In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries ins
talled
# It is defined by the kaggle/python Docker image: https://github.com/kaggl
e/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will
 list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that
gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be sa
ved outside of the current session
```

```
/kaggle/input/nlp-getting-started/sample_submission.csv
/kaggle/input/nlp-getting-started/train.csv
/kaggle/input/nlp-getting-started/test.csv
```

Brief Description of Problem

This is an introductory competition on Kaggle to learn Natural Language Processing (NLP) techniques. We are given tweets and have to determine if they are actually announcing disaster or not. This is a simple binary classification (only 2 categories).

Exploratory Data Analysis

Load Dataset

In [2]:

```
train = pd.read_csv('/kaggle/input/nlp-getting-started/train.csv')
test = pd.read_csv('/kaggle/input/nlp-getting-started/test.csv')
```

In [3]:

```
train.head()
```

Out[3]:

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

In [4]:

```
test.head()
```

Out[4]:

	id	keyword	location	text
0	0	NaN	NaN	Just happened a terrible car crash
1	2	NaN	NaN	Heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	Apocalypse lighting. #Spokane #wildfires
4	11	NaN	NaN	Typhoon Soudelor kills 28 in China and Taiwan

In [5]:

```
len(test)
```

Out[5]:

3263

Since we only want to train on the text, I will remove the columns 'id', 'keyword', and 'location'. The target column will be our labels.

In [6]:
```
train = train.drop(['id', 'keyword', 'location'], axis=1)
test = test.drop(['id', 'keyword', 'location'], axis=1)
```

In [7]:
```
print(train.shape)
print(test.shape)
```

```
(7613, 2)
(3263, 1)
```

In [8]:
```
train['target'].value_counts()
```

Out[8]:
```
0    4342
1    3271
Name: target, dtype: int64
```

Preprocessing Data

We see the training dataset is somewhat imbalanced. I will further clean the training data before handling this imbalance.

I want to look at more entries to see what characters/cleaning may be necessary. I will select 50 random entries from the target set.

In [9]:

```
import random

rand_idx = random.sample(list(train.index),50)

for idx in rand_idx:
    print(train.iloc[idx,0])
```

```
Still blazing ????
Investigators rule catastrophic structural failure resulted in 2014
... http://t.co/AdZ8kbuRt7
Look for my Policy Matters Ohio report on #CLE and Cuyahoga County bli
ght and greening vacant lands soon!  https://t.co/if62SdXVp7
Whirlwind Head Scissor on @alexhammerstone @kttape ktfounder #RemyMarc
el #FroFroFro Û_ https://t.co/B19z8Vi3td
Fear and panic in the air
I want to be free
From desolation and despair
And I feel like everything I sow ? http://t.co/iXW2cUTk1C
Gunshot wound #9 is in the bicep. The only one of the ten wounds that
is not in the chest/torso area.  #KerrickTrial #JonathanFerrell
I rated Catastrophe (2015) 8/10  #IMDb - hilarious! http://t.co/cjrSSR
Y1RT
#Earthquake #Sismo M 1.9 - 15km E of Anchorage Alaska: Time2015-08-06
00:11:16 UTC2015-08-05 16:11:16 -08:00 ... http://t.co/Z0VeR1hVM9
Best believe all the wrong decisions are being made out here in these
Memphis streets during this here rainstorm lol my folk doe
oh yeah my ipod almost exploded last night i was using it while chargi
ng and shit was sparking akxbskdn almost died
Watching 'The Desolation of Smaug' in Spanish is a hell of a drug
I hope they fall off a cliff.
Dramatic Video Shows Plane Landing During Violent Storm http://t.co/rJ
9gkJKJJn
And please don't flood poor @RobertBEnglund's mentions. He's put in hi
s work!
I can't believe @myfriendmina photo bombed a screenshot
Why weren't they taken back to Africa once rescued? #c4news
STAR WARS POWER OF THE JEDI COLLECTION 1 BATTLE DROID HASBRO - Full re
ad by eBay http://t.co/xFguklrlTf http://t.co/FeGu8hWMc4
just got engulfed in a car-induced tidal wave on my run... I thought t
his only happened in the movies ????
My brains going to explode i need to leave this house. Ill be out smok
ing packs if you need me
daviesmutia: Breaking news! Unconfirmed! I just heard a loud bang near
by. in what appears to be a blast of wind from my neighbour's ass.
I understand why broke ppl be mad or always hav an attitude now this s
ht ain't no fun i won't be desolate for long
@jamienye u can't blame it all on coaching management penalties defenc
```

e or injuries. Cursed is probably a good way to put it! #riders
Gut Deutsch musik! The old and rotten the monarchy has collapsed. The new may live. Long live the German Republic! https://t.co/RJjU70rHyu
Emergency units simulate a chemical explosion at NU: Suppose a student in the research labs at Northwestern Û_ http://t.co/ExitLxgIsJ
Watching a man electrocuted on the roof of #mumbailocals is definitely a lesson.. People please learn!! #lessonforlife #marinelines #mumbai
Obama Declares Disaster for Typhoon-Devastated Saipan: Obama signs disaster declaration for Northern Marians a... http://t.co/XDt4VHFn7B
@ThomasHCrown My grandfather was set to be in the first groups of Marines to hit Japan in Operation Olympic. 95% casualty rate predictions
'I tried to save Mido Macia': One of the murder accused has testified he tried to save Mido Macia Ûªs life. http://t.co/vxVfAEEY0q
#PFT Barkevious Mingo missed Browns practice with a mystery injury http://t.co/D7m9KGMPJI
I liked a @YouTube video from @jeromekem http://t.co/Nq89drydbU DJ Hazard - Death Sport
Permits for bear hunting in danger of outnumbering actual bears: The licenses for Florida's fir... http://t.co/FP64YOSJwx #st petersburg
@_itsmegss_ I think it is. well it's bloody barking now
My emotions are a train wreck. My body is a train wreck. I'm a wreck and I thought my surgical wounds were healed!!! this weather ain't helping either):
Dr. Bengston on #wildfire management: ÛÏnumbers and size of fires areas affected and costs of fighting them all show upward trend. Û #smem
@laevantine Fortunately I reworked the plumbing on my emergency chemical shower to draw from the glitter pipe for just such an occasion
I tell my cousins I don't wanna hang out and they text me saying 'we're coming over' honestly do you have a death wish
13,000 people receive #wildfires evacuation orders in California
I had to grill for a school function. One of the grills we had going was pretty much either off or forest fire. No inbetween! Made it work
@Hurricane_Dolce no prob
What the fuck is going on at Trent Bridge?! Reminds me of England's collapse out in the Caribbean back in the 90s...
Storm batters Auckland and Northland: A violent overnight storm has battered Auckland and Northland uprooting... http://t.co/enrPGRgtTs
@bre_morrow neither of them even smoke so I dk what was going on lol
VIDEO: 'We're picking up bodies from water': Rescuers are searching for hundreds of migrants in ... http://t.co/bS6PjT09Tc #Africa #News
@mockingpanems @cuddlesforjen what if he slammed her against the wall

```
for the wrong reason but then he came out of hijack mode and it
Worlds Collide When an American Family Takes Over Britain's Isle of Ma
n in New TLC Show Suddenly Royal http://t.co/OmB3oS54tN via @People
Watch This Airport Get Swallowed Up By A Sandstorm In Under A Minute h
ttp://t.co/mkWyvM3i8r
Posted a new song: 'Earthquake' http://t.co/RfTyyZ4GwJ http://t.co/lau
0Ay7ahV
kotolily_: Breaking news! Unconfirmed! I just heard a loud bang nearb
y. in what appears to be a blast of wind from my neighbour's ass.
Watch This Airport Get Swallowed Up By A Sandstorm In Under A Minute h
ttp://t.co/8L4RFFZD0P
```

So, we see there are a lot of links (http://...) and tweets directed at certain users. Removing html links and strings that begin with @ will be the initial set for cleaning.

In [10]:

```
import re
```

In [11]:

```
def remove_links(sentence):
    link = re.compile(r'https?://\S+')
    return link.sub(r'', sentence)

def remove_targeted_tweets(sentence):
    tgt_twt = re.compile(r'@\S+')
    return tgt_twt.sub(r'', sentence)

def clean_data(data):
    data['text'] = data['text'].apply(lambda x : remove_links(x))
    data['text'] = data['text'].apply(lambda x : remove_targeted_tweets(x
))
    return data
```

In [12]:

```
train_cleaned = clean_data(train)
test_cleaned = clean_data(test)
```

Now I will look at the same subset again

In [13]:

```
for idx in rand_idx:
    print(train_cleaned.iloc[idx,0])
```

```
Still blazing ????
Investigators rule catastrophic structural failure resulted in 2014
...
Look for my Policy Matters Ohio report on #CLE and Cuyahoga County bli
ght and greening vacant lands soon!
Whirlwind Head Scissor on   ktfounder #RemyMarcel #FroFroFro Û_
Fear and panic in the air
I want to be free
From desolation and despair
And I feel like everything I sow ?
Gunshot wound #9 is in the bicep. The only one of the ten wounds that
is not in the chest/torso area.  #KerrickTrial #JonathanFerrell
I rated Catastrophe (2015) 8/10  #IMDb - hilarious!
#Earthquake #Sismo M 1.9 - 15km E of Anchorage Alaska: Time2015-08-06
00:11:16 UTC2015-08-05 16:11:16 -08:00 ...
Best believe all the wrong decisions are being made out here in these
Memphis streets during this here rainstorm lol my folk doe
oh yeah my ipod almost exploded last night i was using it while chargi
ng and shit was sparking akxbskdn almost died
Watching 'The Desolation of Smaug' in Spanish is a hell of a drug
I hope they fall off a cliff.
Dramatic Video Shows Plane Landing During Violent Storm
And please don't flood poor  mentions. He's put in his work!
I can't believe  photo bombed a screenshot
Why weren't they taken back to Africa once rescued? #c4news
STAR WARS POWER OF THE JEDI COLLECTION 1 BATTLE DROID HASBRO - Full re
ad by eBay
just got engulfed in a car-induced tidal wave on my run... I thought t
his only happened in the movies ????
My brains going to explode i need to leave this house. Ill be out smok
ing packs if you need me
daviesmutia: Breaking news! Unconfirmed! I just heard a loud bang near
by. in what appears to be a blast of wind from my neighbour's ass.
I understand why broke ppl be mad or always hav an attitude now this s
ht ain't no fun i won't be desolate for long
 u can't blame it all on coaching management penalties defence or inju
ries. Cursed is probably a good way to put it! #riders
Gut Deutsch musik! The old and rotten the monarchy has collapsed. The
new may live. Long live the German Republic!
Emergency units simulate a chemical explosion at NU: Suppose a student
```

in the research labs at Northwestern Û_
Watching a man electrocuted on the roof of #mumbailocals is definitely a lesson.. People please learn!! #lessonforlife #marinelines #mumbai
Obama Declares Disaster for Typhoon-Devastated Saipan: Obama signs disaster declaration for Northern Marians a...
My grandfather was set to be in the first groups of Marines to hit Japan in Operation Olympic. 95% casualty rate predictions
'I tried to save Mido Macia': One of the murder accused has testified he tried to save Mido Macia Ûªs life.
#PFT Barkevious Mingo missed Browns practice with a mystery injury
I liked a video from DJ Hazard - Death Sport
Permits for bear hunting in danger of outnumbering actual bears: The licenses for Florida's fir... #st petersburg
I think it is. well it's bloody barking now
My emotions are a train wreck. My body is a train wreck. I'm a wreck and I thought my surgical wounds were healed!!! this weather ain't helping either):
Dr. Bengston on #wildfire management: ÛÏnumbers and size of fires areas affected and costs of fighting them all show upward trend. Û #smem
Fortunately I reworked the plumbing on my emergency chemical shower to draw from the glitter pipe for just such an occasion
I tell my cousins I don't wanna hang out and they text me saying 'we're coming over' honestly do you have a death wish
13,000 people receive #wildfires evacuation orders in California
I had to grill for a school function. One of the grills we had going was pretty much either off or forest fire. No inbetween! Made it work no prob
What the fuck is going on at Trent Bridge?! Reminds me of England's collapse out in the Caribbean back in the 90s...
Storm batters Auckland and Northland: A violent overnight storm has battered Auckland and Northland uprooting...
neither of them even smoke so I dk what was going on lol
VIDEO: 'We're picking up bodies from water': Rescuers are searching for hundreds of migrants in ... #Africa #News
what if he slammed her against the wall for the wrong reason but then he came out of hijack mode and it
Worlds Collide When an American Family Takes Over Britain's Isle of Man in New TLC Show Suddenly Royal via
Watch This Airport Get Swallowed Up By A Sandstorm In Under A Minute
Posted a new song: 'Earthquake'
kotolily_: Breaking news! Unconfirmed! I just heard a loud bang nearb

```
y. in what appears to be a blast of wind from my neighbour's ass.
Watch This Airport Get Swallowed Up By A Sandstorm In Under A Minute
```

So, we see the links and targets of tweets have been removed, leaving just the information from the "body" of the tweet.

Next, I will check for duplicates and attempt to remove them as well.

In [14]:

```
train_cleaned = train_cleaned.drop_duplicates(subset='text', keep="first"
)
```

In [15]:

```
print(train_cleaned.shape)
print(test_cleaned.shape)
```

```
(6958, 2)
(3263, 1)
```

Ok, so we have removed duplicate entries and now we can check the class balances again. If they are imbalanced, I will undersample the majority class.

In [16]:

```
train_cleaned['target'].value_counts()
```

Out[16]:

```
0    4104
1    2854
Name: target, dtype: int64
```

In [17]:

```
#Undersmaple majority class
class0 = train_cleaned[train_cleaned['target']==0]
class1 = train_cleaned[train_cleaned['target']==1]

class0_sample = class0.sample(n=class1.shape[0])
```

In [18]:

```
train_cleaned_balanced = pd.concat([class0_sample,class1]).sample(frac=1,
random_state=12345).reset_index(drop=True)
train_cleaned_balanced.head()
```

Out[18]:

	text	target
0	Fire in Pisgah National Forest grows to 375 ac...	1
1	RIZZO IS ON ???????? THAT BALL WAS OBLITERATED	0
2	Do you want to play a game?\n\nIts a GoogleMap...	0
3	Remaining Sections Of Greystone Psychiatric Ho...	0
4	as in dropping the No-Sports show? I don't t...	0

In [19]:

```
train_cleaned_balanced['target'].value_counts()
```

Out[19]:

```
1    2854
0    2854
Name: target, dtype: int64
```

Next I will look to remove "stop words". These are the words in the english language that typically provide little information (such as a, an, the, etc.).

In [20]:

```
from nltk.corpus import stopwords


def remove_stop_words(sentence):
    SENTENCE = sentence.split()
    WORDS = [word for word in SENTENCE if word not in stopwords.words('en
glish')]

    return ' '.join(WORDS)

def clean_stop_words(data):
    data['text'] = data['text'].apply(lambda x : remove_stop_words(x))
    return data
```

In [21]:

```
train_cleaned_balanced = clean_data(train_cleaned_balanced)
test_cleaned = clean_data(test_cleaned)
```

In [22]:

```
train_cleaned_balanced.head()
```

Out[22]:

	text	target
0	Fire in Pisgah National Forest grows to 375 ac...	1
1	RIZZO IS ON ???????? THAT BALL WAS OBLITERATED	0
2	Do you want to play a game?\n\nIts a GoogleMap...	0
3	Remaining Sections Of Greystone Psychiatric Ho...	0
4	as in dropping the No-Sports show? I don't t...	0

The final step in preprocessing the data will be to convert the text into a format the model will be able to understand. This is called tokenization. I have included both the training and test sets in the corpus of text in order to make sure all words are included in both sets. In a real world example, where we do not have the test set in advance, I would only be able to classify words previously seen. Padding extends the observations with blank spaces at the end of the sentence in order for all observations to be the same length.

I used the built in keras tokenizer instead of other formats. There were pros and cons to this decision, with the primary pro being I am able to learn a bit more about tensorflow and tensors which is something I want to learn more about as I study deep learning. The major con and something that likely is affecting the results is the lack of n-grams in this tokenizer. "n-grams" are combinations of words which together may be more informative than when found apart. For instance, the bigram "Hurricane Warning" may immediately indicate a pending disaster while "That restaurants Hurricane cocktail should have come with a warning" does not, but it uses both the words hurricane and warning.

In hindsight I probably should have used bigrams and trigrams, but for this assignment I really wanted to focus on understanding the model architecture rather than trying to perfect my score.

In [23]:

```
import tensorflow as tf
from tensorflow.python.keras.preprocessing.text import Tokenizer
from tensorflow.python.keras.preprocessing.sequence import pad_sequences

text_corpus = pd.concat([train_cleaned_balanced['text'],test_cleaned['tex
t']])

tokenizer = tf.keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(text_corpus)
```

In [24]:

```
max_len = max(len(x.split()) for x in text_corpus)
max_len
```

Out[24]:

```
31
```

In [25]:
```
train_features = train_cleaned_balanced.iloc[:,0]
train_labels = train_cleaned_balanced.iloc[:,1]
test_features = test_cleaned.iloc[:,0]
```

In [26]:
```
train_token = tokenizer.texts_to_sequences(train_features)
test_token = tokenizer.texts_to_sequences(test_features)

train_pad = pad_sequences(train_token, maxlen=max_len, padding='post')
test_pad = pad_sequences(test_token, maxlen=max_len, padding='post')
```

In [27]:
```
train_labels = np.array(train_labels)
```

Model Architecture

I will be be using LSTM, one of the more advanced architectures from the RNN family. The LSTMs let the model remember inputs over longer periods of time. I felt this was useful since most on twitter do not write in full sentences but rather snippets. By remembering how to treat these smaller snippets and improper grammar the model may be able to perform a bit better. I will follow these up by some dense/fully connected layers as I find those generally aid with classification.

In [28]:
```
import keras
from keras.layers import LSTM
from keras.models import Sequential
from keras.layers import Dense, Embedding, Bidirectional, Dropout
from keras import optimizers
```

In [29]:

```
all_words = len(tokenizer.word_index)+1
embedding_units = 100
hidden_units = 64

model0 = Sequential()
model0.add(Embedding(all_words, embedding_units, input_length = max_len))
model0.add(Bidirectional(LSTM(hidden_units)))
model0.add(Dense(128, activation='tanh'))
model0.add(Dense(64, activation='tanh'))
model0.add(Dense(1, activation='sigmoid'))

model0.summary()
```

```
2022-05-26 16:56:25.946155: I tensorflow/core/common_runtime/process_u
til.cc:146] Creating new thread pool with default inter op setting: 2.
Tune using inter_op_parallelism_threads for best performance.
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 31, 100)           1806000
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               84480
_________________________________________________________________
dense (Dense)                (None, 128)               16512
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 1,915,313
Trainable params: 1,915,313
Non-trainable params: 0
_________________________________________________________________
```

In [30]:
```
model0.compile( loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(0.00001),
    metrics=['accuracy', 'Precision', 'Recall'])
```

In [31]:
```
#added a callback for early stopping if it appears to be overfitting based
 on val_loss
call_backs = [tf.keras.callbacks.EarlyStopping(monitor='val_loss', patien
ce=5, verbose=1)]
```

In [32]:

```
history0 = model0.fit(train_pad, train_labels,epochs=50,validation_split=
0.2, callbacks = call_backs)
```

```
2022-05-26 16:56:55.375305: I tensorflow/compiler/mlir/mlir_graph_opti
mization_pass.cc:185] None of the MLIR Optimization Passes are enabled
(registered 2)
```

```
Epoch 1/50
143/143 [==============================] - 15s 68ms/step - loss: 0.693
0 - accuracy: 0.4991 - precision: 0.4471 - recall: 0.0167 - val_loss:
0.6927 - val_accuracy: 0.4939 - val_precision: 0.4750 - val_recall: 0.
0330
Epoch 2/50
143/143 [==============================] - 9s 66ms/step - loss: 0.6923
- accuracy: 0.5239 - precision: 0.6361 - recall: 0.1067 - val_loss: 0.
6921 - val_accuracy: 0.5219 - val_precision: 0.6103 - val_recall: 0.14
41
Epoch 3/50
143/143 [==============================] - 8s 58ms/step - loss: 0.6915
- accuracy: 0.5769 - precision: 0.6765 - recall: 0.2910 - val_loss: 0.
6913 - val_accuracy: 0.5841 - val_precision: 0.6613 - val_recall: 0.35
94
Epoch 4/50
143/143 [==============================] - 8s 57ms/step - loss: 0.6904
- accuracy: 0.6251 - precision: 0.6724 - recall: 0.4846 - val_loss: 0.
6901 - val_accuracy: 0.6095 - val_precision: 0.6836 - val_recall: 0.42
01
Epoch 5/50
143/143 [==============================] - 8s 54ms/step - loss: 0.6889
- accuracy: 0.6202 - precision: 0.7605 - recall: 0.3486 - val_loss: 0.
6885 - val_accuracy: 0.6384 - val_precision: 0.7022 - val_recall: 0.49
13
Epoch 6/50
143/143 [==============================] - 9s 60ms/step - loss: 0.6864
- accuracy: 0.6710 - precision: 0.7389 - recall: 0.5268 - val_loss: 0.
6859 - val_accuracy: 0.6629 - val_precision: 0.6969 - val_recall: 0.58
68
Epoch 7/50
143/143 [==============================] - 8s 57ms/step - loss: 0.6825
- accuracy: 0.6995 - precision: 0.7354 - recall: 0.6212 - val_loss: 0.
6817 - val_accuracy: 0.6778 - val_precision: 0.7023 - val_recall: 0.62
67
Epoch 8/50
143/143 [==============================] - 8s 56ms/step - loss: 0.6761
- accuracy: 0.7100 - precision: 0.7671 - recall: 0.6014 - val_loss: 0.
6747 - val_accuracy: 0.6821 - val_precision: 0.7036 - val_recall: 0.63
89
```

```
Epoch 9/50
143/143 [==============================] - 8s 56ms/step - loss: 0.6656
- accuracy: 0.7146 - precision: 0.7564 - recall: 0.6313 - val_loss: 0.
6635 - val_accuracy: 0.6891 - val_precision: 0.7097 - val_recall: 0.64
93
Epoch 10/50
143/143 [==============================] - 9s 62ms/step - loss: 0.6490
- accuracy: 0.7276 - precision: 0.7673 - recall: 0.6514 - val_loss: 0.
6468 - val_accuracy: 0.6935 - val_precision: 0.7025 - val_recall: 0.68
06
Epoch 11/50
143/143 [==============================] - 8s 55ms/step - loss: 0.6239
- accuracy: 0.7387 - precision: 0.7410 - recall: 0.7322 - val_loss: 0.
6238 - val_accuracy: 0.6926 - val_precision: 0.7184 - val_recall: 0.64
24
Epoch 12/50
143/143 [==============================] - 8s 56ms/step - loss: 0.5898
- accuracy: 0.7503 - precision: 0.7637 - recall: 0.7234 - val_loss: 0.
5952 - val_accuracy: 0.7032 - val_precision: 0.7120 - val_recall: 0.69
10
Epoch 13/50
143/143 [==============================] - 8s 53ms/step - loss: 0.5476
- accuracy: 0.7659 - precision: 0.7754 - recall: 0.7471 - val_loss: 0.
5678 - val_accuracy: 0.7172 - val_precision: 0.7466 - val_recall: 0.66
49
Epoch 14/50
143/143 [==============================] - 9s 61ms/step - loss: 0.5043
- accuracy: 0.7838 - precision: 0.7993 - recall: 0.7568 - val_loss: 0.
5444 - val_accuracy: 0.7242 - val_precision: 0.7524 - val_recall: 0.67
53
Epoch 15/50
143/143 [==============================] - 8s 55ms/step - loss: 0.4650
- accuracy: 0.8003 - precision: 0.8243 - recall: 0.7621 - val_loss: 0.
5300 - val_accuracy: 0.7338 - val_precision: 0.7361 - val_recall: 0.73
61
Epoch 16/50
143/143 [==============================] - 8s 59ms/step - loss: 0.4335
- accuracy: 0.8136 - precision: 0.8367 - recall: 0.7783 - val_loss: 0.
5227 - val_accuracy: 0.7356 - val_precision: 0.7429 - val_recall: 0.72
74
Epoch 17/50
```

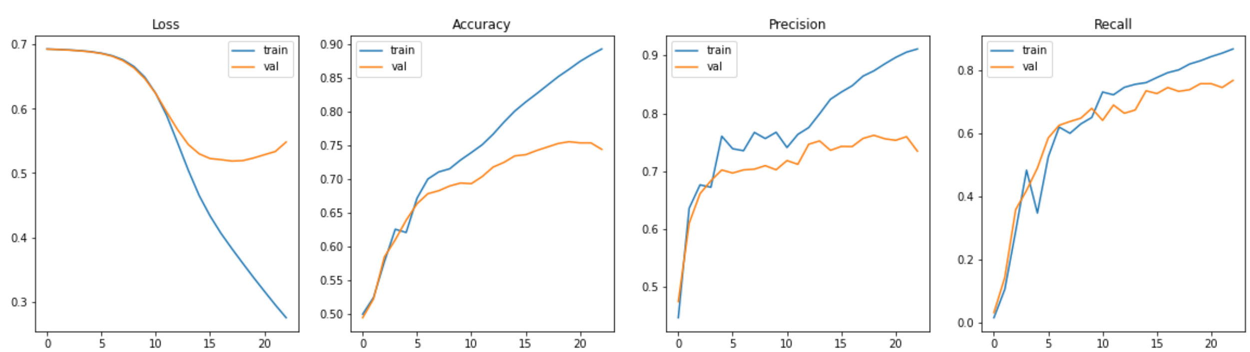
```
143/143 [==============================] - 8s 58ms/step - loss: 0.4065
- accuracy: 0.8259 - precision: 0.8476 - recall: 0.7937 - val_loss: 0.
5209 - val_accuracy: 0.7417 - val_precision: 0.7427 - val_recall: 0.74
65
Epoch 18/50
143/143 [==============================] - 9s 61ms/step - loss: 0.3828
- accuracy: 0.8386 - precision: 0.8643 - recall: 0.8025 - val_loss: 0.
5188 - val_accuracy: 0.7469 - val_precision: 0.7567 - val_recall: 0.73
44
Epoch 19/50
143/143 [==============================] - 9s 60ms/step - loss: 0.3600
- accuracy: 0.8513 - precision: 0.8734 - recall: 0.8209 - val_loss: 0.
5194 - val_accuracy: 0.7522 - val_precision: 0.7621 - val_recall: 0.73
96
Epoch 20/50
143/143 [==============================] - 8s 59ms/step - loss: 0.3378
- accuracy: 0.8622 - precision: 0.8855 - recall: 0.8314 - val_loss: 0.
5235 - val_accuracy: 0.7548 - val_precision: 0.7561 - val_recall: 0.75
87
Epoch 21/50
143/143 [==============================] - 9s 61ms/step - loss: 0.3163
- accuracy: 0.8739 - precision: 0.8966 - recall: 0.8446 - val_loss: 0.
5286 - val_accuracy: 0.7531 - val_precision: 0.7534 - val_recall: 0.75
87
Epoch 22/50
143/143 [==============================] - 10s 67ms/step - loss: 0.295
2 - accuracy: 0.8835 - precision: 0.9057 - recall: 0.8556 - val_loss:
0.5336 - val_accuracy: 0.7531 - val_precision: 0.7597 - val_recall: 0.
7465
Epoch 23/50
143/143 [==============================] - 10s 73ms/step - loss: 0.275
2 - accuracy: 0.8922 - precision: 0.9111 - recall: 0.8687 - val_loss:
0.5484 - val_accuracy: 0.7434 - val_precision: 0.7347 - val_recall: 0.
7691
Epoch 00023: early stopping
```

In [33]:

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(1, 4, figsize=(20, 5))

axs[0].set_title('Loss')
axs[0].plot(history0.history['loss'], label='train')
axs[0].plot(history0.history['val_loss'], label='val')
axs[0].legend()

axs[1].set_title('Accuracy')
axs[1].plot(history0.history['accuracy'], label='train')
axs[1].plot(history0.history['val_accuracy'], label='val')
axs[1].legend()

axs[2].set_title('Precision')
axs[2].plot(history0.history['precision'], label='train')
axs[2].plot(history0.history['val_precision'], label='val')
axs[2].legend()

axs[3].set_title('Recall')
axs[3].plot(history0.history['recall'], label='train')
axs[3].plot(history0.history['val_recall'], label='val')
axs[3].legend()
```

Out[33]:

```
<matplotlib.legend.Legend at 0x7f192cdb48d0>
```



This model did not utilize any dropout for generalization and we can see that there appears to be some overfitting around the 15th epoch and the model stopped before 25 epochs were even completed based on the early stopping callbacks set up. I will next attempt to add some dropout for generalization.

In [34]:

```
model1 = Sequential()
model1.add(Embedding(all_words, embedding_units, input_length = max_len))
model1.add(Bidirectional(LSTM(hidden_units)))
model1.add(Dropout(0.2))
model1.add(Dense(128, activation='tanh'))
model1.add(Dropout(0.2))
model1.add(Dense(64, activation='tanh'))
model1.add(Dropout(0.2))
model1.add(Dense(1, activation='sigmoid'))

model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 31, 100)           1806000   
_________________________________________________________________
bidirectional_1 (Bidirection (None, 128)               84480     
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512     
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256      
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 65        
=================================================================
Total params: 1,915,313
Trainable params: 1,915,313
Non-trainable params: 0
_________________________________________________________________
```

In [35]:
```
model1.compile( loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(0.00001),
    metrics=['accuracy', 'Precision', 'Recall'])
```

In [36]:

```
history1 = model1.fit(train_pad, train_labels,epochs=50,validation_split=
0.2, callbacks = call_backs)
```

```
Epoch 1/50
143/143 [==============================] - 15s 67ms/step - loss: 0.692
8 - accuracy: 0.5131 - precision: 0.5290 - recall: 0.2199 - val_loss:
0.6929 - val_accuracy: 0.4982 - val_precision: 0.5517 - val_recall: 0.
0278
Epoch 2/50
143/143 [==============================] - 9s 61ms/step - loss: 0.6924
- accuracy: 0.5171 - precision: 0.5331 - recall: 0.2581 - val_loss: 0.
6924 - val_accuracy: 0.5158 - val_precision: 0.6075 - val_recall: 0.11
28
Epoch 3/50
143/143 [==============================] - 9s 60ms/step - loss: 0.6913
- accuracy: 0.5414 - precision: 0.5771 - recall: 0.3025 - val_loss: 0.
6918 - val_accuracy: 0.5280 - val_precision: 0.6350 - val_recall: 0.15
10
Epoch 4/50
143/143 [==============================] - 9s 63ms/step - loss: 0.6911
- accuracy: 0.5598 - precision: 0.5929 - recall: 0.3753 - val_loss: 0.
6911 - val_accuracy: 0.5578 - val_precision: 0.6381 - val_recall: 0.28
47
Epoch 5/50
143/143 [==============================] - 9s 60ms/step - loss: 0.6899
- accuracy: 0.5710 - precision: 0.6087 - recall: 0.3920 - val_loss: 0.
6901 - val_accuracy: 0.5884 - val_precision: 0.6688 - val_recall: 0.36
46
Epoch 6/50
143/143 [==============================] - 9s 61ms/step - loss: 0.6883
- accuracy: 0.6102 - precision: 0.6604 - recall: 0.4500 - val_loss: 0.
6889 - val_accuracy: 0.6060 - val_precision: 0.6831 - val_recall: 0.40
80
Epoch 7/50
143/143 [==============================] - 9s 66ms/step - loss: 0.6866
- accuracy: 0.6286 - precision: 0.6734 - recall: 0.4960 - val_loss: 0.
6872 - val_accuracy: 0.6200 - val_precision: 0.6830 - val_recall: 0.46
01
Epoch 8/50
143/143 [==============================] - 9s 61ms/step - loss: 0.6841
- accuracy: 0.6445 - precision: 0.6779 - recall: 0.5478 - val_loss: 0.
6846 - val_accuracy: 0.6349 - val_precision: 0.6853 - val_recall: 0.51
04
```

```
Epoch 9/50
143/143 [==============================] - 9s 61ms/step - loss: 0.6805
- accuracy: 0.6669 - precision: 0.7406 - recall: 0.5114 - val_loss: 0.
6811 - val_accuracy: 0.6497 - val_precision: 0.7037 - val_recall: 0.52
78
Epoch 10/50
143/143 [==============================] - 9s 62ms/step - loss: 0.6750
- accuracy: 0.6875 - precision: 0.7389 - recall: 0.5777 - val_loss: 0.
6757 - val_accuracy: 0.6620 - val_precision: 0.6947 - val_recall: 0.58
85
Epoch 11/50
143/143 [==============================] - 9s 66ms/step - loss: 0.6667
- accuracy: 0.7131 - precision: 0.7474 - recall: 0.6418 - val_loss: 0.
6674 - val_accuracy: 0.6690 - val_precision: 0.7071 - val_recall: 0.58
68
Epoch 12/50
143/143 [==============================] - 9s 61ms/step - loss: 0.6542
- accuracy: 0.7216 - precision: 0.7799 - recall: 0.6159 - val_loss: 0.
6553 - val_accuracy: 0.6813 - val_precision: 0.7112 - val_recall: 0.61
98
Epoch 13/50
143/143 [==============================] - 9s 62ms/step - loss: 0.6355
- accuracy: 0.7416 - precision: 0.7833 - recall: 0.6664 - val_loss: 0.
6378 - val_accuracy: 0.6944 - val_precision: 0.7052 - val_recall: 0.67
71
Epoch 14/50
143/143 [==============================] - 9s 63ms/step - loss: 0.6070
- accuracy: 0.7659 - precision: 0.8006 - recall: 0.7068 - val_loss: 0.
6130 - val_accuracy: 0.7145 - val_precision: 0.7256 - val_recall: 0.69
79
Epoch 15/50
143/143 [==============================] - 9s 66ms/step - loss: 0.5669
- accuracy: 0.7790 - precision: 0.7983 - recall: 0.7454 - val_loss: 0.
5825 - val_accuracy: 0.7312 - val_precision: 0.7514 - val_recall: 0.69
79
Epoch 16/50
143/143 [==============================] - 8s 58ms/step - loss: 0.5195
- accuracy: 0.8003 - precision: 0.8271 - recall: 0.7581 - val_loss: 0.
5537 - val_accuracy: 0.7408 - val_precision: 0.7465 - val_recall: 0.73
61
Epoch 17/50
```

```
143/143 [==============================] - 8s 59ms/step - loss: 0.4757
- accuracy: 0.8167 - precision: 0.8387 - recall: 0.7831 - val_loss: 0.
5346 - val_accuracy: 0.7452 - val_precision: 0.7558 - val_recall: 0.73
09
Epoch 18/50
143/143 [==============================] - 9s 64ms/step - loss: 0.4380
- accuracy: 0.8268 - precision: 0.8502 - recall: 0.7924 - val_loss: 0.
5263 - val_accuracy: 0.7434 - val_precision: 0.7444 - val_recall: 0.74
83
Epoch 19/50
143/143 [==============================] - 9s 62ms/step - loss: 0.4112
- accuracy: 0.8382 - precision: 0.8635 - recall: 0.8025 - val_loss: 0.
5240 - val_accuracy: 0.7461 - val_precision: 0.7424 - val_recall: 0.76
04
Epoch 20/50
143/143 [==============================] - 9s 60ms/step - loss: 0.3869
- accuracy: 0.8463 - precision: 0.8689 - recall: 0.8147 - val_loss: 0.
5212 - val_accuracy: 0.7522 - val_precision: 0.7593 - val_recall: 0.74
48
Epoch 21/50
143/143 [==============================] - 8s 59ms/step - loss: 0.3670
- accuracy: 0.8519 - precision: 0.8764 - recall: 0.8187 - val_loss: 0.
5242 - val_accuracy: 0.7592 - val_precision: 0.7911 - val_recall: 0.71
01
Epoch 22/50
143/143 [==============================] - 10s 68ms/step - loss: 0.348
8 - accuracy: 0.8625 - precision: 0.8884 - recall: 0.8284 - val_loss:
0.5277 - val_accuracy: 0.7469 - val_precision: 0.7412 - val_recall: 0.
7656
Epoch 23/50
143/143 [==============================] - 9s 64ms/step - loss: 0.3335
- accuracy: 0.8695 - precision: 0.8904 - recall: 0.8420 - val_loss: 0.
5259 - val_accuracy: 0.7636 - val_precision: 0.7772 - val_recall: 0.74
48
Epoch 24/50
143/143 [==============================] - 9s 61ms/step - loss: 0.3155
- accuracy: 0.8774 - precision: 0.9010 - recall: 0.8472 - val_loss: 0.
5332 - val_accuracy: 0.7496 - val_precision: 0.7491 - val_recall: 0.75
69
Epoch 25/50
143/143 [==============================] - 9s 60ms/step - loss: 0.3036
```

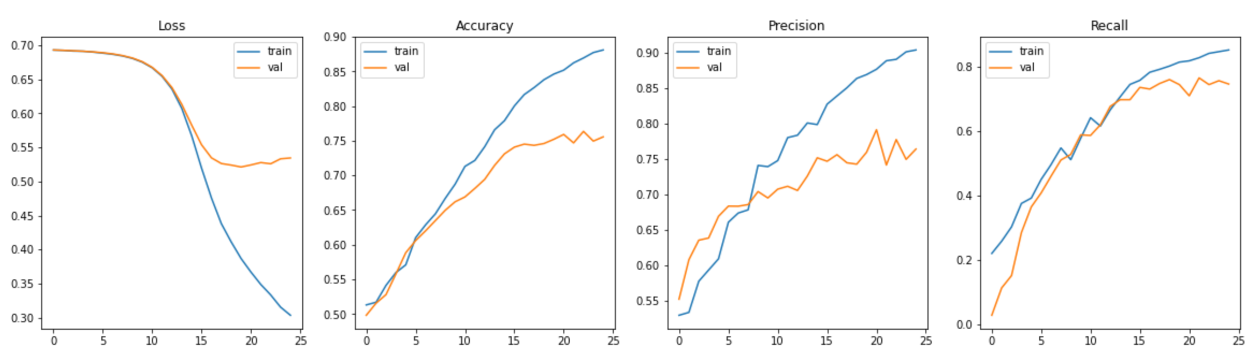
```
- accuracy: 0.8811 - precision: 0.9037 - recall: 0.8525 - val_loss: 0.
5344 - val_accuracy: 0.7557 - val_precision: 0.7638 - val_recall: 0.74
65
Epoch 00025: early stopping
```

In [37]:

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(1, 4, figsize=(20, 5))

axs[0].set_title('Loss')
axs[0].plot(history1.history['loss'], label='train')
axs[0].plot(history1.history['val_loss'], label='val')
axs[0].legend()

axs[1].set_title('Accuracy')
axs[1].plot(history1.history['accuracy'], label='train')
axs[1].plot(history1.history['val_accuracy'], label='val')
axs[1].legend()

axs[2].set_title('Precision')
axs[2].plot(history1.history['precision'], label='train')
axs[2].plot(history1.history['val_precision'], label='val')
axs[2].legend()

axs[3].set_title('Recall')
axs[3].plot(history1.history['recall'], label='train')
axs[3].plot(history1.history['val_recall'], label='val')
axs[3].legend()
```

Out[37]:

```
<matplotlib.legend.Legend at 0x7f192584e050>
```



We see adding some generalization techniques helped the model learn a bit better as the loss wasnt so clearly overfitting to the training set and the accuracy on the validation set did improve some as well. In fact, all of the metrics graphed above are better than the first model. Next, I will change the activation from 'tanh' to 'relu' and see if adjusting that hyperparameter further improves the model.

In [38]:

```
model2 = Sequential()
model2.add(Embedding(all_words, embedding_units, input_length = max_len))
model2.add(Bidirectional(LSTM(hidden_units)))
model2.add(Dropout(0.2))
model2.add(Dense(128, activation='relu'))
model2.add(Dropout(0.2))
model2.add(Dense(64, activation='relu'))
model2.add(Dropout(0.2))
model2.add(Dense(1, activation='sigmoid'))

model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 31, 100)           1806000
_________________________________________________________________
bidirectional_2 (Bidirection (None, 128)               84480
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               16512
_________________________________________________________________
dropout_4 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_5 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 65
=================================================================
Total params: 1,915,313
Trainable params: 1,915,313
Non-trainable params: 0
_________________________________________________________________
```

In [39]:
```
model2.compile( loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(0.00001),
    metrics=['accuracy', 'Precision', 'Recall'])
```

In [40]:
```
history2 = model2.fit(train_pad, train_labels,epochs=50,validation_split=
0.2, callbacks = call_backs)
```

```
Epoch 1/50
143/143 [==============================] - 13s 64ms/step - loss: 0.692
8 - accuracy: 0.5184 - precision: 0.5230 - recall: 0.3951 - val_loss:
0.6930 - val_accuracy: 0.5236 - val_precision: 0.5356 - val_recall: 0.
4184
Epoch 2/50
143/143 [==============================] - 8s 56ms/step - loss: 0.6928
- accuracy: 0.5208 - precision: 0.5244 - recall: 0.4254 - val_loss: 0.
6928 - val_accuracy: 0.5587 - val_precision: 0.5756 - val_recall: 0.47
57
Epoch 3/50
143/143 [==============================] - 9s 62ms/step - loss: 0.6926
- accuracy: 0.5353 - precision: 0.5378 - recall: 0.4868 - val_loss: 0.
6925 - val_accuracy: 0.5718 - val_precision: 0.5589 - val_recall: 0.71
70
Epoch 4/50
143/143 [==============================] - 8s 58ms/step - loss: 0.6920
- accuracy: 0.5515 - precision: 0.5492 - recall: 0.5632 - val_loss: 0.
6921 - val_accuracy: 0.5954 - val_precision: 0.5792 - val_recall: 0.72
40
Epoch 5/50
143/143 [==============================] - 8s 55ms/step - loss: 0.6918
- accuracy: 0.5631 - precision: 0.5635 - recall: 0.5509 - val_loss: 0.
6916 - val_accuracy: 0.6182 - val_precision: 0.6087 - val_recall: 0.68
06
Epoch 6/50
143/143 [==============================] - 8s 55ms/step - loss: 0.6913
- accuracy: 0.5764 - precision: 0.5788 - recall: 0.5549 - val_loss: 0.
6908 - val_accuracy: 0.6270 - val_precision: 0.6147 - val_recall: 0.69
79
Epoch 7/50
143/143 [==============================] - 8s 59ms/step - loss: 0.6907
- accuracy: 0.5777 - precision: 0.5868 - recall: 0.5193 - val_loss: 0.
6899 - val_accuracy: 0.6357 - val_precision: 0.6418 - val_recall: 0.62
85
Epoch 8/50
143/143 [==============================] - 8s 55ms/step - loss: 0.6892
- accuracy: 0.6130 - precision: 0.6240 - recall: 0.5645 - val_loss: 0.
6885 - val_accuracy: 0.6515 - val_precision: 0.6545 - val_recall: 0.65
45
```

```
Epoch 9/50
143/143 [==============================] - 8s 55ms/step - loss: 0.6877
- accuracy: 0.6283 - precision: 0.6503 - recall: 0.5518 - val_loss: 0.
6866 - val_accuracy: 0.6629 - val_precision: 0.6690 - val_recall: 0.65
62
Epoch 10/50
143/143 [==============================] - 8s 56ms/step - loss: 0.6853
- accuracy: 0.6614 - precision: 0.7056 - recall: 0.5514 - val_loss: 0.
6839 - val_accuracy: 0.6751 - val_precision: 0.6909 - val_recall: 0.64
41
Epoch 11/50
143/143 [==============================] - 9s 60ms/step - loss: 0.6822
- accuracy: 0.6684 - precision: 0.7000 - recall: 0.5869 - val_loss: 0.
6803 - val_accuracy: 0.6786 - val_precision: 0.7164 - val_recall: 0.60
07
Epoch 12/50
143/143 [==============================] - 8s 56ms/step - loss: 0.6773
- accuracy: 0.7037 - precision: 0.7368 - recall: 0.6317 - val_loss: 0.
6752 - val_accuracy: 0.6848 - val_precision: 0.7278 - val_recall: 0.59
90
Epoch 13/50
143/143 [==============================] - 8s 56ms/step - loss: 0.6708
- accuracy: 0.7017 - precision: 0.7465 - recall: 0.6089 - val_loss: 0.
6682 - val_accuracy: 0.7014 - val_precision: 0.7345 - val_recall: 0.63
89
Epoch 14/50
143/143 [==============================] - 8s 56ms/step - loss: 0.6611
- accuracy: 0.7297 - precision: 0.7574 - recall: 0.6743 - val_loss: 0.
6584 - val_accuracy: 0.7084 - val_precision: 0.7485 - val_recall: 0.63
54
Epoch 15/50
143/143 [==============================] - 9s 61ms/step - loss: 0.6475
- accuracy: 0.7451 - precision: 0.7771 - recall: 0.6857 - val_loss: 0.
6450 - val_accuracy: 0.7180 - val_precision: 0.7550 - val_recall: 0.65
28
Epoch 16/50
143/143 [==============================] - 8s 54ms/step - loss: 0.6282
- accuracy: 0.7560 - precision: 0.7939 - recall: 0.6901 - val_loss: 0.
6272 - val_accuracy: 0.7303 - val_precision: 0.7463 - val_recall: 0.70
49
Epoch 17/50
```

```
143/143 [==============================] - 8s 57ms/step - loss: 0.6021
- accuracy: 0.7650 - precision: 0.7969 - recall: 0.7098 - val_loss: 0.
6046 - val_accuracy: 0.7285 - val_precision: 0.7628 - val_recall: 0.67
01
Epoch 18/50
143/143 [==============================] - 8s 55ms/step - loss: 0.5706
- accuracy: 0.7806 - precision: 0.8127 - recall: 0.7278 - val_loss: 0.
5791 - val_accuracy: 0.7434 - val_precision: 0.7531 - val_recall: 0.73
09
Epoch 19/50
143/143 [==============================] - 9s 62ms/step - loss: 0.5334
- accuracy: 0.7935 - precision: 0.8201 - recall: 0.7507 - val_loss: 0.
5558 - val_accuracy: 0.7434 - val_precision: 0.7587 - val_recall: 0.72
05
Epoch 20/50
143/143 [==============================] - 8s 55ms/step - loss: 0.4955
- accuracy: 0.8022 - precision: 0.8254 - recall: 0.7656 - val_loss: 0.
5383 - val_accuracy: 0.7443 - val_precision: 0.7545 - val_recall: 0.73
09
Epoch 21/50
143/143 [==============================] - 8s 56ms/step - loss: 0.4682
- accuracy: 0.8066 - precision: 0.8314 - recall: 0.7682 - val_loss: 0.
5300 - val_accuracy: 0.7487 - val_precision: 0.7828 - val_recall: 0.69
44
Epoch 22/50
143/143 [==============================] - 8s 55ms/step - loss: 0.4348
- accuracy: 0.8204 - precision: 0.8458 - recall: 0.7827 - val_loss: 0.
5258 - val_accuracy: 0.7399 - val_precision: 0.7313 - val_recall: 0.76
56
Epoch 23/50
143/143 [==============================] - 9s 61ms/step - loss: 0.4166
- accuracy: 0.8270 - precision: 0.8500 - recall: 0.7932 - val_loss: 0.
5235 - val_accuracy: 0.7382 - val_precision: 0.7351 - val_recall: 0.75
17
Epoch 24/50
143/143 [==============================] - 8s 55ms/step - loss: 0.3953
- accuracy: 0.8384 - precision: 0.8591 - recall: 0.8086 - val_loss: 0.
5231 - val_accuracy: 0.7434 - val_precision: 0.7444 - val_recall: 0.74
83
Epoch 25/50
143/143 [==============================] - 8s 56ms/step - loss: 0.3732
```

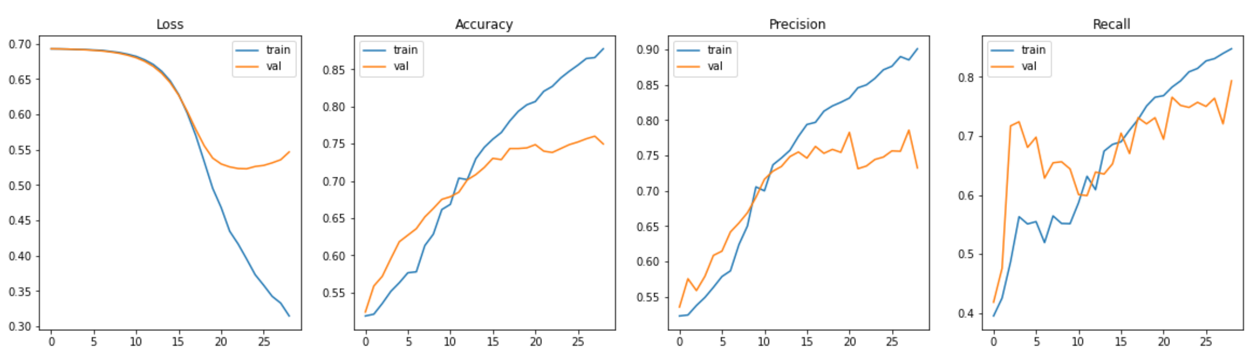
```
- accuracy: 0.8474 - precision: 0.8713 - recall: 0.8143 - val_loss: 0.
5265 - val_accuracy: 0.7487 - val_precision: 0.7479 - val_recall: 0.75
69
Epoch 26/50
143/143 [==============================] - 8s 58ms/step - loss: 0.3583
- accuracy: 0.8555 - precision: 0.8763 - recall: 0.8270 - val_loss: 0.
5279 - val_accuracy: 0.7522 - val_precision: 0.7566 - val_recall: 0.75
00
Epoch 27/50
143/143 [==============================] - 9s 63ms/step - loss: 0.3426
- accuracy: 0.8644 - precision: 0.8900 - recall: 0.8310 - val_loss: 0.
5316 - val_accuracy: 0.7566 - val_precision: 0.7560 - val_recall: 0.76
39
Epoch 28/50
143/143 [==============================] - 8s 57ms/step - loss: 0.3331
- accuracy: 0.8657 - precision: 0.8852 - recall: 0.8398 - val_loss: 0.
5359 - val_accuracy: 0.7601 - val_precision: 0.7860 - val_recall: 0.72
05
Epoch 29/50
143/143 [==============================] - 8s 55ms/step - loss: 0.3148
- accuracy: 0.8776 - precision: 0.9011 - recall: 0.8477 - val_loss: 0.
5470 - val_accuracy: 0.7496 - val_precision: 0.7324 - val_recall: 0.79
34
Epoch 00029: early stopping
```

In [41]:

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(1, 4, figsize=(20, 5))

axs[0].set_title('Loss')
axs[0].plot(history2.history['loss'], label='train')
axs[0].plot(history2.history['val_loss'], label='val')
axs[0].legend()

axs[1].set_title('Accuracy')
axs[1].plot(history2.history['accuracy'], label='train')
axs[1].plot(history2.history['val_accuracy'], label='val')
axs[1].legend()

axs[2].set_title('Precision')
axs[2].plot(history2.history['precision'], label='train')
axs[2].plot(history2.history['val_precision'], label='val')
axs[2].legend()

axs[3].set_title('Recall')
axs[3].plot(history2.history['recall'], label='train')
axs[3].plot(history2.history['val_recall'], label='val')
axs[3].legend()
```

Out[41]:

```
<matplotlib.legend.Legend at 0x7f190423dc90>
```



So, we see the Relu activations didnt really change the performance at all. Finally, looking to see how performance improves with more layers of LSTMs, I will add a couple more and compare the results there.

In [50]:

```
model3 = Sequential()
model3.add(Embedding(all_words, embedding_units, input_length = max_len))
model3.add(Bidirectional(LSTM(hidden_units,return_sequences=True)))
model3.add(Bidirectional(LSTM(hidden_units,return_sequences=True)))
model3.add(Bidirectional(LSTM(hidden_units)))
model3.add(Dropout(0.2))
model3.add(Dense(128, activation='relu'))
model3.add(Dropout(0.2))
model3.add(Dense(64, activation='relu'))
model3.add(Dropout(0.2))
model3.add(Dense(1, activation='sigmoid'))

model3.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 31, 100)           1806000
_________________________________________________________________
bidirectional_8 (Bidirection (None, 31, 128)           84480
_________________________________________________________________
bidirectional_9 (Bidirection (None, 31, 128)           98816
_________________________________________________________________
bidirectional_10 (Bidirectio (None, 128)               98816
_________________________________________________________________
dropout_12 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_15 (Dense)             (None, 128)               16512
_________________________________________________________________
dropout_13 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_16 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_14 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_17 (Dense)             (None, 1)                 65
=================================================================
Total params: 2,112,945
Trainable params: 2,112,945
Non-trainable params: 0
_________________________________________________________________
```

In [51]:

```
model3.compile( loss=tf.keras.losses.BinaryCrossentropy(),
    optimizer=tf.keras.optimizers.Adam(0.00001),
    metrics=['accuracy', 'Precision', 'Recall'])
```

In [52]:
```
history3 = model3.fit(train_pad, train_labels,epochs=50,validation_split=
0.2, callbacks = call_backs)
```

```
Epoch 1/50
143/143 [==============================] - 37s 173ms/step - loss: 0.69
33 - accuracy: 0.4991 - precision: 0.4990 - recall: 0.9794 - val_loss:
0.6931 - val_accuracy: 0.5053 - val_precision: 0.5048 - val_recall: 1.
0000
Epoch 2/50
143/143 [==============================] - 23s 158ms/step - loss: 0.69
32 - accuracy: 0.5002 - precision: 0.4995 - recall: 0.9363 - val_loss:
0.6929 - val_accuracy: 0.5044 - val_precision: 0.5044 - val_recall: 0.
9965
Epoch 3/50
143/143 [==============================] - 20s 143ms/step - loss: 0.69
29 - accuracy: 0.5072 - precision: 0.5032 - recall: 0.9596 - val_loss:
0.6927 - val_accuracy: 0.5044 - val_precision: 0.5044 - val_recall: 0.
9983
Epoch 4/50
143/143 [==============================] - 20s 139ms/step - loss: 0.69
28 - accuracy: 0.5186 - precision: 0.5098 - recall: 0.9153 - val_loss:
0.6924 - val_accuracy: 0.5464 - val_precision: 0.5282 - val_recall: 0.
9427
Epoch 5/50
143/143 [==============================] - 20s 140ms/step - loss: 0.69
26 - accuracy: 0.5420 - precision: 0.5279 - recall: 0.7774 - val_loss:
0.6919 - val_accuracy: 0.5797 - val_precision: 0.5558 - val_recall: 0.
8299
Epoch 6/50
143/143 [==============================] - 20s 137ms/step - loss: 0.69
18 - accuracy: 0.5813 - precision: 0.5561 - recall: 0.7959 - val_loss:
0.6911 - val_accuracy: 0.6305 - val_precision: 0.6250 - val_recall: 0.
6684
Epoch 7/50
143/143 [==============================] - 21s 147ms/step - loss: 0.69
09 - accuracy: 0.6128 - precision: 0.6052 - recall: 0.6440 - val_loss:
0.6897 - val_accuracy: 0.6559 - val_precision: 0.6464 - val_recall: 0.
7014
Epoch 8/50
143/143 [==============================] - 21s 147ms/step - loss: 0.68
85 - accuracy: 0.6529 - precision: 0.6449 - recall: 0.6769 - val_loss:
0.6865 - val_accuracy: 0.6725 - val_precision: 0.6843 - val_recall: 0.
6510
```

```
Epoch 9/50
143/143 [==============================] - 21s 150ms/step - loss: 0.68
31 - accuracy: 0.6875 - precision: 0.6864 - recall: 0.6879 - val_loss:
0.6784 - val_accuracy: 0.6891 - val_precision: 0.7429 - val_recall: 0.
5868
Epoch 10/50
143/143 [==============================] - 23s 162ms/step - loss: 0.66
49 - accuracy: 0.7326 - precision: 0.7501 - recall: 0.6958 - val_loss:
0.6508 - val_accuracy: 0.7058 - val_precision: 0.7166 - val_recall: 0.
6892
Epoch 11/50
143/143 [==============================] - 22s 156ms/step - loss: 0.59
70 - accuracy: 0.7685 - precision: 0.7914 - recall: 0.7278 - val_loss:
0.5658 - val_accuracy: 0.7224 - val_precision: 0.7367 - val_recall: 0.
6997
Epoch 12/50
143/143 [==============================] - 22s 156ms/step - loss: 0.46
73 - accuracy: 0.7970 - precision: 0.8200 - recall: 0.7599 - val_loss:
0.5536 - val_accuracy: 0.7259 - val_precision: 0.7635 - val_recall: 0.
6615
Epoch 13/50
143/143 [==============================] - 22s 154ms/step - loss: 0.40
55 - accuracy: 0.8296 - precision: 0.8609 - recall: 0.7853 - val_loss:
0.5605 - val_accuracy: 0.7312 - val_precision: 0.7582 - val_recall: 0.
6858
Epoch 14/50
143/143 [==============================] - 21s 145ms/step - loss: 0.37
40 - accuracy: 0.8489 - precision: 0.8785 - recall: 0.8090 - val_loss:
0.5650 - val_accuracy: 0.7382 - val_precision: 0.7551 - val_recall: 0.
7118
Epoch 15/50
143/143 [==============================] - 21s 148ms/step - loss: 0.34
84 - accuracy: 0.8640 - precision: 0.8928 - recall: 0.8266 - val_loss:
0.5896 - val_accuracy: 0.7312 - val_precision: 0.7245 - val_recall: 0.
7535
Epoch 16/50
143/143 [==============================] - 21s 144ms/step - loss: 0.32
29 - accuracy: 0.8813 - precision: 0.9118 - recall: 0.8437 - val_loss:
0.5851 - val_accuracy: 0.7513 - val_precision: 0.7664 - val_recall: 0.
7292
Epoch 17/50
```

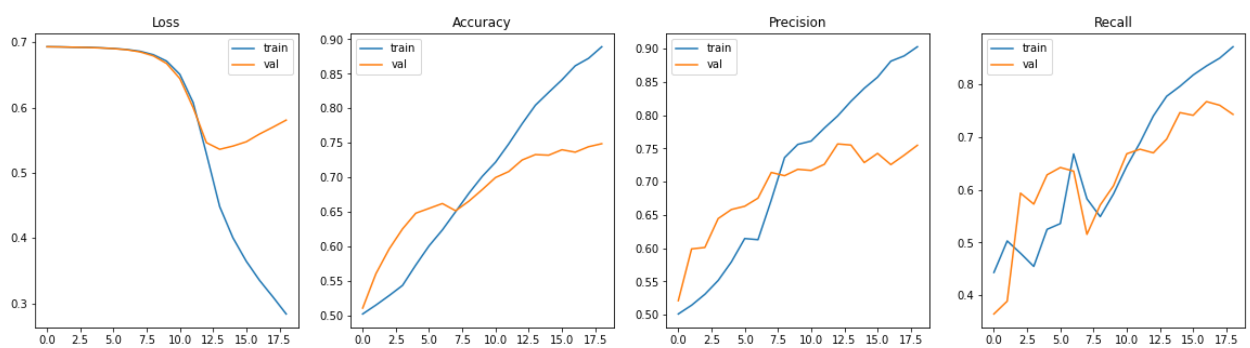
```
143/143 [==============================] - 21s 149ms/step - loss: 0.29
60 - accuracy: 0.8901 - precision: 0.9161 - recall: 0.8582 - val_loss:
0.6282 - val_accuracy: 0.7426 - val_precision: 0.7245 - val_recall: 0.
7899
Epoch 00017: early stopping
```

In [49]:

```python
fig, axs = plt.subplots(1, 4, figsize=(20, 5))

axs[0].set_title('Loss')
axs[0].plot(history3.history['loss'], label='train')
axs[0].plot(history3.history['val_loss'], label='val')
axs[0].legend()

axs[1].set_title('Accuracy')
axs[1].plot(history3.history['accuracy'], label='train')
axs[1].plot(history3.history['val_accuracy'], label='val')
axs[1].legend()

axs[2].set_title('Precision')
axs[2].plot(history3.history['precision'], label='train')
axs[2].plot(history3.history['val_precision'], label='val')
axs[2].legend()

axs[3].set_title('Recall')
axs[3].plot(history3.history['recall'], label='train')
axs[3].plot(history3.history['val_recall'], label='val')
axs[3].legend()
```

Out[49]:

```
<matplotlib.legend.Legend at 0x7f18d1097950>
```



We see adding extra layers of LSTMs does not actually improve the results and appears to make them less consistent on the validation set.

Results and Analysis

We observed similar results across different architectures and hyperparameters. I will use model2 above as the final model for submission as it was arguably the best of the models I compared.

In [53]:
```
preds = model2.predict(test_pad)
```

In [54]:
```
len(preds)
```

Out[54]:
```
3263
```

In [55]:
```
predictions = []

for pred in preds:
    if pred >= 0.5:
        predictions.append(1)
    else:
        predictions.append(0)

predictions[:10]
```

Out[55]:
```
[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
```

In [56]:

```
submission = pd.read_csv("../input/nlp-getting-started/sample_submission.csv")
submission
```

Out[56]:

	id	target
0	0	0
1	2	0
2	3	0
3	9	0
4	11	0
...	...	...
3258	10861	0
3259	10865	0
3260	10868	0
3261	10874	0
3262	10875	0

3263 rows × 2 columns

In [57]:
```
submission['target']=predictions
submission
```

Out[57]:

	id	target
0	0	1
1	2	1
2	3	1
3	9	1
4	11	1
...	...	...
3258	10861	1
3259	10865	1
3260	10868	1
3261	10874	1
3262	10875	1

3263 rows × 2 columns

In [58]:
```
submission.to_csv("submission.csv", index=False)
```

Conclusion

So, I see that over all of the different architectures and parameters my accuracy hovered around 75%. I do think using different methods for cleaning and pre-processing the data may help improve that as well as further analysis study and a better understanding of how best to improve the architecture of the neural network.

References

Along with the documentation for tensorflow, keras, and numpy, I also used these resources found on kaggle:

https://www.kaggle.com/code/msondkar/disaster-tweets-classification-with-an-rnn/notebook (https://www.kaggle.com/code/msondkar/disaster-tweets-classification-with-an-rnn/notebook)

https://www.kaggle.com/code/mattbast/rnn-and-nlp-detect-a-disaster-in-tweets/notebook#Encode-sentences (https://www.kaggle.com/code/mattbast/rnn-and-nlp-detect-a-disaster-in-tweets/notebook#Encode-sentences)