```
from keras_preprocessing.sequence import pad_sequences
import numpy as np
import pandas as pd
from gensim.models.word2vec import Word2Vec
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.layers import Dense, Dropout, Conv1D, MaxPool1D, GlobalMaxPool1D, Embedding, Activ
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.snowball import PorterStemmer
from sklearn import preprocessing
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df·=·pd.read_csv('data.csv',delimiter=',',encoding='latin-1')
df·=·df[['Category','Message']]
df·=·df[pd.notnull(df['Message'])]
df.rename(columns·=·{'Message':'Message'},·inplace·=·True)
df.head()
```

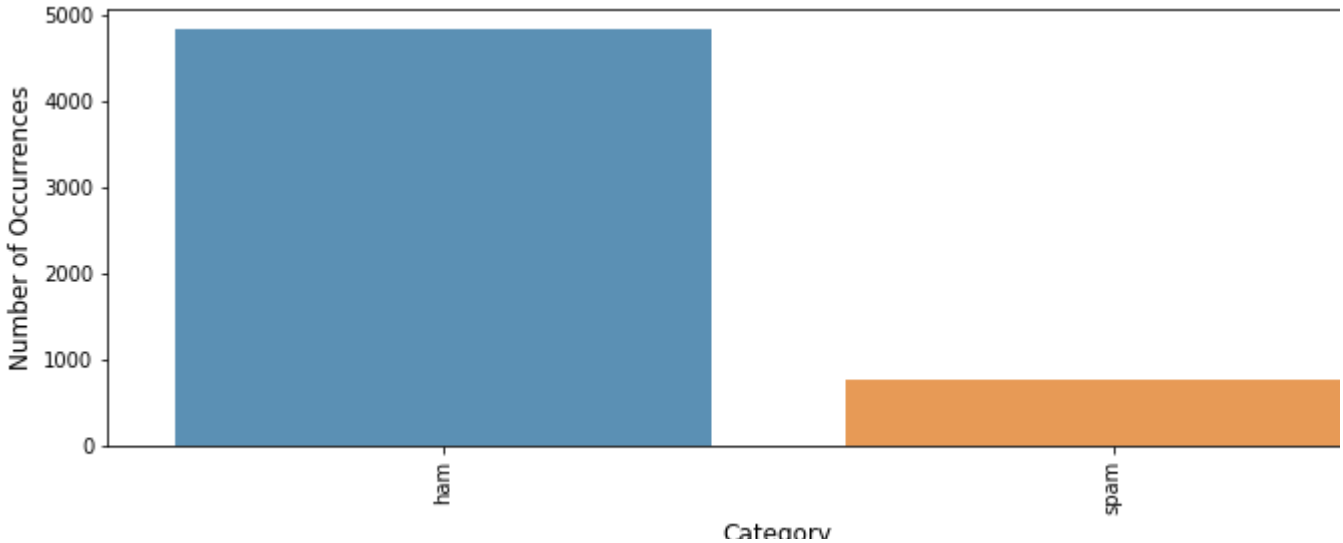
	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
df.shape
```

```
(5572, 2)
```

```
cnt_pro = df['Category'].value_counts()
plt.figure(figsize=(12,4))
sns.barplot(cnt_pro.index, cnt_pro.values, alpha=0.8)
plt.ylabel('Number of Occurrences', fontsize=12)
plt.xlabel('Category', fontsize=12)
plt.xticks(rotation=90)
plt.show();
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass th
  warnings.warn(
```




```
def preprocess_text(sen):
    # Remove punctuations and numbers
    sentence = re.sub('[^a-zA-Z]', ' ', sen)

    # Single character removal
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)

    # Removing multiple spaces
    sentence = re.sub(r'\s+', ' ', sentence)

    stops = stopwords.words('english')
    #print(stops)
    porter = PorterStemmer()
    for word in sentence.split():
        if word in stops:
            sentence = sentence.replace(word, '')
        sentence = sentence.replace(word, porter.stem(word))
    return sentence.lower()


mes = []
for i in df['Message']:
    mes.append(i.split())
print(mes[:2])
```

```
[['Go', 'until', 'jurong', 'point,', 'crazy..', 'Available', 'only', 'in', 'bugis', 'n',
```

```
word2vec_model = Word2Vec(mes, size=500, window=3, min_count=1, workers=16)
print(word2vec_model)
```

```
WARNING:gensim.models.base_any2vec:under 10 jobs per worker: consider setting a smaller
Word2Vec(vocab=15686, size=500, alpha=0.025)
```

```
token = Tokenizer(7229)
token.fit_on_texts(df['Message'])
text = token.texts_to_sequences(df['Message'])
text = pad_sequences(text, 75)
print(text[:2])
```

```
[[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0   49
   472 4427  841  756  659   65    8 1328   87  123  352 1329  148 2993
  1330   67   58 4428  144]
 [   0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0    0    0    0    0    0   46
   337 1500  473    6 1941]]
```

```
le = preprocessing.LabelEncoder()
y = le.fit_transform(df['Category'])
y = to_categorical(y)
y[:2]
```

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

```
x_train, x_test, y_train, y_test = train_test_split(np.array(text), y, test_size=0.2, stratif
```

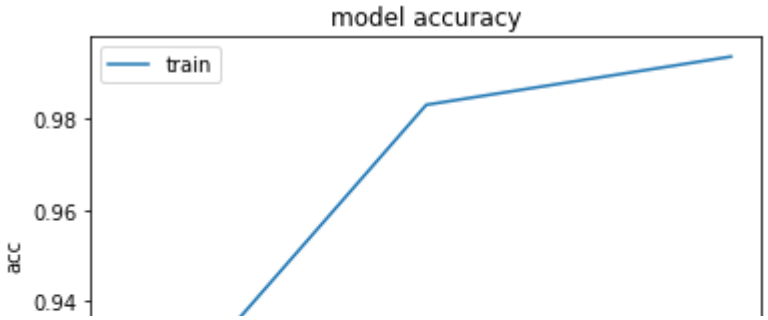
```
keras_model = Sequential()
keras_model.add(word2vec_model.wv.get_keras_embedding(True))
keras_model.add(Dropout(0.2))
keras_model.add(Conv1D(50, 3, activation='relu', padding='same', strides=1))
keras_model.add(Conv1D(50, 3, activation='relu', padding='same', strides=1))
keras_model.add(MaxPool1D())
keras_model.add(Dropout(0.2))
keras_model.add(Conv1D(100, 3, activation='relu', padding='same', strides=1))
keras_model.add(Conv1D(100, 3, activation='relu', padding='same', strides=1))
keras_model.add(MaxPool1D())
keras_model.add(Dropout(0.2))
keras_model.add(Conv1D(200, 3, activation='relu', padding='same', strides=1))
keras_model.add(Conv1D(200, 3, activation='relu', padding='same', strides=1))
keras_model.add(GlobalMaxPool1D())
keras_model.add(Dropout(0.2))
keras_model.add(Dense(200))
keras_model.add(Activation('relu'))
keras_model.add(Dropout(0.2))
keras_model.add(Dense(2))
keras_model.add(Activation('softmax'))
```

```
keras_model.compile(loss='binary_crossentropy', metrics=['acc'], optimizer='adam')
history = keras_model.fit(x_train, y_train, batch_size=16, epochs=3, validation_data=(x_test,
```

```
Epoch 1/3
279/279 [==============================] - 20s 70ms/step - loss: 0.2702 - acc: 0.9069 -
Epoch 2/3
279/279 [==============================] - 19s 69ms/step - loss: 0.0713 - acc: 0.9829 -
Epoch 3/3
279/279 [==============================] - 21s 74ms/step - loss: 0.0253 - acc: 0.9935 -
```

```
plt.plot(history.history['acc'])
plt.title('model accuracy')
plt.ylabel('acc')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.savefig('model_accuracy.png')

# summarize history for loss
plt.plot(history.history['loss'])
#plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.savefig('model_loss.png')
```




```
from tkinter.constants import X
message = ['Congratulations! you have won a $1,000 Walmart gift card. Go to http://bit.ly/123
seq = token.texts_to_sequences(message)

padded = pad_sequences(seq, maxlen=text.shape[1], dtype='int32', value=0)

pred = keras_model.predict(padded)

labels = ['ham','spam']
print(pred, labels[np.argmax(pred)])
```

```
1/1 [==============================] - 0s 166ms/step
[[1.7089591e-12 1.0000000e+00]] spam
```

```
message = ['thanks for accepting my request to connect']
seq = token.texts_to_sequences(message)

padded = pad_sequences(seq, maxlen=text.shape[1], dtype='int32', value=0)

pred = model.predict(padded)

labels = ['ham','spam']
print(pred, labels[np.argmax(pred)])
```

```
1/1 [==============================] - 0s 112ms/step
[[0.86681026 0.13318971]] ham
```

Analysis:

Throughtout my process of evaulating three different neural network models, it has come to my attention that with embeddings added, it's pretty hard to fit the model without it.

Sequential model as itself without CNN or RNN or even any other types of Neural Networks seems to give out very good accuracy but when testing the values to check if the following sentence is spam or not. It fails to give the right result.

All the other models that I tested both CNN and RNN gave very good accuracy with the right results as intended.