Embeddings

What are Embeddings?
An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings make it easier to do machine learning on large inputs like sparse vectors representing words. Ideally, an embedding captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space. An embedding can be learned and reused across models

** Low dimensional = less number of inout features.
** Sparse vectors = featurest hat have mostly zero values. No actual values.

what are its advntages?
Helps represents words as meanigful semantic real values vectors. Also it helps in genralising the issue

TYpes of embeddings

a. Bag of words or count vector
b. TF-ID
c. Glove
d. Fasttext

How many Dimensions should be used for representing words?

Word embeddings generally ask how many dimensions would you like the embeddings to have- there is no optimal number for this.The trade off is between accuracy and the computational concerns.

** More dimanesions means the potential to compute increasingly accurate representation of words.
** But more dimensions also reuire high computational resources- memory , speed during the training and also the inference speed.

Inference speed =
Measures as Frames per second, namely the average iterations per second which can show how fast the model can handle the input. The higher the fast the better.

In [1]:

```
import os
import time
import pandas as pd
import numpy as np
import math
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from sklearn import metrics


from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation, CuDNNGRU, Conv
from keras.layers import Bidirectional, GlobalMaxPool1D
from keras.models import Model
from keras import initializers, regularizers, constraints, optimizers, layers

## CuDNN - Deep Neural Network Library , provides highly tuned implementations
```

In [2]:

```
#import tensorflow as tf; print(tf.reduce_sum(tf.random.normal([1000, 1000])))
import tensorflow as tf;
print(tf.config.list_physical_devices('GPU'))
```

In [22]:

```
train =open('train_csv.csv', 'r',errors='ignore')
```

In [3]:

```
train_df = pd.read_csv("train_csv.csv")
```

In [4]:

```
train_df
```

Out[4]:

	qid	question_text	target
0	00002165364db923c7e6	How did Quebec nationalists see their province...	0
1	000032939017120e6e44	Do you have an adopted dog, how would you enco...	0
2	0000412ca6e4628ce2cf	Why does velocity affect time? Does velocity a...	0
3	000042bf85aa498cd78e	How did Otto von Guericke used the Magdeburg h...	0
4	0000455dfa3e01eae3af	Can I convert montra helicon D to a mountain b...	0
...	...	...	...
1048570	cd76189d120969381147	What info does a Facebook page receive when yo...	0
1048571	cd762e8941f1ab6ae0bf	If no can know God, why believe in something y...	0
1048572	cd76362e61ce44bcff74	Is it true that the lunar cycle affects women'...	0
1048573	cd763d37d3d7cfcd42ce	What is the difference between the old currenc...	0
1048574	cd7642554d107f946d8a	What is the full form of DML?	0

1048575 rows × 3 columns

In [5]:

```
test = open("test.csv", 'r',errors='ignore')
```

In [6]:

```
test_df =pd.read_csv('test.csv')
```

In [7]:

```
test_df
```

Out[7]:

	qid	question_text
0	0000163e3ea7c7a74cd7	Why do so many women become so rude and arroga...
1	00002bd4fb5d505b9161	When should I apply for RV college of engineer...
2	00007756b4a147d2b0b3	What is it really like to be a nurse practitio...
3	000086e4b7e1c7146103	Who are entrepreneurs?
4	0000c4c3fbe8785a3090	Is education really making good people nowadays?
...	...	...
375801	ffff7fa746bd6d6197a9	How many countries listed in gold import in in...
375802	ffffa1be31c43046ab6b	Is there an alternative to dresses on formal p...
375803	ffffae173b6ca6bfa563	Where I can find best friendship quotes in Tel...
375804	ffffb1f7f1a008620287	What are the causes of refraction of light?
375805	fffff85473f4699474b0	Climate change is a worrying topic. How much t...

375806 rows × 2 columns

In [8]:

```
print("Train shape:", train_df.shape)
print("Test shape:" , test_df.shape)
```

```
Train shape: (1048575, 3)
Test shape: (375806, 2)
```

1.Split the training dataset into train and val sample. Cross validation is a time consuming process and so let us do simple train val split. 2.Fill up the missing values in the text column with 'na' 3.Tokenize the text column and convert them to vector sequences 4.Pad the sequence as needed - if the number of words in the text is greater than 'max_len' trunacate them to 'max_len' or if the number of words in the text is lesser than 'max_len' add zeros for remaining values.

In [9]:

```
## Train and val split

train_df , val_df = train_test_split(train_df, test_size=0.1, random_state=2014)

## Some config values
embed_size= 300
max_features = 50000 # how many unique words to use (i.e num rows in embedding vector)
maxlen = 100 # max number of words in a question to use
```

In [10]:

```
## Fill the missing values

train_X = train_df['question_text'].fillna("_na_").values
val_X = val_df['question_text'].fillna("_na_").values
test_X = val_df['question_text'].fillna("_na_").values

## Tokenize the sentences
tokenizer = Tokenizer(num_words= max_features)
tokenizer.fit_on_texts(list(train_X))
train_X = tokenizer.texts_to_sequences(train_X)
val_X = tokenizer.texts_to_sequences(val_X)
test_X = tokenizer.texts_to_sequences(test_X)

## Pad the sentences
train_X = pad_sequences(train_X, maxlen=maxlen)
val_X = pad_sequences(val_X, maxlen=maxlen)
test_X = pad_sequences(test_X, maxlen=maxlen)

## Get the target values
train_y = train_df['target'].values
val_y = val_df['target'].values
```

Birectional layers connects two hidden layers of the opposite directions to same output. Because of this generative deep learning, output layer gets the information from past or backwards and the future or forward states simultaneously.

GLOBALMAXpool1D , Global max pooling operation for 1D temporal data = Downsamples the input representation by taking the maximum value over the time dimension.
** Downsampling means to reduce the height and width of the feature maps as per the requirement.

In [11]:

```
inp = Input(shape=(maxlen,))
x= Embedding(max_features, embed_size)(inp)
x= Bidirectional(CuDNNGRU(64, return_sequences=True))(x)
x=GlobalMaxPool1D()(x)
x = Dense(16, activation="relu")(x)
x = Dropout(0.1)(x)
x = Dense(1, activation="sigmoid")(x)

model = Model(inputs=inp, outputs=x)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

print(model.summary())
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_1 (InputLayer)         [(None, 100)]             0

 embedding (Embedding)        (None, 100, 300)          15000000

 bidirectional (Bidirectiona  (None, 100, 128)          140544
 l)

 global_max_pooling1d (Globa  (None, 128)              0
 lMaxPooling1D)

 dense (Dense)                (None, 16)                2064

 dropout (Dropout)            (None, 16)                0

 dense_1 (Dense)              (None, 1)                 17

=================================================================
Total params: 15,142,625
Trainable params: 15,142,625
Non-trainable params: 0
_________________________________________________________________
None
```

Train the model using train sample and monitor the metric on the valid sample. Changing the epochs, batch_size and model parameters might give us a better model.

In [1]:

```
## Train the model
model.fit(train_X, train_y, batch_size=512, epochs=2, validation_data=(val_X, val_y))
```

```
Train on 1175509 samples, validate on 130613 samples
Epoch 1/2
1175509/1175509 [==============================] - 235s 200us/step - loss: 0.1225 - acc:
0.9538 - val_loss: 0.1073 - val_acc: 0.9563
Epoch 2/2
```

```
1175509/1175509 [==============================] - 232s 197us/step - loss: 0.0983 - acc:
0.9609 - val_loss: 0.1069 - val_acc: 0.9569
```

Now let us get the validation sample predictions and also get the best threshold for F1 score.

In [ ]:

```
pred_noemb_val_y = model.predict([val_X], batch_size=1024, verbose=1)
for thresh in np.arange(0.1, 0.501, 0.01):
    thresh = np.round(thresh, 2)
    print("F1 score at threshold {0} is {1}".format(thresh, metrics.f1_score(val_y, (pred_n
```

```
130613/130613 [==============================] - 6s 46us/step
F1 score at threshold 0.1 is 0.5736396808212038
F1 score at threshold 0.11 is 0.5815422477440525
F1 score at threshold 0.12 is 0.5890422417398578
F1 score at threshold 0.13 is 0.5948687350835322
F1 score at threshold 0.14 is 0.6007970198388634
F1 score at threshold 0.15 is 0.605882094342111
F1 score at threshold 0.16 is 0.612073584568673
F1 score at threshold 0.17 is 0.6171832517672647
F1 score at threshold 0.18 is 0.6215955541266706
F1 score at threshold 0.19 is 0.6253904610937574
F1 score at threshold 0.2 is 0.6289385422079455
F1 score at threshold 0.21 is 0.6333077954414864
F1 score at threshold 0.22 is 0.6377570818781528
F1 score at threshold 0.23 is 0.6392066375374343
F1 score at threshold 0.24 is 0.6413556626745515
F1 score at threshold 0.25 is 0.6430793937569005
F1 score at threshold 0.26 is 0.6459570580173595
F1 score at threshold 0.27 is 0.6481519507186858
F1 score at threshold 0.28 is 0.6500518134715025
F1 score at threshold 0.29 is 0.6513478147081915
F1 score at threshold 0.3 is 0.6529641762654549
F1 score at threshold 0.31 is 0.6532309660908507
F1 score at threshold 0.32 is 0.6534983853606028
F1 score at threshold 0.33 is 0.6543873947296931
F1 score at threshold 0.34 is 0.6545235090799364
F1 score at threshold 0.35 is 0.654829074012612
F1 score at threshold 0.36 is 0.6544216167932113
F1 score at threshold 0.37 is 0.6543140346924983
F1 score at threshold 0.38 is 0.6552703316674238
F1 score at threshold 0.39 is 0.6549077997938381
F1 score at threshold 0.4 is 0.6547069681587449
F1 score at threshold 0.41 is 0.6559758294114231
F1 score at threshold 0.42 is 0.6558875219683655
F1 score at threshold 0.43 is 0.6548672566371682
F1 score at threshold 0.44 is 0.654260063023961
F1 score at threshold 0.45 is 0.653401258615523
F1 score at threshold 0.46 is 0.6520713637738131
F1 score at threshold 0.47 is 0.6512535838467639
F1 score at threshold 0.48 is 0.6506216914932906
F1 score at threshold 0.49 is 0.6488601776507857
F1 score at threshold 0.5 is 0.6479349186483103
```

In [ ]:

```
#Now let us get the test set predictions as well and save them

pred_noemb_test_y = model.predict([test_X], batch_size=1024, verbose=1)

#Now that our model building is done, it might be a good idea to clean up some memory befor

del model, inp, x
import gc; gc.collect()
time.sleep(10)
```

```
56370/56370 [==============================] - 2s 44us/step
```