Natural Language Processing with Tensorflow

Introduction:

In this project I will analyze text using Natural Language Processing. To accomplish this I will use Tensorflow. Why use Tensorflow, according to (Alsora, 1998) "The aim of this framework is to allow to easily design networks for any NLP task, using modular code and always maintaining the same input/output structure. This is extremely useful if your main use case is the deploy of a saved model in production, even if you want to access to this model using different programming languages."

Goal:

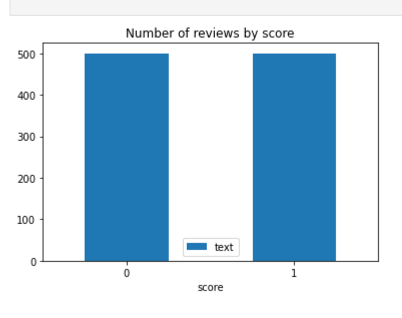
The goal of this project is to build a NLP API within Tensorflow that can text a string of text and predict if the customer either liked, or disliked the product that they reviewed. For this project I will using product reviews pulled from Amazon.

Step One: Import required packages

In [1]:
```
import matplotlib.pyplot as plt
```
In [2]:
```
import tensorflow as tf
```
In [3]:
```
%load_ext tensorboard
```
In [4]:
```
import datetime
```
In [5]:
```
from tensorflow import keras
```
In [6]:
```
from tensorflow.keras import layers
```
In [7]:
```
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
```
In [8]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
```
In [9]:
```
from sklearn.metrics import mean_squared_error
```
In [10]:
```
from sklearn.feature_extraction.text import CountVectorizer
```
In [11]:
```
from sklearn.model_selection import train_test_split
```
In [12]:
```
from keras.preprocessing.sequence import pad_sequences
```
In [13]:
```
from pprint import pprint
```
In [14]:
```
from collections import defaultdict
```
In [15]:
```
import pandas as pd
```
In [16]:
```
pd.set_option('display.max_rows', 1000)
```
In [17]:
```
import numpy as np
```
In [18]:
```
from nltk.tokenize import regexp_tokenize
```
In [19]:
```
import gensim
```
In [20]:
```
from gensim import corpora
```
In [21]:
```
from gensim.corpora.dictionary import Dictionary
```
In [22]:
```
import nltk
```
In [23]:
```
from nltk.corpus import stopwords
```
In [24]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```
In [25]:
```
import string
```
In [26]:
```
import re
```

Step Two: Import and prepare the data (Foy, 2020)

In [30]:
```
data = pd.read_csv('amazon_cells_labelled.csv', sep=",")
```
In [32]:
```
data.rename(columns={'Liked/Disliked' : 'score', 'Review': 'text'}, inplace=True)
```
In [34]:
```
ax = data.groupby('score').count().plot.bar(ylim=0)
ax.set_title('Number of reviews by score')
plt.xticks(rotation=0);
```



The data contains two columns, first a column of cell phone reviews and then a column of 1's and 0's indicating if the customer liked the phone or not. From the graph above it is clear that the number of positive and negative reviews is equal.

To prepare the data first all NA's will be removed, then all special characters will be removed, then all numbers will be removed, and finally all capital letters will be replaced with lowercase letters.

Next remove NA's from the data

In [35]:
```
df = data.dropna(subset=['text'])
```

Next remove any special characters from the text column (Petit, 2020)

In [36]:
```
spec_chars = ["!",'"',"#","%","&","'","(",")",
              "*","+",",","-",".","/",":",";","<",
              "=",">","?","@","[","\\","]","^","_",
              "`","{","|","}","~","–"]
```
In [37]:
```
for char in spec_chars:
    df['text'] = df['text'].str.replace(char, ' ')
```
```
<ipython-input-37-7e4c334a0c8b>:2: FutureWarning: The default value of regex will change from True to False
in a future version. In addition, single character regular expressions will*not* be treated as literal strin
gs when regex=True.
  df['text'] = df['text'].str.replace(char, ' ')
```

Remove numbers from the text column

In [38]:
```
df['text'] = df['text'].str.replace('\d+', '')
```
```
<ipython-input-38-935a25ac7b5f>:1: FutureWarning: The default value of regex will change from True to False
in a future version.
  df['text'] = df['text'].str.replace('\d+', '')
```

Next replace all captial letters with lowercase

In [39]:
```
df['text'] = df['text'].str.lower()
```

Next tokenize the text column, for this part I followed the following guide: (Jedamski, 2020)

To tokenize the words in the reviews I will define a function named tokenize, this function will split each sentence into it's individual words. Once we have the words seperated, we can filter out stopwords and then apply a vectorization to words replacing them with numerical values. We do this because Tensorflow cannot handle lists of string characters.

In [40]:
```
def tokenize(text):
    tokens = re.split('\W+', text)
    return tokens

df['text_tokenized'] = df['text'].apply(lambda x: tokenize(x))
df.head()
```
Out[40]:

	text	score	text_tokenized
0	so there is no way for me to plug it in here i...	0	[so, there, is, no, way, for, me, to, plug, it...
1	good case excellent value	1	[good, case, excellent, value,]
2	great for the jawbone	1	[great, for, the, jawbone,]
3	tied to charger for conversations lasting more...	0	[tied, to, charger, for, conversations, lastin...
4	the mic is great	1	[the, mic, is, great,]

Load stopwords

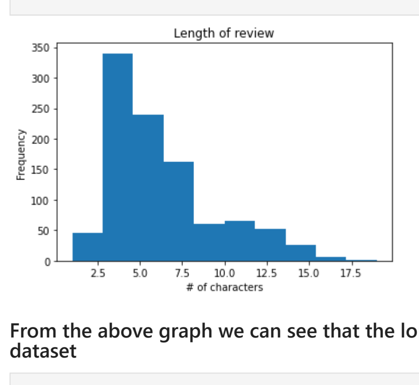
In [41]:
```
stopwords = nltk.corpus.stopwords.words('english')
```

Next remove the stopwords

In [42]:
```
def remove_stopwords(text_tokenized):
    text = [word for word in text_tokenized if word not in stopwords]
    return text

df['text_nostop'] = df['text_tokenized'].apply(lambda x: remove_stopwords(x))
df.head()
```
Out[42]:

	text	score	text_tokenized	text_nostop
0	so there is no way for me to plug it in here i...	0	[so, there, is, no, way, for, me, to, plug, it...	[way, plug, us, unless, go, converter,]
1	good case excellent value	1	[good, case, excellent, value,]	[good, case, excellent, value,]
2	great for the jawbone	1	[great, for, the, jawbone,]	[great, jawbone,]
3	tied to charger for conversations lasting more...	0	[tied, to, charger, for, conversations, lastin...	[tied, charger, conversations, lasting, minute...
4	the mic is great	1	[the, mic, is, great,]	[mic, great,]

In [43]:
```
line_num_words = [len(t_line) for t_line in df['text_nostop']]
```
In [44]:
```
plt.hist(line_num_words)
plt.title('Length of review')
plt.xlabel('# of characters')
plt.ylabel('Frequency');
```



From the above graph we can see that the longest review is 20 words in the final cleaned dataset

In [45]:
```
df['text_final'] = df['text_nostop'].apply(lambda x: ' '.join(x))
```
In [47]:
```
tfidf_vect = TfidfVectorizer()
X_tfidf = tfidf_vect.fit_transform(df['text_final'])
print(X_tfidf.shape)
print(tfidf_vect.get_feature_names())
```
```
(1000, 1686)
['abhor', 'ability', 'able', 'abound', 'absolutel', 'absolutely', 'ac', 'accept', 'acceptable', 'access', 'a
ccessable', 'accessing', 'accessory', 'accessoryone', 'accidentally', 'accompanied', 'according', 'activat
e', 'activated', 'activesync', 'actually', 'ad', 'adapter', 'adapters', 'add', 'addition', 'additional', 'ad
dress', 'adhesive', 'adorable', 'advertised', 'advise', 'aggravating', 'ago', 'alarm', 'allot', 'allow', 'al
lowing', 'allows', 'almost', 'alone', 'along', 'alot', 'also', 'although', 'aluminum', 'always', 'amazed',
'amazing', 'amazon', 'amp', 'ample', 'angeles', 'angle', 'another', 'answer', 'ant', 'antena', 'anti', 'anyo
ne', 'anything', 'anyway', 'anywhere', 'apart', 'apartment', 'apparently', 'appealing', 'appearance', 'appea
rs', 'applifies', 'appointments', 'area', 'arguing', 'armband', 'around', 'arrival', 'arrived', 'asia', 'as
k', 'aspect', 'assumed', 'atleast', 'att', 'attacked', 'attractive', 'audio', 'authentic', 'auto', 'availabl
e', 'average', 'avoid', 'avoiding', 'away', 'awesome', 'awful', 'awkward', 'awsome', 'back', 'background',
'backlight', 'bad', 'balance', 'bar', 'barely', 'bargain', 'bars', 'basement', 'basic', 'basically', 'batter
ies', 'battery', 'beat', 'beats', 'beautiful', 'bed', 'beep', 'beeping', 'behing', 'believe', 'bells', 'bel
t', 'bend', 'best', 'better', 'beware', 'big', 'biggest', 'bills', 'bit', 'bitpim', 'black', 'blackberry',
'blacktop', 'bland', 'blew', 'blue', 'blueant', 'bluetoooth', 'bluetooth', 'bluetooths', 'bmw', 'book', 'boo
king', 'boost', 'boot', 'bose', 'bother', 'bottom', 'bottowm', 'bought', 'bougth', 'boy', 'brand', 'break',
'breakage', 'breaking', 'breaks', 'brilliant', 'broke', 'broken', 'browser', 'browsing', 'bt', 'btv', 'bubbl
ing', 'bucks', 'buds', 'build', 'built', 'bulky', 'bumpers', 'button', 'buttons', 'buy', 'buyer', 'buyers',
'buying', 'buyit', 'buzzing', 'ca', 'cable', 'cables', 'calendar', 'call', 'called', 'calls', 'came', 'camer
a', 'canal', 'cancellation', 'cancelling', 'cannot', 'cant', 'capability', 'capacity', 'car', 'card', 'car
e', 'careful', 'carried', 'carriers', 'carries', 'carry', 'case', 'cases', 'casing', 'cassette', 'cat', 'cat
ching', 'caused', 'causing', 'cbr', 'cds', 'cell', 'cellphone', 'cellphones', 'cellular', 'cent', 'center',
'certain', 'certainly', 'changing', 'channel', 'charge', 'charged', 'charger', 'chargers', 'charges', 'charg
ing', 'charm', 'cheap', 'cheaper', 'cheaply', 'cheapy', 'check', 'checked', 'child', 'china', 'chinese', 'ch
oice', 'choices', 'christmas', 'cingulair', 'cingular', 'clarity', 'classy', 'clear', 'clearer', 'clearly',
'clever', 'clicks', 'clip', 'clipping', 'clips', 'clock', 'colleague', 'color', 'colored', 'colors', 'combin
ation', 'come', 'comes', 'comfort', 'comfortable', 'comfortably', 'comfortible', 'coming', 'comments', 'comm
ercials', 'communicate', 'communication', 'communications', 'commuter', 'company', 'comparably', 'compared',
'compete', 'competitors', 'complain', 'complained', 'complaint', 'complaints', 'completely', 'compliments',
'compromise', 'computer', 'concrete', 'conditions', 'confortable', 'confusing', 'connect', 'connected', 'con
necting', 'connection', 'constantly', 'constructed', 'construction', 'consumer', 'contact', 'contacted', 'co
ntacting', 'contacts', 'continue', 'continues', 'contract', 'control', 'controls', 'contstruct', 'convenien
t', 'conversation', 'conversations', 'converter', 'cool', 'copier', 'copy', 'corded', 'correctly', 'cost',
'costs', 'could', 'couldnt', 'counter', 'counterfeit', 'couple', 'coupon', 'course', 'cover', 'coverage', 'c
overed', 'crack', 'cracked', 'cradle', 'cradles', 'crap', 'crappy', 'crashed', 'crawl', 'creaks', 'crisp',
'cumbersome', 'current', 'currently', 'curve', 'customer', 'cut', 'cute', 'cutouts', 'cuts', 'damage', 'dar
n', 'data', 'date', 'day', 'days', 'dead', 'deaf', 'deal', 'decade', 'decent', 'decision', 'defeats', 'defec
t', 'defective', 'deffinitely', 'definitely', 'definitly', 'delay', 'delivery', 'describe', 'described', 'de
scription', 'design', 'designed', 'designs', 'despite', 'destination', 'destroying', 'detachable', 'detaile
d', 'development', 'device', 'devices', 'dialing', 'died', 'dieing', 'different', 'difficult', 'directed',
'directions', 'directly', 'dirty', 'disapoinment', 'disapointing', 'disappoint', 'disappointed', 'disappoint
ing', 'disappointment', 'discarded', 'discomfort', 'disconnected', 'discount', 'disgusting', 'display', 'dis
pleased', 'disposable', 'dissapointed', 'dissapointing', 'distorted', 'distracting', 'dit', 'division', 'dn
a', 'docking', 'dollar', 'done', 'dont', 'double', 'download', 'downloading', 'dozen', 'dozens', 'drain', 'd
rained', 'drains', 'drawback', 'driving', 'drivng', 'droid', 'drop', 'dropped', 'dropping', 'drops', 'dual',
'due', 'durable', 'dustpan', 'dying', 'ear', 'earbud', 'earbuds', 'earbugs', 'eargels', 'earlier', 'earpad',
'earphone', 'earphones', 'earpiece', 'earpieces', 'ears', 'earset', 'ease', 'easier', 'easily', 'easy', 'ech
o', 'edge', 'effect', 'effective', 'effects', 'effort', 'either', 'electronics', 'elegant', 'else', 'elsewhe
re', 'embarassing', 'embarrassing', 'embedded', 'encourage', 'end', 'ended', 'ends', 'engineered', 'enjoy',
'enough', 'enter', 'entertainment', 'entire', 'env', 'equipment', 'era', 'ergonomic', 'ericson', 'ericsson',
'especially', 'essentially', 'etc', 'europe', 'even', 'eventually', 'ever', 'every', 'everyday', 'everyone',
'everything', 'everywhere', 'exactly', 'exceeds', 'excelent', 'excellent', 'excels', 'except', 'exceptiona
l', 'excessive', 'exchange', 'exchanged', 'excited', 'exclaim', 'excrutiatingly', 'exercise', 'existing', 'e
xpect', 'expectations', 'expected', 'expensive', 'experience', 'experienced', 'explain', 'extended', 'exteri
or', 'external', 'extra', 'extremely', 'eye', 'fabulous', 'face', 'faceplates', 'fact', 'factor', 'failed',
'fails', 'fairly', 'fall', 'falling', 'falls', 'family', 'fantastic', 'far', 'fast', 'faster', 'father', 'fa
vorite', 'feature', 'features', 'fee', 'feel', 'feels', 'feet', 'felt', 'fi', 'figure', 'file', 'finally',
'find', 'finds', 'fine', 'fingers', 'finished', 'fire', 'first', 'fit', 'fits', 'five', 'fixes', 'flash', 'f
law', 'flawed', 'flawless', 'flawlessly', 'flaws', 'flimsy', 'flip', 'flipphones', 'fliptop', 'floor', 'flop
py', 'flops', 'flush', 'fm', 'followed', 'fond', 'fooled', 'forced', 'forever', 'forgeries', 'forget', 'forg
ot', 'form', 'found', 'four', 'fourth', 'fraction', 'free', 'freedom', 'freeway', 'freezes', 'frequently',
'frequentyly', 'friendly', 'friends', 'frog', 'front', 'frustration', 'fry', 'ft', 'fulfills', 'full', 'full
y', 'fun', 'function', 'functional', 'functionality', 'functions', 'funny', 'gadget', 'gadgets', 'games', 'g
arbage', 'garbled', 'gave', 'geeky', 'gels', 'generally', 'gentle', 'genuine', 'get', 'gets', 'getting', 'gi
mmick', 'girl', 'give', 'given', 'giving', 'glad', 'glare', 'glasses', 'glove', 'glued', 'go', 'goes', 'goin
g', 'gonna', 'good', 'gosh', 'got', 'gotten', 'graphics', 'great', 'greater', 'grey', 'grip', 'grtting', 'gu
ess', 'gx', 'hair', 'hand', 'handheld', 'hands', 'handset', 'handsfree', 'handy', 'happened', 'happening',
'happens', 'happier', 'happy', 'hard', 'hardly', 'hat', 'hate', 'hated', 'haul', 'headbands', 'headphones',
'headset', 'headsets', 'hear', 'hearing', 'heavy', 'help', 'helpful', 'hey', 'high', 'highest', 'highly', 'h
ighy', 'hinge', 'hit', 'hitch', 'hold', 'holder', 'holding', 'holds', 'holster', 'home', 'hook', 'hoped', 'h
oping', 'horrible', 'hot', 'hour', 'hours', 'hoursthe', 'house', 'however', 'hs', 'huge', 'humans', 'hummin
g', 'hundred', 'hurt', 'hybrid', 'hype', 'iam', 'idea', 'ideal', 'igo', 'ill', 'im', 'imac', 'images', 'imag
ine', 'immediately', 'important', 'impossible', 'impressed', 'impressive', 'improper', 'improve', 'improveme
nt', 'inches', 'included', 'incoming', 'inconspicuous', 'increase', 'incrediable', 'incredible', 'incredibl
y', 'indoors', 'industrial', 'inexcusable', 'inexpensive', 'infatuated', 'inform', 'infra', 'infuriating',
'insert', 'inside', 'install', 'installed', 'instance', 'instead', 'instruction', 'instructions', 'integrate
d', 'intended', 'interested', 'interface', 'intermittently', 'internet', 'invented', 'investment', 'iphone',
'ipod', 'ipods', 'ir', 'irda', 'iriver', 'issues', 'item', 'items', 'jabra', 'jack', 'jawbone', 'jerks', 'ji
ggle', 'job', 'joke', 'joy', 'juice', 'junk', 'jx', 'keen', 'keep', 'keeping', 'keeps', 'kept', 'key', 'keyb
oard', 'keypad', 'keypads', 'keys', 'killer', 'kind', 'kindle', 'kitchen', 'kits', 'knock', 'know', 'knows',
'krussel', 'lacking', 'land', 'lap', 'laptop', 'large', 'last', 'lasted', 'lasting', 'lasts', 'latch', 'late
ly', 'later', 'latest', 'laughing', 'lc', 'leaf', 'leaks', 'learned', 'least', 'leather', 'left', 'lense',
'leopard', 'less', 'lesson', 'let', 'letting', 'lg', 'life', 'light', 'lightly', 'lights', 'lightweight', 'l
ike', 'liked', 'likes', 'line', 'linked', 'linking', 'linksys', 'listener', 'listening', 'lit', 'literally',
'little', 'living', 'loads', 'lock', 'locked', 'locks', 'logitech', 'long', 'longer', 'look', 'looking', 'lo
oks', 'loop', 'loops', 'loose', 'looses', 'los', 'lose', 'lost', 'lot', 'lots', 'loud', 'louder', 'loudest',
'loudspeaker', 'lousy', 'love', 'loved', 'loves', 'low', 'luck', 'machine', 'made', 'magical', 'magnetic',
'mail', 'mainly', 'maintain', 'maintains', 'major', 'majority', 'make', 'makes', 'making', 'managed', 'manag
ement', 'manual', 'manufacturer', 'many', 'mark', 'market', 'match', 'material', 'max', 'may', 'means', 'mec
hanism', 'media', 'mediocre', 'mega', 'megapixels', 'memory', 'mention', 'mentioned', 'menus', 'mere', 'mes
s', 'message', 'messages', 'messaging', 'messes', 'metal', 'metro', 'mic', 'microphone', 'microsoft', 'migh
t', 'mind', 'mine', 'mini', 'mins', 'minute', 'minutes', 'misleading', 'missed', 'mistake', 'mobile', 'mod
e', 'model', 'modest', 'money', 'monkeys', 'month', 'months', 'morning', 'mostly', 'mother', 'moto', 'moto
r', 'motorola', 'motorolas', 'moving', 'mp', 'mps', 'much', 'muddy', 'muffled', 'multiple', 'music', 'must',
'mute', 'nano', 'navigate', 'near', 'nearly', 'neat', 'need', 'needed', 'needless', 'needs', 'negatively',
'neither', 'network', 'never', 'new', 'next', 'ngage', 'nice', 'nicely', 'nicer', 'night', 'nightmare', 'noi
se', 'noises', 'nokia', 'none', 'normal', 'normally', 'note', 'noted', 'nothing', 'notice', 'noticed', 'numb
er', 'numbers', 'numerous', 'nyc', 'obviously', 'occupied', 'odd', 'oem', 'offering', 'offers', 'official',
'oh', 'ok', 'old', 'one', 'ones', 'online', 'oozes', 'open', 'opens', 'operate', 'operates', 'optimal', 'opt
ion', 'options', 'order', 'ordered', 'ordering', 'orders', 'organizational', 'original', 'originally', 'os',
'others', 'otherwise', 'outgoing', 'outlet', 'outperform', 'outside', 'overall', 'overly', 'overnight', 'ove
rnite', 'override', 'owned', 'owner', 'owning', 'pack', 'package', 'packaged', 'pad', 'pads', 'pain', 'painf
ul', 'pair', 'paired', 'pairing', 'palm', 'palms', 'palmtop', 'pants', 'part', 'particular', 'party', 'passe
d', 'patient', 'pause', 'pay', 'pc', 'pcs', 'pda', 'peachy', 'peeling', 'penny', 'pens', 'people', 'perfec
t', 'perfectly', 'performance', 'performed', 'performing', 'perhaps', 'periodically', 'periods', 'person',
'petroleum', 'phone', 'phones', 'photo', 'pics', 'picture', 'pictures', 'piece', 'pitiful', 'pixel', 'plac
e', 'placed', 'places', 'plan', 'planning', 'plans', 'plantronics', 'plantronincs', 'plastic', 'play', 'play
er', 'players', 'plays', 'pleasantly', 'please', 'pleased', 'pleather', 'plenty', 'plug', 'plugged', 'plug
s', 'plus', 'pocket', 'pockets', 'point', 'poor', 'poorly', 'port', 'portable', 'portraits', 'possesed', 'po
ssibility', 'posted', 'potentially', 'power', 'practical', 'practically', 'practice', 'preferably', 'premiu
m', 'prettier', 'pretty', 'prevents', 'previous', 'price', 'priced', 'pricing', 'prime', 'print', 'probabl
y', 'problem', 'problems', 'procedure', 'procedures', 'produce', 'product', 'products', 'program', 'promise
d', 'prompt', 'promptly', 'properly', 'pros', 'protected', 'protection', 'protective', 'protector', 'protect
s', 'provide', 'provided', 'provides', 'ps', 'psyched', 'puff', 'pull', 'purcashed', 'purchase', 'purchase
d', 'purchases', 'purchasing', 'purpose', 'push', 'pushed', 'put', 'quality', 'quick', 'quickly', 'quiet',
'quit', 'quite', 'qwerty', 'randomly', 'range', 'rare', 'rate', 'rated', 'rather', 'rating', 'razor', 'raz
r', 'reach', 'reaching', 'read', 'reading', 'ready', 'real', 'realize', 'really', 'reason', 'reasonable', 'r
easonably', 'reboots', 'reccomendation', 'reccommend', 'receipt', 'receive', 'received', 'receiving', 'recen
tly', 'reception', 'recessed', 'recharge', 'recieve', 'recognition', 'recognizes', 'recommend', 'recommende
d', 'red', 'refund', 'refurb', 'refuse', 'refused', 'regarding', 'regret', 'regretted', 'relative', 'relativ
ely', 'reliability', 'remorse', 'removing', 'renders', 'reoccure', 'replace', 'replaced', 'replacement', 're
placementr', 'requirements', 'research', 'resistant', 'resolution', 'respect', 'rest', 'restart', 'restockin
g', 'restored', 'rests', 'results', 'return', 'returned', 'returning', 'reverse', 'reversible', 'review', 'r
eviews', 'ride', 'right', 'riingtones', 'ring', 'ringer', 'ringing', 'ringtones', 'rip', 'ripped', 'risk',
'roam', 'rocketed', 'rocks', 'roles', 'room', 'rotating', 'row', 'rubber', 'run', 'runs', 'sa', 'saggy', 'sa
id', 'samsung', 'sanyo', 'satisfied', 'satisifed', 'save', 'saved', 'say', 'saying', 'says', 'scary', 'sch',
'scratch', 'scratched', 'screen', 'screens', 'seamlessly', 'searched', 'seat', 'seconds', 'secure', 'securel
y', 'securly', 'see', 'seeen', 'seem', 'seemed', 'seems', 'seen', 'self', 'seller', 'send', 'sending', 'sens
itive', 'sensor', 'sent', 'seperated', 'series', 'seriously', 'service', 'set', 'setting', 'setup', 'severa
l', 'severe', 'sex', 'shape', 'share', 'sharp', 'shield', 'shifting', 'shine', 'shiny', 'shipment', 'shippe
d', 'shipping', 'shooters', 'short', 'shots', 'shouldve', 'shouting', 'show', 'shows', 'side', 'sides', 'sig
ht', 'signal', 'signals', 'significantly', 'signs', 'sim', 'simple', 'simpler', 'simply', 'since', 'sins',
'sister', 'sitting', 'situations', 'size', 'sizes', 'sketchy', 'skip', 'skype', 'sleek', 'slid', 'slide', 's
lider', 'sliding', 'slim', 'slipping', 'slow', 'slowly', 'small', 'smallest', 'smartphone', 'smell', 'smok
e', 'smoking', 'smoother', 'smoothly', 'smudged', 'snap', 'snug', 'soft', 'software', 'sold', 'solid', 'some
how', 'someone', 'something', 'sometimes', 'somewhat', 'somewhere', 'son', 'songs', 'sony', 'soon', 'soone
r', 'sorry', 'sos', 'sound', 'sounded', 'sounds', 'source', 'sources', 'soyo', 'span', 'speaker', 'speakerph
one', 'specially', 'specs', 'speed', 'spinn', 'spring', 'sprint', 'stand', 'standard', 'star', 'stars', 'sta
rtac', 'started', 'starter', 'starts', 'state', 'stated', 'static', 'station', 'stay', 'stays', 'steep', 'st
eer', 'stereo', 'still', 'stop', 'stopped', 'stops', 'storage', 'store', 'strange', 'strap', 'stream', 'stre
ngth', 'stress', 'strip', 'strong', 'stuck', 'study', 'stuff', 'stupid', 'sturdiness', 'sturdy', 'styles',
'styling', 'stylish', 'submerged', 'sucked', 'sucks', 'sudden', 'suddenly', 'sunglasses', 'super', 'superb',
'superfast', 'supertooth', 'support', 'supposedly', 'suprised', 'sure', 'surefire', 'surprised', 'survived',
'sweetest', 'switch', 'swivel', 'sync', 'synchronization', 'take', 'takes', 'talk', 'talking', 'tape', 'tec
h', 'technology', 'telephone', 'tell', 'terrible', 'texas', 'text', 'thank', 'thanks', 'thats', 'theory', 't
hereplacement', 'thin', 'thing', 'things', 'think', 'third', 'thorn', 'though', 'thought', 'three', 'threw',
'thru', 'thumbs', 'tick', 'ticking', 'tied', 'tight', 'time', 'timeframe', 'timely', 'times', 'tinny', 'tin
y', 'tips', 'tmobile', 'toactivate', 'toast', 'today', 'together', 'toilet', 'told', 'tone', 'tones', 'too
k', 'tool', 'tools', 'tooth', 'top', 'total', 'totally', 'touch', 'touches', 'tracfone', 'tracfonewebsite',
'tracking', 'transceiver', 'transfer', 'transformed', 'transmission', 'transmit', 'transmitters', 'trash',
'travled', 'tremendous', 'treo', 'tricky', 'tried', 'tries', 'trouble', 'truly', 'trunk', 'trust', 'try', 't
rying', 'tungsten', 'turn', 'turned', 'turns', 'tv', 'two', 'type', 'ugly', 'unacceptable', 'unacceptible',
'unbearable', 'uncomfortable', 'understand', 'understanding', 'unfortunately', 'unhappy', 'unintelligible',
'unit', 'units', 'unknown', 'unless', 'unlike', 'unreliable', 'unsatisfactory', 'unusable', 'upbeat', 'updat
e', 'upgrade', 'upload', 'upstairs', 'us', 'usable', 'usage', 'usb', 'use', 'used', 'useful', 'usefulness',
'useless', 'user', 'using', 'usually', 'utter', 'utterly', 'value', 'vc', 'vehicle', 'verizon', 'vi', 'via',
'video', 'videos', 'virgin', 'visor', 'voice', 'voltage', 'volume', 'vx', 'waaay', 'waiting', 'wake', 'walke
d', 'walkman', 'wall', 'wallet', 'want', 'wanted', 'warning', 'warranty', 'waste', 'wasted', 'wasting', 'wat
erproof', 'way', 'weak', 'wear', 'wearing', 'web', 'website', 'websites', 'week', 'weeks', 'weight', 'weir
d', 'well', 'went', 'whatever', 'whatsoever', 'whether', 'whine', 'whistles', 'white', 'whoa', 'whole', 'who
se', 'wi', 'wife', 'wild', 'wind', 'window', 'windows', 'winner', 'wiping', 'wire', 'wired', 'wirefly', 'wir
eless', 'wise', 'wish', 'wit', 'within', 'without', 'wobbly', 'wonder', 'wonderfully', 'wont', 'wood', 'wood
en', 'word', 'work', 'worked', 'working', 'works', 'world', 'worn', 'worst', 'worth', 'worthless', 'worthwhi
le', 'would', 'wow', 'wrong', 'wrongly', 'year', 'years', 'yell', 'yes', 'yet', 'za', 'zero']
```

From looking at the output of tfidf_vect we can see that there are 1686 words in the final cleaned dataset.

The output, X_tfidf is in sparse matrix form, tensorflow cannot handle this object type. X_tfidf must be converted to a Pandas dataframe first.

In [48]:
```
X_features = pd.DataFrame(X_tfidf.toarray())
X_features.head()
```
Out[48]:

	0	1	2	3	4	5	6	7	8	9	...	1676	1677	1678	1679	1680	1681	1682	1683	1684	1685
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 1686 columns

Next export the cleaned dataset to csv

In [49]:
```
X_features.to_csv('data_clean.csv')
```

Building the model in Tensorflow (Load a Pandas.DataFrame | TensorFlow Core, 2021)

In [50]:
```
target = df.pop('score')
```
In [51]:
```
X_train, X_test, y_train, y_test = train_test_split(X_features, target, test_size=0.20, random_state=100)
```

For this project the data is split 80/20, 80% of the words will be used for training the model and 20% will be used for testing.

In [52]:
```
print(df.shape); print(X_train.shape); print(X_test.shape)
```
```
(1000, 4)
(800, 1686)
(200, 1686)
```

Next convert the DataFrame to a Tensor object (Ydobon, 2019)

By converting out vectorized list to a tensorflow data object we will not need to pad our sequences. The reason for this is that each word in our dataset is assigned a value by the vectorization function, then each of these values is placed in it's own cell within the tensor. We will not be feeding the raw strings into Tensorflow, and hence we will not have to worry about the strings having uniform length. I will also not be imposing a stopping criteria on the Tensorflow model, instead I will vary the number the number of epochs and see what effect that on the model accuracy.

In [53]:
```
dataset = tf.data.Dataset.from_tensor_slices((X_features.values, target.values))
```
In [54]:
```
train_dataset = dataset.shuffle(len(df)).batch(1)
```
In [56]:
```
def get_compiled_model():
  model = tf.keras.Sequential([
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(1)
  ])

  model.compile(optimizer='adam',
                loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
                metrics=['accuracy'])
  return model
```

Model Features:

This is a sequential model, featuring three layers. The first two layers have ten nodes each, while the final layer features one node. The model metric is accuracy, and the loss function is BinaryCrossentropy. The nodes where choosen by taking the square root of input features, which is Sqrt(1686) which equals 41. That is then rounded down to the nearest power or ten, which in this case is ten.

In [58]:
```
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
```

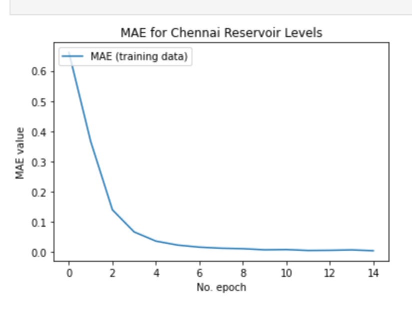
The model will be fit to the training data_set in the next step

In [65]:
```
model = get_compiled_model()
history = model.fit(train_dataset, epochs=15, callbacks=[tensorboard_callback])
model.summary()
```
```
Epoch 1/15
WARNING:tensorflow:Layer dense_6 is casting an input tensor from dtype float64 to the layer's dtype of float
32, which is new behavior in TensorFlow 2.  The layer has dtype float32 because its dtype defaults to float
x.

If you intended to run this layer in float32, you can safely ignore this warning. If in doubt, this warning
is likely only an issue if you are porting a TensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backend.set_floatx('float64')`. To cha
nge just this layer, pass dtype='float64' to the layer constructor. If you are the author of this layer, you
can disable autocasting by passing autocast=False to the base Layer constructor.

   2/1000 [..............................] - ETA: 4:41 - loss: 0.6917 - accuracy: 0.5000WARNING:tensorflow:C
allbacks method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0010s vs `on_train_ba
tch_end` time: 0.5633s). Check your callbacks.
1000/1000 [==============================] - 1s 1ms/step - loss: 0.6575 - accuracy: 0.5620
Epoch 2/15
1000/1000 [==============================] - 1s 616us/step - loss: 0.3446 - accuracy: 0.8590
Epoch 3/15
1000/1000 [==============================] - 1s 593us/step - loss: 0.1292 - accuracy: 0.9570
Epoch 4/15
1000/1000 [==============================] - 1s 612us/step - loss: 0.0565 - accuracy: 0.9840
Epoch 5/15
1000/1000 [==============================] - 1s 619us/step - loss: 0.0302 - accuracy: 0.9910
Epoch 6/15
1000/1000 [==============================] - 1s 601us/step - loss: 0.0184 - accuracy: 0.9950
Epoch 7/15
1000/1000 [==============================] - 1s 604us/step - loss: 0.0130 - accuracy: 0.9980
Epoch 8/15
1000/1000 [==============================] - 1s 665us/step - loss: 0.0090 - accuracy: 0.9970
Epoch 9/15
1000/1000 [==============================] - 1s 617us/step - loss: 0.0077 - accuracy: 0.9970
Epoch 10/15
1000/1000 [==============================] - 1s 672us/step - loss: 0.0096 - accuracy: 0.9960
Epoch 11/15
1000/1000 [==============================] - 1s 624us/step - loss: 0.0068 - accuracy: 0.9960
Epoch 12/15
1000/1000 [==============================] - 1s 654us/step - loss: 0.0073 - accuracy: 0.9980
Epoch 13/15
1000/1000 [==============================] - 1s 651us/step - loss: 0.0067 - accuracy: 0.9970
Epoch 14/15
1000/1000 [==============================] - 1s 704us/step - loss: 0.0057 - accuracy: 0.9970
Epoch 15/15
1000/1000 [==============================] - 1s 599us/step - loss: 0.0064 - accuracy: 0.9960
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 10)                16870
_________________________________________________________________
dense_7 (Dense)              (None, 10)                110
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 11
=================================================================
Total params: 16,991
Trainable params: 16,991
Non-trainable params: 0
_________________________________________________________________
```

Next we need to make sure that the model is not overfitting the data. According to (Tensorflow.org, 2020) "If the validation metric is going in the wrong direction, the model is clearly overfitting." Looking at the model history it is clear that the accuracy drops going from epoch 13 to epoch 14. So yes there is some overfitting going on. This is another reason to use a smaller number of epochs.

Next we will visualize the training for different epochs versus the Mean Absolute Error (Chris & Sara, 2021)

In [64]:
```
plt.plot(history.history['loss'], label='MAE (training data)')
plt.title('MAE for Chennai Reservoir Levels')
plt.ylabel('MAE value')
plt.xlabel('No. epoch')
plt.legend(loc="upper left")
plt.show()
```



From the above graph the MAE reaches a minimum at around 8 epochs

Next create a tensorboard object analyze the model (Get Started with TensorBoard |, 2020)

In [ ]:
```
%tensorboard --logdir logs/fit;
```

Jupyter notebook gives an error when opening the TensorBoard object but it opens successfully on http://localhost:6006/

From looking at the graph of Epoch Accuracy it is clear that 8 epochs is the value that maxes out the accuracy

Next create a new model with 8 epochs

In [66]:
```
model = get_compiled_model()
model.fit(train_dataset, epochs=8, callbacks=[tensorboard_callback])
model.summary()
```
```
Epoch 1/8
WARNING:tensorflow:Layer dense_9 is casting an input tensor from dtype float64 to the layer's dtype of float
32, which is new behavior in TensorFlow 2.  The layer has dtype float32 because its dtype defaults to float
x.

If you intended to run this layer in float32, you can safely ignore this warning. If in doubt, this warning
is likely only an issue if you are porting a TensorFlow 1.X model to TensorFlow 2.

To change all layers to have dtype float64 by default, call `tf.keras.backend.set_floatx('float64')`. To cha
nge just this layer, pass dtype='float64' to the layer constructor. If you are the author of this layer, you
can disable autocasting by passing autocast=False to the base Layer constructor.

   2/1000 [..............................] - ETA: 5:16 - loss: 0.6876 - accuracy: 0.5000  WARNING:tensorflo
w:Callbacks method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0010s vs `on_train
_batch_end` time: 0.6342s). Check your callbacks.
1000/1000 [==============================] - 1s 1ms/step - loss: 0.6513 - accuracy: 0.5380
Epoch 2/8
1000/1000 [==============================] - 1s 607us/step - loss: 0.3430 - accuracy: 0.8710
Epoch 3/8
1000/1000 [==============================] - 1s 639us/step - loss: 0.1295 - accuracy: 0.9560
Epoch 4/8
1000/1000 [==============================] - 1s 606us/step - loss: 0.0619 - accuracy: 0.9830
Epoch 5/8
1000/1000 [==============================] - 1s 618us/step - loss: 0.0339 - accuracy: 0.9920
Epoch 6/8
1000/1000 [==============================] - 1s 712us/step - loss: 0.0222 - accuracy: 0.9950
Epoch 7/8
1000/1000 [==============================] - 1s 672us/step - loss: 0.0151 - accuracy: 0.9970
Epoch 8/8
1000/1000 [==============================] - 1s 620us/step - loss: 0.0122 - accuracy: 0.9950
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 10)                16870
_________________________________________________________________
dense_10 (Dense)             (None, 10)                110
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 11
=================================================================
Total params: 16,991
Trainable params: 16,991
Non-trainable params: 0
_________________________________________________________________
```

From the above model 8 epochs are enough to achive an accuarcy of 99.7%

Network Architecture

The model uses the following architecture:

Activation: Relu, this is a rectified linear activation function. It outputs the input value for positive values, and 0 for negative values. This is used due to score being either 1, or 0. The model cannot output a negative model with this activation. (Brownlee, 2020)

number of nodes per layer: For this take the square root of the number of features and then try values near to this and see how it effects the accuracy. Ten nodes for each of the first two layers, and one node in the final layer produce an accuracy of over 99%

Loss function = BinaryCrossentropy. This problem is a binary classification problem, and so BinaryCrossentropy is perfect for this type of problem.

Optimizer = Adam. "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters". (Kingma & Ba, 2017)

Network Functionality:

This is a very functional neural network. It produces very high levels of accuracy over a large range of words. Taking this api and porting it to a different set of data would be very easy.

Course of action:

The goal of this project was to show that it is possible to predict if a review would be positive or negative based on analyzing key words. The results of the model applied to the training data show that a high level of accuracy can be achived. A recommended course of action would be for a company to use this information to train bots to create fake positive reviews. This would allow the company to increase profits by having people that check reviews online before making purchases see these created reviews and trust tat they are real reviews.

References

N. (2020, June 2). 4 Ways to Use Pandas to Select Columns in a Dataframe. Datagy. https://datagy.io/pandas-select-columns/

Removing URL from a column in Pandas Dataframe. (2018, August 23). Stack Overflow. https://stackoverflow.com/questions/51994254/removing-url-from-a-column-in-pandas-dataframe

Category: nltk. (2020, June 26). Pythonspot. https://pythonspot.com/category/nltk/

Team, K. (2020). Keras documentation: The Sequential model. K. Team. https://keras.io/guides/sequential_model/

Petit, U. (2020, November 13). Simplify your Dataset Cleaning with Pandas - Towards Data Science. Medium. https://towardsdatascience.com/simplify-your-dataset-cleaning-with-pandas-75951b23568e

Load a pandas.DataFrame | TensorFlow Core. (2021). TensorFlow. https://www.tensorflow.org/tutorials/load_data/pandas_dataframe

Ydobon, A. (2019, September 26). [Tensorflow 2.0] Load Pandas dataframe to Tensorflow. Medium. https://financial-engineering.medium.com/tensorflow-2-0-load-pandas-dataframe-to-tensorflow-202c5d48966b

Jedamski, D. (2020, August 10). Compare all methods using key performance metrics - Advanced NLP with Python for Machine Learning. LinkedIn. https://www.linkedin.com/learning/advanced-nlp-with-python-for-machine-learning/compare-all-methods-using-key-performance-metrics?u=2045532

Get started with TensorBoard |. (2020). TensorFlow. https://www.tensorflow.org/tensorboard/get_started

Brownlee, J. (2020, August 20). A Gentle Introduction to the Rectified Linear Unit (ReLU). Machine Learning Mastery. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Kingma, D., & Ba, J. (2017, January 30). Adam: A method for stochastic optimization. Retrieved March 11, 2021, from https://arxiv.org/abs/1412.6980

Chris, & Sara. (2021, January 26). Visualizing training performance with TensorFlow 2 and Keras. Retrieved March 11, 2021, from https://www.machinecurve.com/index.php/2019/10/08/how-to-visualize-the-training-process-in-keras/