In [5]:

import seaborn as sns

import matplotlib.pyplot as plt

from urllib.parse import urlparse

from nltk.corpus import stopwords

import re

import nltk

from wordcloud import WordCloud

nltk.download("stopwords")

from sklearn.model\_selection import train\_test\_split, RandomizedSearchCV, RepeatedStratifiedKFold

from sklearn.ensemble import RandomForestClassifier

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from sklearn.metrics import confusion\_matrix, accuracy\_score, roc\_curve, roc\_auc\_score, recall\_score, f1\_score

from lightgbm import LGBMClassifier

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, GRU, Embedding

from tensorflow.keras.optimizers import Adam,SGD,RMSprop

import optuna

import time

import warnings

warnings.filterwarnings('ignore')

[nltk\_data] Downloading package stopwords to

[nltk\_data] C:\Users\jkong\AppData\Roaming\nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

In [7]:

import pandas as pd

import numpy as np

In [11]:

data=pd.read\_json('Sarcasm\_Headlines\_Dataset\_v2.json', lines=True)

In [12]:

data.head()

Out[12]:

|  | **is\_sarcastic** | **headline** | **article\_link** |
| --- | --- | --- | --- |
| **0** | 1 | thirtysomething scientists unveil doomsday clo... | https://www.theonion.com/thirtysomething-scien... |
| **1** | 0 | dem rep. totally nails why congress is falling... | https://www.huffingtonpost.com/entry/donna-edw... |
| **2** | 0 | eat your veggies: 9 deliciously different recipes | https://www.huffingtonpost.com/entry/eat-your-... |
| **3** | 1 | inclement weather prevents liar from getting t... | https://local.theonion.com/inclement-weather-p... |
| **4** | 1 | mother comes pretty close to using word 'strea... | https://www.theonion.com/mother-comes-pretty-c... |

In [13]:

print('Null Values\n', data.isnull().sum())

print('\nData Shape: ', data.shape)

Null Values

is\_sarcastic 0

headline 0

article\_link 0

dtype: int64

Data Shape: (28619, 3)

In [14]:

def find\_num\_words(data):

wordlist=[]

splitting\_list=data.str.split()

for i in range(len(splitting\_list)):

for j in range(len(splitting\_list[i])):

wordlist.append(splitting\_list[i][j])

wordset=set(wordlist)

print(len(wordset), len(wordlist))

find\_num\_words(data.headline)

38234 287620

In [16]:

print(data.is\_sarcastic.value\_counts(), '\n')

fig, ax=plt.subplots(1,2, figsize=(19, 5))

f1=sns.countplot(data.is\_sarcastic, ax=ax[0]);

f1.set\_title('Count of real and fake data')

f1.set\_ylabel('Count')

f1.set\_xlabel('Target')

f2=plt.pie(data['is\_sarcastic'].value\_counts().values, explode=[0,0], labels=data.is\_sarcastic.value\_counts().index, autopct='%1.1f%%')

fig.show()

0 14985

1 13634

Name: is\_sarcastic, dtype: int64



In [19]:

def get\_netloc(url):

scheme, netloc, path, params, query, fragment=urlparse(url)

return netloc

link=[]

for i in range(len(data.article\_link)):

link.append(get\_netloc(data.article\_link[i]))

data['link']=link

In [20]:

data.link = data["link"].str.replace("local.theonion.com","www.theonion.com").replace("politics.theonion.com","www.theonion.com").replace("entertainment.theonion.com","www.theonion.com").replace("sports.theonion.com","www.theonion.com").replace("ogn.theonion.com","www.theonion.com")

data.link = data.link.str.replace('www.huffingtonpost.comhttp:','www.huffingtonpost.com').replace('www.huffingtonpost.comhttps:','www.huffingtonpost.com')

data.link.value\_counts()

Out[20]:

www.huffingtonpost.com 14985

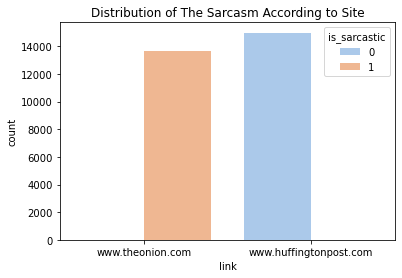
www.theonion.com 13634

Name: link, dtype: int64

In [21]:

ax=sns.countplot(x='link', hue='is\_sarcastic', data=data, palette='pastel')

plt.title('Distribution of The Sarcasm According to Site');



In [22]:

def remove\_punctuations(text):

return re.sub('\[[^]]\*\]', '', text)

#Removing special characters

def remove\_specialchars(text):

return re.sub("[^a-zA-Z]"," ",text)

#Removal of stopwords and lemmatization

def remove\_stopwords\_and\_lemmatization(text):

final\_text = []

text = text.lower()

text = nltk.word\_tokenize(text)

for word in text:

if word not in set(stopwords.words('english')):

lemma = nltk.WordNetLemmatizer()

word = lemma.lemmatize(word)

final\_text.append(word)

return " ".join(final\_text)

def cleaning(text):

text = remove\_punctuations(text)

text = remove\_specialchars(text)

text = remove\_stopwords\_and\_lemmatization(text)

return text

data['headline']=data['headline'].apply(cleaning)

In [23]:

text = " ".join(review for review in data.headline)

print ("There are {} words in the combination of all review.".format(len(text)))

# Generate a word cloud image

wordcloud = WordCloud(background\_color="white").generate(text)

# Display the generated image:

# the matplotlib way:

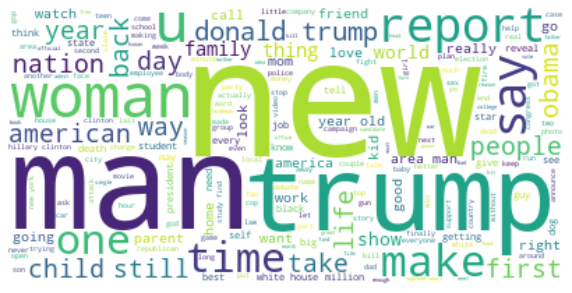
plt.figure(figsize=(10,10))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.show()

There are 1417022 words in the combination of all review.



In [24]:

data.head()

Out[24]:

|  | **is\_sarcastic** | **headline** | **article\_link** | **link** |
| --- | --- | --- | --- | --- |
| **0** | 1 | thirtysomething scientist unveil doomsday cloc... | https://www.theonion.com/thirtysomething-scien... | www.theonion.com |
| **1** | 0 | dem rep totally nail congress falling short ge... | https://www.huffingtonpost.com/entry/donna-edw... | www.huffingtonpost.com |
| **2** | 0 | eat veggie deliciously different recipe | https://www.huffingtonpost.com/entry/eat-your-... | www.huffingtonpost.com |
| **3** | 1 | inclement weather prevents liar getting work | https://local.theonion.com/inclement-weather-p... | www.theonion.com |
| **4** | 1 | mother come pretty close using word streaming ... | https://www.theonion.com/mother-comes-pretty-c... | www.theonion.com |

In [25]:

sentences=data['headline'].values.tolist()

target=data['is\_sarcastic'].values.tolist()

In [26]:

print(sentences[:2])

print(target[:2])

['thirtysomething scientist unveil doomsday clock hair loss', 'dem rep totally nail congress falling short gender racial equality']

[1, 0]

In [27]:

num\_words=1000

tokenizer=Tokenizer(num\_words=num\_words)

tokenizer.fit\_on\_texts(sentences)

tokens=tokenizer.texts\_to\_sequences(sentences)

numTokens=[len(token) for token in tokens]

numTokens=np.array(numTokens)

print('Tokens mean', np.mean(numTokens))

print('Max', np.max(numTokens))

print('Argmax', np.argmax(numTokens))

Tokens mean 3.7437017366085468

Max 31

Argmax 7302

In [28]:

sentences[7302]

Out[28]:

'hot wheel ranked number one toy rolling ramp knocking domino send marble funnel dropping onto teeter totter yank string causing pulley system raise wooden block propelling series twine roller unwind spring launching tennis ball across room inching tire slope hit power switch activating table fan blow toy ship nail attached across kiddie pool popping water balloon fill cup weighing lever force basketball track nudging broomstick axis rotate allowing golf ball roll sideways coffee mug tumble row hardcover book handle catch hook attached lever cause wooden mallet slam serving spoon catapulting small ball cup attached ribbon lazy susan spin push battery incline plane tipping salt shaker season omelet'

In [30]:

max\_tokens=int(np.mean(numTokens)+2\*np.std(numTokens))

print('Calculated maximum number of tokens :', max\_tokens)

print('What percentage of the data fits this average? :', np.sum(numTokens < max\_tokens)/len(numTokens))

Calculated maximum number of tokens : 7

What percentage of the data fits this average? : 0.9099898668716587

In [32]:

padding\_data=pad\_sequences(tokens, maxlen=max\_tokens)

print(padding\_data.shape)

print(len(target))

(28619, 7)

28619

In [34]:

X\_train, X\_test, y\_train, y\_test=train\_test\_split(padding\_data, data.is\_sarcastic, random\_state=1, test\_size=0.15, stratify=data.is\_sarcastic)

X\_train, X\_val, y\_train, y\_val=train\_test\_split(X\_train, y\_train, random\_state=1, test\_size=0.1, stratify=y\_train)

In [36]:

def objective(trial):

L2=trial.suggest\_float('1', 1e-5, 1e-2, log=True)

EMBEDDING\_SIZE=trial.suggest\_int('embedding\_size', 10, 50, step=10)

BATCH\_SIZE=trial.suggest\_int('batch\_size', 16, 64, step=8)

EPOCHS=trial.suggest\_int('epochs', 10, 30, step=10)

LR=trial.suggest\_float('learning\_rate', 1e-5, 1e-2, log=True)

OPT=trial.suggest\_categorical('optimizer', [Adam, SGD, RMSprop])

model=Sequential()

model.add(Embedding(input\_dim=num\_words,

output\_dim=EMBEDDING\_SIZE,

input\_length=max\_tokens,

name='embedding\_layer'))

model.add(GRU(units=128, return\_sequences=True, kernel\_regularizer = tf.keras.regularizers.l2(l=L2)))

model.add(GRU(units=64,return\_sequences=True, kernel\_regularizer = tf.keras.regularizers.l2(l=L2)))

model.add(GRU(units=32))

model.add(Dense(16, activation="relu"))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer=OPT(lr=LR), loss='binary\_crossentropy', metrics=['accuracy'])

H = model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=EPOCHS, batch\_size=BATCH\_SIZE)

val\_loss, val\_acc = model.evaluate(X\_test,y\_test)

return val\_loss

study = optuna.create\_study()

start = time.time()

study.optimize(objective, n\_trials=5)

end = time.time()

[I 2022-02-02 19:51:34,158] A new study created in memory with name: no-name-b54d278e-0346-441b-a3a3-8783d0ebf3ec

Epoch 1/30

913/913 [==============================] - 13s 10ms/step - loss: 0.7552 - accuracy: 0.5213 - val\_loss: 0.7548 - val\_accuracy: 0.5236

Epoch 2/30

913/913 [==============================] - 9s 9ms/step - loss: 0.7547 - accuracy: 0.5236 - val\_loss: 0.7545 - val\_accuracy: 0.5236

Epoch 3/30

913/913 [==============================] - 8s 9ms/step - loss: 0.7544 - accuracy: 0.5236 - val\_loss: 0.7543 - val\_accuracy: 0.5236

Epoch 4/30

913/913 [==============================] - 9s 9ms/step - loss: 0.7543 - accuracy: 0.5236 - val\_loss: 0.7542 - val\_accuracy: 0.5236

Epoch 5/30

913/913 [==============================] - 9s 10ms/step - loss: 0.7542 - accuracy: 0.5236 - val\_loss: 0.7542 - val\_accuracy: 0.5236

Epoch 6/30

913/913 [==============================] - 9s 10ms/step - loss: 0.7542 - accuracy: 0.5236 - val\_loss: 0.7541 - val\_accuracy: 0.5236

Epoch 7/30

913/913 [==============================] - 10s 10ms/step - loss: 0.7541 - accuracy: 0.5236 - val\_loss: 0.7541 - val\_accuracy: 0.5236

Epoch 8/30

913/913 [==============================] - 10s 11ms/step - loss: 0.7541 - accuracy: 0.5236 - val\_loss: 0.7540 - val\_accuracy: 0.5236

Epoch 9/30

913/913 [==============================] - 10s 11ms/step - loss: 0.7540 - accuracy: 0.5236 - val\_loss: 0.7540 - val\_accuracy: 0.5236

Epoch 10/30

913/913 [==============================] - 10s 11ms/step - loss: 0.7540 - accuracy: 0.5236 - val\_loss: 0.7540 - val\_accuracy: 0.5236

Epoch 11/30

913/913 [==============================] - 10s 11ms/step - loss: 0.7540 - accuracy: 0.5236 - val\_loss: 0.7539 - val\_accuracy: 0.5236

Epoch 12/30

913/913 [==============================] - 10s 11ms/step - loss: 0.7539 - accuracy: 0.5236 - val\_loss: 0.7539 - val\_accuracy: 0.5236

Epoch 13/30

913/913 [==============================] - 10s 11ms/step - loss: 0.7539 - accuracy: 0.5236 - val\_loss: 0.7538 - val\_accuracy: 0.5236

Epoch 14/30

913/913 [==============================] - 10s 11ms/step - loss: 0.7538 - accuracy: 0.5236 - val\_loss: 0.7538 - val\_accuracy: 0.5236

Epoch 15/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7538 - accuracy: 0.5236 - val\_loss: 0.7538 - val\_accuracy: 0.5236

Epoch 16/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7538 - accuracy: 0.5236 - val\_loss: 0.7537 - val\_accuracy: 0.5236

Epoch 17/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7537 - accuracy: 0.5236 - val\_loss: 0.7537 - val\_accuracy: 0.5236

Epoch 18/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7537 - accuracy: 0.5236 - val\_loss: 0.7536 - val\_accuracy: 0.5236

Epoch 19/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7537 - accuracy: 0.5236 - val\_loss: 0.7536 - val\_accuracy: 0.5236

Epoch 20/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7536 - accuracy: 0.5236 - val\_loss: 0.7536 - val\_accuracy: 0.5236

Epoch 21/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7536 - accuracy: 0.5236 - val\_loss: 0.7535 - val\_accuracy: 0.5236

Epoch 22/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7535 - accuracy: 0.5236 - val\_loss: 0.7535 - val\_accuracy: 0.5236

Epoch 23/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7535 - accuracy: 0.5236 - val\_loss: 0.7534 - val\_accuracy: 0.5236

Epoch 24/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7535 - accuracy: 0.5236 - val\_loss: 0.7534 - val\_accuracy: 0.5236

Epoch 25/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7534 - accuracy: 0.5236 - val\_loss: 0.7534 - val\_accuracy: 0.5236

Epoch 26/30

913/913 [==============================] - 12s 13ms/step - loss: 0.7534 - accuracy: 0.5236 - val\_loss: 0.7533 - val\_accuracy: 0.5236

Epoch 27/30

913/913 [==============================] - 12s 13ms/step - loss: 0.7533 - accuracy: 0.5236 - val\_loss: 0.7533 - val\_accuracy: 0.5236

Epoch 28/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7533 - accuracy: 0.5236 - val\_loss: 0.7532 - val\_accuracy: 0.5236

Epoch 29/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7533 - accuracy: 0.5236 - val\_loss: 0.7532 - val\_accuracy: 0.5236

Epoch 30/30

913/913 [==============================] - 11s 12ms/step - loss: 0.7532 - accuracy: 0.5236 - val\_loss: 0.7531 - val\_accuracy: 0.5236

135/135 [==============================] - 1s 4ms/step - loss: 0.7531 - accuracy: 0.5236

[I 2022-02-02 19:56:50,040] Trial 0 finished with value: 0.7531344294548035 and parameters: {'1': 0.0002596327568419324, 'embedding\_size': 50, 'batch\_size': 24, 'epochs': 30, 'learning\_rate': 0.00041085377781099647, 'optimizer': <class 'keras.optimizer\_v2.gradient\_descent.SGD'>}. Best is trial 0 with value: 0.7531344294548035.

Epoch 1/20

1369/1369 [==============================] - 20s 12ms/step - loss: 0.8108 - accuracy: 0.5229 - val\_loss: 0.8103 - val\_accuracy: 0.5236

Epoch 2/20

1369/1369 [==============================] - 15s 11ms/step - loss: 0.8100 - accuracy: 0.5236 - val\_loss: 0.8097 - val\_accuracy: 0.5236

Epoch 3/20

1369/1369 [==============================] - 15s 11ms/step - loss: 0.8095 - accuracy: 0.5236 - val\_loss: 0.8092 - val\_accuracy: 0.5236

Epoch 4/20

1369/1369 [==============================] - 16s 11ms/step - loss: 0.8090 - accuracy: 0.5236 - val\_loss: 0.8088 - val\_accuracy: 0.5236

Epoch 5/20

1369/1369 [==============================] - 16s 12ms/step - loss: 0.8087 - accuracy: 0.5236 - val\_loss: 0.8085 - val\_accuracy: 0.5236

Epoch 6/20

1369/1369 [==============================] - 17s 12ms/step - loss: 0.8083 - accuracy: 0.5236 - val\_loss: 0.8081 - val\_accuracy: 0.5236

Epoch 7/20

1369/1369 [==============================] - 17s 12ms/step - loss: 0.8080 - accuracy: 0.5236 - val\_loss: 0.8078 - val\_accuracy: 0.5236

Epoch 8/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8076 - accuracy: 0.5236 - val\_loss: 0.8074 - val\_accuracy: 0.5236

Epoch 9/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8073 - accuracy: 0.5236 - val\_loss: 0.8071 - val\_accuracy: 0.5236

Epoch 10/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8070 - accuracy: 0.5236 - val\_loss: 0.8068 - val\_accuracy: 0.5236

Epoch 11/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8067 - accuracy: 0.5236 - val\_loss: 0.8064 - val\_accuracy: 0.5236

Epoch 12/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8063 - accuracy: 0.5236 - val\_loss: 0.8061 - val\_accuracy: 0.5236

Epoch 13/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8060 - accuracy: 0.5236 - val\_loss: 0.8058 - val\_accuracy: 0.5236

Epoch 14/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8057 - accuracy: 0.5236 - val\_loss: 0.8055 - val\_accuracy: 0.5236

Epoch 15/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8053 - accuracy: 0.5236 - val\_loss: 0.8051 - val\_accuracy: 0.5236

Epoch 16/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8050 - accuracy: 0.5236 - val\_loss: 0.8048 - val\_accuracy: 0.5236

Epoch 17/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8047 - accuracy: 0.5236 - val\_loss: 0.8045 - val\_accuracy: 0.5236

Epoch 18/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8044 - accuracy: 0.5236 - val\_loss: 0.8042 - val\_accuracy: 0.5236

Epoch 19/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8040 - accuracy: 0.5236 - val\_loss: 0.8038 - val\_accuracy: 0.5236

Epoch 20/20

1369/1369 [==============================] - 14s 10ms/step - loss: 0.8037 - accuracy: 0.5236 - val\_loss: 0.8035 - val\_accuracy: 0.5236

135/135 [==============================] - 0s 3ms/step - loss: 0.8035 - accuracy: 0.5236

[I 2022-02-02 20:01:45,659] Trial 1 finished with value: 0.8035165667533875 and parameters: {'1': 0.0006825726855771489, 'embedding\_size': 10, 'batch\_size': 16, 'epochs': 20, 'learning\_rate': 0.000760926844213793, 'optimizer': <class 'keras.optimizer\_v2.gradient\_descent.SGD'>}. Best is trial 0 with value: 0.7531344294548035.

Epoch 1/30

913/913 [==============================] - 14s 11ms/step - loss: 0.5869 - accuracy: 0.6825 - val\_loss: 0.5636 - val\_accuracy: 0.6971

Epoch 2/30

913/913 [==============================] - 10s 11ms/step - loss: 0.5222 - accuracy: 0.7376 - val\_loss: 0.5241 - val\_accuracy: 0.7296

Epoch 3/30

913/913 [==============================] - 10s 11ms/step - loss: 0.5099 - accuracy: 0.7431 - val\_loss: 0.5241 - val\_accuracy: 0.7296

Epoch 4/30

913/913 [==============================] - 10s 11ms/step - loss: 0.5039 - accuracy: 0.7482 - val\_loss: 0.5234 - val\_accuracy: 0.7328

Epoch 5/30

913/913 [==============================] - 10s 11ms/step - loss: 0.5010 - accuracy: 0.7501 - val\_loss: 0.5208 - val\_accuracy: 0.7337

Epoch 6/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4991 - accuracy: 0.7524 - val\_loss: 0.5192 - val\_accuracy: 0.7402

Epoch 7/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4963 - accuracy: 0.7549 - val\_loss: 0.5277 - val\_accuracy: 0.7349

Epoch 8/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4930 - accuracy: 0.7549 - val\_loss: 0.5687 - val\_accuracy: 0.7316

Epoch 9/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4881 - accuracy: 0.7595 - val\_loss: 0.5157 - val\_accuracy: 0.7370

Epoch 10/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4839 - accuracy: 0.7647 - val\_loss: 0.5219 - val\_accuracy: 0.7370

Epoch 11/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4805 - accuracy: 0.7648 - val\_loss: 0.5212 - val\_accuracy: 0.7431

Epoch 12/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4770 - accuracy: 0.7677 - val\_loss: 0.5329 - val\_accuracy: 0.7370

Epoch 13/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4752 - accuracy: 0.7677 - val\_loss: 0.5237 - val\_accuracy: 0.7390

Epoch 14/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4736 - accuracy: 0.7715 - val\_loss: 0.5283 - val\_accuracy: 0.7452

Epoch 15/30

913/913 [==============================] - 11s 12ms/step - loss: 0.4710 - accuracy: 0.7718 - val\_loss: 0.5137 - val\_accuracy: 0.7427

Epoch 16/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4705 - accuracy: 0.7739 - val\_loss: 0.5289 - val\_accuracy: 0.7349

Epoch 17/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4682 - accuracy: 0.7723 - val\_loss: 0.5251 - val\_accuracy: 0.7333

Epoch 18/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4644 - accuracy: 0.7759 - val\_loss: 0.5167 - val\_accuracy: 0.7423

Epoch 19/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4615 - accuracy: 0.7778 - val\_loss: 0.5441 - val\_accuracy: 0.7320

Epoch 20/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4589 - accuracy: 0.7806 - val\_loss: 0.5274 - val\_accuracy: 0.7296

Epoch 21/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4572 - accuracy: 0.7824 - val\_loss: 0.5276 - val\_accuracy: 0.7365

Epoch 22/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4554 - accuracy: 0.7841 - val\_loss: 0.5370 - val\_accuracy: 0.7320

Epoch 23/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4534 - accuracy: 0.7850 - val\_loss: 0.5480 - val\_accuracy: 0.7349

Epoch 24/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4505 - accuracy: 0.7879 - val\_loss: 0.5346 - val\_accuracy: 0.7287

Epoch 25/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4495 - accuracy: 0.7880 - val\_loss: 0.5371 - val\_accuracy: 0.7300

Epoch 26/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4452 - accuracy: 0.7894 - val\_loss: 0.5505 - val\_accuracy: 0.7287

Epoch 27/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4447 - accuracy: 0.7887 - val\_loss: 0.5319 - val\_accuracy: 0.7287

Epoch 28/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4415 - accuracy: 0.7929 - val\_loss: 0.5624 - val\_accuracy: 0.7263

Epoch 29/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4377 - accuracy: 0.7951 - val\_loss: 0.5660 - val\_accuracy: 0.7291

Epoch 30/30

913/913 [==============================] - 10s 11ms/step - loss: 0.4362 - accuracy: 0.7947 - val\_loss: 0.5679 - val\_accuracy: 0.7304

135/135 [==============================] - 0s 3ms/step - loss: 0.5817 - accuracy: 0.7240

[I 2022-02-02 20:06:54,556] Trial 2 finished with value: 0.5817269682884216 and parameters: {'1': 0.0005422882272544079, 'embedding\_size': 30, 'batch\_size': 24, 'epochs': 30, 'learning\_rate': 0.0010289068259304612, 'optimizer': <class 'keras.optimizer\_v2.rmsprop.RMSprop'>}. Best is trial 2 with value: 0.5817269682884216.

Epoch 1/10

548/548 [==============================] - 12s 14ms/step - loss: 0.6071 - accuracy: 0.6609 - val\_loss: 0.5589 - val\_accuracy: 0.7193

Epoch 2/10

548/548 [==============================] - 7s 12ms/step - loss: 0.5252 - accuracy: 0.7356 - val\_loss: 0.5314 - val\_accuracy: 0.7222

Epoch 3/10

548/548 [==============================] - 7s 12ms/step - loss: 0.5128 - accuracy: 0.7434 - val\_loss: 0.5249 - val\_accuracy: 0.7205

Epoch 4/10

548/548 [==============================] - 7s 12ms/step - loss: 0.5068 - accuracy: 0.7484 - val\_loss: 0.5302 - val\_accuracy: 0.7283

Epoch 5/10

548/548 [==============================] - 7s 13ms/step - loss: 0.5040 - accuracy: 0.7498 - val\_loss: 0.5267 - val\_accuracy: 0.7345

Epoch 6/10

548/548 [==============================] - 7s 13ms/step - loss: 0.5011 - accuracy: 0.7509 - val\_loss: 0.5484 - val\_accuracy: 0.7271

Epoch 7/10

548/548 [==============================] - 7s 12ms/step - loss: 0.4986 - accuracy: 0.7500 - val\_loss: 0.5455 - val\_accuracy: 0.7382

Epoch 8/10

548/548 [==============================] - 7s 12ms/step - loss: 0.4976 - accuracy: 0.7536 - val\_loss: 0.5211 - val\_accuracy: 0.7341

Epoch 9/10

548/548 [==============================] - 7s 12ms/step - loss: 0.4959 - accuracy: 0.7548 - val\_loss: 0.5352 - val\_accuracy: 0.7230

Epoch 10/10

548/548 [==============================] - 7s 12ms/step - loss: 0.4957 - accuracy: 0.7530 - val\_loss: 0.5190 - val\_accuracy: 0.7324

135/135 [==============================] - 0s 3ms/step - loss: 0.5217 - accuracy: 0.7286

[I 2022-02-02 20:08:08,130] Trial 3 finished with value: 0.5217283368110657 and parameters: {'1': 0.0014116120766505167, 'embedding\_size': 10, 'batch\_size': 40, 'epochs': 10, 'learning\_rate': 0.0031928026186228923, 'optimizer': <class 'keras.optimizer\_v2.rmsprop.RMSprop'>}. Best is trial 3 with value: 0.5217283368110657.

Epoch 1/30

913/913 [==============================] - 14s 11ms/step - loss: 1.3844 - accuracy: 0.5220 - val\_loss: 1.3836 - val\_accuracy: 0.5236

Epoch 2/30

913/913 [==============================] - 9s 10ms/step - loss: 1.3829 - accuracy: 0.5236 - val\_loss: 1.3822 - val\_accuracy: 0.5236

Epoch 3/30

913/913 [==============================] - 9s 10ms/step - loss: 1.3815 - accuracy: 0.5236 - val\_loss: 1.3809 - val\_accuracy: 0.5236

Epoch 4/30

913/913 [==============================] - 10s 10ms/step - loss: 1.3802 - accuracy: 0.5236 - val\_loss: 1.3796 - val\_accuracy: 0.5236

Epoch 5/30

913/913 [==============================] - 10s 11ms/step - loss: 1.3790 - accuracy: 0.5236 - val\_loss: 1.3784 - val\_accuracy: 0.5236

Epoch 6/30

913/913 [==============================] - 9s 10ms/step - loss: 1.3778 - accuracy: 0.5236 - val\_loss: 1.3772 - val\_accuracy: 0.5236

Epoch 7/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3766 - accuracy: 0.5236 - val\_loss: 1.3760 - val\_accuracy: 0.5236

Epoch 8/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3754 - accuracy: 0.5236 - val\_loss: 1.3748 - val\_accuracy: 0.5236

Epoch 9/30

913/913 [==============================] - 10s 11ms/step - loss: 1.3742 - accuracy: 0.5236 - val\_loss: 1.3736 - val\_accuracy: 0.5236

Epoch 10/30

913/913 [==============================] - 12s 13ms/step - loss: 1.3730 - accuracy: 0.5236 - val\_loss: 1.3724 - val\_accuracy: 0.5236

Epoch 11/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3719 - accuracy: 0.5236 - val\_loss: 1.3713 - val\_accuracy: 0.5236

Epoch 12/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3707 - accuracy: 0.5236 - val\_loss: 1.3701 - val\_accuracy: 0.5236

Epoch 13/30

913/913 [==============================] - 10s 11ms/step - loss: 1.3695 - accuracy: 0.5236 - val\_loss: 1.3689 - val\_accuracy: 0.5236

Epoch 14/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3684 - accuracy: 0.5236 - val\_loss: 1.3678 - val\_accuracy: 0.5236

Epoch 15/30

913/913 [==============================] - 10s 11ms/step - loss: 1.3672 - accuracy: 0.5236 - val\_loss: 1.3666 - val\_accuracy: 0.5236

Epoch 16/30

913/913 [==============================] - 10s 11ms/step - loss: 1.3661 - accuracy: 0.5236 - val\_loss: 1.3655 - val\_accuracy: 0.5236

Epoch 17/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3650 - accuracy: 0.5236 - val\_loss: 1.3643 - val\_accuracy: 0.5236

Epoch 18/30

913/913 [==============================] - 10s 11ms/step - loss: 1.3638 - accuracy: 0.5236 - val\_loss: 1.3632 - val\_accuracy: 0.5236

Epoch 19/30

913/913 [==============================] - 10s 11ms/step - loss: 1.3627 - accuracy: 0.5236 - val\_loss: 1.3621 - val\_accuracy: 0.5236

Epoch 20/30

913/913 [==============================] - 12s 13ms/step - loss: 1.3615 - accuracy: 0.5236 - val\_loss: 1.3609 - val\_accuracy: 0.5236

Epoch 21/30

913/913 [==============================] - 10s 11ms/step - loss: 1.3604 - accuracy: 0.5236 - val\_loss: 1.3598 - val\_accuracy: 0.5236

Epoch 22/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3593 - accuracy: 0.5236 - val\_loss: 1.3587 - val\_accuracy: 0.5236

Epoch 23/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3582 - accuracy: 0.5236 - val\_loss: 1.3576 - val\_accuracy: 0.5236

Epoch 24/30

913/913 [==============================] - 12s 13ms/step - loss: 1.3570 - accuracy: 0.5236 - val\_loss: 1.3564 - val\_accuracy: 0.5236

Epoch 25/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3559 - accuracy: 0.5236 - val\_loss: 1.3553 - val\_accuracy: 0.5236

Epoch 26/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3548 - accuracy: 0.5236 - val\_loss: 1.3542 - val\_accuracy: 0.5236

Epoch 27/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3537 - accuracy: 0.5236 - val\_loss: 1.3531 - val\_accuracy: 0.5236

Epoch 28/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3525 - accuracy: 0.5236 - val\_loss: 1.3519 - val\_accuracy: 0.5236

Epoch 29/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3514 - accuracy: 0.5236 - val\_loss: 1.3508 - val\_accuracy: 0.5236

Epoch 30/30

913/913 [==============================] - 11s 12ms/step - loss: 1.3503 - accuracy: 0.5236 - val\_loss: 1.3497 - val\_accuracy: 0.5236

135/135 [==============================] - 1s 4ms/step - loss: 1.3497 - accuracy: 0.5236

[I 2022-02-02 20:13:30,475] Trial 4 finished with value: 1.3497391939163208 and parameters: {'1': 0.003316134755131107, 'embedding\_size': 30, 'batch\_size': 24, 'epochs': 30, 'learning\_rate': 0.00013948793904019777, 'optimizer': <class 'keras.optimizer\_v2.gradient\_descent.SGD'>}. Best is trial 3 with value: 0.5217283368110657.

In [37]:

best\_params=study.best\_params

print(best\_params)

print('model took %0.2f seconds to train'%(end-start))

{'1': 0.0014116120766505167, 'embedding\_size': 10, 'batch\_size': 40, 'epochs': 10, 'learning\_rate': 0.0031928026186228923, 'optimizer': <class 'keras.optimizer\_v2.rmsprop.RMSprop'>}

model took 1316.32 seconds to train

In [38]:

EMBEDDING\_SIZE = 40

model = Sequential()

model.add(Embedding(input\_dim = num\_words,

output\_dim=EMBEDDING\_SIZE,

input\_length=max\_tokens,

name='embedding\_layer'))

model.add(GRU(units=128, return\_sequences=True, kernel\_regularizer = tf.keras.regularizers.l2(l=1.367675698620851e-05)))

model.add(GRU(units=64,return\_sequences=True, kernel\_regularizer = tf.keras.regularizers.l2(l=1.367675698620851e-05)))

model.add(GRU(units=32))

#model.add(Dense(16, activation="relu"))

model.add(Dense(1, activation='sigmoid'))

model.summary()

Model: "sequential\_5"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_layer (Embedding) (None, 7, 40) 40000

gru\_15 (GRU) (None, 7, 128) 65280

gru\_16 (GRU) (None, 7, 64) 37248

gru\_17 (GRU) (None, 32) 9408

dense\_10 (Dense) (None, 1) 33

=================================================================

Total params: 151,969

Trainable params: 151,969

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [39]:

model.compile(loss="binary\_crossentropy", optimizer = RMSprop(lr=8.74355057651944e-05), metrics=["accuracy"])

history = model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=30, batch\_size=40)

Epoch 1/30

548/548 [==============================] - 11s 13ms/step - loss: 0.6807 - accuracy: 0.5685 - val\_loss: 0.6398 - val\_accuracy: 0.6387

Epoch 2/30

548/548 [==============================] - 7s 12ms/step - loss: 0.5897 - accuracy: 0.6854 - val\_loss: 0.5718 - val\_accuracy: 0.7065

Epoch 3/30

548/548 [==============================] - 7s 12ms/step - loss: 0.5412 - accuracy: 0.7211 - val\_loss: 0.5434 - val\_accuracy: 0.7185

Epoch 4/30

548/548 [==============================] - 7s 12ms/step - loss: 0.5207 - accuracy: 0.7330 - val\_loss: 0.5342 - val\_accuracy: 0.7246

Epoch 5/30

548/548 [==============================] - 7s 12ms/step - loss: 0.5096 - accuracy: 0.7419 - val\_loss: 0.5275 - val\_accuracy: 0.7333

Epoch 6/30

548/548 [==============================] - 7s 13ms/step - loss: 0.5034 - accuracy: 0.7449 - val\_loss: 0.5262 - val\_accuracy: 0.7337

Epoch 7/30

548/548 [==============================] - 8s 16ms/step - loss: 0.4994 - accuracy: 0.7490 - val\_loss: 0.5223 - val\_accuracy: 0.7370

Epoch 8/30

548/548 [==============================] - 10s 19ms/step - loss: 0.4961 - accuracy: 0.7496 - val\_loss: 0.5197 - val\_accuracy: 0.7394

Epoch 9/30

548/548 [==============================] - 10s 18ms/step - loss: 0.4930 - accuracy: 0.7560 - val\_loss: 0.5212 - val\_accuracy: 0.7415

Epoch 10/30

548/548 [==============================] - 10s 18ms/step - loss: 0.4917 - accuracy: 0.7542 - val\_loss: 0.5192 - val\_accuracy: 0.7394

Epoch 11/30

548/548 [==============================] - 10s 19ms/step - loss: 0.4901 - accuracy: 0.7555 - val\_loss: 0.5166 - val\_accuracy: 0.7415

Epoch 12/30

548/548 [==============================] - 11s 19ms/step - loss: 0.4883 - accuracy: 0.7557 - val\_loss: 0.5178 - val\_accuracy: 0.7328

Epoch 13/30

548/548 [==============================] - 11s 20ms/step - loss: 0.4870 - accuracy: 0.7558 - val\_loss: 0.5169 - val\_accuracy: 0.7402

Epoch 14/30

548/548 [==============================] - 11s 20ms/step - loss: 0.4863 - accuracy: 0.7586 - val\_loss: 0.5158 - val\_accuracy: 0.7435

Epoch 15/30

548/548 [==============================] - 11s 20ms/step - loss: 0.4851 - accuracy: 0.7587 - val\_loss: 0.5211 - val\_accuracy: 0.7349

Epoch 16/30

548/548 [==============================] - 11s 20ms/step - loss: 0.4839 - accuracy: 0.7591 - val\_loss: 0.5177 - val\_accuracy: 0.7431

Epoch 17/30

548/548 [==============================] - 9s 17ms/step - loss: 0.4829 - accuracy: 0.7595 - val\_loss: 0.5140 - val\_accuracy: 0.7435

Epoch 18/30

548/548 [==============================] - 9s 17ms/step - loss: 0.4814 - accuracy: 0.7613 - val\_loss: 0.5145 - val\_accuracy: 0.7431

Epoch 19/30

548/548 [==============================] - 9s 17ms/step - loss: 0.4801 - accuracy: 0.7614 - val\_loss: 0.5131 - val\_accuracy: 0.7419

Epoch 20/30

548/548 [==============================] - 9s 17ms/step - loss: 0.4786 - accuracy: 0.7639 - val\_loss: 0.5144 - val\_accuracy: 0.7365

Epoch 21/30

548/548 [==============================] - 9s 17ms/step - loss: 0.4766 - accuracy: 0.7635 - val\_loss: 0.5140 - val\_accuracy: 0.7398

Epoch 22/30

548/548 [==============================] - 9s 17ms/step - loss: 0.4746 - accuracy: 0.7642 - val\_loss: 0.5112 - val\_accuracy: 0.7398

Epoch 23/30

548/548 [==============================] - 10s 18ms/step - loss: 0.4722 - accuracy: 0.7682 - val\_loss: 0.5114 - val\_accuracy: 0.7333

Epoch 24/30

548/548 [==============================] - 11s 20ms/step - loss: 0.4701 - accuracy: 0.7692 - val\_loss: 0.5276 - val\_accuracy: 0.7337

Epoch 25/30

548/548 [==============================] - 11s 20ms/step - loss: 0.4681 - accuracy: 0.7696 - val\_loss: 0.5107 - val\_accuracy: 0.7452

Epoch 26/30

548/548 [==============================] - 11s 20ms/step - loss: 0.4667 - accuracy: 0.7711 - val\_loss: 0.5123 - val\_accuracy: 0.7439

Epoch 27/30

548/548 [==============================] - 11s 20ms/step - loss: 0.4651 - accuracy: 0.7731 - val\_loss: 0.5121 - val\_accuracy: 0.7476

Epoch 28/30

548/548 [==============================] - 9s 17ms/step - loss: 0.4637 - accuracy: 0.7728 - val\_loss: 0.5129 - val\_accuracy: 0.7456

Epoch 29/30

548/548 [==============================] - 9s 16ms/step - loss: 0.4624 - accuracy: 0.7742 - val\_loss: 0.5141 - val\_accuracy: 0.7460

Epoch 30/30

548/548 [==============================] - 8s 15ms/step - loss: 0.4607 - accuracy: 0.7751 - val\_loss: 0.5300 - val\_accuracy: 0.7287

In [40]:

import matplotlib.pyplot as plt

In [41]:

def plot\_graphs(history, string):

plt.plot(history.history[string])

plt.plot(history.history['val\_'+string])

plt.xlabel('Epochs')

plt.ylabel(string)

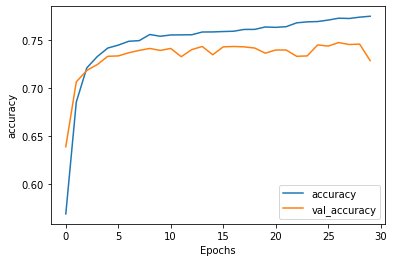
plt.legend([string, 'val\_'+string])

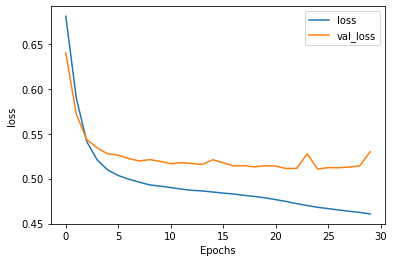
plt.show()

In [42]:

plot\_graphs(history, 'accuracy')

plot\_graphs(history, 'loss')





In [ ]: