Image segmentation and Caption Generator.
Name : Neeraj Padarthi
Class ID: 20
LAB: 4

Objective

- Creating image segmentation using clustering.
- Create your own Show and Tell Model using your dataset. (Describe why did you choose the dataset for creating the model)
- Generate captions for your own dataset using the show and Tell model.
- Report your accuracy in BLEU and ROGUE measures.
- Checking the accuracy before and the image segmentation

Image Segmentation

- We have performed the image segmentation based on the clustering.
- Basically, clustering is a concept of dividing the data groups from the data set into similar groups of data, so that the data points are similar to the data points in that group than those in the other group.
- K-Means clustering is used on the images.
- Firstly, we will assign k initial clusters.
- Next we will randomly assign each data point to each of the k clusters.
- Now we would be calculating the centers of these clusters.
- We need to calculate the distance from the center of each cluster.
- Based on this distance, the points are re-assigned to the nearest cluster.
- Finally, we would be calculating the center from the newly formed clusters.

Image Segmentation - approach

- Importing the necessary libraries such as sklearn(for instantiating the K-Means clustering), glob(for file name pattern matching - reading the images from directories) and matplotlib(for reading and saving the images).
- We would be dividing the images by 255 to bring the pixel values to between 0 and 1
- As it is 3-dimensional images, for K-Means clustering we would be converting the images into a 2-dimensional array whose shape will be in (length*width, channel)
- Once it is converted into 2-dimensional images we would fit k-means algorithm on the reshaped array of elements.
- we have used 2 functions with K-Means clustering which are cluster_centers_function, and labels_function.
- The cluster_centers_function of k-means is used to return the cluster centers
- labels_ function will return the label for each pixel.

```
from sklearn.cluster import KMeans
import glob
from PIL import Image
import matplotlib.pyplot as plt
```

```
for img in glob.glob("*.jpg"):
    try:
        print(img)
        x =img[:img.find('.')]
        pic = plt.imread(img)/255
        print(pic.shape)
        plt.imshow(pic)
        pic_n = pic.reshape(pic.shape[0]*pic.shape[1], pic.shape[2])
        pic_n.shape
        kmeans = KMeans(n_clusters=8, random_state=0).fit(pic_n)
        pic2show = kmeans.cluster_centers_[kmeans.labels_]
        cluster_pic = pic2show.reshape(pic.shape[0], pic.shape[1], pic.shape[2])
        plt.imshow(cluster_pic)
        plt.savefig(x+".png")
    except:
        pass
```

Image1


NEXT
4 MILES

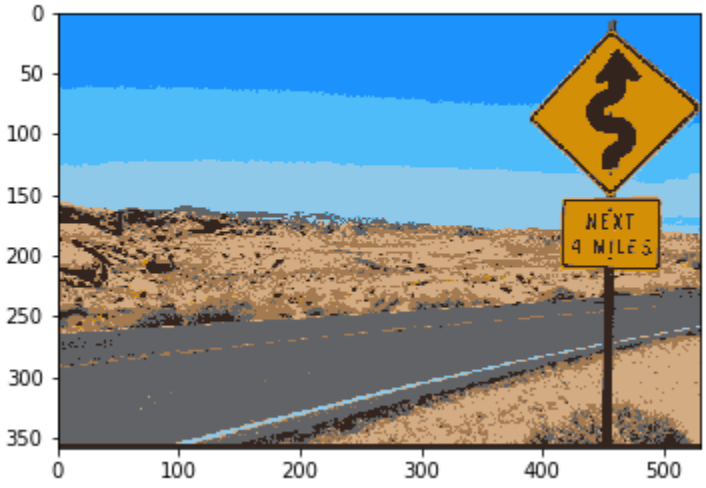

0
50
100
150
200
250
300
350
0
100
200
300
400
500
NEXT
4 MILES

Image2


STOP
NO
STOPPING
ANY
TIME

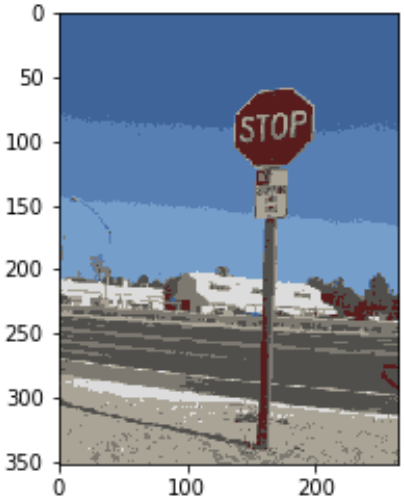
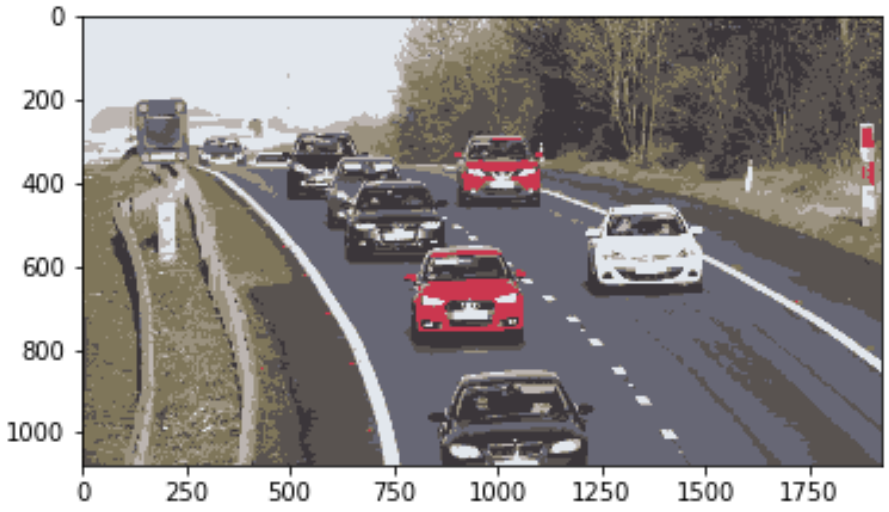

0
50
100
150
200
250
300
350
0
100
200
STOP

Image3



Creating Show and tell Model(Lab3 Continuation)

Extract Features

- Dataset images are feed into this model(VGG MODEL) to extract the image features from it.
- Over here we have removed the last layer to capture the features. Hence to avoid the classification in the last layer.
- Once the features are extracted we have saved it as Features.pkl

```
def extract_features(directory):
    # load the model
    model = VGG16()
    # re-structure the model
    model.layers.pop()
    model = Model(inputs=model.inputs, outputs=model.layers[-1].output)
    # summarize
    print(model.summary())
    # extract features from each photo
    features = dict()
    for name in listdir(directory):
        # load an image from file
        for imagePath in name:
            if imagePath == directory + '.DS_Store':
                continue
        filename = directory + '/' + name
        image = load_img(filename, target_size=(224, 224))
        # convert the image pixels to a numpy array
        image = img_to_array(image)
        # reshape data for the model
        image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
        # prepare the image for the VGG model
        image = preprocess_input(image)
        # get features
        feature = model.predict(image, verbose=0)
        # get image id
        image_id = name.split('.')[0]
        # store feature
        features[image_id] = feature
        print('>%s' % name)
    return features

# extract features from all images
directory = 'dataset'
features = extract_features(directory)
print('Extracted Features: %d' % len(features))
# save to file
dump(features, open('features.pkl', 'wb'))
```

```
extract_features()
featureExtraction × tst ×
>184.jpg
Extracted Features: 713
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Preprocessing the data

- Downloaded the SBU DataSet.
- Merging the two Input files into Pandas Data Frame columns(Captions Name, Image Links)
- Each Column data is respective to the 2 text files from the SBU Dataset.
- Performed using strip at the new line and append to a list.
- Pandas Data Frame is created on the top the two lists respectively.
- Search is performed to get the corresponding Image data.
- Logic implemented was created a DF with WRT keywords and finally all the Search DF's are merged into a new DF
- Searched with Sign, Exit, Mile and Route keywords.
- Once the data is fetched, iterated the records and downloaded the images for the keys.
- Used counter for image names.
- Once the keywords are fetched, I have a created a new CSV which contains the respective counter ID and the caption.
- Over here I have taken 720 images, split into the train(600) and test(120)set. THe test and train set contains only the image names which are the ID column from the caption's CSV. Also, K-means clustering also requires the captions, so iterated over the ID and captions to generate the train, test, and caption text files.

	A	B
1	id	caption
2	0	RICK'S FAVORITE ROAD SIGN IN GERMANY; IT MEANS "NO SPEED LIMIT.&QUOT
3	1	SIGN ON CURRENT N78 ON JUNCTION WITH ROAD LEADING TO NEWBRIDGE, NEAR KILCULLEN JU
4	2	LOOKING IN FROM THE ENTRANCE THIS OFFICE IS JUST TO THE LEFT OF THE SIGN IN ROOM AT S
5	3	A WHITE PINE BONSAI TREE THEY HAVE. THE SIGN SAYS IT'S BEEN IN TRAINING SINCE 1625!
6	4	A ROAD IN CAMP PENDLETON. SEE THE RANGE SIGN AHEAD.
7	5	DUTCH BUS SERVING IN HAVANNA (CUBA) NOW. THE SIGN STILL SAYS "DELFT STATI
8	6	YOU CAN TELL THIS IS A STUDENT RESIDENCE BY THE ROAD SIGN SECRETED IN THE GARDEN.
9	7	SIGN ABOVE GROUND EVEN THOUGH TRAIN IS BELOW GROUND
10	8	A STREET SIGN IN GINZA.
11	9	A GREAT SIGN IN BOUROUGHS MARKET
12	10	THE LARGEST LIVING CHRISTMAS TREE IN THE COUNTRY (OR SO THE SIGN SAYS)
13	11	MY SISTER IN FRONT OF THE STREET SIGN IN OUR FRONT YARD.
14	12	THAT SIGN IS A BIT IN THE DITCH AND MOSTLY UP TO IT'S NECK IN SNOW. OH... THIS IS THE ROAD
15	13	A ROAD SIGN ALONG THE HIGHWAY IN BOLIVIA.
16	14	THIS IS THE ONLY SIGN ABOVE THE LOCAL NEWSPAPER OFFICE. IT'S A BANNER SCREWED TO THE
17	15	THE SIGN ON THE LIGHT POLE IN THIS MOOSE LAKE GAS STATION PARKING LOT READS:NOCONGF
18	16	MY COMPANY HEADQUARTERS. ONCE SAW A WOMAN DOING THE SIGN OF THE CROSS AND SAY
19	17	EVEN THE BUS STOP SIGN IS IN FRENCH. FEELS LIKE HOME :)
20	18	THIS SIGN WAS JUST LYING ON THE GROUND NEAR THE GUARD TOWER.
21	19	OLD PACIFIC CREST TRAIL SYSTEM SIGN FRAMED IN A TREE
22	20	DOUBLE EX... ROAD ARROW AND ROAD SIGN NEAR OCOTILLO , CA.
23	21	A SIGN IN SPANISH IS OMINOUSLY WRITTEN OVER IN ENGLISH USING RED SPRAY PAINT.
24	22	PABST SIGN ABOVE THE BAR
25	23	MY VOTE FOR COOLEST SIGN AND WALL IN YELAPA.
26	24	NO FISHING SIGN WITH REPLICA OF COLUMBUS'S SHIP THE NINA IN THE BACKGROUND
27	25	SUKHUMI ROAD SIGN IN ZUGDIDI
28	26	A SIGN ON THE SIDE OF ROAD SOMEWHERE NEAR GAINESVILLE.
29	27	PICTURE IN AUCKLAND, THE SIGN IN THE BACKGROUND IS CHARTING THE COUNTDOWN TO THE V
30	28	ICED OVER PASSENGER SIGN WINDOW
31	29	WOODEN SIGN FOR THE SOUTHERN CROSS HOTEL FROM THE BALCONY ABOVE DUVAL.
32	30	THIS VINTAGE SIGN IS ON A BUILDING IN THE AUSTRALIAN EXHIBIT AT THE COLUMBUS ZOO AND
33	31	I DECIDED TO GET A FEW NICE SHOTS OF THE SINGLECAR DMU SITTING BESIDE THE STATION SIGI

)

In [2]:
```
import pandas as pd
import nltk
import numpy as np
import cv2
import os
import matplotlib.pyplot as plot
nltk.download('punkt')
nltk.download('wordnet')
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] =15,9
```

In [3]:
```
colA = []
colB = []

with open('dataset/SBU_captioned_photo_dataset_captions.txt') as f:
    for line in f:
        inner_list = [line.strip() for line in line.split('/n')]
        colA.append(inner_list)

with open('dataset/SBU_captioned_photo_dataset_urls.txt') as f:
    for line in f:
        inner_list = [line.strip() for line in line.split('/n')]
        colB.append(inner_list)
```

In [6]:
```
newDF['tokenized_sents'] = newDF.apply(lambda row: nltk.word_tokenize(row['caption']),axis=1)
```

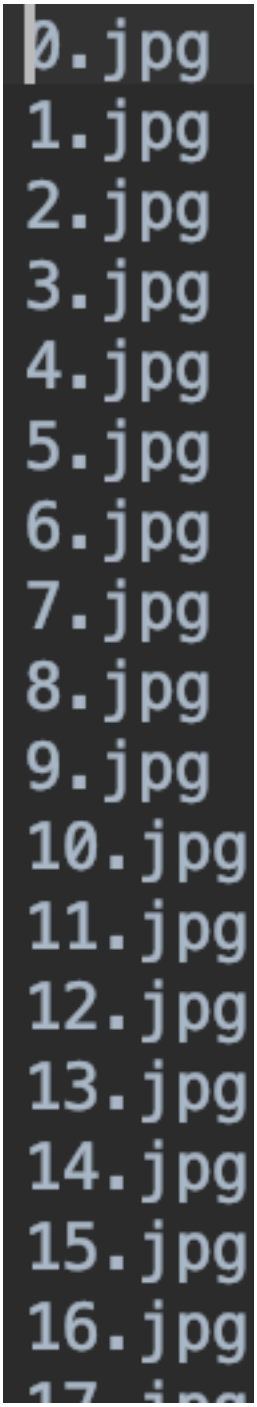
In [7]:
```
lemmatizer = nltk.stem.WordNetLemmatizer()
def lemmatize_text(text):
    return [lemmatizer.lemmatize(w) for w in nltk.tokenize.WhitespaceTokenizer().tokenize(text)]

newDF['text_lemmatized'] = newDF.caption.apply(lemmatize_text)
```

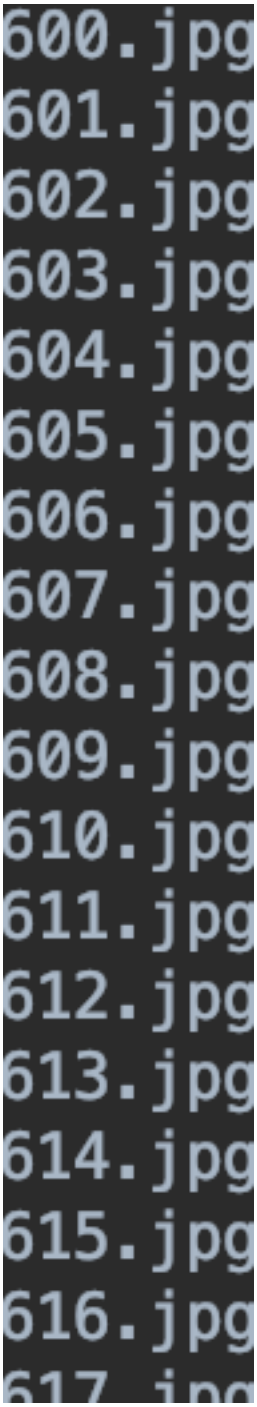
In [94]:
```
1 img_count = 0
2 idCol=[]
3 caption=[]
```

In [95]:
```
1 for index, row in signdf[0:200].iterrows():
2     newpath = 'temp/' + str(img_count) + '.jpg'
3     request = urllib.request.Request(row['link'], None, headers)
4     response = urllib.request.urlopen(request)
5     image = Image.open(response)
6     image.save(newpath)
7     idCol.append(img_count)
8     caption.append(row['caption'])
9     img_count = img_count + 1
```

```
x=pd.read_csv('imageCaption.csv')
z=x.id.apply(lambda x:str(x)+'.jpg')
x.to_csv('tst2.txt', header=None, index=None, sep=' ')
z[0:600].to_csv('train.txt', header=None, index=None)
z[600:].to_csv('tst.txt', header=None, index=None)
```

```
11 "MY SISTER IN FRONT OF THE STREET SIGN IN OUR FRONT YARD."
12 "THAT SIGN IS A BIT IN THE DITCH AND MOSTLY UP TO IT'S NECK IN SNOW. OH... THIS IS THE ROAD TOO."
13 "A ROAD SIGN ALONG THE HIGHWAY IN BOLIVIA."
14 "THIS IS THE ONLY SIGN ABOVE THE LOCAL NEWSPAPER OFFICE. IT'S A BANNER SCREWED TO THE DOOR FRAME!"
15 "THE SIGN ON THE LIGHT POLE IN THIS MOOSE LAKE GAS STATION PARKING LOT READS:NOCONGREGATINGOR CRUISINGVIOLATORS
16 "MY COMPANY HEADQUARTERS.  ONCE SAW A WOMAN DOING THE SIGN OF THE CROSS AND SAYING IGLESIA! IN FRONT."
17 "EVEN THE BUS STOP SIGN IS IN FRENCH. FEELS LIKE HOME :)"
18 "THIS SIGN WAS JUST LYING ON THE GROUND NEAR THE GUARD TOWER."
19 "OLD PACIFIC CREST TRAIL SYSTEM SIGN FRAMED IN A TREE"
20 "DOUBLE EX...  ROAD ARROW AND ROAD SIGN NEAR OCOTILLO , CA."
21 "A SIGN IN SPANISH IS OMINOUSLY WRITTEN OVER IN ENGLISH USING RED SPRAY PAINT."
22 "PABST SIGN ABOVE THE BAR"
23 "MY VOTE FOR COOLEST SIGN AND WALL IN YELAPA."
24 "NO FISHING SIGN WITH REPLICA OF COLUMBUS'S SHIP THE NINA IN THE BACKGROUND"
25 "SUKHUMI ROAD SIGN IN ZUGDIDI"
26 "A SIGN ON THE SIDE OF ROAD SOMEWHERE NEAR GAINESVILLE."
```


0.jpg
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
6.jpg
7.jpg
8.jpg
9.jpg
10.jpg
11.jpg
12.jpg
13.jpg
14.jpg
15.jpg
16.jpg


600.jpg
601.jpg
602.jpg
603.jpg
604.jpg
605.jpg
606.jpg
607.jpg
608.jpg
609.jpg
610.jpg
611.jpg
612.jpg
613.jpg
614.jpg
615.jpg
616.jpg
617.jpg

Model

- It mainly consists of 3 steps which are:-

1. Preprocessing Model.
2. Defining the Model.
3. Fitting the Model.

- Preprocessing Data

1. Firstly, we will load the descritption.txt, train images, and features.pkl so that we can use it in fitting the model.
2. Now we will train the model on the training data set's images and the caption through which we would be deciding the best model to fit.
3. Basically, the generation of the words is based on the previous word that has generated. We will be using the strings startseq and endseq. These tokens between the start and end sequence are added to the loaded descriptions.
4. To_lines and tokenizer are used in converting the dictionary of descriptions into a list of strings.
5. Overall the training process is followed as splitting the description into words. The model is trained on one word and the photo and it generates the next word. Now, the first two words of the description will be provided to the model as input with the image to generate the next word. This is how the model will be trained. Generation of the words stops with the end sequence.
6. The model will now output a prediction as a probability distribution over other words in the dictionary. output data is encoded using one-hot

```
# load doc into memory
def load_doc(filename):
    # open the file as read only
    file = open(filename, 'r')
    # read all text
    text = file.read()
    # close the file
    file.close()
    return text

# load a pre-defined list of photo identifiers
```

```python
# load a pre-defined list of photo identifiers
def load_set(filename):
    doc = load_doc(filename)
    dataset = list()
    # process line by line
    for line in doc.split('\n'):
        # skip empty lines
        if len(line) < 1:
            continue
        # get the image identifier
        identifier = line.split('.')[0]
        dataset.append(identifier)
    return set(dataset)

# load clean descriptions into memory
def load_clean_descriptions(filename, dataset):
    # load document
    doc = load_doc(filename)
    descriptions = dict()
    for line in doc.split('\n'):
        # split line by white space
        tokens = line.split()
        # split id from description
        image_id, image_desc = tokens[0], tokens[1:]
        # skip images not in the set
        if image_id in dataset:
            # create list
            if image_id not in descriptions:
                descriptions[image_id] = list()
            # wrap description in tokens
            desc = 'startseq ' + ' '.join(image_desc) + ' endseq'
            # store
            descriptions[image_id].append(desc)
    return descriptions
```

```python
# create sequences of images, input sequences and output words for an image
def create_sequences(tokenizer, max_length, descriptions, photos):
    X1, X2, y = list(), list(), list()
    # walk through each image identifier
    for key, desc_list in descriptions.items():
        # walk through each description for the image
        for desc in desc_list:
            # encode the sequence
            seq = tokenizer.texts_to_sequences([desc])[0]
            # split one sequence into multiple X,y pairs
            for i in range(1, len(seq)):
                # split into input and output pair
                in_seq, out_seq = seq[:i], seq[i]
                # pad input sequence
                in_seq = pad_sequences([in_seq], maxlen=max_length)[0]
                # encode output sequence
                out_seq = to_categorical([out_seq], num_classes=vocab_size)[0]
                # store
                X1.append(photos[key][0])
                X2.append(in_seq)
                y.append(out_seq)
    return array(X1), array(X2), array(y)
```

- Defining the Model

1. Basically, it requires the feature extractor model and sequence processor, model
2. Feature extractor model requires input photo features of a vector of 4096 elements.
3. Sequence processor model needs input sequences with a predefined length which is fed to embedding layer and next by an LSTM layer with 256 memory units.
4. Above 2 models produce 256 element vector and we have avoided the overfitting by giving 50% dropout layer.
5. Finally, the decoded model combines both using an addition operation and fed to a dense 256 neuron layer that can predict the entire output vocabulary for the next word.

```
# define the captioning model
def define_model(vocab_size, max_length):
    # feature extractor model
    inputs1 = Input(shape=(4096,))
    fe1 = Dropout(0.5)(inputs1)
    fe2 = Dense(256, activation='relu')(fe1)
    # sequence model
    inputs2 = Input(shape=(max_length,))
    se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
    se2 = Dropout(0.5)(se1)
    se3 = LSTM(256)(se2)
    # decoder model
    decoder1 = add([fe2, se3])
    decoder2 = Dense(256, activation='relu')(decoder1)
    outputs = Dense(vocab_size, activation='softmax')(decoder2)
    # tie it together [image, seq] [word]
    model = Model(inputs=[inputs1, inputs2], outputs=outputs)
    model.compile(loss='categorical_crossentropy', optimizer='adam')
    # summarize model
    print(model.summary())
    plot_model(model, to_file='model.png', show_shapes=True)
    return model
```

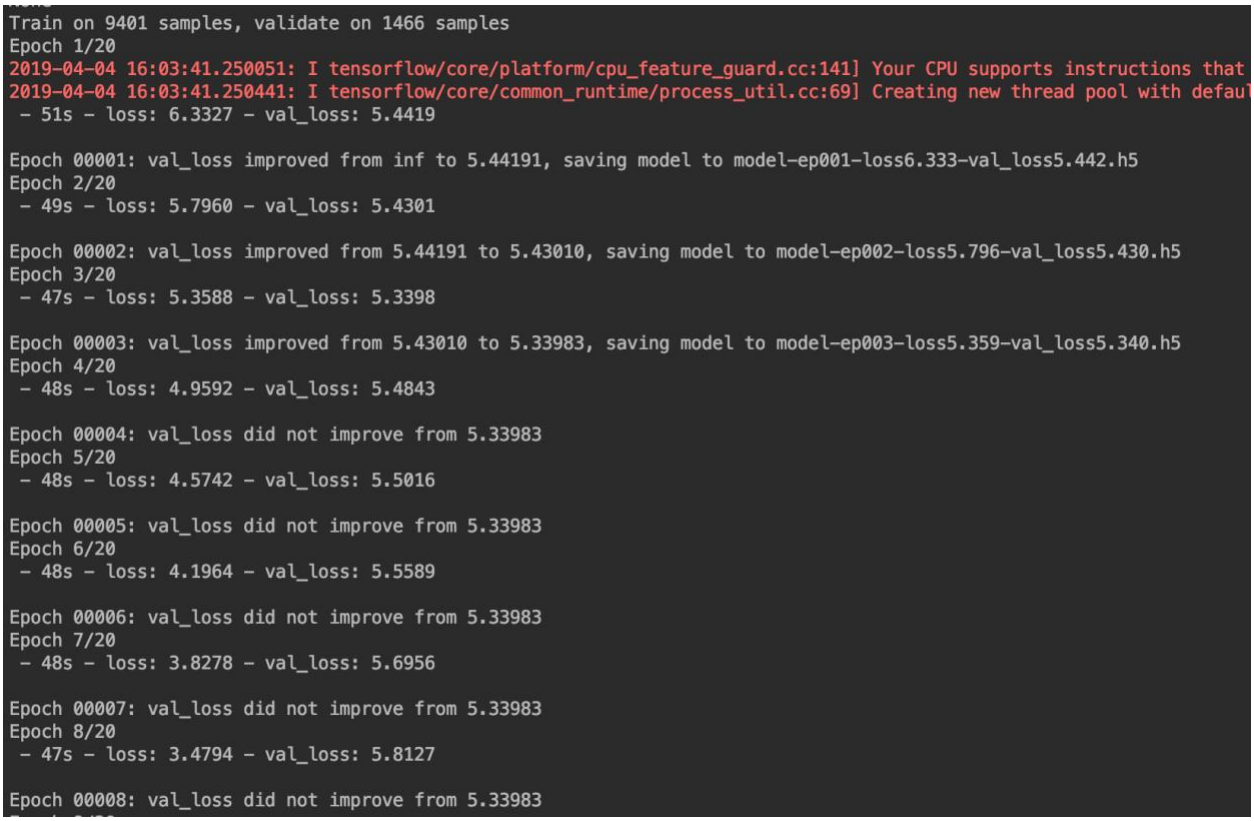
- Fitting the model

1. Now we would save the best-trained model by checking the accuracy of the trained model on the holdout development dataset.
2. Once the best model is selected we will fit the model for 20 epochs.


```
# define the captioning model
def define_model(vocab_size, max_length):
    # feature extractor model
    inputs1 = Input(shape=(4096,))
    fe1 = Dropout(0.5)(inputs1)
    fe2 = Dense(256, activation='relu')(fe1)
    # sequence model
    inputs2 = Input(shape=(max_length,))
    se1 = Embedding(vocab_size, 256, mask_zero=True)(inputs2)
    se2 = Dropout(0.5)(se1)
    se3 = LSTM(256)(se2)
    # decoder model
    decoder1 = add([fe2, se3])
    decoder2 = Dense(256, activation='relu')(decoder1)
    outputs = Dense(vocab_size, activation='softmax')(decoder2)
    # tie it together [image, seq] [word]
    model = Model(inputs=[inputs1, inputs2], outputs=outputs)
    model.compile(loss='categorical_crossentropy', optimizer='adam')
    # summarize model
    print(model.summary())
    plot_model(model, to_file='model.png', show_shapes=True)
    return model
```

```
# descriptions
train_descriptions = load_clean_descriptions('descriptions.txt', train)
print('Descriptions: train=%d' % len(train_descriptions))
# photo features
train_features = load_photo_features('features.pkl', train)
print('Photos: train=%d' % len(train_features))
# prepare tokenizer
tokenizer = create_tokenizer(train_descriptions)
vocab_size = len(tokenizer.word_index) + 1
print('Vocabulary Size: %d' % vocab_size)
# determine the maximum sequence length
max_length = max_length(train_descriptions)
print('Description Length: %d' % max_length)
# prepare sequences
X1train, X2train, ytrain = create_sequences(tokenizer, max_length, train_descriptions, train_features)

# dev dataset

# load test set
filename = 'devImages.txt'
test = load_set(filename)
print('Dataset: %d' % len(test))
# descriptions
test_descriptions = load_clean_descriptions('descriptions.txt', test)
print('Descriptions: test=%d' % len(test_descriptions))
# photo features
test_features = load_photo_features('features.pkl', test)
print('Photos: test=%d' % len(test_features))
# prepare sequences
X1test, X2test, ytest = create_sequences(tokenizer, max_length, test_descriptions, test_features)

# fit model

# define the model
model = define_model(vocab_size, max_length)
# define checkpoint callback
filepath = 'model-ep{epoch:03d}-loss{loss:.3f}-val_loss{val_loss:.3f}.h5'
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
# fit model
model.fit([X1train, X2train], ytrain, epochs=20, verbose=2, callbacks=[checkpoint], validation_data=([X1test, X2test], ytest))
model.save("final.h5")
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         (None, 27)                0         
_________________________________________________________________
input_1 (InputLayer)         (None, 4096)              0         
_________________________________________________________________
embedding_1 (Embedding)      (None, 27, 256)           615424    
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 27, 256)           0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               1048832   
_________________________________________________________________
lstm_1 (LSTM)                (None, 256)               525312    
_________________________________________________________________
add_1 (Add)                  (None, 256)               0         
                                                                 
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792     
_________________________________________________________________
dense_3 (Dense)              (None, 2404)              617828    
=================================================================
Total params: 2,873,188
```



Model Evaluation

- Reported the model's accuracy using the Pandas data frame.
- Created 2 DataFrames, one which contains the SBU caption Data Set and other one is Show and tell models prediction output data frame.
- I have performed tokenization on the caption and the caption prediction columns which are generated by the model.
- Used row level Lambda function applied Corpus Bleu scoring methods bypassing all the tokenized predicted columns and caption column as the parameter to the function and created a new column bleu_score which contains the accuracy for each of the caption generated.
- BLEU is a score for comparing a candidate translation to reference translations.
- Used row level Lambda function applied Rouge scoring methods bypassing all the tokenized predicted columns and caption column as the parameter to the function and created a new column called bleu_precision, blue_recall and blue_f1score which contains the Precision, Recall and F1Score for each of the caption generated.
- Rouge is a Metrics for evaluating automatic summarization of texts.

```
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.translate.bleu_score import sentence_bleu
from PyRouge.pyrouge import Rouge
```

```
df1 = pd.read_csv('imageCaption.csv')
df2= pd.read_csv('ShowTellCapt.csv')
```

```
df2.head()
```

	Unnamed: 0	id	pred_caption
0	0	600	startseq a exit sign in an window door seen on...
1	1	601	startseq a sign on a exit at the exit endseq
2	2	602	startseq a painted sign on the window of the b...
3	3	603	startseq a right under a side of the road on t...
4	4	604	startseq this sign headquarters me written in ...

```
df2['pred_caption']=df2.pred_caption.apply(lambda x:x[9:])
```

```
df2['pred_caption']=df2.pred_caption.apply(lambda x:x[0:x.find(' endseq')])
```

```
df2['pred_caption']=df2['pred_caption'].str.upper()
```

```
df2.head()
```

	Unnamed: 0	id	pred_caption
0	0	600	A EXIT SIGN IN AN WINDOW DOOR SEEN ON THE WATE...
1	1	601	A SIGN ON A EXIT AT THE EXIT
2	2	602	A PAINTED SIGN ON THE WINDOW OF THE BUILDING I...
3	3	603	A RIGHT UNDER A SIDE OF THE ROAD ON THE HUGE S...
4	4	604	THIS SIGN HEADQUARTERS ME WRITTEN IN TIPPEX NO...

```
finalDf = pd.merge(df1, df2, on='id')
```

```
finalDf.head()
```

	id	caption	Unnamed: 0	pred_caption
0	600	I LOVE THE SIGN ABOVE THE DOOR AND HOW THAT WA...	0	A EXIT SIGN IN AN WINDOW DOOR SEEN ON THE WATE...
1	601	THIS IS THE EXIT SIGN ABOVE THE DOOR THAT TAKE...	1	A SIGN ON A EXIT AT THE EXIT
2	602	EXIT SIGN IN THE MEN'S LOUNGE	2	A PAINTED SIGN ON THE WINDOW OF THE BUILDING I...
3	603	EMERGENCY EXIT DOOR HAS A GAP LARGE ENOUGH THA...	3	A RIGHT UNDER A SIDE OF THE ROAD ON THE HUGE S...
4	604	THE GREEN EXIT SIGN IS EMPHASIZED BECAUSE IT I...	4	THIS SIGN HEADQUARTERS ME WRITTEN IN TIPPEX NO...

```
finalDf['tokenized_caption'] = finalDf.apply(lambda row: nltk.word_tokenize(row['caption']),axis=1)
finalDf['tokenized_pred_caption'] = finalDf.apply(lambda row: nltk.word_tokenize(row['pred_caption']),axis=1)
```

```
finalDf['bleu_score'] =finalDf.apply(lambda row : sentence_bleu(row['tokenized_pred_caption'],row['tokenized_capti
```

```
/Users/neerajpadarthi/anaconda3/envs/work/lib/python3.5/site-packages/nltk/translate/bleu_score.py:503: UserWarning:
The hypothesis contains 0 counts of 2-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
/Users/neerajpadarthi/anaconda3/envs/work/lib/python3.5/site-packages/nltk/translate/bleu_score.py:503: UserWarning:
The hypothesis contains 0 counts of 3-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
/Users/neerajpadarthi/anaconda3/envs/work/lib/python3.5/site-packages/nltk/translate/bleu_score.py:503: UserWarning:
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
```

```
finalDf.head()
```

	id	caption	Unnamed: 0	pred_caption	tokenized_caption	tokenized_pred_caption	bleu_score
0	600	I LOVE THE SIGN ABOVE THE DOOR AND HOW THAT WA...	0	A EXIT SIGN IN AN WINDOW DOOR SEEN ON THE WATE...	[I, LOVE, THE, SIGN, ABOVE, THE, DOOR, AND, HO...	[A, EXIT, SIGN, IN, AN, WINDOW, DOOR, SEEN, ON...	0.047619
1	601	THIS IS THE EXIT SIGN ABOVE THE DOOR THAT TAKE...	1	A SIGN ON A EXIT AT THE EXIT	[THIS, IS, THE, EXIT, SIGN, ABOVE, THE, DOOR, ...	[A, SIGN, ON, A, EXIT, AT, THE, EXIT]	0.000000
2	602	EXIT SIGN IN THE MEN'S LOUNGE	2	A PAINTED SIGN ON THE WINDOW OF THE BUILDING I...	[EXIT, SIGN, IN, THE, MEN, 'S, LOUNGE]	[A, PAINTED, SIGN, ON, THE, WINDOW, OF, THE, B...	0.000000
3	603	EMERGENCY EXIT DOOR HAS A GAP LARGE ENOUGH THA...	3	A RIGHT UNDER A SIDE OF THE ROAD ON THE HUGE S...	[EMERGENCY, EXIT, DOOR, HAS, A, GAP, LARGE, EN...	[A, RIGHT, UNDER, A, SIDE, OF, THE, ROAD, ON, ...	0.055556
4	604	THE GREEN EXIT SIGN IS EMPHASIZED BECAUSE IT I...	4	THIS SIGN HEADQUARTERS ME WRITTEN IN TIPPEX NO...	[THE, GREEN, EXIT, SIGN, IS, EMPHASIZED, BECAU...	[THIS, SIGN, HEADQUARTERS, ME, WRITTEN, IN, TI...	0.000000

```
finalDf['rouge_fscore']=finalDf.apply(lambda row : r.rouge_l([row['tokenized_pred_caption']],row['tokenized_caption
finalDf['rouge_precision']=finalDf.apply(lambda row : r.rouge_l([row['tokenized_pred_caption']],row['tokenized_capt
finalDf['rouge_recall']=finalDf.apply(lambda row : r.rouge_l([row['tokenized_pred_caption']],row['tokenized_caption
```

```
finalDf.head()
```

	id	caption	Unnamed: 0	pred_caption	tokenized_caption	tokenized_pred_caption	bleu_score	rouge_fscore	rouge_precision	rouge_recall
0	600	I LOVE THE SIGN ABOVE THE DOOR AND HOW THAT WA...	0	A EXIT SIGN IN AN WINDOW DOOR SEEN ON THE WATE...	[I, LOVE, THE, SIGN, ABOVE, THE, DOOR, AND, HO...	[A, EXIT, SIGN, IN, AN, WINDOW, DOOR, SEEN, ON...	0.047619	0.023257	0.071429	0.013889
1	601	THIS IS THE EXIT SIGN ABOVE THE DOOR THAT TAKE...	1	A SIGN ON A EXIT AT THE EXIT	[THIS, IS, THE, EXIT, SIGN, ABOVE, THE, DOOR, ...	[A, SIGN, ON, A, EXIT, AT, THE, EXIT]	0.000000	0.021054	0.125000	0.011494
2	602	EXIT SIGN IN THE MEN'S LOUNGE	2	A PAINTED SIGN ON THE WINDOW OF THE BUILDING I...	[EXIT, SIGN, IN, THE, MEN, 'S, LOUNGE]	[A, PAINTED, SIGN, ON, THE, WINDOW, OF, THE, B...	0.000000	0.000001	0.000000	0.000000
3	603	EMERGENCY EXIT DOOR HAS A GAP LARGE ENOUGH THA...	3	A RIGHT UNDER A SIDE OF THE ROAD ON THE HUGE S...	[EMERGENCY, EXIT, DOOR, HAS, A, GAP, LARGE, EN...	[A, RIGHT, UNDER, A, SIDE, OF, THE, ROAD, ON, ...	0.055556	0.020409	0.058824	0.012346
4	604	THE GREEN EXIT SIGN IS EMPHASIZED BECAUSE IT I...	4	THIS SIGN HEADQUARTERS ME WRITTEN IN TIPPEX NO...	[THE, GREEN, EXIT, SIGN, IS, EMPHASIZED, BECAU...	[THIS, SIGN, HEADQUARTERS, ME, WRITTEN, IN, TI...	0.000000	0.000001	0.000000	0.000000

```python
def extract_features(filename):
    # load the model
    model = VGG16()
    # re-structure the model
    model.layers.pop()
    model = Model(inputs=model.inputs, outputs=model.layers[-1].output)
    # load the photo
    image = load_img(filename, target_size=(224, 224))
    # convert the image pixels to a numpy array
    image = img_to_array(image)
    # reshape data for the model
    image = image.reshape((1, image.shape[0], image.shape[1], image.shape[
    # prepare the image for the VGG model
    image = preprocess_input(image)
    # get features
    feature = model.predict(image, verbose=0)
    return feature

# map an integer to a word
def word_for_id(integer, tokenizer):
    for word, index in tokenizer.word_index.items():
        if index == integer:
            return word
    return None

# generate a description for an image
def generate_desc(model, tokenizer, photo, max_length):
    # seed the generation process
    in_text = 'startseq'
    # iterate over the whole length of the sequence
    for i in range(max_length):
        # integer encode input sequence
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        # pad input
        sequence = pad_sequences([sequence], maxlen=max_length)
        # predict next word
        yhat = model.predict([photo,sequence], verbose=0)
        # convert probability to integer
        yhat = argmax(yhat)
        # map integer to word
        word = word_for_id(yhat, tokenizer)
        # stop if we cannot map the word
        if word is None:
            break
        # append as input for generating the next word
        in_text += ' ' + word
        # stop if we predict the end of the sequence
        if word == 'endseq':
            break
    return in_text
```

```
/Users/neerajpadarthi/anaconda3/bin/python3.6 "/Users/neerajpadarthi/Neer
Using TensorFlow backend.
2019-04-04 20:21:58.539547: I tensorflow/core/platform/cpu_feature_guard.
2019-04-04 20:21:58.540721: I tensorflow/core/common_runtime/process_util
startseq of door door road with in tree airport endseq

Process finished with exit code 0
```

Checking the Accuracy before the segmentation

```
720
startseq this sign on the vintage of road is the road of south south own kilcullen missouri endseq
BLEU-1: 0.123418
/Users/neerajpadarthi/anaconda3/lib/python3.6/site-packages/nltk/translate/bleu_score.py:523: UserWarning:
The hypothesis contains 0 counts of 2-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
/Users/neerajpadarthi/anaconda3/lib/python3.6/site-packages/nltk/translate/bleu_score.py:523: UserWarning:
The hypothesis contains 0 counts of 3-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
/Users/neerajpadarthi/anaconda3/lib/python3.6/site-packages/nltk/translate/bleu_score.py:523: UserWarning:
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
BLEU-2: 0.000000
BLEU-3: 0.000000
BLEU-4: 0.000000
```

	id	caption	pred_caption	tokenized_caption	tokenized_pred_caption	bleu_score	rouge_fscore	rouge_precision	rouge_recall
0	711	YELLOW BOARD THAT SAYS THERE IS WAVY ROAD AHEAD	AMP AMP I I I I I I I I I I I I I I I I I I ...	[YELLOW, BOARD, THAT, SAYS, THERE, IS, WAVY, R...	[AMP, AMP, I, I, I, I, I, I, I, I, I, I, I, I,...	0	0.030770	0.038462	0.025641
1	712	TRAFFIC STANDING WITH RED SIGNAL THAT SAYS TO ...	INTERSECTION ROUTE BY THE ROAD SOMEWHERE 22 RACE	[TRAFFIC, STANDING, WITH, RED, SIGNAL, THAT, S...	[INTERSECTION, ROUTE, BY, THE, ROAD, SOMEWHERE...	0	0.000001	0.000000	0.000000
2	713	BLUE BOARD TELLING TO TAKE DIFFERENT DIRECTION...	NO ROAD SIGN IN THE ROAD 66	[BLUE, BOARD, TELLING, TO, TAKE, DIFFERENT, DI...	[NO, ROAD, SIGN, IN, THE, ROAD, 66]	0	0.000001	0.000000	0.000000
3	714	GREEN BOARD THAT IS TELLING HOW FAR IS PARRISH...	THE SIGN 66 SIGN PARK IN GARY 66 PARK	[GREEN, BOARD, THAT, IS, TELLING, HOW, FAR, IS...	[THE, SIGN, 66, SIGN, PARK, IN, GARY, 66, PARK]	0	0.000001	0.000000	0.000000
4	715	WIDE ROAD WITH BLUE SKY WITH RED STOP SIGNAL T...	THE STREET SIGN IN THE ROAD	[WIDE, ROAD, WITH, BLUE, SKY, WITH, RED, STOP,...	[THE, STREET, SIGN, IN, THE, ROAD]	0	0.000001	0.000000	0.000000
5	716	GREEN BOARD WITH WIDE BLUE SKY WITH LEFT RIGHT...	THE ROAD SIGN FOR SIGN 66	[GREEN, BOARD, WITH, WIDE, BLUE, SKY, WITH, LE...	[THE, ROAD, SIGN, FOR, SIGN, 66]	0	0.000001	0.000000	0.000000
6	717	GREEN BOARD ON THE BRIDGE TELLING WJERE TO GO ...	FUNNY ROAD SIGN WITH THE GREAT EXIT IN ENGLAND	[GREEN, BOARD, ON, THE, BRIDGE, TELLING, WJERE...	[FUNNY, ROAD, SIGN, WITH, THE, GREAT, EXIT, IN...	0	0.000001	0.000000	0.000000
7	718	GREEN TRAFFIC BOARD WITH YELLOW EXIT SIGN WHIC...	FUNNY SIGN BY THE ROAD SIGN PARK	[GREEN, TRAFFIC, BOARD, WITH, YELLOW, EXIT, SI...	[FUNNY, SIGN, BY, THE, ROAD, SIGN, PARK]	0	0.000001	0.000000	0.000000
8	719	BLUE LANE DIRECTION BOARD WHICH TELLS TO KEEP ...	NO NO ROAD SIGN WITH REPLICA OF EVERY EVERY AR...	[BLUE, LANE, DIRECTION, BOARD, WHICH, TELLS, T...	[NO, NO, ROAD, SIGN, WITH, REPLICA, OF, EVERY,...	0	0.000001	0.000000	0.000000
9	720	BLUE BOARD THAT SAYS TO TAKE DIFFERENT ROUES T...	SUKHUMI SIGNS ON THE SIGN FOR THE SIGNS STREET...	[BLUE, BOARD, THAT, SAYS, TO, TAKE, DIFFERENT,...	[SUKHUMI, SIGNS, ON, THE, SIGN, FOR, THE, SIGN...	0	0.000001	0.000000	0.000000

Checking the Accuracy after the segmentation

	id	caption	pred_caption	tokenized_caption	tokenized_pred_caption	bleu_score	rouge_fscore	rouge_precision	rouge_recall
0	711	YELLOW BOARD THAT SAYS THERE IS WAVY ROAD AHEAD	THIS SIGN BY A BAR OF OHIO IN A STREET OF IS	[YELLOW, BOARD, THAT, SAYS, THERE, IS, WAVY, R...	[THIS, SIGN, BY, A, BAR, OF, OHIO, IN, A, STRE...	0	0.078432	0.166667	0.051282
1	712	TRAFFIC STANDING WITH RED SIGNAL THAT SAYS TO ...	IN THE HORN ON THE STREET SIGN IN THE STREET	[TRAFFIC, STANDING, WITH, RED, SIGNAL, THAT, S...	[IN, THE, HORN, ON, THE, STREET, SIGN, IN, THE...	0	0.000001	0.000000	0.000000
2	713	BLUE BOARD TELLING TO TAKE DIFFERENT DIRECTION...	THIS SIGN BY THE WINDOW OF A PEOPLE OF THE STR...	[BLUE, BOARD, TELLING, TO, TAKE, DIFFERENT, DI...	[THIS, SIGN, BY, THE, WINDOW, OF, A, PEOPLE, O...	0	0.026668	0.071429	0.016393
3	714	GREEN BOARD THAT IS TELLING HOW FAR IS PARRISH...	THIS SIGN ON A STREET OF A SIGN OF IT	[GREEN, BOARD, THAT, IS, TELLING, HOW, FAR, IS...	[THIS, SIGN, ON, A, STREET, OF, A, SIGN, OF, IT]	0	0.027779	0.100000	0.016129
4	715	WIDE ROAD WITH BLUE SKY WITH RED STOP SIGNAL T...	A SIGN ON THE STREET OF THE SIGN OF	[WIDE, ROAD, WITH, BLUE, SKY, WITH, RED, STOP,...	[A, SIGN, ON, THE, STREET, OF, THE, SIGN, OF]	0	0.033899	0.111111	0.020000
5	716	GREEN BOARD WITH WIDE BLUE SKY WITH LEFT RIGHT...	THIS SIGN ON SPANISH ON ROAD IS IS ROAD IS THE...	[GREEN, BOARD, WITH, WIDE, BLUE, SKY, WITH, LE...	[THIS, SIGN, ON, SPANISH, ON, ROAD, IS, IS, RO...	0	0.000001	0.000000	0.000000
6	717	GREEN BOARD ON THE BRIDGE TELLING WJERE TO GO ...	THIS SIGN ON A VINTAGE OF THIS IS TO THE COLUM...	[GREEN, BOARD, ON, THE, BRIDGE, TELLING, WJERE...	[THIS, SIGN, ON, A, VINTAGE, OF, THIS, IS, TO,...	0	0.026317	0.052632	0.017544
7	718	GREEN TRAFFIC BOARD WITH YELLOW EXIT SIGN WHIC...	THIS SIGN ON A VINTAGE OF ROAD OVER THE COLUMB...	[GREEN, TRAFFIC, BOARD, WITH, YELLOW, EXIT, SI...	[THIS, SIGN, ON, A, VINTAGE, OF, ROAD, OVER, T...	0	0.022223	0.071429	0.013158
8	719	BLUE LANE DIRECTION BOARD WHICH TELLS TO KEEP ...	QUITE DRIVEN IN RESCUE BLUE OF THE SIGN OF A S...	[BLUE, LANE, DIRECTION, BOARD, WHICH, TELLS, T...	[QUITE, DRIVEN, IN, RESCUE, BLUE, OF, THE, SIG...	0	0.021979	0.062500	0.013333
9	720	BLUE BOARD THAT SAYS TO TAKE DIFFERENT ROUES T...	THIS SIGN ON THE VINTAGE OF ROAD IS THE ROAD O...	[BLUE, BOARD, THAT, SAYS, TO, TAKE, DIFFERENT,...	[THIS, SIGN, ON, THE, VINTAGE, OF, ROAD, IS, T...	0	0.000001	0.000000	0.000000

Conclusion

- We can see that using the clustering segmentation the accuracy has improved.