Double-click (or enter) to edit

```
[2] #Mounting google drive to get access to data uploaded in drive having dataset
    from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[3] # !unzip "/content/drive/MyDrive/NLP_Da/20_newsgroup.zip" -d "/content/drive/MyDrive/NLP_Da/Dataset"
```

```
[4] import os
    import sys
    import numpy as np
    import keras
    from keras.preprocessing.text import Tokenizer
    from keras.utils.np_utils import to_categorical
    from keras.preprocessing.sequence import pad_sequences
    from keras.layers import Activation, Conv2D, Input, Embedding, Reshape, MaxPool2D, Concatenate, Flatten, Dropout, Dense, Conv1D
    from keras.layers import MaxPool1D
    from keras.models import Model
    from keras.callbacks import ModelCheckpoint
    from keras.optimizers import Adam
```

```
[5] # just to make sure the dataset is added properly
    !ls '/content/drive/MyDrive/NLP_Da/Dataset/20_newsgroup'
```

```
alt.atheism               rec.autos           sci.space
comp.graphics             rec.motorcycles     soc.religion.christian
comp.os.ms-windows.misc   rec.sport.baseball  talk.politics.guns
comp.sys.ibm.pc.hardware  rec.sport.hockey    talk.politics.mideast
comp.sys.mac.hardware     sci.crypt           talk.politics.misc
comp.windows.x            sci.electronics     talk.religion.misc
misc.forsale              sci.med
```

```
# the dataset path
TEXT_DATA_DIR = r'/content/drive/MyDrive/NLP_Da/Dataset/20_newsgroup'
#the path for Glove embeddings
GLOVE_DIR = r'/content/drive/MyDrive/NLP_Da/Dataset/Glove'
# make the max word length to be constant
MAX_WORDS = 10000
MAX_SEQUENCE_LENGTH = 1000
# the percentage of train test split to be applied
VALIDATION_SPLIT = 0.20
# the dimension of vectors to be used
EMBEDDING_DIM = 100
# filter sizes of the different conv layers
filter_sizes = [3,4,5]
num_filters = 512
embedding_dim = 100
# dropout probability
drop = 0.5
batch_size = 30
epochs = 2
```

DATASET STRUCTURE

The dataset is present in a hierarchical structure, i.e. all files of a given class are located in their respective folders and each datapoint has its own '.txt' file.

- First we go through the entire dataset to build our text list and label list.
- Followed by this we tokenize the entire data using Tokenizer, which is a part of keras.preprocessing.text.
- We then add padding to the sequences to make them of a uniform length.

```
[7] ## preparing dataset
    texts = []  # list of text samples
    labels_index = {}  # dictionary mapping label name to numeric id
    labels = []  # list of label ids
    for name in sorted(os.listdir(TEXT_DATA_DIR)):
        path = os.path.join(TEXT_DATA_DIR, name)
        if os.path.isdir(path):
            label_id = len(labels_index)
            labels_index[name] = label_id
            for fname in sorted(os.listdir(path)):
                if fname.isdigit():
                    fpath = os.path.join(path, fname)
                    if sys.version_info < (3,):
                        f = open(fpath)
                    else:
                        f = open(fpath, encoding='latin-1')
                    t = f.read()
                    i = t.find('\n\n')  # skip header
                    if 0 < i:
                        t = t[i:]
                    texts.append(t)
                    f.close()
                    labels.append(label_id)
    print(labels_index)

    print('Found %s texts.' % len(texts))
```

```
{'alt.atheism': 0, 'comp.graphics': 1, 'comp.os.ms-windows.misc': 2, 'comp.sys.ibm.pc.hardware': 3, 'comp.sys.mac.hardware': 4, 'comp.windows.x': 5, 'misc.forsale': 6, 'rec.autos': 7, '
Found 19997 texts.
```

[8]
```
print(texts[1000])
```

```
            Carderock Division, Naval Surface Warfare Center
              (formerly the David Taylor Research Center)
                         Bethesda, Maryland

SPONSOR: NESS (Navy Engineering Software System) is sponsoring a
one-day Navy Scientific Visualization and Virtual Reality Seminar.
The purpose of the seminar is to present and exchange information for
Navy-related scientific visualization and virtual reality programs,
research, developments, and applications.

PRESENTATIONS: Presentations are solicited on all aspects of
Navy-related scientific visualization and virtual reality.  All
current work, works-in-progress, and proposed work by Navy
organizations will be considered.  Four types of presentations are
available.

     1. Regular presentation: 20-30 minutes in length
     2. Short presentation: 10 minutes in length
     3. Video presentation: a stand-alone videotape (author need not
        attend the seminar)
     4. Scientific visualization or virtual reality demonstration (BYOH)

Accepted presentations will not be published in any proceedings,
however, viewgraphs and other materials will be reproduced for
seminar attendees.

ABSTRACTS: Authors should submit a one page abstract and/or videotape to:

     Robert Lipman
     Naval Surface Warfare Center, Carderock Division
     Code 2042
     Bethesda, Maryland  20084-5000

     VOICE (301) 227-3618;  FAX (301) 227-5753
     E-MAIL  lipman@oasys.dt.navy.mil

Authors should include the type of presentation, their affiliations,
addresses, telephone and FAX numbers, and addresses.  Multi-author
papers should designate one point of contact.

DEADLINES: The abstact submission deadline is April 30, 1993.
Notification of acceptance will be sent by May 14, 1993.
Materials for reproduction must be received by June 1, 1993.

For further information, contact Robert Lipman at the above address.

          PLEASE DISTRIBUTE AS WIDELY AS POSSIBLE, THANKS.



Robert Lipman                     | Internet: lipman@oasys.dt.navy.mil
David Taylor Model Basin - CDNSWC |       or: lip@ocean.dt.navy.mil
Computational Signatures and      | Voicenet: (301) 227-3618
   Structures Group, Code 2042    | Factsnet: (301) 227-5753
Bethesda, Maryland  20084-5000    | Phishnet: stockings@long.legs

The sixth sick shiek's sixth sheep's sick.
```

[9]
```
tokenizer  = Tokenizer(num_words = MAX_WORDS)
tokenizer.fit_on_texts(texts)
sequences =  tokenizer.texts_to_sequences(texts)

word_index = tokenizer.word_index
print("unique words : {}".format(len(word_index)))

data = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)

labels = to_categorical(np.asarray(labels))
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)
print(labels)
```

```
unique words : 174074
Shape of data tensor: (19997, 1000)
Shape of label tensor: (19997, 20)
[[1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 1.]]
```

[10]
```
x_test=data[1000]
y_test=labels[1000]
print(x_test.shape, y_test.shape)
```

```
(1000,) (20,)
```

[11]
```
labels[1000]
```

```
array([0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0.], dtype=float32)
```

[12]
```
# split the data into a training set and a validation set
indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
nb_validation_samples = int(VALIDATION_SPLIT * data.shape[0])

x_train = data[:-nb_validation_samples]
y_train = labels[:-nb_validation_samples]
x_val = data[-nb_validation_samples:]
y_val = labels[-nb_validation_samples:]
```

Since we have our train-validation split ready, our next step is to create an embedding matrix from the precomputed Glove embeddings. For convenience we are freezing the embedding layer i.e we will not be fine tuning the word embeddings. Feel free to test it out for better accuracy on very specific examples. From what can be seen, the Glove embeddings are universal features and tend to perform great in general.

[13]
```
embeddings_index = {}
f = open(os.path.join(GLOVE_DIR, 'glove.6B.100d.txt'))
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()

print('Found %s word vectors.' % len(embeddings_index))
```

```
Found 400000 word vectors.
```

[14]
```
embedding_matrix = np.zeros((len(word_index) + 1, EMBEDDING_DIM))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
```

[15]
```
from keras.layers import Embedding

embedding_layer = Embedding(len(word_index) + 1,
                            EMBEDDING_DIM,
                            weights=[embedding_matrix],
                            input_length=MAX_SEQUENCE_LENGTH,
                            trainable=False)
```

```
inputs = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedding = embedding_layer(inputs)

print(embedding.shape)
reshape = Reshape((MAX_SEQUENCE_LENGTH,EMBEDDING_DIM,1))(embedding)
print(reshape.shape)

conv_0 = Conv2D(num_filters, kernel_size=(filter_sizes[0], embedding_dim), padding='valid', kernel_initializer='normal', activation='relu')(reshape)
conv_1 = Conv2D(num_filters, kernel_size=(filter_sizes[1], embedding_dim), padding='valid', kernel_initializer='normal', activation='relu')(reshape)
conv_2 = Conv2D(num_filters, kernel_size=(filter_sizes[2], embedding_dim), padding='valid', kernel_initializer='normal', activation='relu')(reshape)

maxpool_0 = MaxPool2D(pool_size=(MAX_SEQUENCE_LENGTH - filter_sizes[0] + 1, 1), strides=(1,1), padding='valid')(conv_0)
maxpool_1 = MaxPool2D(pool_size=(MAX_SEQUENCE_LENGTH - filter_sizes[1] + 1, 1), strides=(1,1), padding='valid')(conv_1)
maxpool_2 = MaxPool2D(pool_size=(MAX_SEQUENCE_LENGTH - filter_sizes[2] + 1, 1), strides=(1,1), padding='valid')(conv_2)

concatenated_tensor = Concatenate(axis=1)([maxpool_0, maxpool_1, maxpool_2])
flatten = Flatten()(concatenated_tensor)
dropout = Dropout(drop)(flatten)
output = Dense(units=20, activation='softmax')(dropout)

# this creates a model that includes
model = Model(inputs=inputs, outputs=output)

checkpoint = ModelCheckpoint('weights_cnn_sentece.hdf5', monitor='val_acc', verbose=1, save_best_only=True, mode='auto')
adam = Adam(lr=1e-4, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)

model.compile(optimizer=adam, loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
(None, 1000, 100)
(None, 1000, 100, 1)
Model: "model"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 1000)]	0	
embedding (Embedding)	(None, 1000, 100)	17407500	input_1[0][0]
reshape (Reshape)	(None, 1000, 100, 1)	0	embedding[0][0]
conv2d (Conv2D)	(None, 998, 1, 512)	154112	reshape[0][0]
conv2d_1 (Conv2D)	(None, 997, 1, 512)	205312	reshape[0][0]
conv2d_2 (Conv2D)	(None, 996, 1, 512)	256512	reshape[0][0]
max_pooling2d (MaxPooling2D)	(None, 1, 1, 512)	0	conv2d[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 1, 1, 512)	0	conv2d_1[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 512)	0	conv2d_2[0][0]
concatenate (Concatenate)	(None, 3, 1, 512)	0	max_pooling2d[0][0] max_pooling2d_1[0][0] max_pooling2d_2[0][0]
flatten (Flatten)	(None, 1536)	0	concatenate[0][0]
dropout (Dropout)	(None, 1536)	0	flatten[0][0]
dense (Dense)	(None, 20)	30740	dropout[0][0]

```
Total params: 18,054,176
Trainable params: 646,676
Non-trainable params: 17,407,500
_________________________________________________________________
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/optimizer_v2.py:375: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  "The `lr` argument is deprecated, use `learning_rate` instead.")
```

```
print("Traning Model...")
model.fit(x_train, y_train, batch_size=batch_size, epochs=50, verbose=1, callbacks=[checkpoint], validation_data=(x_val, y_val))
```

```
534/534 [==============================] - 324s 607ms/step - loss: 0.2283 - accuracy: 0.9419 - val_loss: 0.6186 - val_accuracy: 0.8075
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 32/50
534/534 [==============================] - 324s 607ms/step - loss: 0.2105 - accuracy: 0.9456 - val_loss: 0.6166 - val_accuracy: 0.8092
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 33/50
534/534 [==============================] - 324s 607ms/step - loss: 0.2023 - accuracy: 0.9496 - val_loss: 0.6157 - val_accuracy: 0.8105
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 34/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1956 - accuracy: 0.9472 - val_loss: 0.6179 - val_accuracy: 0.8105
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 35/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1849 - accuracy: 0.9536 - val_loss: 0.6253 - val_accuracy: 0.8097
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 36/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1856 - accuracy: 0.9501 - val_loss: 0.6219 - val_accuracy: 0.8052
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 37/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1726 - accuracy: 0.9542 - val_loss: 0.6178 - val_accuracy: 0.8090
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 38/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1614 - accuracy: 0.9612 - val_loss: 0.6222 - val_accuracy: 0.8072
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 39/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1604 - accuracy: 0.9591 - val_loss: 0.6234 - val_accuracy: 0.8070
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 40/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1579 - accuracy: 0.9585 - val_loss: 0.6233 - val_accuracy: 0.8070
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 41/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1593 - accuracy: 0.9596 - val_loss: 0.6227 - val_accuracy: 0.8070
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 42/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1512 - accuracy: 0.9600 - val_loss: 0.6293 - val_accuracy: 0.8125
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 43/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1542 - accuracy: 0.9558 - val_loss: 0.6202 - val_accuracy: 0.8115
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 44/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1506 - accuracy: 0.9615 - val_loss: 0.6289 - val_accuracy: 0.8102
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 45/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1421 - accuracy: 0.9629 - val_loss: 0.6291 - val_accuracy: 0.8117
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 46/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1351 - accuracy: 0.9628 - val_loss: 0.6285 - val_accuracy: 0.8085
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 47/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1361 - accuracy: 0.9630 - val_loss: 0.6331 - val_accuracy: 0.8117
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 48/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1317 - accuracy: 0.9614 - val_loss: 0.6330 - val_accuracy: 0.8130
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 49/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1293 - accuracy: 0.9626 - val_loss: 0.6320 - val_accuracy: 0.8117
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 50/50
534/534 [==============================] - 324s 607ms/step - loss: 0.1250 - accuracy: 0.9651 - val_loss: 0.6358 - val_accuracy: 0.8082
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
<keras.callbacks.History at 0x7f54e04ffb90>
```

```
score, acc = model.evaluate(x_train, y_train)
print("Loss: ", score)
print("Accuracy: ", acc*100)
```

```
500/500 [==============================] - 242s 484ms/step - loss: 0.0728 - accuracy: 0.9759
Loss:  0.07276985049247742
Accuracy:  97.59345054626465
```

```
[19] x_test=x_test.reshape(1, 1000)
pred=model.predict(x_test).argmax()
```

```
[20] print("Actual label: ", y_test.argmax())
print("Predicted label: ", pred)
```

```
Actual label:  1
Predicted label:  1
```

```
[21] print(labels[1000].argmax())
```

```
10
```

```
labels_index
```

```
{'alt.atheism': 0,
 'comp.graphics': 1,
 'comp.os.ms-windows.misc': 2,
 'comp.sys.ibm.pc.hardware': 3,
 'comp.sys.mac.hardware': 4,
 'comp.windows.x': 5,
 'misc.forsale': 6,
 'rec.autos': 7,
 'rec.motorcycles': 8,
 'rec.sport.baseball': 9,
 'rec.sport.hockey': 10,
 'sci.crypt': 11,
 'sci.electronics': 12,
 'sci.med': 13,
 'sci.space': 14,
 'soc.religion.christian': 15,
 'talk.politics.guns': 16,
 'talk.politics.mideast': 17,
 'talk.politics.misc': 18,
 'talk.religion.misc': 19}
```

```
[23] print(texts[1000])
```

```
            Carderock Division, Naval Surface Warfare Center
              (formerly the David Taylor Research Center)
                         Bethesda, Maryland

SPONSOR: NESS (Navy Engineering Software System) is sponsoring a
one-day Navy Scientific Visualization and Virtual Reality Seminar.
```

```
The purpose of the seminar is to present and exchange information for
Navy-related scientific visualization and virtual reality programs,
research, developments, and applications.

PRESENTATIONS: Presentations are solicited on all aspects of
Navy-related scientific visualization and virtual reality.  All
current work, works-in-progress, and proposed work by Navy
organizations will be considered.  Four types of presentations are
available.

     1. Regular presentation: 20-30 minutes in length
     2. Short presentation: 10 minutes in length
     3. Video presentation: a stand-alone videotape (author need not
        attend the seminar)
     4. Scientific visualization or virtual reality demonstration (BYOH)

Accepted presentations will not be published in any proceedings,
however, viewgraphs and other materials will be reproduced for
seminar attendees.

ABSTRACTS: Authors should submit a one page abstract and/or videotape to:

     Robert Lipman
     Naval Surface Warfare Center, Carderock Division
     Code 2042
     Bethesda, Maryland  20084-5000

     VOICE (301) 227-3618;  FAX (301) 227-5753
     E-MAIL  lipman@oasys.dt.navy.mil

Authors should include the type of presentation, their affiliations,
addresses, telephone and FAX numbers, and addresses.  Multi-author
papers should designate one point of contact.

DEADLINES: The abstact submission deadline is April 30, 1993.
Notification of acceptance will be sent by May 14, 1993.
Materials for reproduction must be received by June 1, 1993.

For further information, contact Robert Lipman at the above address.

          PLEASE DISTRIBUTE AS WIDELY AS POSSIBLE, THANKS.



Robert Lipman                     | Internet: lipman@oasys.dt.navy.mil
David Taylor Model Basin - CDNSWC |       or: lip@ocean.dt.navy.mil
Computational Signatures and      | Voicenet: (301) 227-3618
   Structures Group, Code 2042    | Factsnet: (301) 227-5753
Bethesda, Maryland  20084-5000    | Phishnet: stockings@long.legs

The sixth sick shiek's sixth sheep's sick.
```

Double-click (or enter) to edit

```
a=0
while(True):
    a=a+1
    a=a-1
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-24-2de2149b9a12> in <module>()
      2 while(True):
      3     a=a+1
----> 4     a=a-1

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW