```
# CNN Model


# import necessary libraries
import logging

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging

from keras.models import Sequential
from keras.layers import Dense, Embedding, Flatten, Dropout
from keras.layers.pooling import MaxPooling1D
from keras.layers.convolutional import Conv1D
from keras.regularizers import l2
from keras.callbacks import EarlyStopping
import gensim
from keras.preprocessing.sequence import pad_sequences
import numpy as np
import keras
import json, multiprocessing
import pandas as pd
import numpy as np
import gensim
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x
via the `%tensorflow_version 1.x` magic: more info.

```
2019-11-10 03:36:06,223 : DEBUG : {'transport_params': None, 'ignore_ext': False,
2019-11-10 03:36:06,571 : INFO : 'pattern' package not found; tag filters are not
```

```
# model hyper parameters
hyperparameters = {"EMBEDDING_DIM":100,
"SEQUENCE_LENGTH_PERCENTILE" : 90,
"n_layers" : 2,
"hidden_units" : 500,
"batch_size" :100,
"pretrained_embedding" : False,
# if we have pre-trained embeddings, specify if they are static or non-static embeddi
"TRAINABLE_EMBEDDINGS" : True,
"patience" : 2,
"dropout_rate": 0.3,
"n_filters" :100,
"window_size" : 8,
"dense_activation" : "relu",
"l2_penalty" : 0.0003,
"epochs" : 10,
"VALIDATION_SPLIT" : 0.1}


def token_to_index(token, dictionary):
    """
```

```
    Given a token and a gensim dictionary, return the token index
    if in the dictionary, None otherwise.
    Reserve index 0 for padding.
    """
    if token not in dictionary.token2id:
        return None
    return dictionary.token2id[token] + 1



def texts_to_indices(text, dictionary):
    """
    Given a list of tokens (text) and a gensim dictionary, return a list
    of token ids.
    """
    result = list(map(lambda x: token_to_index(x, dictionary), text))
    return list(filter(None, result))



# define function to train model that takes in training dataset, training labels, hyp
def train(train_texts, train_labels, dictionary, hyperparameters, model_file=None,EMB
    """
    Train a word-level CNN text classifier.
    :param train_texts: tokenized and normalized texts, a list of token lists, [['sen
    :param train_labels: the label for each train text
    :param dictionary: A gensim dictionary object for the training text tokens
    :param model_file: An optional output location for the ML model file
    :param EMBEDDINGS_MODEL_FILE: An optinal location for pre-trained word embeddings
    :return: the produced keras model, the validation accuracy, and the size of the t
    """
    assert len(train_texts)==len(train_labels)
    # compute the max sequence length
    lengths=list(map(lambda x: len(x), train_texts))
    a = np.array(lengths)
    MAX_SEQUENCE_LENGTH = int(np.percentile(a, hyperparameters["SEQUENCE_LENGTH_PERCE
    # convert all texts to dictionary indices
    train_texts_indices = list(map(lambda x: texts_to_indices(x, dictionary), train_t
    # pad or truncate the texts
    x_data = pad_sequences(train_texts_indices, maxlen=int(MAX_SEQUENCE_LENGTH))
    # convert the train labels to one-hot encoded vectors
    train_labels = keras.utils.to_categorical(train_labels)
    y_data = train_labels

    model = Sequential()

    # create embeddings matrix from word2vec pre-trained embeddings, if provided
    if hyperparameters["pretrained_embedding"]:
        embeddings_index = gensim.models.KeyedVectors.load_word2vec_format(EMBEDDINGS
        embedding_matrix = np.zeros((len(dictionary) + 1, hyperparameters["EMBEDDING_
        for word, i in dictionary.token2id.items():
            embedding_vector = embeddings_index[word] if word in embeddings_index els
```

```
            if embedding_vector is not None:
                # words not found in embedding index will be all-zeros.
                embedding_matrix[i] = embedding_vector
        model.add(Embedding(len(dictionary) + 1,
                            hyperparameters["EMBEDDING_DIM"],
                            weights=[embedding_matrix],
                            input_length=MAX_SEQUENCE_LENGTH,
                            trainable=hyperparameters["TRAINABLE_EMBEDDINGS"]))
    else:
        model.add(Embedding(len(dictionary) + 1,
                            hyperparameters["EMBEDDING_DIM"],
                            input_length=MAX_SEQUENCE_LENGTH))
    # add drop out for the input layer, why do you think this might help?
    model.add(Dropout(hyperparameters["dropout_rate"]))
    # add a 1 dimensional conv layer
    # a rectified linear activation unit, returns input if input > 0 else 0
    model.add(Conv1D(filters=hyperparameters["n_filters"],
                     kernel_size=hyperparameters["window_size"],
                     activation='relu'))
    # add a max pooling layer
    model.add(MaxPooling1D(MAX_SEQUENCE_LENGTH - hyperparameters["window_size"] + 1))
    model.add(Flatten())

    # add 0 or more fully connected layers with drop out
    for _ in range(hyperparameters["n_layers"]):
        model.add(Dropout(hyperparameters["dropout_rate"]))
        model.add(Dense(hyperparameters["hidden_units"],
                        activation=hyperparameters["dense_activation"],
                        kernel_regularizer=l2(hyperparameters["l2_penalty"]),
                        bias_regularizer=l2(hyperparameters["l2_penalty"]),
                        kernel_initializer='glorot_uniform',
                        bias_initializer='zeros'))

    # add the last fully connected layer with softmax activation
    model.add(Dropout(hyperparameters["dropout_rate"]))
    model.add(Dense(len(train_labels[0]),
                    activation='softmax',
                    kernel_regularizer=l2(hyperparameters["l2_penalty"]),
                    bias_regularizer=l2(hyperparameters["l2_penalty"]),
                    kernel_initializer='glorot_uniform',
                    bias_initializer='zeros'))

    # compile the model, provide an optimizer
    model.compile(loss='categorical_crossentropy',
                  optimizer='rmsprop',
                  metrics=['accuracy'])

    # print a summary
    print(model.summary())


    # train the model with early stopping
```

```
    early_stopping = EarlyStopping(patience=hyperparameters["patience"])
    Y = np.array(y_data)

    fit = model.fit(x_data,
                    Y,
                    batch_size=hyperparameters["batch_size"],
                    epochs=hyperparameters["epochs"],
                    validation_split=hyperparameters["VALIDATION_SPLIT"],
                    verbose=1,
                    callbacks=[early_stopping])

    print(fit.history.keys())
    val_accuracy = fit.history['acc'][-1]
    print(val_accuracy)
    # save the model

    if model_file:
        model.save(model_file)
    return model, val_accuracy, len(train_labels)


def tokenize(text):
    # for each token in the text (the result of text.split(),
    # apply a function that strips punctuation and converts to lower case.
    tokens = map(lambda x: x.strip(',.&').lower(), text.split())
    # get rid of empty tokens
    tokens = list(filter(None, tokens))
    return tokens


# from google.colab import drive
# drive.mount('/content/drive', force_remount=True)
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=

Enter your authorization code:
··········
Mounted at /content/drive
```

```
# read the first 500,000 yelp reviews
df = pd.read_csv("yelp_reviews.csv", encoding='utf-8', engine='python', error_bad_lin
```

```
Skipping line 59684: unexpected end of data
```

```
# Create a list of text reviews from the text column in the reviews dataframe
text = df['text'].values.tolist()
# Tokenize the reviews
texts = list(map(tokenize, text))
# Create a list of labels from the label column in the reviews dataframe
labels = df['stars'].values.tolist()
```

```
# Create a vocabulary from the tokenized texts
mydict = gensim.corpora.Dictionary(texts)
mydict.save('yelp.dict')


#### MODEL 1 : Bag of word representation - Word level
#Epochs = 10
#Batch_Size = 100
#Dropout_rate = 0.3
#Dense_activation function = relu


# train the model
train(texts, labels, mydict, hyperparameters, model_file='yelp_cnn.model')
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 229, 100)          23191000
_________________________________________________________________
dropout_5 (Dropout)          (None, 229, 100)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 222, 100)          80100
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 1, 100)            0
_________________________________________________________________
flatten_2 (Flatten)          (None, 100)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_4 (Dense)              (None, 500)               50500
_________________________________________________________________
dropout_7 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_5 (Dense)              (None, 500)               250500
_________________________________________________________________
dropout_8 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 3006
=================================================================
Total params: 23,575,106
Trainable params: 23,575,106
Non-trainable params: 0
_________________________________________________________________
None
Train on 131459 samples, validate on 14607 samples
Epoch 1/10
131459/131459 [==============================] - 22s 171us/step - loss: 0.9952 -
Epoch 2/10
131459/131459 [==============================] - 22s 164us/step - loss: 0.8014 -
Epoch 3/10
131459/131459 [==============================] - 21s 163us/step - loss: 0.7558 -
Epoch 4/10
131459/131459 [==============================] - 21s 163us/step - loss: 0.7272 -
Epoch 5/10
131459/131459 [==============================] - 22s 164us/step - loss: 0.7036 -
Epoch 6/10
131459/131459 [==============================] - 21s 163us/step - loss: 0.6818 -
Epoch 7/10
131459/131459 [==============================] - 21s 163us/step - loss: 0.6642 -
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
0.7380323908085421
(<keras.engine.sequential.Sequential at 0x7fc165d2a6a0>,
 0.7380323908085421,
 146066)
```

```
#### MODEL 2 : Bag of word representation - Word level : Change NUMBER OF EPOCHS
#Epochs = 20
#Batch_Size = 100
```

```
#Dropout_rate = 0.3
#Dense_activation function = relu


hyperparameters["epochs"] = 20
train(texts, labels, mydict, hyperparameters, model_file='yelp_cnn_epochs.model')
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 229, 100)          23191000
_________________________________________________________________
dropout_13 (Dropout)         (None, 229, 100)          0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 222, 100)          80100
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 1, 100)            0
_________________________________________________________________
flatten_4 (Flatten)          (None, 100)               0
_________________________________________________________________
dropout_14 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_10 (Dense)             (None, 500)               50500
_________________________________________________________________
dropout_15 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_11 (Dense)             (None, 500)               250500
_________________________________________________________________
dropout_16 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_12 (Dense)             (None, 6)                 3006
=================================================================
Total params: 23,575,106
Trainable params: 23,575,106
Non-trainable params: 0
_________________________________________________________________
None
Train on 131459 samples, validate on 14607 samples
Epoch 1/20
131459/131459 [==============================] - 23s 174us/step - loss: 0.9821 -
Epoch 2/20
131459/131459 [==============================] - 22s 166us/step - loss: 0.8001 -
Epoch 3/20
131459/131459 [==============================] - 22s 166us/step - loss: 0.7547 -
Epoch 4/20
131459/131459 [==============================] - 22s 166us/step - loss: 0.7251 -
Epoch 5/20
131459/131459 [==============================] - 22s 165us/step - loss: 0.7006 -
Epoch 6/20
131459/131459 [==============================] - 22s 165us/step - loss: 0.6813 -
Epoch 7/20
131459/131459 [==============================] - 22s 165us/step - loss: 0.6617 -
Epoch 8/20
131459/131459 [==============================] - 22s 165us/step - loss: 0.6465 -
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
0.7476323413746769
(<keras.engine.sequential.Sequential at 0x7fc1659c67b8>,
 0.7476323413746769,
 146066)
```

MODEL 3 : Bag of word representation - Word level : Keep number of epochs same a
Change BATCH_SIZE

```
#### Change BATCH SIZE
#Epochs = 20
#Batch_Size = 200
#Dropout_rate = 0.3
#Dense_activation function = relu


hyperparameters["batch_size"] = 200
train(texts, labels, mydict, hyperparameters, model_file='yelp_cnn_batch.model')
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 229, 100)          23191000
_________________________________________________________________
dropout_17 (Dropout)         (None, 229, 100)          0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 222, 100)          80100
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 1, 100)            0
_________________________________________________________________
flatten_5 (Flatten)          (None, 100)               0
_________________________________________________________________
dropout_18 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_13 (Dense)             (None, 500)               50500
_________________________________________________________________
dropout_19 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_14 (Dense)             (None, 500)               250500
_________________________________________________________________
dropout_20 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_15 (Dense)             (None, 6)                 3006
=================================================================
Total params: 23,575,106
Trainable params: 23,575,106
Non-trainable params: 0
_________________________________________________________________
None
Train on 131459 samples, validate on 14607 samples
Epoch 1/20
131459/131459 [==============================] - 15s 114us/step - loss: 1.0409 -
Epoch 2/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.8057 -
Epoch 3/20
131459/131459 [==============================] - 14s 107us/step - loss: 0.7477 -
Epoch 4/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.7094 -
Epoch 5/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.6799 -
Epoch 6/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.6526 -
Epoch 7/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.6267 -
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
0.7555663740442945
(<keras.engine.sequential.Sequential at 0x7fc0577926a0>,
 0.7555663740442945,
 146066)
```

```
#### MODEL 4 : Bag of word representation - Word level : Keep number of epochs and ba
#### Change DENSE ACTIVATION FUNCTION
#Epochs = 20
```

```
#Batch_Size = 200
#Dropout_rate = 0.3
#Dense_activation function = softplus


hyperparameters["dense_activation"] = "softplus"
train(texts, labels, mydict, hyperparameters, model_file='yelp_cnn_activation.model')
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_7 (Embedding)      (None, 229, 100)          23191000
_________________________________________________________________
dropout_25 (Dropout)         (None, 229, 100)          0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 222, 100)          80100
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 1, 100)            0
_________________________________________________________________
flatten_7 (Flatten)          (None, 100)               0
_________________________________________________________________
dropout_26 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_19 (Dense)             (None, 500)               50500
_________________________________________________________________
dropout_27 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_20 (Dense)             (None, 500)               250500
_________________________________________________________________
dropout_28 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_21 (Dense)             (None, 6)                 3006
=================================================================
Total params: 23,575,106
Trainable params: 23,575,106
Non-trainable params: 0
_________________________________________________________________
None
Train on 131459 samples, validate on 14607 samples
Epoch 1/20
131459/131459 [==============================] - 15s 115us/step - loss: 1.2869 -
Epoch 2/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.9183 -
Epoch 3/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.8250 -
Epoch 4/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.7834 -
Epoch 5/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.7522 -
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
0.6976167477475144
(<keras.engine.sequential.Sequential at 0x7fbfb7d6e390>,
 0.6976167477475144,
 146066)
```

```
#### MODEL 5 : Bag of word representation - Word level : Keep number of epochs and ba
```

```
#### Change dense activation function back to relu, Change DROPOUT RATE
#Epochs = 20
#Batch_Size = 200
#Dropout_rate = 0.4
#Dense_activation function = relu


hyperparameters["dense_activation"] = "relu"
hyperparameters["dropout_rate"] = 0.4
train(texts, labels, mydict, hyperparameters, model_file='yelp_cnn_dropout.model')
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_10 (Embedding)     (None, 229, 100)          23191000
_________________________________________________________________
dropout_37 (Dropout)         (None, 229, 100)          0
_________________________________________________________________
conv1d_10 (Conv1D)           (None, 222, 100)          80100
_________________________________________________________________
max_pooling1d_10 (MaxPooling (None, 1, 100)            0
_________________________________________________________________
flatten_10 (Flatten)         (None, 100)               0
_________________________________________________________________
dropout_38 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_28 (Dense)             (None, 500)               50500
_________________________________________________________________
dropout_39 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_29 (Dense)             (None, 500)               250500
_________________________________________________________________
dropout_40 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_30 (Dense)             (None, 6)                 3006
=================================================================
Total params: 23,575,106
Trainable params: 23,575,106
Non-trainable params: 0
_________________________________________________________________
None
Train on 131459 samples, validate on 14607 samples
Epoch 1/20
131459/131459 [==============================] - 15s 117us/step - loss: 1.0907 -
Epoch 2/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.8379 -
Epoch 3/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.7869 -
Epoch 4/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.7569 -
Epoch 5/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.7374 -
Epoch 6/20
131459/131459 [==============================] - 14s 105us/step - loss: 0.7164 -
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
0.7145573908146315
(<keras.engine.sequential.Sequential at 0x7fbfb7f892b0>,
 0.7145573908146315,
 146066)
```

```
# let's create a bag-of-words using both 1-grams and 2-grams
def uni_and_bigrams(text):
    # our unigrams are our tokens
    unigrams=tokenize(text)
    # the bigrams just contatenate 2 adjacent tokens with _ in between
```

```
    bigrams=list(map(lambda x: '_'.join(x), zip(unigrams, unigrams[1:])))
    # returning a list containing all 1 and 2-grams
    return unigrams+bigrams


tokenized_texts=list(map(uni_and_bigrams, text))


# let's see what the new dictionary looks like
my_bigram_dict = gensim.corpora.Dictionary(tokenized_texts)
```

```
2019-11-10 03:45:39,294 : INFO : adding document #0 to Dictionary(0 unique tokens
2019-11-10 03:45:41,540 : INFO : adding document #10000 to Dictionary(390165 unic
2019-11-10 03:45:43,812 : INFO : adding document #20000 to Dictionary(650860 unic
2019-11-10 03:45:46,217 : INFO : adding document #30000 to Dictionary(872596 unic
2019-11-10 03:45:48,652 : INFO : adding document #40000 to Dictionary(1085476 uni
2019-11-10 03:45:51,050 : INFO : adding document #50000 to Dictionary(1277001 uni
2019-11-10 03:45:53,528 : INFO : built Dictionary(1449082 unique tokens: ['$69',
```

```
#### MODEL 6 : Bag of word representation - NGRAM level : Keep number of epochs, batc
#### Change dropout back to 0.3
#Epochs = 20
#Batch_Size = 200
#Dropout_rate = 0.3
#Dense_activation function = relu


hyperparameters["batch_size"] = 200
hyperparameters["epochs"] = 20
hyperparameters["dense_activation"] = "relu"
hyperparameters["dropout_rate"] = 0.3


train(tokenized_texts, labels, my_bigram_dict, hyperparameters, model_file='yelp_cnn_
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 453, 100)          144908300
_________________________________________________________________
dropout_5 (Dropout)          (None, 453, 100)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 446, 100)          80100
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 1, 100)            0
_________________________________________________________________
flatten_2 (Flatten)          (None, 100)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_4 (Dense)              (None, 500)               50500
_________________________________________________________________
dropout_7 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_5 (Dense)              (None, 500)               250500
_________________________________________________________________
dropout_8 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 3006
=================================================================
Total params: 145,292,406
Trainable params: 145,292,406
Non-trainable params: 0
_________________________________________________________________
None
/usr/local/lib/python3.6/dist-packages/tensorflow_core/python/framework/indexed_s
  num_elements)
Train on 53713 samples, validate on 5969 samples
Epoch 1/20
53713/53713 [==============================] - 18s 326us/step - loss: 1.2723 - ac
Epoch 2/20
53713/53713 [==============================] - 15s 287us/step - loss: 0.9024 - ac
Epoch 3/20
53713/53713 [==============================] - 16s 297us/step - loss: 0.6949 - ac
Epoch 4/20
53713/53713 [==============================] - 16s 297us/step - loss: 0.5233 - ac
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
0.811796025526716
(<keras.engine.sequential.Sequential at 0x7fe51de13128>,
 0.811796025526716,
 59682)
```