-B	1/4 T X 1/4 T SGL FERRULE COMP TYPE|3000PSI CWP BLUNT STEM	PARKER-HANNIFI	4Z-V4LR-B	PARKER-HANNIFI
VALVE, NEEDLE BRS 3/8T WHITEY B1VS6	P-H 6A-V6LR-B GYROLOK 33126	PARKER-HANNIFI	6A-V6LR-B	PARKER-HANNIFI
VALVE, NEEDLE SS 316 3/8T WHITEY SS1VS6	3000PSI MAX P-H 6A-V6LR-SS	PARKER-HANNIFI	6A-V6LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 316 3/8 P-H 6F-V8LR-SS	3/8 X 3/8 FNPT CONN 5000PSIG	PARKER-HANNIFI	6F-V8LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 316 3/8 P-H 6M-V6LR-SS	3/8 X 3/8 MPT CONN 5000PSI INLINE TYPE	PARKER-HANNIFI	6M-V6LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 316 3/8 P-H 6Z-V4LR-SS	3/8 X 3/8 TUBE CONN 5000PSI V4 SERIES	PARKER-HANNIFI	6Z-V4LR-SS	PARKER-HANNIFI
VALVE, NEEDLE BRZ 3/8 P-H 6ZV6LR-B		PARKER-HANNIFI	6ZV6LR-B	PARKER-HANNIFI
VALVE, NEEDLE SS 316 1/2 P-H 8A-V8LR-SS	1/2 T X 1/2 T	PARKER-HANNIFI	8A-V8LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 316 1/2 P-H 8F-V12LR-SS	1/2 X 1/2 FNPT CONN 5000PSIG	PARKER-HANNIFI	8F-V12LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 1/2 MPT PARKER	NO 8M-V8LR-SS	PARKER-HANNIFI	8M-V8LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 1/2 IN P-H 8Z-V6LN		PARKER-HANNIFI	8Z-V6LN	PARKER-HANNIFI
VALVE, NEEDLE SS 316 1/2 P-H 8Z-V6LR-SS	1/2 X 1/2 TUBE CONN 5000PSIG COMPRESSION|TYPE	PARKER-HANNIFI	8Z-V6LR-SS	PARKER-HANNIFI
VALVE, NEEDLE SS 3/8 P-H 8Z-V8LN-B	5000PSI T-HANDLE OPER	PARKER-HANNIFI	8Z-V8LN-B	PARKER-HANNIFI
VALVE, NEEDLE 1/2 P-H 8Z-V8LR-SS	*BELOIT 265577-002	PARKER-HANNIFI	8Z-V8LR-SS	PARKER-HANNIFI
VALVE, NEEDLE STL 1/4 FPT P-H N-400-S	5000 PSI MAX 5 GPM	PARKER-HANNIFI	N-400-S	PARKER-HANNIFI
VALVE, NEEDLE 3/8 IN IPS P-H N-600-S10	3000 PSI ADJUSTMENT UP TO 8 GPM|HAM-LET H300SS-L-R3/8RS	PARKER-HANNIFI	N-600-S10	PARKER-HANNIFI
VALVE, NEEDLE BRZ 3/4 NPT 2000LB 25GPM	P-H N1200B DELTROL EN35B GL&amp;V VAL0031384|*BELOIT MC2-75-118D METSO 641205260A|*LAMB 0001027, 0004034, 0024044	PARKER-HANNIFI	N1200B	PARKER-HANNIFI

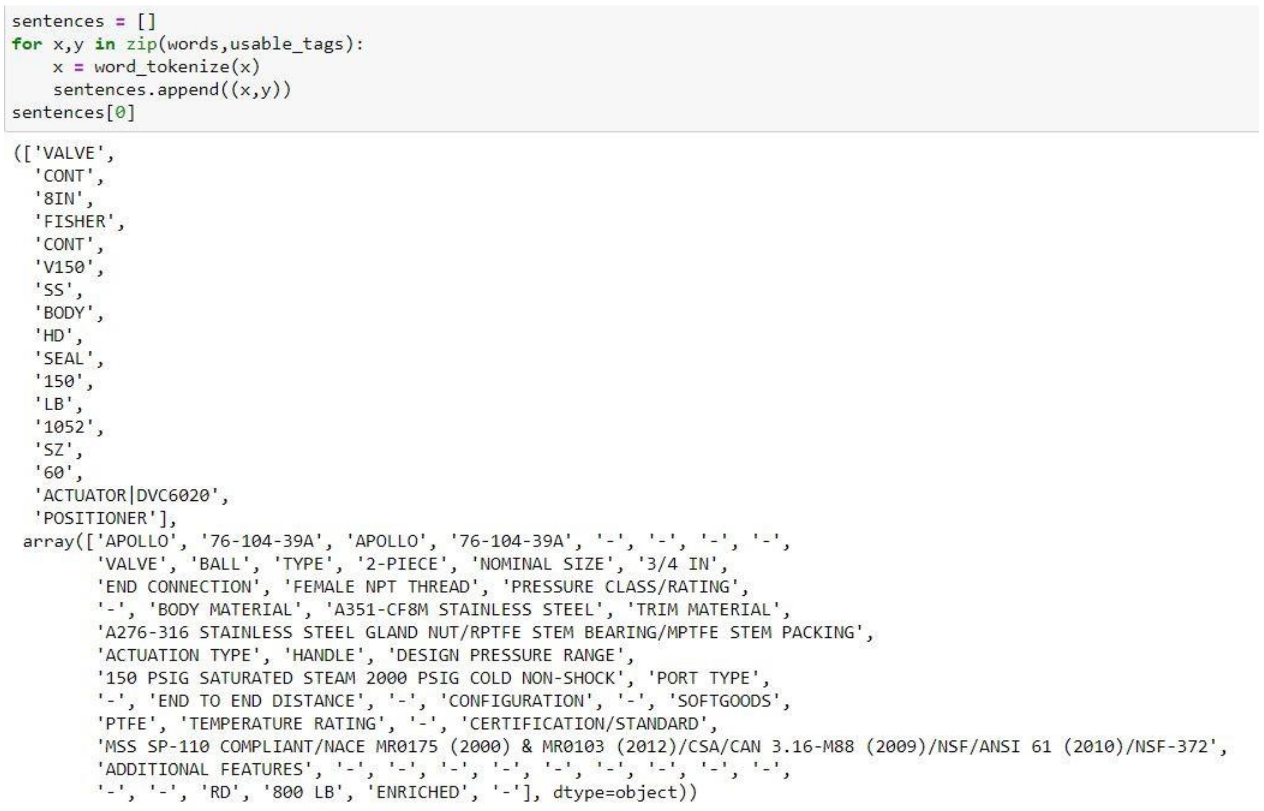
The image shown here is of the dataset given which is to classify a code like description into entities.However there are some abbreviations used between the description and the final classified form(eg-BRS in the description becomes BRASS finally).Therefor a traditional NER approach is not feasible for the given dataset.Therefore an approach of machine translation was followed where the description and the final classified form were treated as 2 sentences from 2 different languages.

Pre-Processing of the dataset.

As the dataset is in the csv format the pandas library has been used to get the data according to words and corresponding labels.

The descriptions were concatenated into our input sentences and were tokenized later on.A similar process was done on the entity values of each column.
Also some columns such as product code and URL link were dropped from the data to improve the accuracy as they don't have a logical basis of prediction.

Finally they were merged into ordered pairs as shown below.

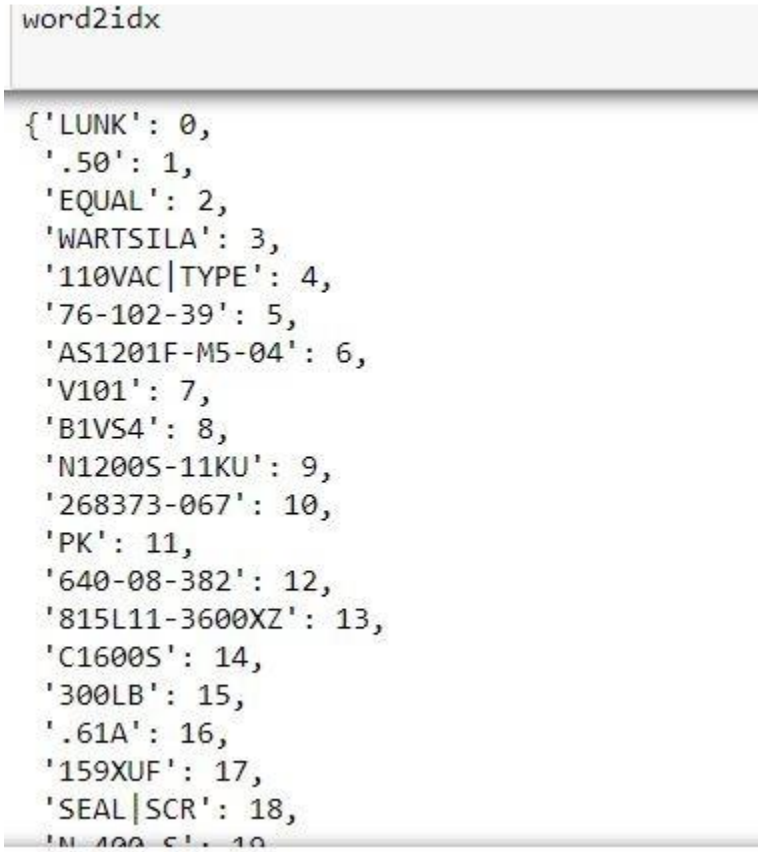


```python
sentences = []
for x,y in zip(words,usable_tags):
    x = word_tokenize(x)
    sentences.append((x,y))
sentences[0]
```

```
(['VALVE',
  'CONT',
  '8IN',
  'FISHER',
  'CONT',
  'V150',
  'SS',
  'BODY',
  'HD',
  'SEAL',
  '150',
  'LB',
  '1052',
  'SZ',
  '60',
  'ACTUATOR|DVC6020',
  'POSITIONER'],
 array(['APOLLO', '76-104-39A', 'APOLLO', '76-104-39A', '-', '-', '-', '-',
        'VALVE', 'BALL', 'TYPE', '2-PIECE', 'NOMINAL SIZE', '3/4 IN',
        'END CONNECTION', 'FEMALE NPT THREAD', 'PRESSURE CLASS/RATING',
        '-', 'BODY MATERIAL', 'A351-CF8M STAINLESS STEEL', 'TRIM MATERIAL',
        'A276-316 STAINLESS STEEL GLAND NUT/RPTFE STEM BEARING/MPTFE STEM PACKING',
        'ACTUATION TYPE', 'HANDLE', 'DESIGN PRESSURE RANGE',
        '150 PSIG SATURATED STEAM 2000 PSIG COLD NON-SHOCK', 'PORT TYPE',
        '-', 'END TO END DISTANCE', '-', 'CONFIGURATION', '-', 'SOFTGOODS',
        'PTFE', 'TEMPERATURE RATING', '-', 'CERTIFICATION/STANDARD',
        'MSS SP-110 COMPLIANT/NACE MR0175 (2000) & MR0103 (2012)/CSA/CAN 3.16-M88 (2009)/NSF/ANSI 61 (2010)/NSF-372',
        'ADDITIONAL FEATURES', '-', '-', '-', '-', '-', '-', '-', '-', '-',
        '-', '-', 'RD', '800 LB', 'ENRICHED', '-'], dtype=object))
```

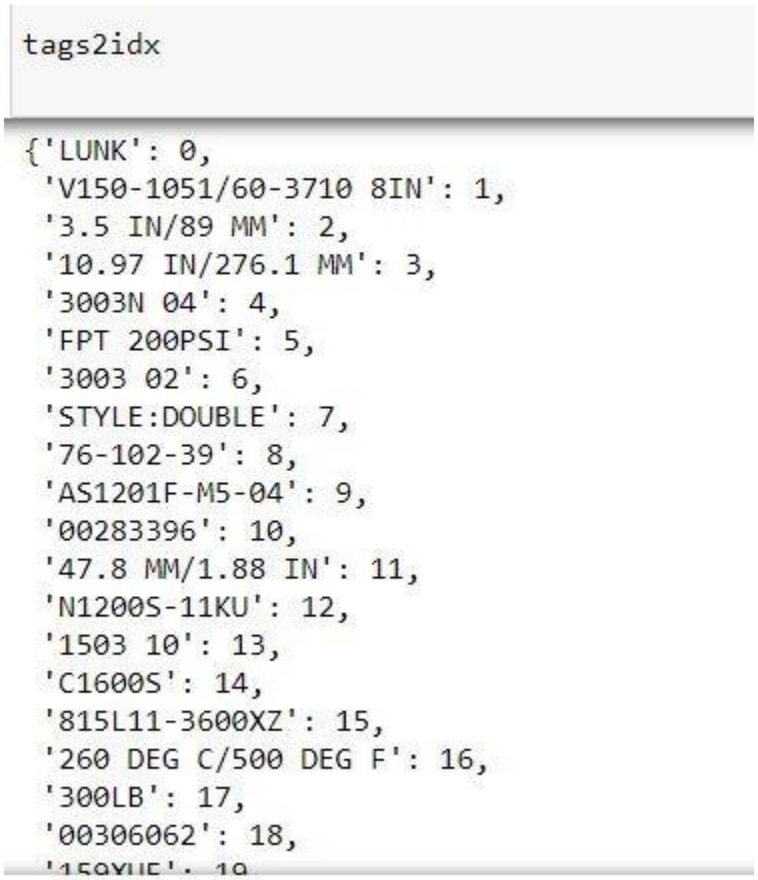
It can be seen above that the description and the corresponding classified values are in the form of an initial sentence and translation.

After this to begin the process of feeding them into a network two dictionaries are created for the inputs as well as the outputs.



```
word2idx
```

```
{'LUNK': 0,
 '.50': 1,
 'EQUAL': 2,
 'WARTSILA': 3,
 '110VAC|TYPE': 4,
 '76-102-39': 5,
 'AS1201F-M5-04': 6,
 'V101': 7,
 'B1VS4': 8,
 'N1200S-11KU': 9,
 '268373-067': 10,
 'PK': 11,
 '640-08-382': 12,
 '815L11-3600XZ': 13,
 'C1600S': 14,
 '300LB': 15,
 '.61A': 16,
 '159XUF': 17,
 'SEAL|SCR': 18,
```

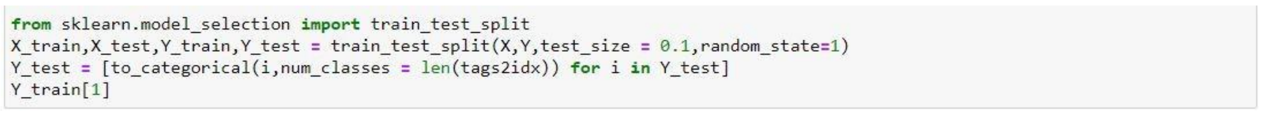


```
tags2idx
```

```
{'LUNK': 0,
 'V150-1051/60-3710 8IN': 1,
 '3.5 IN/89 MM': 2,
 '10.97 IN/276.1 MM': 3,
 '3003N 04': 4,
 'FPT 200PSI': 5,
 '3003 02': 6,
 'STYLE:DOUBLE': 7,
 '76-102-39': 8,
 'AS1201F-M5-04': 9,
 '00283396': 10,
 '47.8 MM/1.88 IN': 11,
 'N1200S-11KU': 12,
 '1503 10': 13,
 'C1600S': 14,
 '815L11-3600XZ': 15,
 '260 DEG C/500 DEG F': 16,
 '300LB': 17,
 '00306062': 18,
```

Now using these dictionaries the sentences are converted into numeric form and stored as features and labels separately.The features are padded to a common length.

They're divided into training and test sets using the sklearn library as shown



```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.1,random_state=1)
Y_test = [to_categorical(i,num_classes = len(tags2idx)) for i in Y_test]
Y_train[1]
```

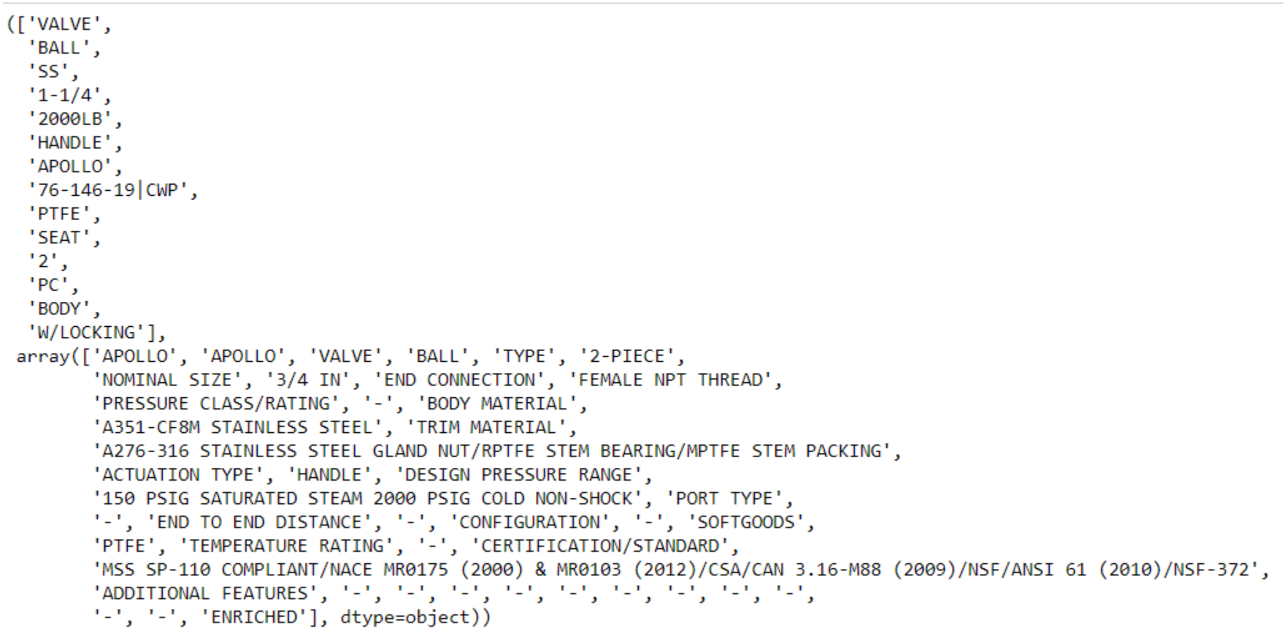
The labels are converted into the one-hot form to help with the softmax identifier.

Idea behind Translation :

Ideally a NER problem should not require something as complex as translation . However on clear observation of the data a few inconsistencies can be noticed.

In the data shown below it can be noticed that 'SS' on the description becomes 'STAINLESS STEEL' finally . Similar situations can be observed for certain other material names and company Names.

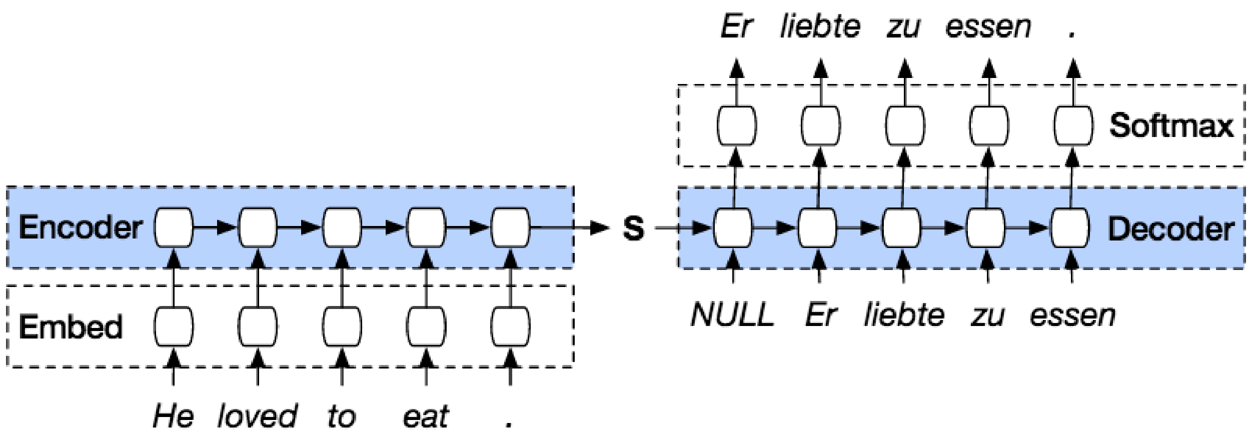
An obvious approach would be to hardcode the words with their respective abbreviations. However This approach is extremely inefficient and not very flexible when applying the model on a bigger set of data with more such cases. Thereby translation helps in making the model understand the appropriate relation between the abbreviation and the word so that the prediction is more general and flexible.



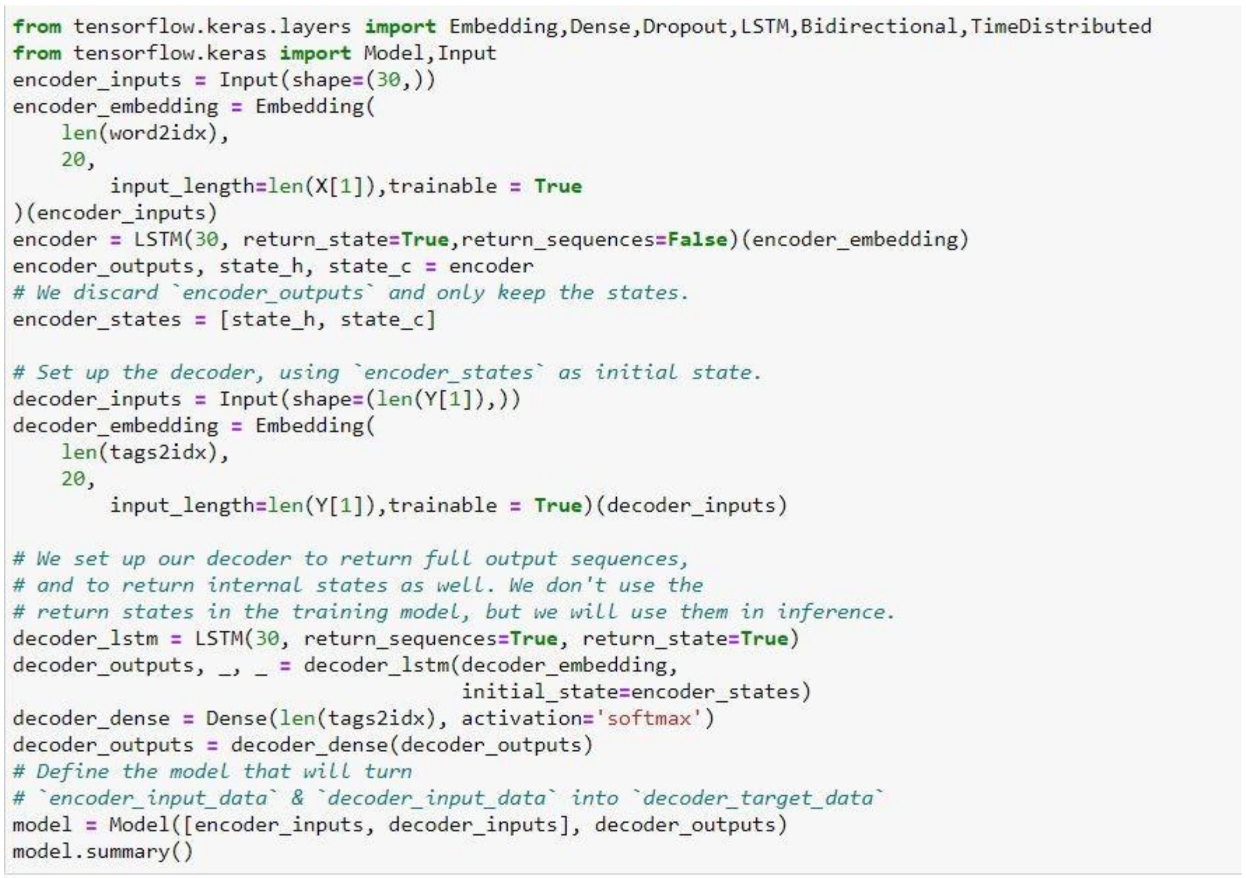
```
(['VALVE',
  'BALL',
  'SS',
  '1-1/4',
  '2000LB',
  'HANDLE',
  'APOLLO',
  '76-146-19|CWP',
  'PTFE',
  'SEAT',
  '2',
  'PC',
  'BODY',
  'W/LOCKING'],
 array(['APOLLO', 'APOLLO', 'VALVE', 'BALL', 'TYPE', '2-PIECE',
        'NOMINAL SIZE', '3/4 IN', 'END CONNECTION', 'FEMALE NPT THREAD',
        'PRESSURE CLASS/RATING', '-', 'BODY MATERIAL',
        'A351-CF8M STAINLESS STEEL', 'TRIM MATERIAL',
        'A276-316 STAINLESS STEEL GLAND NUT/RPTFE STEM BEARING/MPTFE STEM PACKING',
        'ACTUATION TYPE', 'HANDLE', 'DESIGN PRESSURE RANGE',
        '150 PSIG SATURATED STEAM 2000 PSIG COLD NON-SHOCK', 'PORT TYPE',
        '-', 'END TO END DISTANCE', '-', 'CONFIGURATION', '-', 'SOFTGOODS',
        'PTFE', 'TEMPERATURE RATING', '-', 'CERTIFICATION/STANDARD',
        'MSS SP-110 COMPLIANT/NACE MR0175 (2000) & MR0103 (2012)/CSA/CAN 3.16-M88 (2009)/NSF/ANSI 61 (2010)/NSF-372',
        'ADDITIONAL FEATURES', '-', '-', '-', '-', '-', '-', '-', '-', '-',
        '-', '-', 'ENRICHED'], dtype=object))
```

Constructing and Training the Neural Network Model.

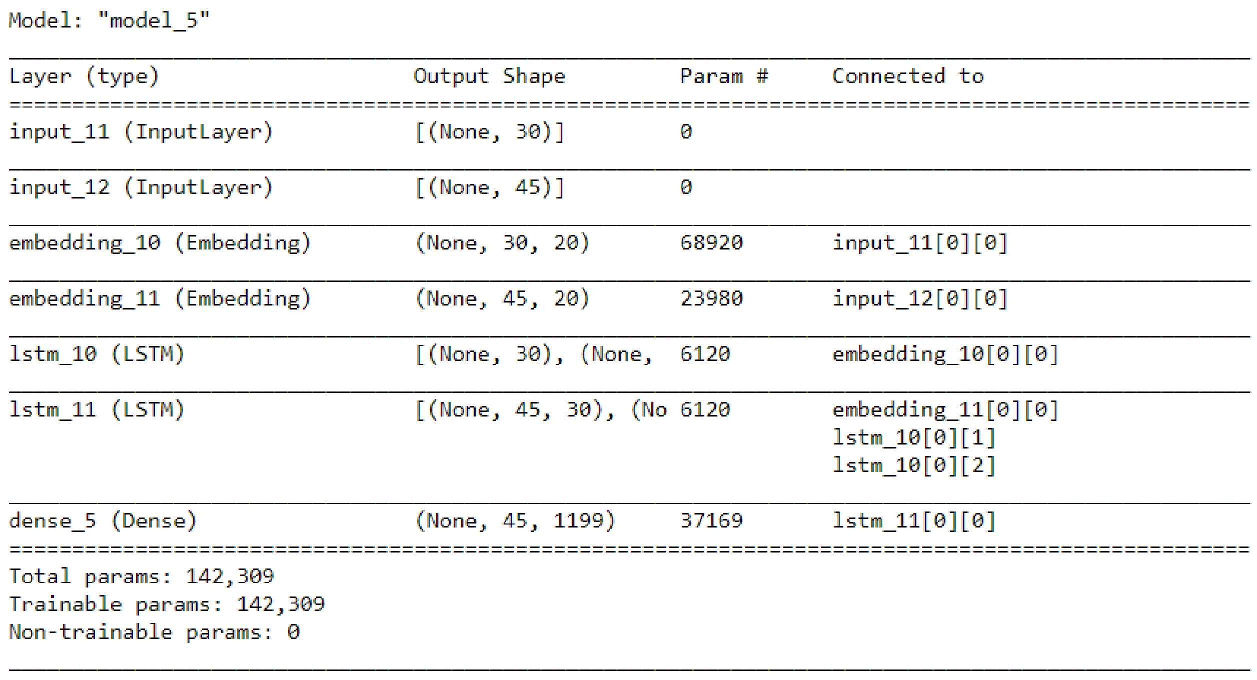
For translation purpose a seq2seq model has been utilised . The model will have an encoder which shall pass a thought vector to the decoder as shown.



The above model illustrates the idea behind the encoder and decoder where s is the thought vector.



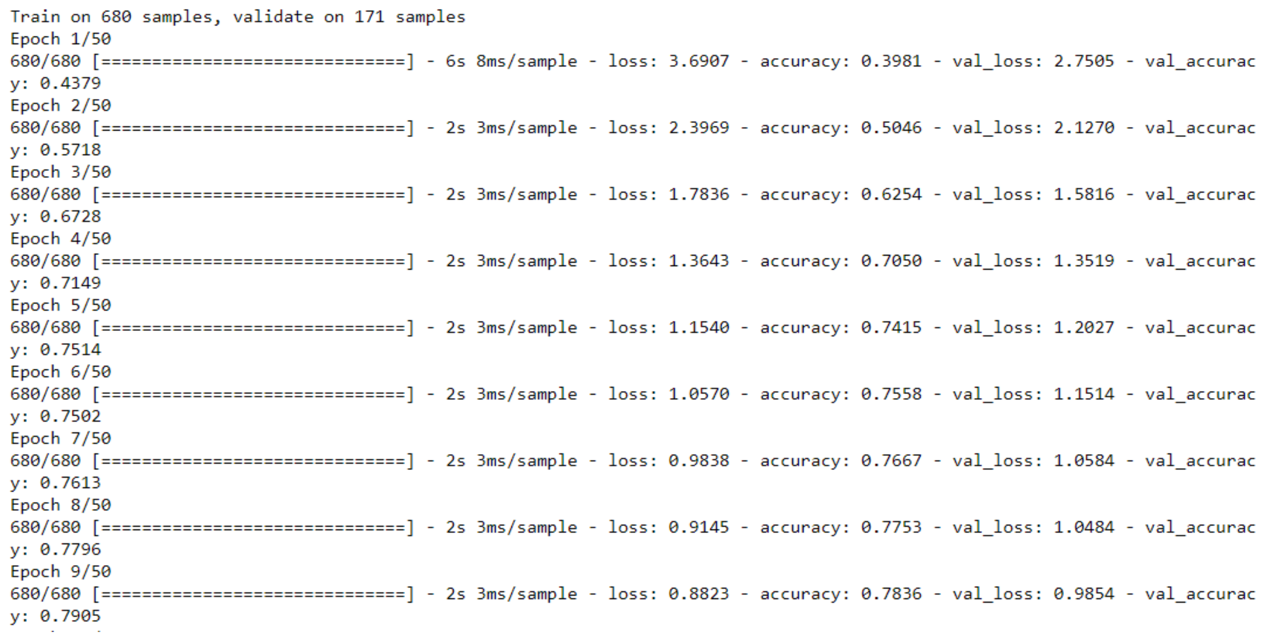
```
from tensorflow.keras.layers import Embedding,Dense,Dropout,LSTM,Bidirectional,TimeDistributed
from tensorflow.keras import Model,Input
encoder_inputs = Input(shape=(30,))
encoder_embedding = Embedding(
    len(word2idx),
    20,
        input_length=len(X[1]),trainable = True
)(encoder_inputs)
encoder = LSTM(30, return_state=True,return_sequences=False)(encoder_embedding)
encoder_outputs, state_h, state_c = encoder
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]

# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(len(Y[1]),))
decoder_embedding = Embedding(
    len(tags2idx),
    20,
        input_length=len(Y[1]),trainable = True)(decoder_inputs)

# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_lstm = LSTM(30, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_embedding,
                                     initial_state=encoder_states)
decoder_dense = Dense(len(tags2idx), activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
model.summary()
```



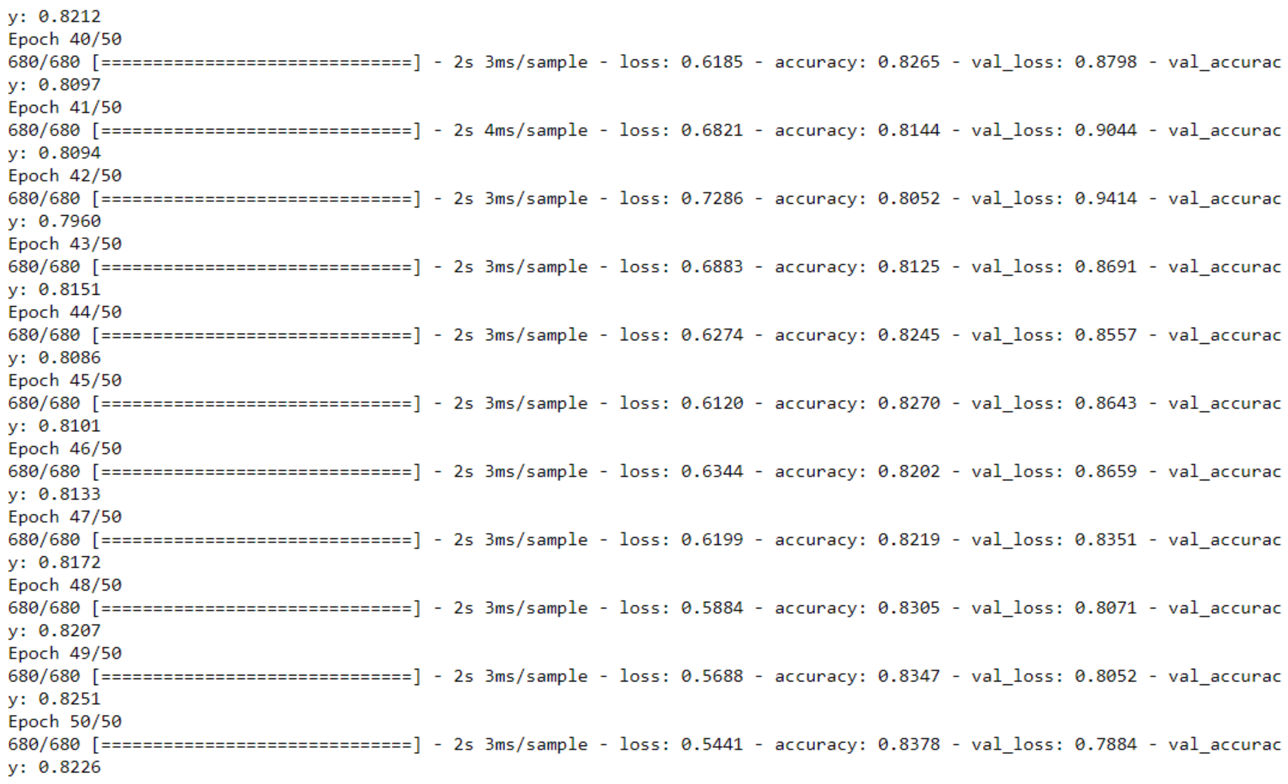
```
Model: "model_5"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_11 (InputLayer)           [(None, 30)]         0
__________________________________________________________________________________________________
input_12 (InputLayer)           [(None, 45)]         0
__________________________________________________________________________________________________
embedding_10 (Embedding)        (None, 30, 20)       68920       input_11[0][0]
__________________________________________________________________________________________________
embedding_11 (Embedding)        (None, 45, 20)       23980       input_12[0][0]
__________________________________________________________________________________________________
lstm_10 (LSTM)                  [(None, 30), (None,  6120        embedding_10[0][0]
__________________________________________________________________________________________________
lstm_11 (LSTM)                  [(None, 45, 30), (No 6120        embedding_11[0][0]
                                                                 lstm_10[0][1]
                                                                 lstm_10[0][2]
__________________________________________________________________________________________________
dense_5 (Dense)                 (None, 45, 1199)     37169       lstm_11[0][0]
==================================================================================================
Total params: 142,309
Trainable params: 142,309
Non-trainable params: 0
__________________________________________________________________________________________________
```

The model is a complex one with a high number of parameters.

This model shall use the adam optimizer and categorical cross entropy as the loss function. It will train on 50 epochs of data with a batch size of 32.



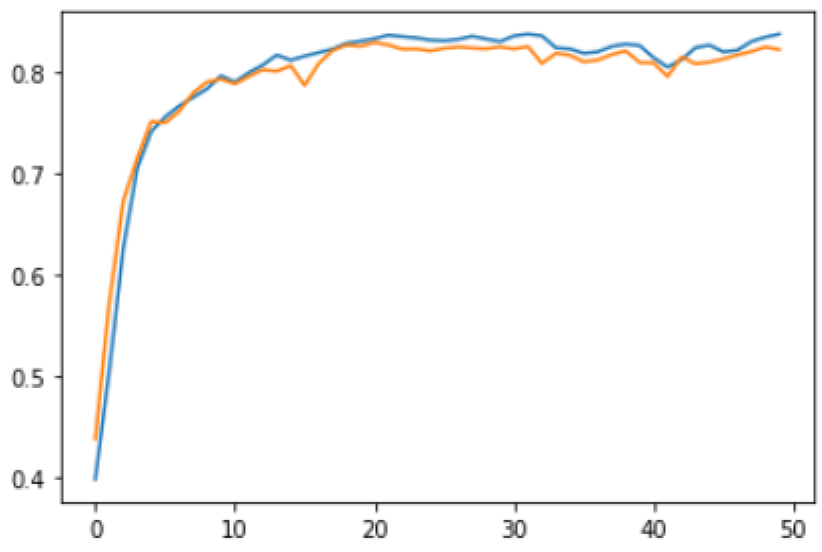
```
Train on 680 samples, validate on 171 samples
Epoch 1/50
680/680 [==============================] - 6s 8ms/sample - loss: 3.6907 - accuracy: 0.3981 - val_loss: 2.7505 - val_accurac
y: 0.4379
Epoch 2/50
680/680 [==============================] - 2s 3ms/sample - loss: 2.3969 - accuracy: 0.5046 - val_loss: 2.1270 - val_accurac
y: 0.5718
Epoch 3/50
680/680 [==============================] - 2s 3ms/sample - loss: 1.7836 - accuracy: 0.6254 - val_loss: 1.5816 - val_accurac
y: 0.6728
Epoch 4/50
680/680 [==============================] - 2s 3ms/sample - loss: 1.3643 - accuracy: 0.7050 - val_loss: 1.3519 - val_accurac
y: 0.7149
Epoch 5/50
680/680 [==============================] - 2s 3ms/sample - loss: 1.1540 - accuracy: 0.7415 - val_loss: 1.2027 - val_accurac
y: 0.7514
Epoch 6/50
680/680 [==============================] - 2s 3ms/sample - loss: 1.0570 - accuracy: 0.7558 - val_loss: 1.1514 - val_accurac
y: 0.7502
Epoch 7/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.9838 - accuracy: 0.7667 - val_loss: 1.0584 - val_accurac
y: 0.7613
Epoch 8/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.9145 - accuracy: 0.7753 - val_loss: 1.0484 - val_accurac
y: 0.7796
Epoch 9/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.8823 - accuracy: 0.7836 - val_loss: 0.9854 - val_accurac
y: 0.7905
```



```
y: 0.8212
Epoch 40/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.6185 - accuracy: 0.8265 - val_loss: 0.8798 - val_accurac
y: 0.8097
Epoch 41/50
680/680 [==============================] - 2s 4ms/sample - loss: 0.6821 - accuracy: 0.8144 - val_loss: 0.9044 - val_accurac
y: 0.8094
Epoch 42/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.7286 - accuracy: 0.8052 - val_loss: 0.9414 - val_accurac
y: 0.7960
Epoch 43/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.6883 - accuracy: 0.8125 - val_loss: 0.8691 - val_accurac
y: 0.8151
Epoch 44/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.6274 - accuracy: 0.8245 - val_loss: 0.8557 - val_accurac
y: 0.8086
Epoch 45/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.6120 - accuracy: 0.8270 - val_loss: 0.8643 - val_accurac
y: 0.8101
Epoch 46/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.6344 - accuracy: 0.8202 - val_loss: 0.8659 - val_accurac
y: 0.8133
Epoch 47/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.6199 - accuracy: 0.8219 - val_loss: 0.8351 - val_accurac
y: 0.8172
Epoch 48/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.5884 - accuracy: 0.8305 - val_loss: 0.8071 - val_accurac
y: 0.8207
Epoch 49/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.5688 - accuracy: 0.8347 - val_loss: 0.8052 - val_accurac
y: 0.8251
Epoch 50/50
680/680 [==============================] - 2s 3ms/sample - loss: 0.5441 - accuracy: 0.8378 - val_loss: 0.7884 - val_accurac
y: 0.8226
```

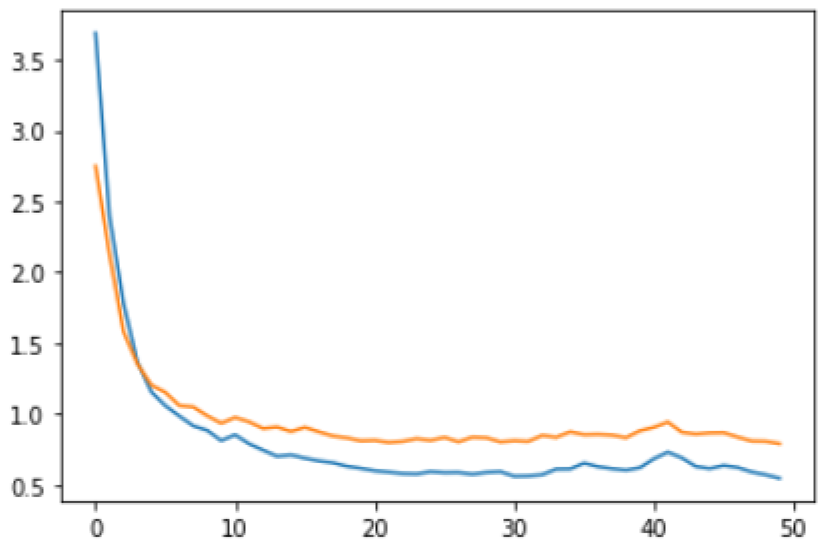
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

```
]:  [<matplotlib.lines.Line2D at 0x181eb44d548>]
```



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

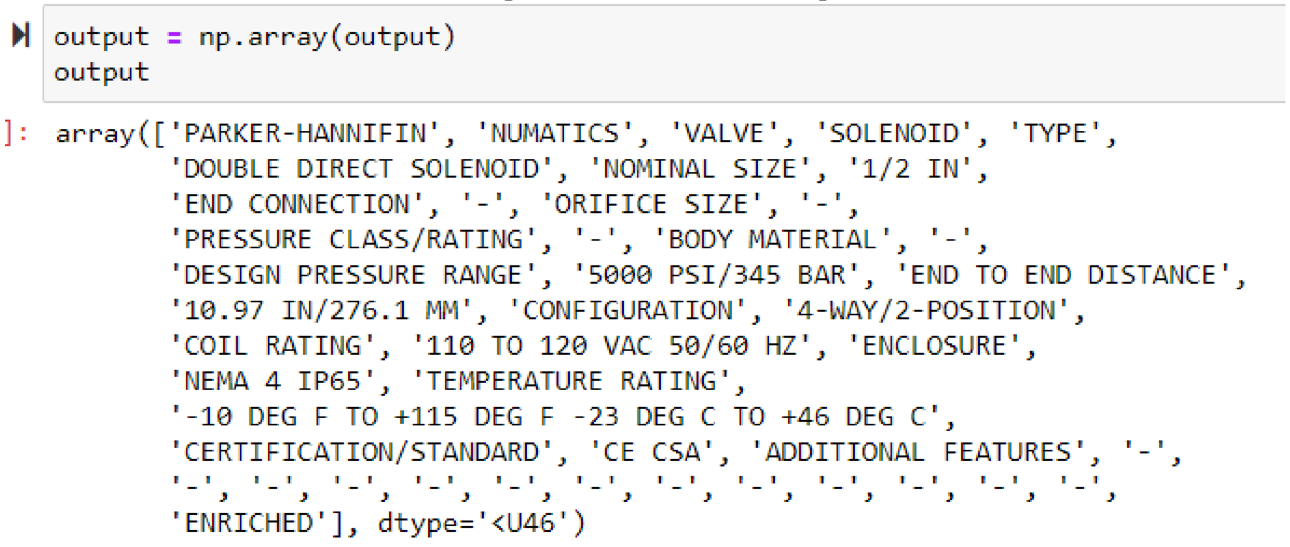
```
]:  [<matplotlib.lines.Line2D at 0x181d9055488>]
```



Around 82% of validation accuracy is reached at the end of 50 epochs.This model was trained using teacher forcing which helped in the high accuracy seen.

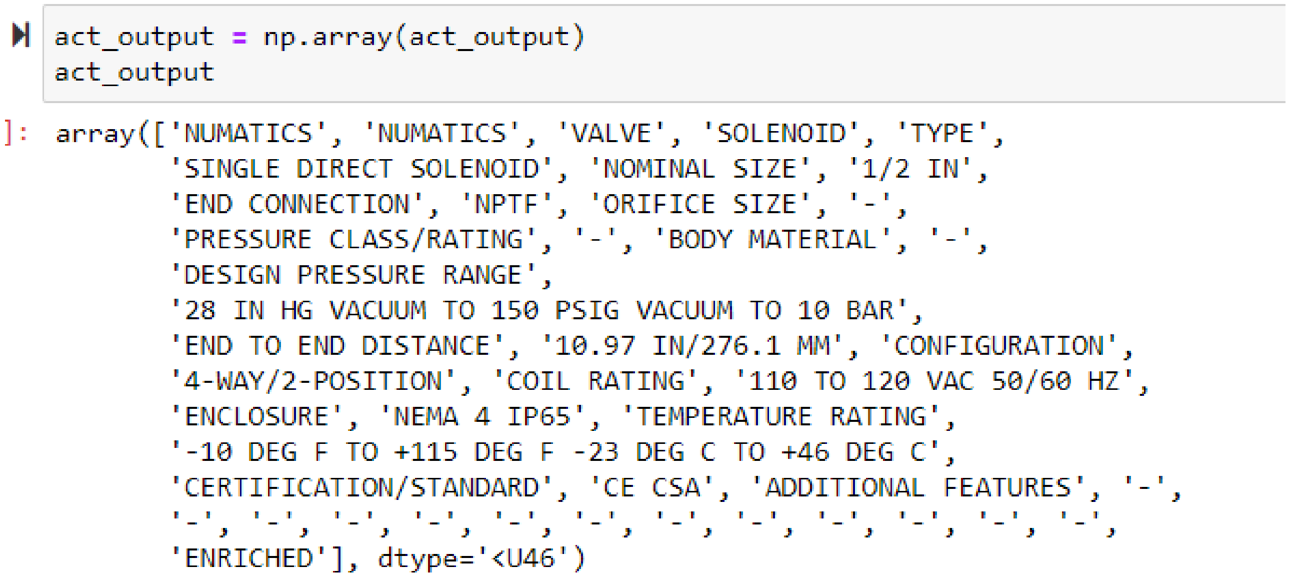
PREDICTIONS

The predictions done by this model are pretty accurate as shown. However certain discrepancies continue to exist which can be attributed to the unorganized nature of the data and certain features Such as the measurements that are not given in the initial description.



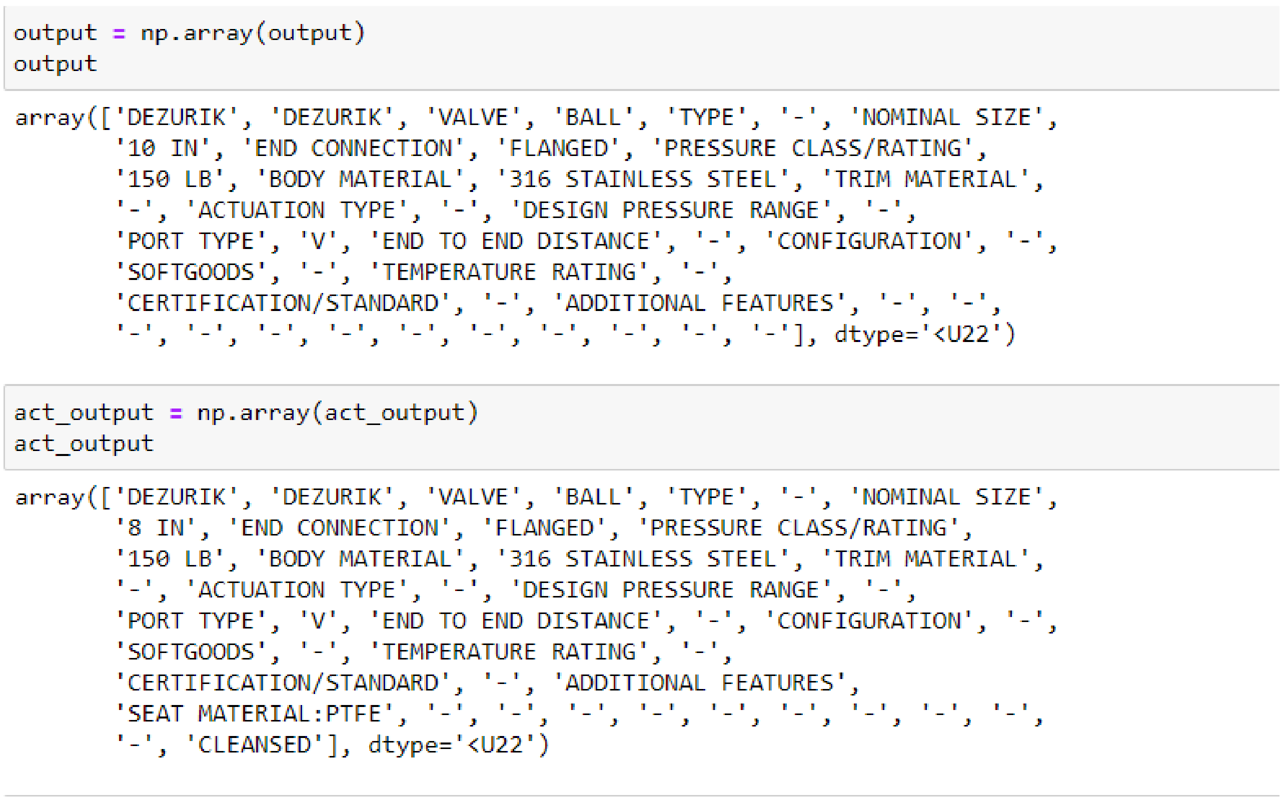
```
output = np.array(output)
output
```

```
]: array(['PARKER-HANNIFIN', 'NUMATICS', 'VALVE', 'SOLENOID', 'TYPE',
          'DOUBLE DIRECT SOLENOID', 'NOMINAL SIZE', '1/2 IN',
          'END CONNECTION', '-', 'ORIFICE SIZE', '-',
          'PRESSURE CLASS/RATING', '-', 'BODY MATERIAL', '-',
          'DESIGN PRESSURE RANGE', '5000 PSI/345 BAR', 'END TO END DISTANCE',
          '10.97 IN/276.1 MM', 'CONFIGURATION', '4-WAY/2-POSITION',
          'COIL RATING', '110 TO 120 VAC 50/60 HZ', 'ENCLOSURE',
          'NEMA 4 IP65', 'TEMPERATURE RATING',
          '-10 DEG F TO +115 DEG F -23 DEG C TO +46 DEG C',
          'CERTIFICATION/STANDARD', 'CE CSA', 'ADDITIONAL FEATURES', '-',
          '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-',
          'ENRICHED'], dtype='<U46')
```



```
act_output = np.array(act_output)
act_output
```

```
]: array(['NUMATICS', 'NUMATICS', 'VALVE', 'SOLENOID', 'TYPE',
          'SINGLE DIRECT SOLENOID', 'NOMINAL SIZE', '1/2 IN',
          'END CONNECTION', 'NPTF', 'ORIFICE SIZE', '-',
          'PRESSURE CLASS/RATING', '-', 'BODY MATERIAL', '-',
          'DESIGN PRESSURE RANGE',
          '28 IN HG VACUUM TO 150 PSIG VACUUM TO 10 BAR',
          'END TO END DISTANCE', '10.97 IN/276.1 MM', 'CONFIGURATION',
          '4-WAY/2-POSITION', 'COIL RATING', '110 TO 120 VAC 50/60 HZ',
          'ENCLOSURE', 'NEMA 4 IP65', 'TEMPERATURE RATING',
          '-10 DEG F TO +115 DEG F -23 DEG C TO +46 DEG C',
          'CERTIFICATION/STANDARD', 'CE CSA', 'ADDITIONAL FEATURES', '-',
          '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-', '-',
          'ENRICHED'], dtype='<U46')
```



```
output = np.array(output)
output
```

```
array(['DEZURIK', 'DEZURIK', 'VALVE', 'BALL', 'TYPE', '-', 'NOMINAL SIZE',
       '10 IN', 'END CONNECTION', 'FLANGED', 'PRESSURE CLASS/RATING',
       '150 LB', 'BODY MATERIAL', '316 STAINLESS STEEL', 'TRIM MATERIAL',
       '-', 'ACTUATION TYPE', '-', 'DESIGN PRESSURE RANGE', '-',
       'PORT TYPE', 'V', 'END TO END DISTANCE', '-', 'CONFIGURATION', '-',
       'SOFTGOODS', '-', 'TEMPERATURE RATING', '-',
       'CERTIFICATION/STANDARD', '-', 'ADDITIONAL FEATURES', '-', '-',
       '-', '-', '-', '-', '-', '-', '-', '-', '-', '-'], dtype='<U22')
```

```
act_output = np.array(act_output)
act_output
```

```
array(['DEZURIK', 'DEZURIK', 'VALVE', 'BALL', 'TYPE', '-', 'NOMINAL SIZE',
       '8 IN', 'END CONNECTION', 'FLANGED', 'PRESSURE CLASS/RATING',
       '150 LB', 'BODY MATERIAL', '316 STAINLESS STEEL', 'TRIM MATERIAL',
       '-', 'ACTUATION TYPE', '-', 'DESIGN PRESSURE RANGE', '-',
       'PORT TYPE', 'V', 'END TO END DISTANCE', '-', 'CONFIGURATION', '-',
       'SOFTGOODS', '-', 'TEMPERATURE RATING', '-',
       'CERTIFICATION/STANDARD', '-', 'ADDITIONAL FEATURES',
       'SEAT MATERIAL:PTFE', '-', '-', '-', '-', '-', '-', '-', '-', '-',
       '-', 'CLEANSED'], dtype='<U22')
```

CONCLUSION

The model can be seen to have a proper fit with a above average validation accuracy.It also reaches the high accuracy by the 20th epoch which shows the smooth nature of the training process.Also the training process is relatively quick .The high loss observed can be attributed as a common phenomenon in translation based problems due to the large possibilities that the softmax faces.This network is thereby extremely successful in performing Named Entity Recognition through machine translation.

However this model could be improved by incorporating attention into the seq2seq model.Also if the dataset could be tuned to improve the accuracy.