Faculty of Computing & Information Technology

INDUS UNIVERSITY

Artificial Intelligence (Lab)

Project Report

Title:

Next Word Prediction

Student's Name	*Student's ID*
MUHAMMAD OWAIS QADRI	*519-2020*
RAO MUHAMMAD NOMAN	*25-2020*
MUHAMMAD MUDASSIR	*174-2020*

Submitted to: Ms. Komal Chohan

ACKNOWLEDGMENT

We are grateful because we managed to complete our final lab project within the time given by our teacher **Miss Komal Chohan**. This project could not be achieved without the effort and cooperation of our group members, Group members, **Muhammad Owais Qadri**, **Rao Muhammad Noman**, and **Muhammad Mudassir**.

We also sincerely thank our teacher, Miss Komal Chohan, for the guidance and encouragement in finishing this project and for teaching us in this course.

Finally, we would like to express our gratitude to our friends and respondents for their support and willingness directly or indirectly to spend some time with us to fill in the questionnaires.

CONTENT

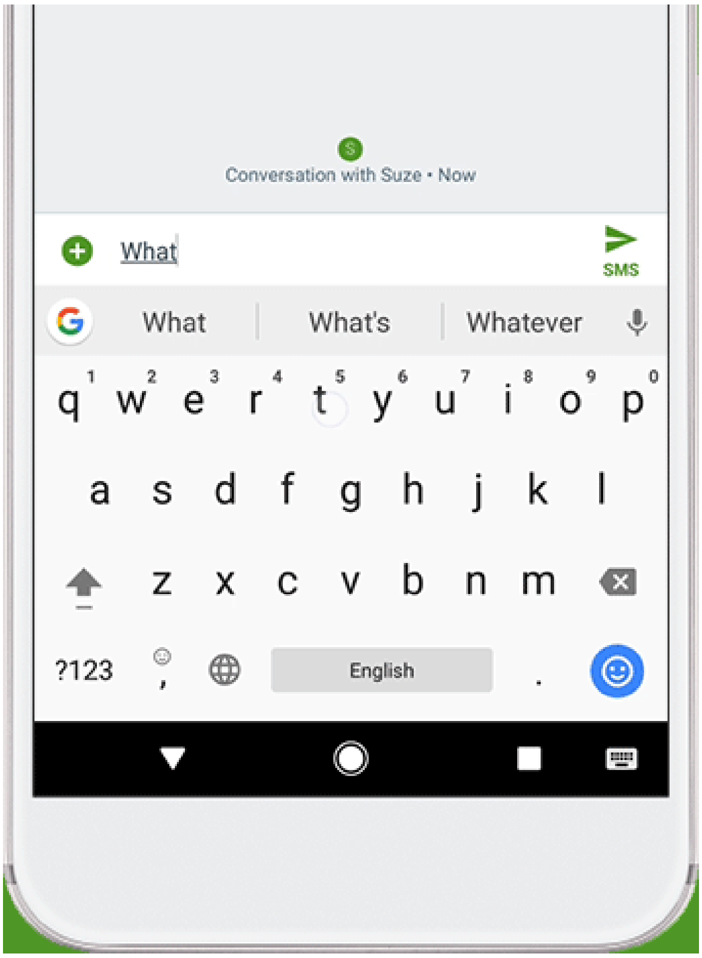
I. Abstract

Writing long sentences is a bit boring, however with next-word prediction within the keyboard technology has created this easy. Next Word Prediction is in addition, referred to as Language Modeling. It's the endeavor of predicting what word comes straightaway. It's every one of the critical assignments of human language technology and has various applications. Our Aim of creating this model to predict 10 or more than 10 word as fast as possible utilizing minimum time. As RNN is Long short time memory it will understand past text and predict the words which may be helpful for the user to frame sentences and this technique uses letter to letter prediction means it predict a letter after letter to create a word.

II. Introduction

Natural Language Processing (NLP) is a significant part of artificial Intelligence, which incorporates AI, which contributes to finding productive approaches to speak with people and gain from the associations with them. One such commitment is to give portable clients anticipated" next words," as they type along within applications, with an end goal to assist message conveyance by having the client select a proposed word as opposed to composing it.

Next Word Prediction or what is also called Language Modeling is the task of predicting what word comes next. It is one of the fundamental tasks of NLP and has many applications. You might be using it daily when you write texts or emails without realizing it.



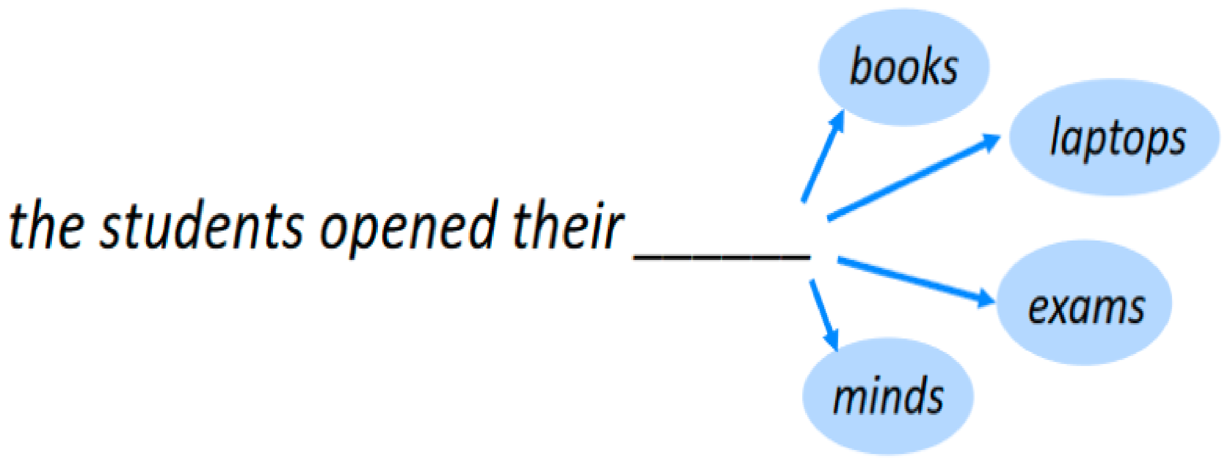
III. Objectives of the project

As writing an essay and framing a big paragraph are time-consuming it will help end-users to frame important parts of the paragraph and help users to focus on the topic instead of wasting time on what to type next.

Predicting the next word for a movie website that can predict movies name by just their initials, So that we uses Recurrent neural networks. Since basic recurrent neural networks have a lot of flows, we go for LSTM.

WE set up a multi-layer LSTM in TensorFlow with 100 units per layer and 2 LSTM layers. The input to the LSTM is the last 5 words and the target for LSTM is the next word. The final layer in the model is a dense layer that predicts the likelihood of each word.

Here we can make sure of having longer memory of what words are important with help of those three gates we saw earlier.



IV. Hardware/ Software Requirements

Hardware

- A Windows 7 or above x64 Operating system (here we use windows 10 x64 bit OS)
- 4 Gigabytes of RAM or better

Software

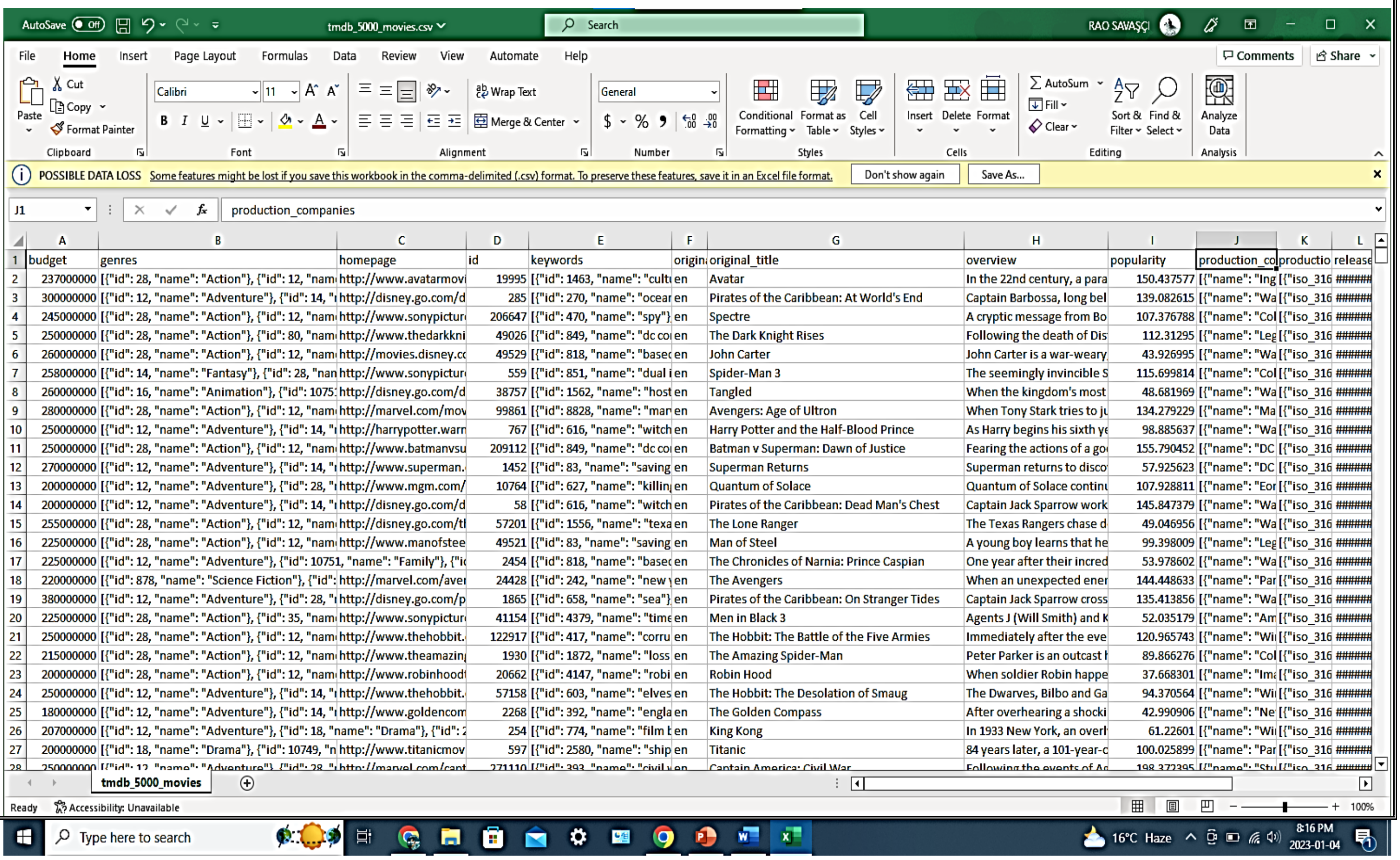
- Jupyter notebook (Anaconda), Google Colab

==

V. WORK ANALYSIS

DATA DOWNLOADING:

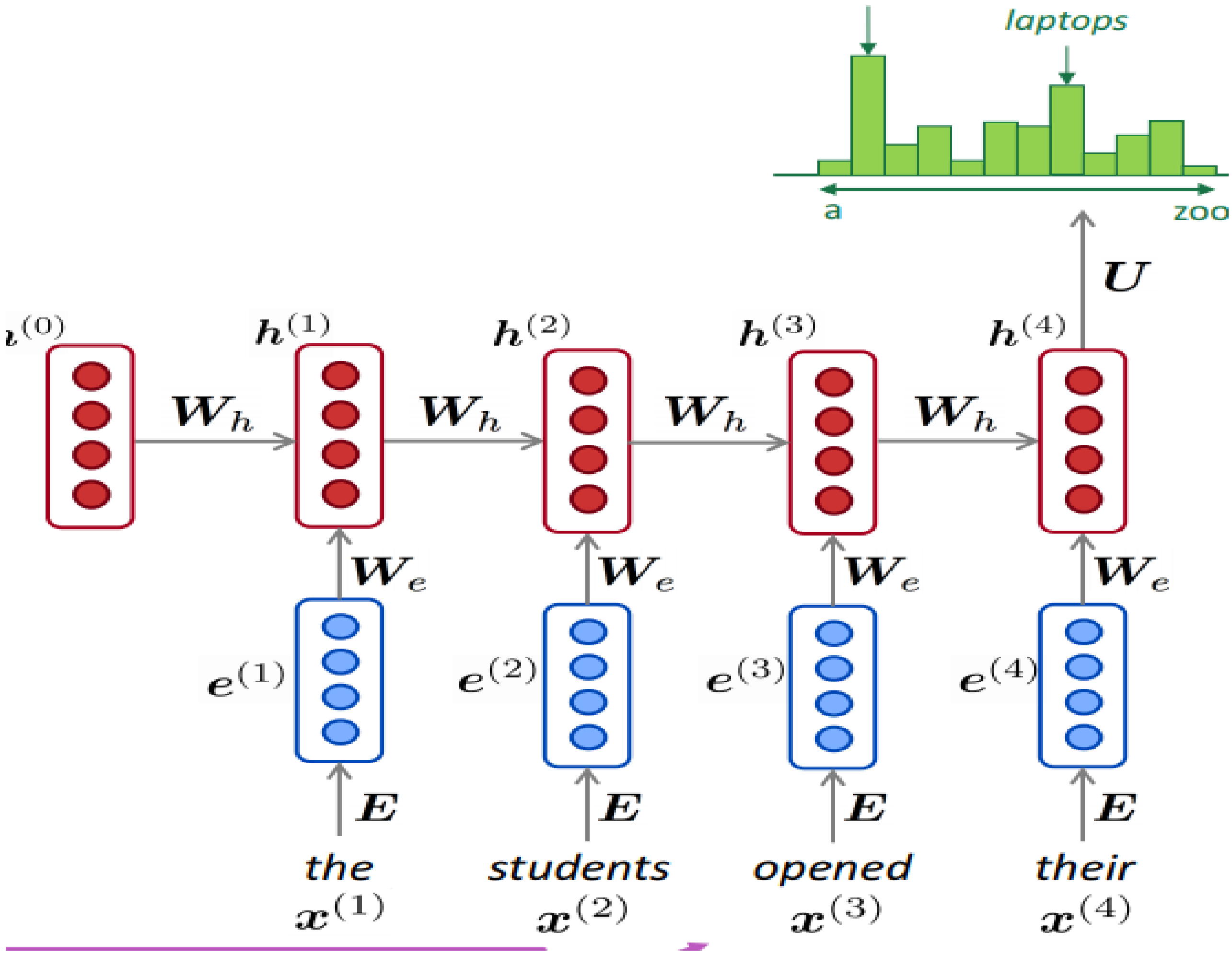
We used the movies dataset which contain 5000 movies. The dataset is quite huge with a total of 5000 movies description. But we use only original title column for the purpose of testing and building a next word prediction model.



	A	B	C	D	E	F	G	H	I	J	K	L
1	budget	genres	homepage	id	keywords	origin	original_title	overview	popularity	production_co	productio	release
2	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://www.avatarmovi	19995	[{"id": 1463, "name": "cultu	en	Avatar	In the 22nd century, a para	150.437577	[{"name": "Ing	[{"iso_316	######
3	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "I	http://disney.go.com/d	285	[{"id": 270, "name": "ocear	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long bel	139.082615	[{"name": "Wa	[{"iso_316	######
4	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://www.sonypictur	206647	[{"id": 470, "name": "spy"}	en	Spectre	A cryptic message from Bo	107.376788	[{"name": "Col	[{"iso_316	######
5	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam	http://www.thedarkkni	49026	[{"id": 849, "name": "dc co	en	The Dark Knight Rises	Following the death of Dis	112.31295	[{"name": "Leg	[{"iso_316	######
6	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://movies.disney.cc	49529	[{"id": 818, "name": "basec	en	John Carter	John Carter is a war-weary	43.926995	[{"name": "Wa	[{"iso_316	######
7	258000000	[{"id": 14, "name": "Fantasy"}, {"id": 28, "nan	http://www.sonypictur	559	[{"id": 851, "name": "dual i	en	Spider-Man 3	The seemingly invincible S	115.699814	[{"name": "Col	[{"iso_316	######
8	260000000	[{"id": 16, "name": "Animation"}, {"id": 1075	http://disney.go.com/d	38757	[{"id": 1562, "name": "host	en	Tangled	When the kingdom's most	48.681969	[{"name": "Wa	[{"iso_316	######
9	280000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://marvel.com/mov	99861	[{"id": 8828, "name": "mar	en	Avengers: Age of Ultron	When Tony Stark tries to ju	134.279229	[{"name": "Ma	[{"iso_316	######
10	250000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "I	http://harrypotter.warn	767	[{"id": 616, "name": "witch	en	Harry Potter and the Half-Blood Prince	As Harry begins his sixth ye	98.885637	[{"name": "Wa	[{"iso_316	######
11	250000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://www.batmanvsu	209112	[{"id": 849, "name": "dc co	en	Batman v Superman: Dawn of Justice	Fearing the actions of a go	155.790452	[{"name": "DC	[{"iso_316	######
12	270000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "I	http://www.superman.	1452	[{"id": 83, "name": "saving	en	Superman Returns	Superman returns to disco	57.925623	[{"name": "DC	[{"iso_316	######
13	200000000	[{"id": 12, "name": "Adventure"}, {"id": 28, "I	http://www.mgm.com/	10764	[{"id": 627, "name": "killin	en	Quantum of Solace	Quantum of Solace contin	107.928811	[{"name": "Eor	[{"iso_316	######
14	200000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "I	http://disney.go.com/d	58	[{"id": 616, "name": "witch	en	Pirates of the Caribbean: Dead Man's Chest	Captain Jack Sparrow work	145.847379	[{"name": "Wa	[{"iso_316	######
15	255000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://disney.go.com/tl	57201	[{"id": 1556, "name": "texa	en	The Lone Ranger	The Texas Rangers chase d	49.046956	[{"name": "Wa	[{"iso_316	######
16	225000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://www.manofstee	49521	[{"id": 83, "name": "saving	en	Man of Steel	A young boy learns that he	99.398009	[{"name": "Leg	[{"iso_316	######
17	225000000	[{"id": 12, "name": "Adventure"}, {"id": 10751, "name": "Family"}, {"i		2454	[{"id": 818, "name": "basec	en	The Chronicles of Narnia: Prince Caspian	One year after their incred	53.978602	[{"name": "Wa	[{"iso_316	######
18	220000000	[{"id": 878, "name": "Science Fiction"}, {"id":	http://marvel.com/ave	24428	[{"id": 242, "name": "new y	en	The Avengers	When an unexpected ener	144.448633	[{"name": "Par	[{"iso_316	######
19	380000000	[{"id": 12, "name": "Adventure"}, {"id": 28, "I	http://disney.go.com/p	1865	[{"id": 658, "name": "sea"}	en	Pirates of the Caribbean: On Stranger Tides	Captain Jack Sparrow cross	135.413856	[{"name": "Wa	[{"iso_316	######
20	225000000	[{"id": 28, "name": "Action"}, {"id": 35, "nam	http://www.sonypictur	41154	[{"id": 4379, "name": "time	en	Men in Black 3	Agents J (Will Smith) and K	52.035179	[{"name": "Am	[{"iso_316	######
21	250000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://www.thehobbit.	122917	[{"id": 417, "name": "corru	en	The Hobbit: The Battle of the Five Armies	Immediately after the eve	120.965743	[{"name": "Wi	[{"iso_316	######
22	215000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://www.theamazin	1930	[{"id": 1872, "name": "loss	en	The Amazing Spider-Man	Peter Parker is an outcast	89.866276	[{"name": "Col	[{"iso_316	######
23	200000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam	http://www.robinhood	20662	[{"id": 4147, "name": "robi	en	Robin Hood	When soldier Robin happe	37.668301	[{"name": "Im	[{"iso_316	######
24	250000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "I	http://www.thehobbit.	57158	[{"id": 603, "name": "elves	en	The Hobbit: The Desolation of Smaug	The Dwarves, Bilbo and Ga	94.370564	[{"name": "Wi	[{"iso_316	######
25	180000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "I	http://www.goldencom	2268	[{"id": 392, "name": "engla	en	The Golden Compass	After overhearing a shocki	42.990906	[{"name": "Ne	[{"iso_316	######
26	207000000	[{"id": 12, "name": "Adventure"}, {"id": 18, "name": "Drama"}, {"id":		254	[{"id": 774, "name": "film t	en	King Kong	In 1933 New York, an overl	61.22601	[{"name": "Wi	[{"iso_316	######
27	200000000	[{"id": 18, "name": "Drama"}, {"id": 10749, "n	http://www.titanicmov	597	[{"id": 2580, "name": "ship	en	Titanic	84 years later, a 101-year-c	100.025899	[{"name": "Par	[{"iso_316	######

MODEL ARCHITECTURE:

For this task we use a RNN since we would like to predict each word by looking at words that come before it and RNNs are able to maintain a hidden state that can transfer information from one time step to the next.



HOW RNN WORKS:

A simple RNN has a weights matrix W_h and an Embedding to hidden matrix We that is the shared at each timestep. Each hidden state is calculated as and the output at any timestep depends on the hidden state as So using this architecture the RNN can "theoretically" use information from the past in predicting future.

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e \boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \mathrm{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

LSTM

Since basic recurrent neural networks have a lot of flows, we go for LSTM.

WE set up a multi-layer LSTM in TensorFlow with 100 units per layer and 2 LSTM layers. The input to the LSTM is the last 5 words and the target for LSTM is the next word. The final layer in the model is a dense layer that predicts the likelihood of each word.

```
2]  model = tf.keras.Sequential([
        tf.keras.layers.Embedding(vocab_size, 14),
        tf.keras.layers.LSTM(100, return_sequences=True),
        tf.keras.layers.LSTM(100),
        tf.keras.layers.Dense(100, activation='relu'),
        tf.keras.layers.Dense(vocab_size, activation='softmax'),
    ])
```

```
3]  model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 14)          70630

 lstm (LSTM)                 (None, None, 100)         46000

 lstm_1 (LSTM)               (None, 100)               80400

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 5045)              509545

=================================================================
Total params: 716,675
Trainable params: 716,675
Non-trainable params: 0
_________________________________________________________________
```

MODEL TRAINING:

The loss function I used was sequence_loss. The model was trained for 150 epoc

We looked at both train loss and the train accuracy to measure the progress of training. accuracy is the typical metric used to measure the performance of a model.

Higher the accuracy, the better the model is. After training for 150 epochs, the model attained a accuracy of 63%.

VI. Code snippet

nwp.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ 2s [1]

```
import pandas as pd
import tensorflow as tf
import numpy as np
```

✓ 3m [2]

```
#url = "https://github.com//pritishmishra703//NextWordPredictor//blob//master//tmdb_5000_movies.csv"
#df = pd.read_csv(url)
from google.colab import files
uploaded = files.upload()
```

Choose Files tmdb_5000_movies.csv

- **tmdb_5000_movies.csv**(text/csv) - 5698602 bytes, last modified: 6/10/2021 - 100% done

```
Saving tmdb_5000_movies.csv to tmdb_5000_movies.csv
```

✓ 0s [3]

```
import pandas as pd
import io
df = pd.read_csv(io.BytesIO(uploaded['tmdb_5000_movies.csv']))

df.head(3)
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_companies	production_countries
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.437577	[{"name": "Ingenious Film Partners", "id": 289...	[{"iso_3166_1": "US", "name": "United States o...
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt Disney Pictures", "id": 2}, {"...	[{"iso_3166_1": "US", "name": "United States o...
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818,	en	Spectre	A cryptic message from Bond's past sends him	107.376788	[{"name": "Columbia Pictures", "id": 5}, {"nam...	[{"iso_3166_1": "GB", "name": "United Kingdom"...

✓ 0s [4]

```
df = df['original_title']
```

✓ 0s [5]

```
df
```

```
0                                         Avatar
1       Pirates of the Caribbean: At World's End
2                                        Spectre
3                          The Dark Knight Rises
4                                    John Carter
                          ...
4798                                 El Mariachi
4799                                   Newlyweds
4800                   Signed, Sealed, Delivered
4801                            Shanghai Calling
4802                           My Date with Drew
Name: original_title, Length: 4803, dtype: object
```

```
[6]  movie_name = df.to_list()
```

```
[7]  movie_name
```

```
'Firewall',
'Absolute Power',
'G.I. Jane',
'The Game',
'Silent Hill',
'The Replacements',
'American Reunion',
'The Negotiator',
'Into the Storm',
'Beverly Hills Cop III',
'Gremlins 2: The New Batch',
'The Judge',
'The Peacemaker',
'Resident Evil: Apocalypse',
'Bridget Jones: The Edge of Reason',
'Out of Time',
'On Deadly Ground',
'The Adventures of Sharkboy and Lavagirl',
'The Beach',
'Raising Helen',
'Ninja Assassin',
'For Love of the Game',
'Striptease',
'Marmaduke',
'Hereafter',
'Murder by Numbers',
'Assassins',
'Hannibal Rising',
'The Story of Us',
'The Iron Giant',
'The Life Aquatic with Steve Zissou',
'Free State of Jones',
'The Life of David Gale',
'Man of the House',
'Run All Night',
'Eastern Promises',
'Into the Blue',
'Joan of Arc',
'Your Highness',
'Dream House',
'Mad City',
"Baby's Day Out",
'The Scarlet Letter',
'Fair Game',
'Domino',
'Jade',
'Gamer',
'Beautiful Creatures',
'Death to Smoochy',
'Zoolander 2',
'The Big Bounce',
'What Planet Are You From?',
...]
```

[8]

```
tokenizer = tf.keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(movie_name)
seq = tokenizer.texts_to_sequences(movie_name)
```

[9]

```
seq[:10]
```

```
[[1564],
 [210, 2, 1, 431, 47, 432, 72],
 [1565],
 [1, 52, 211, 1566],
 [212, 601],
 [213, 8, 21],
 [1567],
 [902, 146, 2, 1568],
 [110, 214, 4, 1, 433, 53, 147],
 [173, 340, 261, 85, 2, 903]]
```

[10]

```
tokenizer.word_index
```

```
 'hulk': 944,
 'within': 945,
 'commander': 946,
 'breaking': 947,
 'incredible': 948,
 'ant': 949,
 'worlds': 950,
 'p': 951,
 'da': 952,
 'rio': 953,
 'silver': 954,
 'pi': 955,
 "charlie's": 956,
 'stuart': 957,
 'dinosaurs': 996,
 'walter': 997,
 'tattoo': 998,
 'atlantis': 999,
 'intelligence': 1000,
 ...}
```

[11]

```
X = []
y = []
total_words_dropped = 0

for i in seq:
    if len(i) > 1:
        for index in range(1, len(i)):
            X.append(i[:index])
            y.append(i[index])
    else:
        total_words_dropped += 1

print("Total Single Words Dropped are:", total_words_dropped)
```



```
Total Single Words Dropped are: 1003
```

```
[12]  1  X[:10]
```

```
[[210],
 [210, 2],
 [210, 2, 1],
 [210, 2, 1, 431],
 [210, 2, 1, 431, 47],
 [210, 2, 1, 431, 47, 432],
 [1],
 [1, 52],
 [1, 52, 211],
 [212]]
```

```
[13]  1  y[:10]
```

```
[2, 1, 431, 47, 432, 72, 52, 211, 1566, 601]
```

```
[14]  1  X = tf.keras.preprocessing.sequence.pad_sequences(X)
```

```
[15]  1  X
```

```
array([[  0,   0,   0, ...,   0,   0, 210],
       [  0,   0,   0, ...,   0, 210,   2],
       [  0,   0,   0, ..., 210,   2,   1],
       ...,
       [  0,   0,   0, ...,   0,   0,  14],
       [  0,   0,   0, ...,   0,  14, 300],
       [  0,   0,   0, ...,  14, 300,  11]], dtype=int32)
```

```
[16]  1  X.shape
```

```
(8483, 14)
```

```
[17]  1  y = tf.keras.utils.to_categorical(y)
```

```
[18]  1  y
```

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

```
[19]  1  y.shape
```

```
(8483, 5045)
```

```
[20]  1  vocab_size = len(tokenizer.word_index) + 1
```

[21]

```
vocab_size
```

```
5045
```

[22]

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, 14),
    tf.keras.layers.LSTM(100, return_sequences=True),
    tf.keras.layers.LSTM(100),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(vocab_size, activation='softmax'),
])
```

[23]

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 14)          70630

 lstm (LSTM)                 (None, None, 100)         46000

 lstm_1 (LSTM)               (None, 100)               80400

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 5045)              509545

=================================================================
Total params: 716,675
Trainable params: 716,675
Non-trainable params: 0
_________________________________________________________________
```

[24]

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.004),
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```

[25]

```
model.fit(X, y, epochs=150)
```

```
Epoch 1/150
266/266 [==============================] - 14s 36ms/step - loss: 7.7075 - accuracy: 0.0569
Epoch 2/150
266/266 [==============================] - 9s 36ms/step - loss: 7.0084 - accuracy: 0.0611
Epoch 3/150
266/266 [==============================] - 9s 36ms/step - loss: 6.7492 - accuracy: 0.0756
Epoch 4/150
266/266 [==============================] - 9s 35ms/step - loss: 6.5865 - accuracy: 0.0885
Epoch 5/150
266/266 [==============================] - 9s 36ms/step - loss: 6.4603 - accuracy: 0.0963
Epoch 6/150
266/266 [==============================] - 9s 35ms/step - loss: 6.3455 - accuracy: 0.1004
Epoch 7/150
266/266 [==============================] - 9s 35ms/step - loss: 6.2370 - accuracy: 0.1044
```

.....
....
...
..
.

```
266/266 [==============================] - 10s 36ms/step - loss: 1.8243 - accuracy: 0.5553
Epoch 134/150
266/266 [==============================] - 10s 36ms/step - loss: 1.8100 - accuracy: 0.5591
Epoch 135/150
266/266 [==============================] - 10s 36ms/step - loss: 1.8013 - accuracy: 0.5611
Epoch 136/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7899 - accuracy: 0.5711
Epoch 137/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7927 - accuracy: 0.5645
Epoch 138/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7707 - accuracy: 0.5741
Epoch 139/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7690 - accuracy: 0.5681
Epoch 140/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7819 - accuracy: 0.5641
Epoch 141/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7756 - accuracy: 0.5664
Epoch 142/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7922 - accuracy: 0.5661
Epoch 143/150
266/266 [==============================] - 9s 36ms/step - loss: 1.7828 - accuracy: 0.5638
Epoch 144/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7575 - accuracy: 0.5760
Epoch 145/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7359 - accuracy: 0.5792
Epoch 146/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7150 - accuracy: 0.5889
Epoch 147/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7934 - accuracy: 0.5635
Epoch 148/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7302 - accuracy: 0.5763
Epoch 149/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7216 - accuracy: 0.5807
Epoch 150/150
266/266 [==============================] - 10s 36ms/step - loss: 1.7054 - accuracy: 0.5866
<keras.callbacks.History at 0x7f0ceb1e1c10>
```

[26]

```
model.save('nwp.h5')
```

[27]

```
vocab_array = np.array(list(tokenizer.word_index.keys()))
```

[28]

```
vocab_array
```

```
array(['the', 'of', 'a', ..., 'signed', 'sealed', 'delivered'],
      dtype='<U14')
```

[29]

```
def make_prediction(text, n_words):
    for i in range(n_words):
        text_tokenize = tokenizer.texts_to_sequences([text])
        text_padded = tf.keras.preprocessing.sequence.pad_sequences(text_tokenize, maxlen=14)
        prediction = np.squeeze(np.argmax(model.predict(text_padded), axis=-1))
        prediction = str(vocab_array[prediction - 1])
        print(vocab_array[np.argsort(model.predict(text_padded)) - 1].ravel()[:-3])
        text += " " + prediction
    return text
```

```
make_prediction("Spiderman", 4)
```

```
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
['budapest' "knight's" 'père' ... 'romano' 'nest' 'problems']
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
['budapest' "d'amélie" 'poulain' ... 'with' 'have' '3d']
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
['budapest' 'timothy' 'ri¢h' ... 'raider' '2' 'ever']
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 20ms/step
['budapest' 'china' 'parfait' ... '2' 'all' 'women']
'Spiderman holliday's revenge over the'
```

VII. Results

```
make_prediction("Harry", 4)
```

```
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 35ms/step
['senior' 'behind' 'dumber' ... 'go' 'day' 'like']
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 86ms/step
['budapest' 'spies' 'lines' ... 'on' 'of' 'all']
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 53ms/step
['delivered' 'vincent' 'where's' ... 'women' 'a' 'too']
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
['budapest' 'migrateur' 'crisis' ... 'order' 'chamber' "philosopher's"]
'Harry potter and the goblet'
```

```
make_prediction("Jack", 3)
```

```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
['budapest' 'words' 'match' ... 'the' 'and' 'times']
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
['delivered' 'runaways' 'wardrobe' ... 'girl' 'woman' 'the']
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
['budapest' 'newton' '39' ... 'over' 'lines' 'down']
'Jack ryan shadow recruit'
```

VIII. Conclusion & Future Work

- Understanding the paragraph using machine learning algorithms like RNN can help soon to understand and frame paragraphs and stories on their own.
- Creating lyrics and songs can be a major field in which this algorithm can help the end-users to predict the next phrase in songs considering the model is train on a music lyrics data set.
- As more data we can train the model which will reevaluate the weights to understand the core features of paragraphs/sentences to predict good results.

Standard RNNs and other language models become less exact when the hole between the specific circumstance and the word to be anticipated increments. Here's when LSTM comes being used to handle the drawn-out reliance issue since it has memory cells to recall the past setting. You can study LSTM Neural Net. Our task in this project is to train and try an algorithm that best fit this task and mostly we are looking forward to implementing an LSTM to get good accuracy as this task is quite complex because we must predict the user's future text which he will be thinking.

This project report presents how the system is predicting and correcting the next/target words using some mechanisms and using TensorFlow closed-loop system, the scalability of a trained system can be increased and using the perplexity concept the system will decide that the sentence is having more misspelled, and the performance of the system can be increased.

THANK YOU